unit3_part_1-datawrangling

November 3, 2022

1 data wrangling-I

[1]:
```
import pandas as pd
import numpy as np
```

#to upload files in content folder

[57]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving subjects-count-course.xlsx to subjects-count-course (1).xlsx
```

2 Reading a xls having details regarding number of core/dse/sec subjects

[2]:
```
f1 = pd.ExcelFile('D:\cs(h)Vsem Data analysis and visulaization␣
↪2021\programs\pandas\subjects-count-course.xlsx')
```

[3]:
```
DF=pd.read_excel(f1,sheet_name=0)
```

[5]:
```
DF
```

[5]:

	Course	Semester	Corepaper	SEC	DSE
0	PSCS	I	3	NIL	NIL
1	PSCS	II	3	NIL	NIL
2	PSCS	III	4	1	NIL
3	PSCS	IV	4	1	NIL
4	PSCS	V	4	1	1
5	PSCS	VI	4	1	1
6	CSHons	I	4	2	NIL
7	CSHons	II	4	2	NIL
8	CSHons	III	4	2	NIL
9	CSHons	IV	4	2	NIL
10	CSHons	V	4	NIL	2
11	CSHons	VI	4	NIL	2
12	Bcom	I	4	2	NIL

```
13    Bcom       II          4    2  NIL
14    Bcom      III          4    3    1
15    Bcom       IV          4    3  NIL
16  lifesc        I          5  NIL    1
17  lifesc       II          5    0    1
```

3 hierarchical indexing

3.1 setting one column as row index:single level index

[4]:
```
DF1=DF.set_index(DF.columns[0])
```

[7]:
```
DF1.index
```

[7]:
```
Index(['PSCS', 'PSCS', 'PSCS', 'PSCS', 'PSCS', 'PSCS', 'CSHons', 'CSHons',
       'CSHons', 'CSHons', 'CSHons', 'Bcom', 'Bcom', 'Bcom', 'Bcom',
       'lifesc', 'lifesc'],
      dtype='object', name='Course')
```

[5]:
```
DF1
```

[5]:

Course	Semester	Corepaper	SEC	DSE
PSCS	I	3	NIL	NIL
PSCS	II	3	NIL	NIL
PSCS	III	4	1	NIL
PSCS	IV	4	1	NIL
PSCS	V	4	1	1
PSCS	VI	4	1	1
CSHons	I	4	2	NIL
CSHons	II	4	2	NIL
CSHons	III	4	2	NIL
CSHons	IV	4	2	NIL
CSHons	V	4	NIL	2
CSHons	VI	4	NIL	2
Bcom	I	4	2	NIL
Bcom	II	4	2	NIL
Bcom	III	4	3	1
Bcom	IV	4	3	NIL
lifesc	I	5	NIL	1
lifesc	II	5	0	1

3.2 setting two columns as row index:hierarchical level index

[6]:
```
DF2=DF.set_index(keys=[DF.columns[0],DF.columns[1]])
```

[9]:
```
I=DF2.index
```

[12]:
```
I.levels
```

[12]:
```
FrozenList([['Bcom', 'CSHons', 'PSCS', 'lifesc'], ['I', 'II', 'III', 'IV', 'V',
'VI']])
```

[13]:
```
I.codes
```

[13]:
```
FrozenList([[2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 3, 3], [0, 1, 2, 3,
4, 5, 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 0, 1]])
```

[10]:
```
DF2
```

[10]:

Course	Semester	Corepaper	SEC	DSE
PSCS	I	3	NIL	NIL
	II	3	NIL	NIL
	III	4	1	NIL
	IV	4	1	NIL
	V	4	1	1
	VI	4	1	1
CSHons	I	4	2	NIL
	II	4	2	NIL
	III	4	2	NIL
	IV	4	2	NIL
	V	4	NIL	2
	VI	4	NIL	2
Bcom	I	4	2	NIL
	II	4	2	NIL
	III	4	3	1
	IV	4	3	NIL
lifesc	I	5	NIL	1
	II	5	0	1

[14]:
```
DF2.index.names
```

[14]:
```
FrozenList(['Course', 'Semester'])
```

3.3 Accessing values using multiindex using tuple()

[15]:
```
DF2.loc[('Bcom','I')]
```

[15]:
```
Corepaper      4
SEC            2
DSE          NIL
Name: (Bcom, I), dtype: object
```

[48]:
```
DF2
```

[48]:
```
                corepaper  SEC  DSE
course Semester
PSCS   I                3  NIL  NIL
       II               3  NIL  NIL
       III              4    1  NIL
       IV               4    1  NIL
       V                4    1    1
       VI               4    1    1
CSHons I                4    2  NIL
       II               4    2  NIL
       III              4    2  NIL
       IV               4    2  NIL
       V                4  NIL    2
       VI               4  NIL    2
Bcom   I                4    2  NIL
       II               4    2  NIL
       III              4    3    1
       IV               4    3  NIL
lifesc I                5  NIL    1
       II               5    0    1
```

[16]:
```
DF2.shape
```

[16]:
```
(18, 3)
```

3.4 accessing rows of a particular course index

[17]:
```
DF2.loc['Bcom']
```

[17]:
```
          Corepaper SEC  DSE
Semester
I                 4   2  NIL
II                4   2  NIL
III               4   3    1
IV                4   3  NIL
```

[18]:
```
DF2.loc[['Bcom','PSCS'],:]
```

[18]:
```
                Corepaper  SEC  DSE
Course Semester
Bcom   I                4    2  NIL
       II               4    2  NIL
       III              4    3    1
       IV               4    3  NIL
PSCS   I                3  NIL  NIL
       II               3  NIL  NIL
       III              4    1  NIL
```

```
        IV                4    1  NIL
        V                 4    1    1
        VI                4    1    1
```

[19]:
```
DF2.loc[('Bcom','I'),:]
```

[19]:
```
Corepaper      4
SEC            2
DSE          NIL
Name: (Bcom, I), dtype: object
```

3.5 find the output?

[20]:
```
DF2.loc[('Bcom',['I','II']),:]
```

[20]:
```
                 Corepaper SEC  DSE
Course Semester
Bcom   I                 4   2  NIL
       II                4   2  NIL
```

[21]:
```
type(I)
```

[21]:
```
pandas.core.indexes.multi.MultiIndex
```

[23]:
```
I.levels[0].names
```

[23]:
```
FrozenList(['Course'])
```

[56]:
```
DF2
```

[56]:
```
                 corepaper  SEC  DSE
course Semester
PSCS   I                 3  NIL  NIL
       II                3  NIL  NIL
       III               4    1  NIL
       IV                4    1  NIL
       V                 4    1    1
       VI                4    1    1
CSHons I                 4    2  NIL
       II                4    2  NIL
       III               4    2  NIL
       IV                4    2  NIL
       V                 4  NIL    2
       VI                4  NIL    2
Bcom   I                 4    2  NIL
       II                4    2  NIL
       III               4    3    1
       IV                4    3  NIL
```

```
lifesc I                5  NIL    1
       II               5    0    1
```

[24]:
```
DF2.values
```

[24]:
```
array([[3, 'NIL', 'NIL'],
       [3, 'NIL', 'NIL'],
       [4, 1, 'NIL'],
       [4, 1, 'NIL'],
       [4, 1, 1],
       [4, 1, 1],
       [4, 2, 'NIL'],
       [4, 2, 'NIL'],
       [4, 2, 'NIL'],
       [4, 2, 'NIL'],
       [4, 'NIL', 2],
       [4, 'NIL', 2],
       [4, 2, 'NIL'],
       [4, 2, 'NIL'],
       [4, 3, 1],
       [4, 3, 'NIL'],
       [5, 'NIL', 1],
       [5, 0, 1]], dtype=object)
```

[25]:
```
DF2.index[0]
```

[25]:
```
('PSCS', 'I')
```

4 dropping a particular level in Hierarcchy

[27]:
```
DF2.droplevel(1)
```

[27]:

Course	Corepaper	SEC	DSE
PSCS	3	NIL	NIL
PSCS	3	NIL	NIL
PSCS	4	1	NIL
PSCS	4	1	NIL
PSCS	4	1	1
PSCS	4	1	1
CSHons	4	2	NIL
CSHons	4	2	NIL
CSHons	4	2	NIL
CSHons	4	2	NIL
CSHons	4	NIL	2
CSHons	4	NIL	2
Bcom	4	2	NIL

```
Bcom            4    2  NIL
Bcom            4    3    1
Bcom            4    3  NIL
lifesc          5  NIL    1
lifesc          5    0    1
```

5 unstacking inner level

[28]:
```
DF2.unstack()
```

[28]:
```
         Corepaper                          SEC                          \
Semester         I   II  III   IV    V   VI    I   II  III   IV    V   VI
Course
Bcom           4.0  4.0  4.0  4.0  NaN  NaN    2    2    3    3  NaN  NaN
CSHons         4.0  4.0  4.0  4.0  4.0  4.0    2    2    2    2  NIL  NIL
PSCS           3.0  3.0  4.0  4.0  4.0  4.0  NIL  NIL    1    1    1    1
lifesc         5.0  5.0  NaN  NaN  NaN  NaN  NIL    0  NaN  NaN  NaN  NaN

          DSE
Semester    I   II  III   IV    V   VI
Course
Bcom      NIL  NIL    1  NIL  NaN  NaN
CSHons    NIL  NIL  NIL  NIL    2    2
PSCS      NIL  NIL  NIL  NIL    1    1
lifesc      1    1  NaN  NaN  NaN  NaN
```

[29]:
```
DF2.unstack().stack()
```

[29]:
```
                Corepaper  SEC  DSE
Course Semester
Bcom   I              4.0    2  NIL
       II             4.0    2  NIL
       III            4.0    3    1
       IV             4.0    3  NIL
CSHons I              4.0    2  NIL
       II             4.0    2  NIL
       III            4.0    2  NIL
       IV             4.0    2  NIL
       V              4.0  NIL    2
       VI             4.0  NIL    2
PSCS   I              3.0  NIL  NIL
       II             3.0  NIL  NIL
       III            4.0    1  NIL
       IV             4.0    1  NIL
       V              4.0    1    1
       VI             4.0    1    1
```

```
lifesc I              5.0  NIL    1
       II             5.0    0    1
```

6 making two levels in columns

[30]:
```
DF2.columns.values
```

[30]:
```
array(['Corepaper', 'SEC', 'DSE'], dtype=object)
```

[31]:
```
newcol = zip(DF2.columns.values,[100,50,100])
newcol
```

[31]:
```
<zip at 0x20caa4cdd80>
```

[32]:
```
DF2.columns = pd.MultiIndex.from_tuples(newcol, names=['CL1','CL2'])
```

[33]:
```
DF2
```

[33]:

CL1		Corepaper	SEC	DSE
CL2		100	50	100
Course	Semester			
PSCS	I	3	NIL	NIL
	II	3	NIL	NIL
	III	4	1	NIL
	IV	4	1	NIL
	V	4	1	1
	VI	4	1	1
CSHons	I	4	2	NIL
	II	4	2	NIL
	III	4	2	NIL
	IV	4	2	NIL
	V	4	NIL	2
	VI	4	NIL	2
Bcom	I	4	2	NIL
	II	4	2	NIL
	III	4	3	1
	IV	4	3	NIL
lifesc	I	5	NIL	1
	II	5	0	1

[34]:
```
DF2.columns
```

[34]:
```
MultiIndex([('Corepaper', 100),
            (      'SEC',  50),
            (      'DSE', 100)],
           names=['CL1', 'CL2'])
```

6.1 retreiving values of row/colum index

[35]:
```
DF2.index.get_level_values(0)
```

[35]:
```
Index(['PSCS', 'PSCS', 'PSCS', 'PSCS', 'PSCS', 'PSCS', 'CSHons', 'CSHons',
       'CSHons', 'CSHons', 'CSHons', 'CSHons', 'Bcom', 'Bcom', 'Bcom', 'Bcom',
       'lifesc', 'lifesc'],
      dtype='object', name='Course')
```

[37]:
```
DF2.columns.get_level_values(0)
```

[37]:
```
Index(['Corepaper', 'SEC', 'DSE'], dtype='object', name='CL1')
```

6.2 accessing a particular column label

[38]:
```
DF2.columns.levels
```

[38]:
```
FrozenList([['Corepaper', 'DSE', 'SEC'], [50, 100]])
```

[40]:
```
DF2.columns.levels[1][1]
```

[40]:
```
100
```

[44]:
```
type(DF2['SEC'])
```

[44]:
```
pandas.core.frame.DataFrame
```

[46]:
```
DF2['SEC']
```

[46]:
```
CL2                50
Course Semester
PSCS   I          NIL
       II         NIL
       III          1
       IV           1
       V            1
       VI           1
CSHons I            2
       II           2
       III          2
       IV           2
       V          NIL
       VI         NIL
Bcom   I            2
       II           2
       III          3
       IV           3
lifesc I          NIL
```

```
        II          0
```

[45]:
```
type(DF2['SEC'][50])
```

[45]:
```
pandas.core.series.Series
```

7 dropping a column level

[47]:
```
DF2.droplevel(1,axis=1)
```

[47]:
```
CL1              Corepaper  SEC  DSE
Course Semester
PSCS   I                 3  NIL  NIL
       II                3  NIL  NIL
       III               4    1  NIL
       IV                4    1  NIL
       V                 4    1    1
       VI                4    1    1
CSHons I                 4    2  NIL
       II                4    2  NIL
       III               4    2  NIL
       IV                4    2  NIL
       V                 4  NIL    2
       VI                4  NIL    2
Bcom   I                 4    2  NIL
       II                4    2  NIL
       III               4    3    1
       IV                4    3  NIL
lifesc I                 5  NIL    1
       II                5    0    1
```

8 Reordering and Sorting Levels

[49]:
```
DF2.swaplevel('Course', 'Semester')
```

[49]:
```
CL1              Corepaper  SEC  DSE
CL2                    100   50  100
Semester Course
I        PSCS            3  NIL  NIL
II       PSCS            3  NIL  NIL
III      PSCS            4    1  NIL
IV       PSCS            4    1  NIL
V        PSCS            4    1    1
VI       PSCS            4    1    1
I        CSHons          4    2  NIL
II       CSHons          4    2  NIL
```

```
III      CSHons          4    2  NIL
IV       CSHons          4    2  NIL
V        CSHons          4  NIL    2
VI       CSHons          4  NIL    2
I        Bcom            4    2  NIL
II       Bcom            4    2  NIL
III      Bcom            4    3    1
IV       Bcom            4    3  NIL
I        lifesc          5  NIL    1
II       lifesc          5    0    1
```

[50]:
```
DF2.sort_index(level=1)
```

[50]:
```
CL1             Corepaper  SEC  DSE
CL2                   100   50  100
Course Semester
Bcom   I                4    2  NIL
CSHons I                4    2  NIL
PSCS   I                3  NIL  NIL
lifesc I                5  NIL    1
Bcom   II               4    2  NIL
CSHons II               4    2  NIL
PSCS   II               3  NIL  NIL
lifesc II               5    0    1
Bcom   III              4    3    1
CSHons III              4    2  NIL
PSCS   III              4    1  NIL
Bcom   IV               4    3  NIL
CSHons IV               4    2  NIL
PSCS   IV               4    1  NIL
CSHons V                4  NIL    2
PSCS   V                4    1    1
CSHons VI               4  NIL    2
PSCS   VI               4    1    1
```

[78]:
```
DF2
```

[78]:
```
CL1             corepaper  SEC  DSE
CL2                   100  100   50
course Semester
PSCS   I                3  NIL  NIL
       II               3  NIL  NIL
       III              4    1  NIL
       IV               4    1  NIL
       V                4    1    1
       VI               4    1    1
CSHons I                4    2  NIL
```

```
       II              4    2  NIL
       III             4    2  NIL
       IV              4    2  NIL
       V               4  NIL    2
       VI              4  NIL    2
Bcom   I               4    2  NIL
       II              4    2  NIL
       III             4    3    1
       IV              4    3  NIL
lifesc I               5  NIL    1
       II              5    0    1
```

9 sorting each row on second column level

[51]:
```
DF2.sort_index(level=1,axis=1)
```

[51]:
```
CL1                SEC Corepaper  DSE
CL2                50         100  100
Course Semester
PSCS   I          NIL           3  NIL
       II         NIL           3  NIL
       III          1           4  NIL
       IV           1           4  NIL
       V            1           4    1
       VI           1           4    1
CSHons I            2           4  NIL
       II           2           4  NIL
       III          2           4  NIL
       IV           2           4  NIL
       V          NIL           4    2
       VI         NIL           4    2
Bcom   I            2           4  NIL
       II           2           4  NIL
       III          3           4    1
       IV           3           4  NIL
lifesc I          NIL           5    1
       II           0           5    1
```

10 Get Summary Statistics by Level

[52]:
```
DF2=DF2.replace({'NIL':None})
```

10.1 query is to find total papers in each course

[53]:
```
DF2
```

[53]:
```
CL1             Corepaper  SEC  DSE
CL2                   100  50   100
Course Semester
PSCS   I                3  NaN  NaN
       II               3  NaN  NaN
       III              4  1.0  NaN
       IV               4  1.0  NaN
       V                4  1.0  1.0
       VI               4  1.0  1.0
CSHons I                4  2.0  NaN
       II               4  2.0  NaN
       III              4  2.0  NaN
       IV               4  2.0  NaN
       V                4  NaN  2.0
       VI               4  NaN  2.0
Bcom   I                4  2.0  NaN
       II               4  2.0  NaN
       III              4  3.0  1.0
       IV               4  3.0  NaN
lifesc I                5  NaN  1.0
       II               5  0.0  1.0
```

[56]:
```
DF2.sum(level='Course',skipna=True).astype(int)
```

[56]:
```
CL1    Corepaper SEC DSE
CL2          100 50  100
Course
PSCS          22   4   2
CSHons        24   8   4
Bcom          16  10   1
lifesc        10   0   2
```

10.2 find total papers of marks 100 and 50 each

[59]:
```
DF2.sum(level=0, axis=1,skipna=True)
```

[59]:
```
CL1             Corepaper  SEC  DSE
Course Semester
PSCS   I              3.0  0.0  0.0
       II             3.0  0.0  0.0
       III            4.0  1.0  0.0
       IV             4.0  1.0  0.0
       V              4.0  1.0  1.0
       VI             4.0  1.0  1.0
```

```
CSHons I              4.0  2.0  0.0
       II             4.0  2.0  0.0
       III            4.0  2.0  0.0
       IV             4.0  2.0  0.0
       V              4.0  0.0  2.0
       VI             4.0  0.0  2.0
Bcom   I              4.0  2.0  0.0
       II             4.0  2.0  0.0
       III            4.0  3.0  1.0
       IV             4.0  3.0  0.0
lifesc I              5.0  0.0  1.0
       II             5.0  0.0  1.0
```

11 Combining and merging dataframes merge() join() concat()

- creating DF 1 and 2

[7]:
```
df1 = pd.DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'a', 'b'],
                    'data1': range(7)})
df2 = pd.DataFrame({'key': ['a', 'b', 'd'],
                    'data2': range(3)})
```

[8]:
```
df1
```

[8]:
```
  key  data1
0   b      0
1   b      1
2   a      2
3   c      3
4   a      4
5   a      5
6   b      6
```

[9]:
```
df2
```

[9]:
```
  key  data2
0   a      0
1   b      1
2   d      2
```

12 Combining: Merge DataFrame or named Series objects with a database-style join. on common attributes

how : {'left', 'right', 'outer', 'inner'}, default 'inner'

12.1 merge() by default inner joins on all common columns with same values

[13]:
```
pd.merge(df1, df2)
```

[13]:

	key	data1	data2
0	b	0	1
1	b	1	1
2	b	6	1
3	a	2	0
4	a	4	0
5	a	5	0

[64]:
```
pd.merge(df1, df2).sort_values('key')
```

[64]:

	key	data1	data2
3	a	2	0
4	a	4	0
5	a	5	0
0	b	0	1
1	b	1	1
2	b	6	1

[65]:
```
pd.merge(df1,df2,how='outer')
```

[65]:

	key	data1	data2
0	b	0.0	1.0
1	b	1.0	1.0
2	b	6.0	1.0
3	a	2.0	0.0
4	a	4.0	0.0
5	a	5.0	0.0
6	c	3.0	NaN
7	d	NaN	2.0

12.2 adding more columns to dataframe

[11]:
```
df1['new']=np.arange(len(df1))
```

[12]:
```
df2['new']=np.arange(len(df2))+1
```

[6]:
```
df1
```

[6]:

	key	data1	new
0	b	0	0
1	b	1	1
2	a	2	2
3	c	3	3
4	a	4	4

```
5   a      5    5
6   b      6    6
```

[8]:
```
df2
```

[8]:
```
  key  data2  new
0   a      0    1
1   b      1    2
2   d      2    3
```

12.3 Find the output?

[70]:
```
pd.merge(df1, df2)
```

[70]:
```
  key  data1  new  data2
0   b      1    1      1
```

12.4 By deafult, joining is on all common attributes on bot DF. for joining on the specified column as using 'on', other common attributes are renamed as att_x to differentiate

[18]:
```
pd.merge(df1, df2,on=['key'])
```

[18]:
```
  key  data1  new_x  data2  new_y
0   b      0      0      1      1
1   b      1      1      1      1
2   b      6      6      1      1
3   a      2      2      0      0
4   a      4      4      0      0
5   a      5      5      0      0
```

[72]:
```
pd.merge(df1, df2,on=['key','new'])
```

[72]:
```
  key  data1  new  data2
0   b      1    1      1
```

12.5 joining on mentioned attribute and if any other common attribute then suffice it with user-specified name as per apperance

[19]:
```
pd.merge(df1, df2, on='key', suffixes=('_df1', '_df2'))
```

[19]:
```
  key  data1  new_df1  data2  new_df2
0   b      0        0      1        1
1   b      1        1      1        1
2   b      6        6      1        1
3   a      2        2      0        0
4   a      4        4      0        0
```

```
5   a      5        5      0        0
```

12.6 joining over two different attributes in two data frames

[74]:
```
pd.merge(df1, df2, left_on='data1', right_on='data2')
```

[74]:
```
  key_x  data1  new_x key_y  data2  new_y
0     b      0      0     a      0      0
1     b      1      1     b      1      1
2     a      2      2     d      2      2
```

[75]:
```
df1
```

[75]:
```
  key  data1  new
0   b      0    0
1   b      1    1
2   a      2    2
3   c      3    3
4   a      4    4
5   a      5    5
6   b      6    6
```

[87]:
```
df2
```

[87]:
```
  key  data2  new
0   a      0    0
1   b      1    1
2   d     10    2
```

[84]:
```
df2.iloc[2,1]=10
```

13 merging on index

(index of one DF and column of another DF)

[20]:
```
pd.merge(df1, df2, left_on='data1', right_index=True)
```

[20]:
```
  key_x  data1  new_x key_y  data2  new_y
0     b      0      0     a      0      0
1     b      1      1     b      1      1
2     a      2      2     d      2      2
```

[77]:
```
DFnew=pd.merge(df1, df2, left_on='data1', right_index=True)
```

[80]:
```
DFnew
```

[80]:

	key_x	new_x	key_y	data2	new_y
0	b	0	a	0	0
1	b	1	b	1	1
2	a	2	d	2	2

[ ]:
```
type(DFnew)
```

[ ]:
```
pandas.core.frame.DataFrame
```

[79]:
```
DFnew.drop('data1',axis=1,inplace=True)
```

[86]:
```
df1
```

[86]:

	key	data1	new
0	b	0	0
1	b	1	1
2	a	2	2
3	c	3	3
4	a	4	4
5	a	5	5
6	b	6	6

13.1 find the output?

[21]:
```
pd.merge(df1, df2, right_on='data2', left_index=True, how='outer')
```

[21]:

	key_x	data1	new_x	key_y	data2	new_y
0.0	b	0	0	a	0	0.0
1.0	b	1	1	b	1	1.0
2.0	a	2	2	d	2	2.0
NaN	c	3	3	NaN	3	NaN
NaN	a	4	4	NaN	4	NaN
NaN	a	5	5	NaN	5	NaN
NaN	b	6	6	NaN	6	NaN

[89]:
```
pd.merge(df1, df2, how='outer', left_index=True, right_index=True)
```

[89]:

	key_x	data1	new_x	key_y	data2	new_y
0	b	0	0	a	0.0	0.0
1	b	1	1	b	1.0	1.0
2	a	2	2	d	10.0	2.0
3	c	3	3	NaN	NaN	NaN
4	a	4	4	NaN	NaN	NaN
5	a	5	5	NaN	NaN	NaN
6	b	6	6	NaN	NaN	NaN

13.2 merging of dataframe with multilevel index and DF with single level index

[90]:
```
DF2
```

[90]:
```
CL1             Corepaper  SEC  DSE
CL2                   100  50   100
Course Semester
PSCS   I                3  NaN  NaN
       II               3  NaN  NaN
       III              4  1.0  NaN
       IV               4  1.0  NaN
       V                4  1.0  1.0
       VI               4  1.0  1.0
CSHons I                4  2.0  NaN
       II               4  2.0  NaN
       III              4  2.0  NaN
       IV               4  2.0  NaN
       V                4  NaN  2.0
       VI               4  NaN  2.0
Bcom   I                4  2.0  NaN
       II               4  2.0  NaN
       III              4  3.0  1.0
       IV               4  3.0  NaN
lifesc I                5  NaN  1.0
       II               5  0.0  1.0
```

[95]:
```
DF1
```

[95]:
```
        Semester  Corepaper  SEC  DSE     new
Course
PSCS           I          3  NIL  NIL    PSCS
PSCS          II          3  NIL  NIL    PSCS
PSCS         III          4    1  NIL    PSCS
PSCS          IV          4    1  NIL    PSCS
PSCS           V          4    1    1    PSCS
PSCS          VI          4    1    1    PSCS
CSHons         I          4    2  NIL  CSHons
CSHons        II          4    2  NIL  CSHons
CSHons       III          4    2  NIL  CSHons
CSHons        IV          4    2  NIL  CSHons
CSHons         V          4  NIL    2  CSHons
CSHons        VI          4  NIL    2  CSHons
Bcom           I          4    2  NIL    Bcom
Bcom          II          4    2  NIL    Bcom
Bcom         III          4    3    1    Bcom
Bcom          IV          4    3  NIL    Bcom
lifesc         I          5  NIL    1  lifesc
lifesc        II          5    0    1  lifesc
```

[ ]:
```
len(DF1)
```

[ ]: 18

[92]:
```
DF1['new']=DF1.index
```

[22]:
```
pd.merge(DF1, DF2, left_on=['new','Semester'], right_index=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\Users\SHARAN~1\AppData\Local\Temp/ipykernel_30116/3260519049.py in <module>
----> 1 pd.merge(DF1, DF2, left_on=['new','Semester'], right_index=True)

c:\Users\Sharanjit␣
 ↪Kaur\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\core\res␣ape\merge.
 ↪py in merge(left, right, how, on, left_on, right_on, left_index, right_index,␣
 ↪sort, suffixes, copy, indicator, validate)
     72     validate=None,
     73 ) -> "DataFrame":
---> 74     op = _MergeOperation(
     75         left,
     76         right,

c:\Users\Sharanjit␣
 ↪Kaur\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\core\res␣ape\merge.
 ↪py in __init__(self, left, right, how, on, left_on, right_on, axis,␣
 ↪left_index, right_index, sort, suffixes, copy, indicator, validate)
    666             self.right_join_keys,
    667             self.join_names,
--> 668         ) = self._get_merge_keys()
    669
    670         # validate the merge keys dtypes. We may need to coerce

c:\Users\Sharanjit␣
 ↪Kaur\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\core\res␣ape\merge.
 ↪py in _get_merge_keys(self)
   1056                     join_names.append(None)
   1057                 else:
-> 1058                     left_keys.append(left._get_label_or_level_values(k))
   1059                     join_names.append(k)
   1060             if isinstance(self.right.index, MultiIndex):

c:\Users\Sharanjit␣
 ↪Kaur\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\core\gen␣eric.
 ↪py in _get_label_or_level_values(self, key, axis)
   1682             values = self.axes[axis].get_level_values(key)._values
   1683         else:
-> 1684             raise KeyError(key)
```

```
   1685
   1686         # Check for duplicates

KeyError: 'new'
```

14 Problem is to merge DF2 on column with that of index of DF1. So adding a new col having Semester val I,II and in DF1 making Semester as index

14.1 changing index of DF1 as column and column as 'semester' index

[96]:
```
DF1.reset_index(level=0,inplace=True)
```

[100]:
```
DF2
```

[100]:
```
CL1             Corepaper  SEC  DSE
CL2                   100  50   100
Course Semester
PSCS   I                3  NaN  NaN
       II               3  NaN  NaN
       III              4  1.0  NaN
       IV               4  1.0  NaN
       V                4  1.0  1.0
       VI               4  1.0  1.0
CSHons I                4  2.0  NaN
       II               4  2.0  NaN
       III              4  2.0  NaN
       IV               4  2.0  NaN
       V                4  NaN  2.0
       VI               4  NaN  2.0
Bcom   I                4  2.0  NaN
       II               4  2.0  NaN
       III              4  3.0  1.0
       IV               4  3.0  NaN
lifesc I                5  NaN  1.0
       II               5  0.0  1.0
```

[98]:
```
DF1.set_index('Semester',inplace=True)
```

[24]:
```
DF2
```

[24]:
```
                Corepaper  SEC  DSE
Course Semester
PSCS   I                3  NIL  NIL
       II               3  NIL  NIL
       III              4    1  NIL
```

```
       IV                4    1  NIL
       V                 4    1    1
       VI                4    1    1
CSHons I                 4    2  NIL
       II                4    2  NIL
       III               4    2  NIL
       IV                4    2  NIL
       V                 4  NIL    2
       VI                4  NIL    2
Bcom   I                 4    2  NIL
       II                4    2  NIL
       III               4    3    1
       IV                4    3  NIL
lifesc I                 5  NIL    1
       II                5    0    1
```

14.2 adding a newcol having Semester values by mapping any col value

[31]:
```
dict1={1:'I',2:'II',3:'III',4:'IV','NIL':'VI'}
```

[33]:
```
DFnew=DF2[DF2['SEC'].notna().values.reshape(18,1)]['SEC']
```

[34]:
```
DFnew.map(dict1)
```

[34]:
```
Course  Semester
PSCS    I            VI
        II           VI
        III           I
        IV            I
        V             I
        VI            I
CSHons  I            II
        II           II
        III          II
        IV           II
        V            VI
        VI           VI
Bcom    I            II
        II           II
        III         III
        IV          III
lifesc  I            VI
        II          NaN
Name: SEC, dtype: object
```

14.3 while mapping take care of the levels

map(), applymap() and apply() methods are used to modify the data whereas their usage is slightly different. * pandas.Series.map(): in the data series element * pandas.DataFrame.applymap() on the each element of data frame * pandas.Series.apply(): used on each element whereas pandas.DataFrame.apply() is applied on row/column wise

[35]:
```
dict1={1:'I',2:'II',3:'III',4:'IV','NaN':'VI'}
DF2['newcol']=DF2['SEC'].map(dict1)
```

14.4 example on map() apply() applymap()

[60]:
```
import pandas as pd
import numpy as np
data = [(30,50,70,80), (21,41,61,81),(5,5,8,9)]
df = pd.DataFrame(data, columns = ['f1','f2','f3','f4'])
print(df)
```

```
   f1  f2  f3  f4
0  30  50  70  80
1  21  41  61  81
2   5   5   8   9
```

[61]:
```
df['f1']=df['f1'].map(lambda x: x/100)
```

[64]:
```
df
```

[64]:

	f1	f2	f3	f4
0	0.0030	50	70	80
1	0.0021	41	61	81
2	0.0005	5	8	9

[63]:
```
df['f1']=df['f1'].apply(lambda x: x/100)
```

[65]:
```
df3 = df.apply(lambda x: x/10)
df3
```

[65]:

	f1	f2	f3	f4
0	0.00030	5.0	7.0	8.0
1	0.00021	4.1	6.1	8.1
2	0.00005	0.5	0.8	0.9

[66]:
```
df3=df.applymap(lambda a: str(a)+".00")
df3
```

[66]:

	f1	f2	f3	f4
0	0.003.00	50.00	70.00	80.00
1	0.0021.00	41.00	61.00	81.00
2	0.0005.00	5.00	8.00	9.00

14.5 merging will preserve the index level as in left side

[ ]:
```
pd.merge(DF2, DF1, left_on=['newcol'], right_index=True)
```

15 Explicitly using join clause: Join columns of another DataFrame either on index or on a key column. Efficiently join multiple DataFrame objects by index at once by passing a list.

[76]:
```
df1
```

[76]:

	key	data1	new
0	b	0	0
1	b	1	1
2	a	2	2
3	c	3	3
4	a	4	4
5	a	5	5
6	b	6	6

[68]:
```
#df2=df2.append(df2.iloc[2])
df2
```

[68]:

	key	data1	new
0	a	0	1
1	b	1	2
2	d	2	10
9	a	0	1

[15]:
```
df2.append(df2.iloc[2])
df2.iloc[2,:]=['d',2,3]
```

[74]:
```
df2
```

[74]:

	key	data1	new
0	a	0	1
1	b	1	2
2	d	2	10
9	a	0	1

[70]:
```
pd.merge(df1,df2)
```

[70]:
```
Empty DataFrame
Columns: [key, data1, new]
Index: []
```

15.1 by default join is done matching row-index of left side table with that of RHSfor preserving its row index

- lsuffix/rsuffix is to rename common columns in LHS/RHS table by specified word

[75]:
```
df1.join(df2,lsuffix='left')
```

[75]:

	keyleft	data1left	newleft	key	data1	new
0	b	0	0	a	0.0	1.0
1	b	1	1	b	1.0	2.0
2	a	2	2	d	2.0	10.0
3	c	3	3	NaN	NaN	NaN
4	a	4	4	NaN	NaN	NaN
5	a	5	5	NaN	NaN	NaN
6	b	6	6	NaN	NaN	NaN

[80]:
```
df2.join(df1,rsuffix='right',how='outer')
```

[80]:

	key	data1	new	keyright	data1right	newright
0	a	0.0	1.0	b	0.0	0.0
1	b	1.0	2.0	b	1.0	1.0
2	d	2.0	10.0	a	2.0	2.0
3	NaN	NaN	NaN	c	3.0	3.0
4	NaN	NaN	NaN	a	4.0	4.0
5	NaN	NaN	NaN	a	5.0	5.0
6	NaN	NaN	NaN	b	6.0	6.0
9	a	0.0	1.0	NaN	NaN	NaN

[24]:
```
df2=df2.append(df2.iloc[0,:])
```

[26]:
```
df2.index=[0,1,2,9]
```

[86]:
```
df2
```

[86]:

	key	data1	new1
0	a	0	1
1	b	1	2
2	d	2	10
9	a	0	1

[83]:
```
df1
```

[83]:

	key	data1	new
0	b	0	0
1	b	1	1
2	a	2	2
3	c	3	3
4	a	4	4
5	a	5	5

```
6   b      6    6
```

15.2 renaming column new to

[81]:
```
df2.rename(columns={'new':'new1'},inplace=True)
df2.iloc[2,2]=10
```

15.3 column LHS and row index of RHS

[84]:
```
df1.join(df2, how='inner',lsuffix='left',on=['new'])
```

[84]:

	keyleft	data1left	new	key	data1	new1
0	b	0	0	a	0	1
1	b	1	1	b	1	2
2	a	2	2	d	2	10

15.4 Join one DF with more than one DFs

15.4.1 creating a new DF

[87]:
```
df3=df2.copy()
df3['data1']*=4
```

15.4.2 renaming its columns

[88]:
```
df3.columns=['a','b','c']
```

[89]:
```
df3['d']=df3['b']-df3['c']
```

[90]:
```
df3
```

[90]:

	a	b	c	d
0	a	0	1	-1
1	b	4	2	2
2	d	8	10	-2
9	a	0	1	-1

15.4.3 renaming columns of DF2 to remove same column names as in other DF df1

[96]:
```
df2.columns=['x','y','z']
df2
```

[96]:

	x	y	z
0	a	0	1
1	b	1	2
2	d	2	10
9	a	0	1

[95]:
```
df1
```

[95]:

	key	data1	new
0	b	0	0
1	b	1	1
2	a	2	2
3	c	3	3
4	a	4	4
5	a	5	5
6	b	6	6

[94]:
```
df3
```

[94]:

	a	b	c	d
0	a	0	1	-1
1	b	4	2	2
2	d	8	10	-2
9	a	0	1	-1

[48]:
```
df3=df3.append(df3.iloc[3,:])
```

15.4.4 now joining three DFs : note on the basis of common row index

[93]:
```
df1.join([df2,df3])
```

[93]:

	key	data1	new	x	y	z	a	b	c	d
0	b	0.0	0.0	a	0.0	1.0	a	0.0	1.0	-1.0
1	b	1.0	1.0	b	1.0	2.0	b	4.0	2.0	2.0
2	a	2.0	2.0	d	2.0	10.0	d	8.0	10.0	-2.0
3	c	3.0	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	a	4.0	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	a	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	b	6.0	6.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN

[97]:
```
df2.join([df1,df3])
```

[97]:

	x	y	z	key	data1	new	a	b	c	d
0	a	0.0	1.0	b	0.0	0.0	a	0.0	1.0	-1.0
1	b	1.0	2.0	b	1.0	1.0	b	4.0	2.0	2.0
2	d	2.0	10.0	a	2.0	2.0	d	8.0	10.0	-2.0
9	a	0.0	1.0	NaN	NaN	NaN	a	0.0	1.0	-1.0

[51]:
```
df2
```

[51]:

	x	y	z
0	a	0	1
1	b	1	2
2	d	2	10

```
4  b  0   0
9  a  0   1
```

16 join() and merge()

Commonality: 1. used to combines two dataframes

Difference:

1. join method combines two dataframes on the basis index values wheras the versatile merge method allows to specify columns beside the index to join on for both dataframesList item
2. join can be used to merge more than two DFs at a time, but not feasible with merge

17 Make sure to use proper stmt to copy two DFs

[98]:
```
DF2
```

[98]:

Course	Semester	Corepaper	SEC	DSE
PSCS	I	3	NIL	NIL
	II	3	NIL	NIL
	III	4	1	NIL
	IV	4	1	NIL
	V	4	1	1
	VI	4	1	1
CSHons	I	4	2	NIL
	II	4	2	NIL
	III	4	2	NIL
	IV	4	2	NIL
	V	4	NIL	2
	VI	4	NIL	2
Bcom	I	4	2	NIL
	II	4	2	NIL
	III	4	3	1
	IV	4	3	NIL
lifesc	I	5	NIL	1
	II	5	0	1

17.0.1 what is done using following statment?

[60]:
```
DF3=DF2
```

[61]:
```
DF4=DF2.copy(deep=True)
```

[62]:
```
DF3.rename(index={'lifesc':'ZooHons'},inplace='True')
```

[63]:
```
DF2
```

[63]:
```
                  Corepaper  SEC  DSE
Course  Semester
PSCS    I                 3  NIL  NIL
        II                3  NIL  NIL
        III               4    1  NIL
        IV                4    1  NIL
        V                 4    1    1
        VI                4    1    1
CSHons  I                 4    2  NIL
        II                4    2  NIL
        III               4    2  NIL
        IV                4    2  NIL
        V                 4  NIL    2
        VI                4  NIL    2
Bcom    I                 4    2  NIL
        II                4    2  NIL
        III               4    3    1
        IV                4    3  NIL
ZooHons I                 5  NIL    1
        II                5    0    1
```

[64]:
```
DF3.loc[['Bcom']]
```

[64]:
```
                 Corepaper SEC  DSE
Course Semester
Bcom   I                 4   2  NIL
       II                4   2  NIL
       III               4   3    1
       IV                4   3  NIL
```

[ ]:
```
DF2
```

[ ]:
```
                  corepaper  SEC  DSE newcol
course  Semester
PSCS    I                 3  NIL  NIL     VI
        II                3  NIL  NIL     VI
        III               4    1  NIL      I
        IV                4    1  NIL      I
        V                 4    1    1      I
        VI                4    1    1      I
CSHons  I                 4    2  NIL     II
        II                4    2  NIL     II
        III               4    2  NIL     II
        IV                4    2  NIL     II
        V                 4  NIL    2     VI
        VI                4  NIL    2     VI
Bcom    I                 4    2  NIL     II
```

```
         II                4    2  NIL     II
         III               4    3    1    III
         IV                4    3    4    III
ZooHons  I                 5  NIL    1     VI
         II                5    0    1    NaN
```

17.0.2 simple =(assignment) refers to original memory address whereas copy makes a new copy so that changes in the copied object are not reflected to original object

18 concatenate(): Concatenating along axis for binding/stacking

- if axis=0 then append in other DF
- if axis=1 then append coulmns for matched index

[99]:
```
df1
```

[99]:
```
  key  data1  new
0   b      0    0
1   b      1    1
2   a      2    2
3   c      3    3
4   a      4    4
5   a      5    5
6   b      6    6
```

[100]:
```
df2
```

[100]:
```
   x  y   z
0  a  0   1
1  b  1   2
2  d  2  10
9  a  0   1
```

[101]:
```
df2.columns=df1.columns
```

[102]:
```
pd.concat([df1,df2])
```

[102]:
```
  key  data1  new
0   b      0    0
1   b      1    1
2   a      2    2
3   c      3    3
4   a      4    4
5   a      5    5
6   b      6    6
0   a      0    1
1   b      1    2
```

```
2   d      2   10
9   a      0    1
```

[103]:
```
pd.concat([df1,df1],ignore_index=True)
```

[103]:
```
    key  data1  new
0    b      0    0
1    b      1    1
2    a      2    2
3    c      3    3
4    a      4    4
5    a      5    5
6    b      6    6
7    b      0    0
8    b      1    1
9    a      2    2
10   c      3    3
11   a      4    4
12   a      5    5
13   b      6    6
```

[70]:
```
df1
```

[70]:
```
  key  data1  new
0   b      0    0
1   b      1    1
2   a      2    2
3   c      3    3
4   a      4    4
5   a      5    5
6   b      6    6
```

[73]:
```
df2
```

[73]:
```
  key  data1  new
0   a      0    1
1   b      1    2
2   d      2   10
4   b      0    0
9   a      0    1
```

[ ]:
```
df1.columns.values
```

[ ]:
```
array(['key', 'data1', 'new'], dtype=object)
```

18.0.1 columns are added on the basis of matching index as Nan is added (axis=1)

[104]:
```
pd.concat([df1,df2],axis=1)
```

[104]:
```
   key  data1  new  key  data1   new
0    b    0.0  0.0    a    0.0   1.0
1    b    1.0  1.0    b    1.0   2.0
2    a    2.0  2.0    d    2.0  10.0
3    c    3.0  3.0  NaN    NaN   NaN
4    a    4.0  4.0  NaN    NaN   NaN
5    a    5.0  5.0  NaN    NaN   NaN
6    b    6.0  6.0  NaN    NaN   NaN
9  NaN    NaN  NaN    a    0.0   1.0
```

18.0.2 distinuishing between columns of two DFs and putting a level

[105]:
```
dftemp=pd.concat([df1, df2], axis=1, keys=['level1', 'level2'])
```

[107]:
```
dftemp.columns.levels
```

[107]:
```
FrozenList([['level1', 'level2'], ['key', 'data1', 'new']])
```

[106]:
```
dftemp
```

[106]:
```
  level1            level2
     key data1  new    key data1   new
0      b   0.0  0.0      a   0.0   1.0
1      b   1.0  1.0      b   1.0   2.0
2      a   2.0  2.0      d   2.0  10.0
3      c   3.0  3.0    NaN   NaN   NaN
4      a   4.0  4.0    NaN   NaN   NaN
5      a   5.0  5.0    NaN   NaN   NaN
6      b   6.0  6.0    NaN   NaN   NaN
9    NaN   NaN  NaN      a   0.0   1.0
```

18.0.3 concatenate two DFs side by side by simply as for strings using dictionary

[108]:
```
dftemp1=pd.concat({'level1': df1, 'level2': df2}, axis=1)
```

[86]:
```
dftemp.dtypes
```

[86]:
```
level1  key       object
        data1    float64
        new      float64
level2  key       object
        data1    float64
        new      float64
dtype: object
```

[109]:
```
dftemp1
```

[109]:

	level1			level2		
	key	data1	new	key	data1	new
0	b	0.0	0.0	a	0.0	1.0
1	b	1.0	1.0	b	1.0	2.0
2	a	2.0	2.0	d	2.0	10.0
3	c	3.0	3.0	NaN	NaN	NaN
4	a	4.0	4.0	NaN	NaN	NaN
5	a	5.0	5.0	NaN	NaN	NaN
6	b	6.0	6.0	NaN	NaN	NaN
9	NaN	NaN	NaN	a	0.0	1.0

18.0.4 why comparing Nan values result in False?

[110]:
```
dftemp==dftemp1
```

[110]:

	level1			level2		
	key	data1	new	key	data1	new
0	True	True	True	True	True	True
1	True	True	True	True	True	True
2	True	True	True	True	True	True
3	True	True	True	False	False	False
4	True	True	True	False	False	False
5	True	True	True	False	False	False
6	True	True	True	False	False	False
9	False	False	False	True	True	True

18.1 naming to coulmn levels

[111]:
```
pd.concat([df1, df2], axis=1, keys=['level1', 'level2'],
          names=['upper', 'lower'])
```

[111]:

upper	level1			level2		
lower	key	data1	new	key	data1	new
0	b	0.0	0.0	a	0.0	1.0
1	b	1.0	1.0	b	1.0	2.0
2	a	2.0	2.0	d	2.0	10.0
3	c	3.0	3.0	NaN	NaN	NaN
4	a	4.0	4.0	NaN	NaN	NaN
5	a	5.0	5.0	NaN	NaN	NaN
6	b	6.0	6.0	NaN	NaN	NaN
9	NaN	NaN	NaN	a	0.0	1.0

[88]:
```
df1
```

[88]:
```
  key  data1  new
0   b      0    0
1   b      1    1
2   a      2    2
3   c      3    3
4   a      4    4
5   a      5    5
6   b      6    6
```

[89]:
```
df2
```

[89]:
```
  key  data1  new
0   a      0    1
1   b      1    2
2   d      2   10
4   b      0    0
9   a      0    1
```

19 queries: row index of the DFs are rollnumer

- Find students names appearing in both tests (intersection)
- Find all the students appearing in either of the tests (union)

19.1 Find the output?

[112]:
```
S=pd.merge(df1,df2,how='outer')['key']
```

[91]:
```
S.unique()
```

[91]:
```
array(['b', 'a', 'c', 'd'], dtype=object)
```

20 stack() unstack() of Multilevel indexing

[113]:
```
DF1=pd.read_excel(f1,sheet_name=1,na_values=['NIL'])
```

[114]:
```
DF1.columns=['course','subject type','Year I','Year II','Year III']
```

[115]:
```
DF1
```

[115]:
```
    course subject type  Year I  Year II  Year III
0     PSCS         Core     6.0      6.0       8.0
1     PSCS          Sec     0.0      2.0       2.0
2     PSCS           DS     0.0      0.0       2.0
3     PSCS         AECC     2.0      0.0       0.0
4     PSCS           GE     NaN      NaN       NaN
5   CSHons         Core     6.0      6.0       4.0
```

```
6   CSHons          Sec     0.0     2.0     2.0
7   CSHons           DS     0.0     0.0     4.0
8   CSHons         AECC     2.0     0.0     0.0
9   CSHons           GE     2.0     2.0     0.0
10    Bcom         Core     6.0     6.0     4.0
11    Bcom          Sec     0.0     2.0     2.0
12    Bcom           DS     0.0     0.0     4.0
13    Bcom         AECC     2.0     0.0     0.0
14    Bcom           GE     2.0     2.0     0.0
```

20.1 Stack() coverts column levels to row level, in case only one level at column then output is a series. In case MLevel, inner most colum is changed to row index

[116]:
```
DF2=DF1.stack()
```

[117]:
```
DF2
```

[117]:
```
0   course          PSCS
    subject type    Core
    Year I           6.0
    Year II          6.0
    Year III         8.0
                     …
14  course          Bcom
    subject type      GE
    Year I           2.0
    Year II          2.0
    Year III         0.0
Length: 72, dtype: object
```

[99]:
```
DF2.index
```

[99]:
```
MultiIndex([( 0,       'course'),
            ( 0, 'subject type'),
            ( 0,       'Year I'),
            ( 0,      'Year II'),
            ( 0,     'Year III'),
            ( 1,       'course'),
            ( 1, 'subject type'),
            ( 1,       'Year I'),
            ( 1,      'Year II'),
            ( 1,     'Year III'),
            ( 2,       'course'),
            ( 2, 'subject type'),
            ( 2,       'Year I'),
            ( 2,      'Year II'),
```

```
( 2,     'Year III'),
( 3,       'course'),
( 3, 'subject type'),
( 3,       'Year I'),
( 3,      'Year II'),
( 3,     'Year III'),
( 4,       'course'),
( 4, 'subject type'),
( 5,       'course'),
( 5, 'subject type'),
( 5,       'Year I'),
( 5,      'Year II'),
( 5,     'Year III'),
( 6,       'course'),
( 6, 'subject type'),
( 6,       'Year I'),
( 6,      'Year II'),
( 6,     'Year III'),
( 7,       'course'),
( 7, 'subject type'),
( 7,       'Year I'),
( 7,      'Year II'),
( 7,     'Year III'),
( 8,       'course'),
( 8, 'subject type'),
( 8,       'Year I'),
( 8,      'Year II'),
( 8,     'Year III'),
( 9,       'course'),
( 9, 'subject type'),
( 9,       'Year I'),
( 9,      'Year II'),
( 9,     'Year III'),
(10,       'course'),
(10, 'subject type'),
(10,       'Year I'),
(10,      'Year II'),
(10,     'Year III'),
(11,       'course'),
(11, 'subject type'),
(11,       'Year I'),
(11,      'Year II'),
(11,     'Year III'),
(12,       'course'),
(12, 'subject type'),
(12,       'Year I'),
(12,      'Year II'),
```

```
                (12,     'Year III'),
                (13,       'course'),
                (13, 'subject type'),
                (13,       'Year I'),
                (13,      'Year II'),
                (13,     'Year III'),
                (14,       'course'),
                (14, 'subject type'),
                (14,       'Year I'),
                (14,      'Year II'),
                (14,     'Year III')],
               )
```

[100]:
```
DF2
```

[100]:
```
0   course          PSCS
    subject type    Core
    Year I           6.0
    Year II          6.0
    Year III         8.0
                    …
14  course          Bcom
    subject type      GE
    Year I           2.0
    Year II          2.0
    Year III         0.0
Length: 72, dtype: object
```

[119]:
```
DF2[(14,'course')]
```

[119]:
```
'Bcom'
```

[120]:
```
dftemp
```

[120]:

	level1			level2		
	key	data1	new	key	data1	new
0	b	0.0	0.0	a	0.0	1.0
1	b	1.0	1.0	b	1.0	2.0
2	a	2.0	2.0	d	2.0	10.0
3	c	3.0	3.0	NaN	NaN	NaN
4	a	4.0	4.0	NaN	NaN	NaN
5	a	5.0	5.0	NaN	NaN	NaN
6	b	6.0	6.0	NaN	NaN	NaN
9	NaN	NaN	NaN	a	0.0	1.0

[121]:
```
dftemp.stack()
```

[121]:

		level1	level2
0	key	b	a
	data1	0.0	0.0
	new	0.0	1.0
1	key	b	b
	data1	1.0	1.0
	new	1.0	2.0
2	key	a	d
	data1	2.0	2.0
	new	2.0	10.0
3	key	c	NaN
	data1	3.0	NaN
	new	3.0	NaN
4	key	a	NaN
	data1	4.0	NaN
	new	4.0	NaN
5	key	a	NaN
	data1	5.0	NaN
	new	5.0	NaN
6	key	b	NaN
	data1	6.0	NaN
	new	6.0	NaN
9	key	NaN	a
	data1	NaN	0.0
	new	NaN	1.0

21 setting first two columns as index

[103]:
```
DF1.columns
```

[103]:
```
Index(['course', 'subject type', 'Year I', 'Year II', 'Year III'],
dtype='object')
```

[122]:
```
DF1
```

[122]:

	course	subject type	Year I	Year II	Year III
0	PSCS	Core	6.0	6.0	8.0
1	PSCS	Sec	0.0	2.0	2.0
2	PSCS	DS	0.0	0.0	2.0
3	PSCS	AECC	2.0	0.0	0.0
4	PSCS	GE	NaN	NaN	NaN
5	CSHons	Core	6.0	6.0	4.0
6	CSHons	Sec	0.0	2.0	2.0
7	CSHons	DS	0.0	0.0	4.0
8	CSHons	AECC	2.0	0.0	0.0
9	CSHons	GE	2.0	2.0	0.0
10	Bcom	Core	6.0	6.0	4.0

```
11    Bcom          Sec     0.0      2.0       2.0
12    Bcom           DS     0.0      0.0       4.0
13    Bcom         AECC     2.0      0.0       0.0
14    Bcom           GE     2.0      2.0       0.0
```

21.0.1 renaming specific columns

[97]:
```
DF1.rename(columns={'Unnamed: 0':'Course','Unnamed: 1':'Types'},inplace=True)
```

[123]:
```
DF2=DF1.set_index(keys=[DF1.columns[0],DF1.columns[1]])
```

[124]:
```
DF2
```

[124]:
```
                     Year I  Year II  Year III
course subject type
PSCS   Core             6.0      6.0       8.0
       Sec              0.0      2.0       2.0
       DS               0.0      0.0       2.0
       AECC             2.0      0.0       0.0
       GE               NaN      NaN       NaN
CSHons Core             6.0      6.0       4.0
       Sec              0.0      2.0       2.0
       DS               0.0      0.0       4.0
       AECC             2.0      0.0       0.0
       GE               2.0      2.0       0.0
Bcom   Core             6.0      6.0       4.0
       Sec              0.0      2.0       2.0
       DS               0.0      0.0       4.0
       AECC             2.0      0.0       0.0
       GE               2.0      2.0       0.0
```

21.1 stack: changing columns as rows in the DF

- As there may be multiple indices, stacking means converting (also called rotating or pivoting) the innermost column index into the innermost row index.
- Unstacking: exactly the inverse operation of stacking— it will convert the innermost row index back into the innermost column index.

21.2 row having NA will be igonred

[125]:
```
type(DF2)
```

[125]:
```
pandas.core.frame.DataFrame
```

[136]:
```
DF2
```

[136]:
```
                     Year I  Year II  Year III
course subject type
```

```
PSCS   Core             6.0      6.0      8.0
       Sec              0.0      2.0      2.0
       DS               0.0      0.0      2.0
       AECC             2.0      0.0      0.0
       GE               NaN      NaN      NaN
CSHons Core             6.0      6.0      4.0
       Sec              0.0      2.0      2.0
       DS               0.0      0.0      4.0
       AECC             2.0      0.0      0.0
       GE               2.0      2.0      0.0
Bcom   Core             6.0      6.0      4.0
       Sec              0.0      2.0      2.0
       DS               0.0      0.0      4.0
       AECC             2.0      0.0      0.0
       GE               2.0      2.0      0.0
```

[127]:
```
dftemp.stack().unstack()
```

[127]:

	level1			level2		
	key	data1	new	key	data1	new
0	b	0.0	0.0	a	0.0	1.0
1	b	1.0	1.0	b	1.0	2.0
2	a	2.0	2.0	d	2.0	10.0
3	c	3.0	3.0	NaN	NaN	NaN
4	a	4.0	4.0	NaN	NaN	NaN
5	a	5.0	5.0	NaN	NaN	NaN
6	b	6.0	6.0	NaN	NaN	NaN
9	NaN	NaN	NaN	a	0.0	1.0

[137]:
```
DF3=DF2.stack()
```

21.3 DF3 is a series with one column and multilevel indices

[112]:
```
type(DF3)
```

[112]:
```
pandas.core.series.Series
```

[135]:
```
DF3
```

[135]:
```
course  subject type
PSCS    Core          Year I      6.0
                      Year II     6.0
                      Year III    8.0
        Sec           Year I      0.0
                      Year II     2.0
                      Year III    2.0
        DS            Year I      0.0
                      Year II     0.0
```

```
                        Year III    2.0
          AECC          Year I      2.0
                        Year II     0.0
                        Year III    0.0
CSHons    Core          Year I      6.0
                        Year II     6.0
                        Year III    4.0
          Sec           Year I      0.0
                        Year II     2.0
                        Year III    2.0
          DS            Year I      0.0
                        Year II     0.0
                        Year III    4.0
          AECC          Year I      2.0
                        Year II     0.0
                        Year III    0.0
          GE            Year I      2.0
                        Year II     2.0
                        Year III    0.0
Bcom      Core          Year I      6.0
                        Year II     6.0
                        Year III    4.0
          Sec           Year I      0.0
                        Year II     2.0
                        Year III    2.0
          DS            Year I      0.0
                        Year II     0.0
                        Year III    4.0
          AECC          Year I      2.0
                        Year II     0.0
                        Year III    0.0
          GE            Year I      2.0
                        Year II     2.0
                        Year III    0.0
dtype: float64
```

[114]:
```
DF3.index.names
```

[114]:
```
FrozenList(['course', 'subject type', None])
```

[116]:
```
DF3.index.names=['Course', 'Types', 'Semester']
```

21.4 unstackin displays row index values as sorted, By default innermost row index level is changed to lowest column level. but any other row level may also be specified

[130]:
```
DF3.unstack()
```

[130]:
```
                     Year I  Year II  Year III
course subject type
Bcom   AECC             2.0      0.0       0.0
       Core             6.0      6.0       4.0
       DS               0.0      0.0       4.0
       GE               2.0      2.0       0.0
       Sec              0.0      2.0       2.0
CSHons AECC             2.0      0.0       0.0
       Core             6.0      6.0       4.0
       DS               0.0      0.0       4.0
       GE               2.0      2.0       0.0
       Sec              0.0      2.0       2.0
PSCS   AECC             2.0      0.0       0.0
       Core             6.0      6.0       8.0
       DS               0.0      0.0       2.0
       Sec              0.0      2.0       2.0
```

[118]:
```
type(DF3.unstack())
```

[118]:
```
pandas.core.frame.DataFrame
```

21.5 Preserving of missing values

[142]:
```
DF3=DF2.stack(dropna=False)
```

[143]:
```
DF3
```

[143]:
```
course  subject type
PSCS    Core           Year I      6.0
                       Year II     6.0
                       Year III    8.0
        Sec            Year I      0.0
                       Year II     2.0
                       Year III    2.0
        DS             Year I      0.0
                       Year II     0.0
                       Year III    2.0
        AECC           Year I      2.0
                       Year II     0.0
                       Year III    0.0
        GE             Year I      NaN
                       Year II     NaN
```

```
                          Year III    NaN
      CSHons  Core        Year I      6.0
                          Year II     6.0
                          Year III    4.0
              Sec         Year I      0.0
                          Year II     2.0
                          Year III    2.0
              DS          Year I      0.0
                          Year II     0.0
                          Year III    4.0
              AECC        Year I      2.0
                          Year II     0.0
                          Year III    0.0
              GE          Year I      2.0
                          Year II     2.0
                          Year III    0.0
      Bcom    Core        Year I      6.0
                          Year II     6.0
                          Year III    4.0
              Sec         Year I      0.0
                          Year II     2.0
                          Year III    2.0
              DS          Year I      0.0
                          Year II     0.0
                          Year III    4.0
              AECC        Year I      2.0
                          Year II     0.0
                          Year III    0.0
              GE          Year I      2.0
                          Year II     2.0
                          Year III    0.0
      dtype: float64
```

[133]:
```
DF3.unstack()
```

[133]:

course	subject	type	Year I	Year II	Year III
Bcom	AECC		2.0	0.0	0.0
	Core		6.0	6.0	4.0
	DS		0.0	0.0	4.0
	GE		2.0	2.0	0.0
	Sec		0.0	2.0	2.0
CSHons	AECC		2.0	0.0	0.0
	Core		6.0	6.0	4.0
	DS		0.0	0.0	4.0
	GE		2.0	2.0	0.0
	Sec		0.0	2.0	2.0

```
PSCS    AECC            2.0     0.0     0.0
        Core            6.0     6.0     8.0
        DS              0.0     0.0     2.0
        GE              NaN     NaN     NaN
        Sec             0.0     2.0     2.0
```

21.6 incase types of subjects per year per course are to be displayed together for comparison

[126]:
```
DF3
```

[126]:
```
course  subject type
PSCS    Core          Year I      6.0
                      Year II     6.0
                      Year III    8.0
        Sec           Year I      0.0
                      Year II     2.0
                      Year III    2.0
        DS            Year I      0.0
                      Year II     0.0
                      Year III    2.0
        AECC          Year I      2.0
                      Year II     0.0
                      Year III    0.0
        GE            Year I      NaN
                      Year II     NaN
                      Year III    NaN
CSHons  Core          Year I      6.0
                      Year II     6.0
                      Year III    4.0
        Sec           Year I      0.0
                      Year II     2.0
                      Year III    2.0
        DS            Year I      0.0
                      Year II     0.0
                      Year III    4.0
        AECC          Year I      2.0
                      Year II     0.0
                      Year III    0.0
        GE            Year I      2.0
                      Year II     2.0
                      Year III    0.0
Bcom    Core          Year I      6.0
                      Year II     6.0
                      Year III    4.0
        Sec           Year I      0.0
                      Year II     2.0
```

```
                      Year III    2.0
          DS          Year I      0.0
                      Year II     0.0
                      Year III    4.0
          AECC        Year I      2.0
                      Year II     0.0
                      Year III    0.0
          GE          Year I      2.0
                      Year II     2.0
                      Year III    0.0
dtype: float64
```

[127]:
```
DF4=DF3.unstack(level=0)
```

[129]:
```
DF4
```

[129]:
```
course                Bcom  CSHons  PSCS
subject type
AECC         Year I    2.0     2.0   2.0
             Year II   0.0     0.0   0.0
             Year III  0.0     0.0   0.0
Core         Year I    6.0     6.0   6.0
             Year II   6.0     6.0   6.0
             Year III  4.0     4.0   8.0
DS           Year I    0.0     0.0   0.0
             Year II   0.0     0.0   0.0
             Year III  4.0     4.0   2.0
GE           Year I    2.0     2.0   NaN
             Year II   2.0     2.0   NaN
             Year III  0.0     0.0   NaN
Sec          Year I    0.0     0.0   0.0
             Year II   2.0     2.0   2.0
             Year III  2.0     2.0   2.0
```

[124]:
```
DF4.columns
```

[124]:
```
Index(['Bcom', 'CSHons', 'PSCS'], dtype='object', name='course')
```

[125]:
```
DF4.index
```

[125]:
```
MultiIndex([('AECC',   'Year I'),
            ('AECC',  'Year II'),
            ('AECC', 'Year III'),
            ('Core',   'Year I'),
            ('Core',  'Year II'),
            ('Core', 'Year III'),
            (  'DS',   'Year I'),
```

```
 (  'DS',  'Year II'),
 (  'DS', 'Year III'),
 (  'GE',   'Year I'),
 (  'GE',  'Year II'),
 (  'GE', 'Year III'),
 ( 'Sec',   'Year I'),
 ( 'Sec',  'Year II'),
 ( 'Sec', 'Year III')],
names=['subject type', None])
```