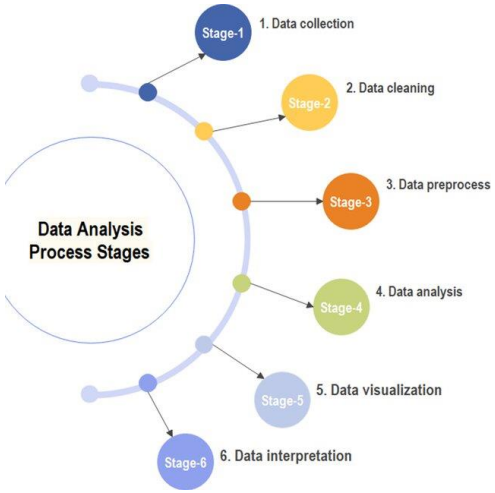
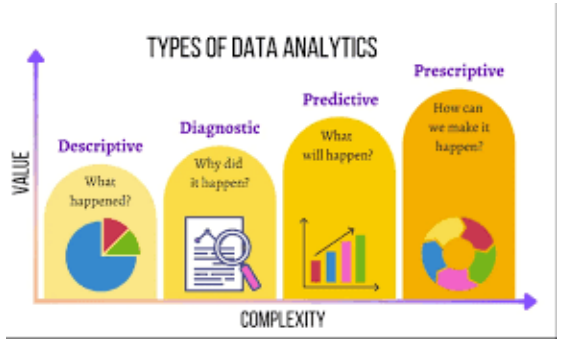
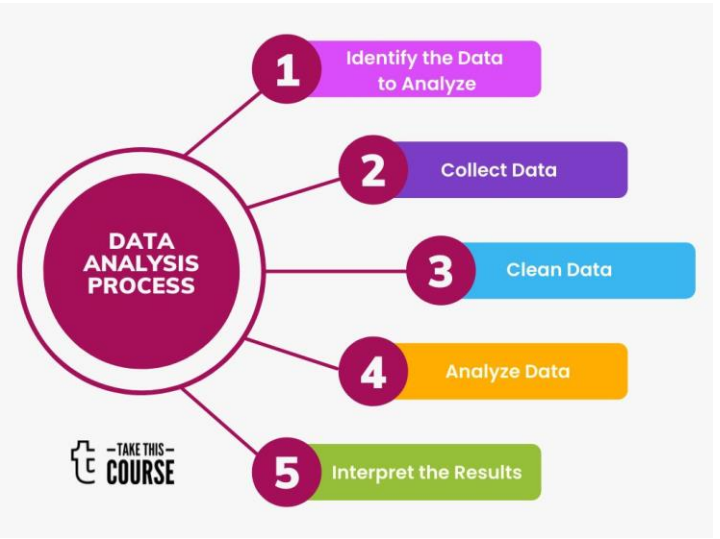




PHÂN TÍCH DỮ LIỆU

(Data Analysis)



Lê Văn Hạnh
levanhanhvn@gmail.com

NỘI DUNG MÔN HỌC

PHẦN 1 TỔNG QUAN & THU THẬP DỮ LIỆU CHO VIỆC PHÂN TÍCH

1. Khoa học dữ liệu
2. Thu thập dữ liệu
3. Tìm hiểu dữ liệu

PHẦN 2: TIỀN XỬ LÝ DỮ LIỆU *(Data Preprocessing)*

4. Nhiệm vụ chính trong tiền xử lý dữ liệu
5. PANDAS
6. Thao tác với các định dạng khác nhau của tập tin dữ liệu
7. Làm sạch và Chuẩn bị dữ liệu
8. Sắp xếp dữ liệu: nối, kết hợp và định hình lại
9. Tổng hợp dữ liệu và các tác vụ trên nhóm

PHẦN 3 TRỰC QUAN HÓA DỮ LIỆU *(Data Visualization)*

10. Đồ thị và Biểu đồ
11. Vẽ đồ thị và Trực quan hóa

PHẦN 2
TIỀN XỬ LÝ DỮ LIỆU *(Data Preprocessing)*

Chương 8

SẮP XẾP DỮ LIỆU:

NỐI, KẾT HỢP & ĐỊNH HÌNH LẠI

(Data Wrangling: Join, Combine & Reshape)

Lê Văn Hạnh
levanhanhvn@gmail.com

NỘI DUNG CHƯƠNG 8

1. Chỉ mục phân cấp

- Lập chỉ mục phân cấp
- Sắp xếp lại và sắp xếp theo mức
- Thống kê tóm tắt theo mức
- Lập chỉ mục với các cột của

2. Kết hợp và hợp nhất các bộ dữ liệu

- Nối kết DataFrame theo kiểu cơ sở dữ liệu quan hệ
- Hợp nhất dữ liệu dựa trên chỉ mục
- Nối dữ liệu dọc theo một trục
- Kết hợp dữ liệu với sự trùng lắp của chỉ mục

3. Định hình lại và xoay trục dữ liệu

- Định hình lại dữ liệu bằng lập chỉ mục phân cấp
- Xoay trục dữ liệu từ định dạng "dài" sang định dạng "rộng"
- Xoay trục dữ liệu từ định dạng "rộng" sang định dạng "dài"

1. CHỈ MỤC PHÂN CẤP *(Hierarchical Indexing)*

1.1. Lập chỉ mục phân cấp Lập chỉ mục phân cấp là một tính năng quan trọng của *pandas* cho phép có nhiều (hai hoặc nhiều) cấp độ chỉ mục trên một trục.

- Lập chỉ mục phân cấp đóng vai trò quan trọng trong việc định hình lại dữ liệu và các tác vụ dựa trên nhóm như hình thành bảng tổng hợp (*pivot table*).

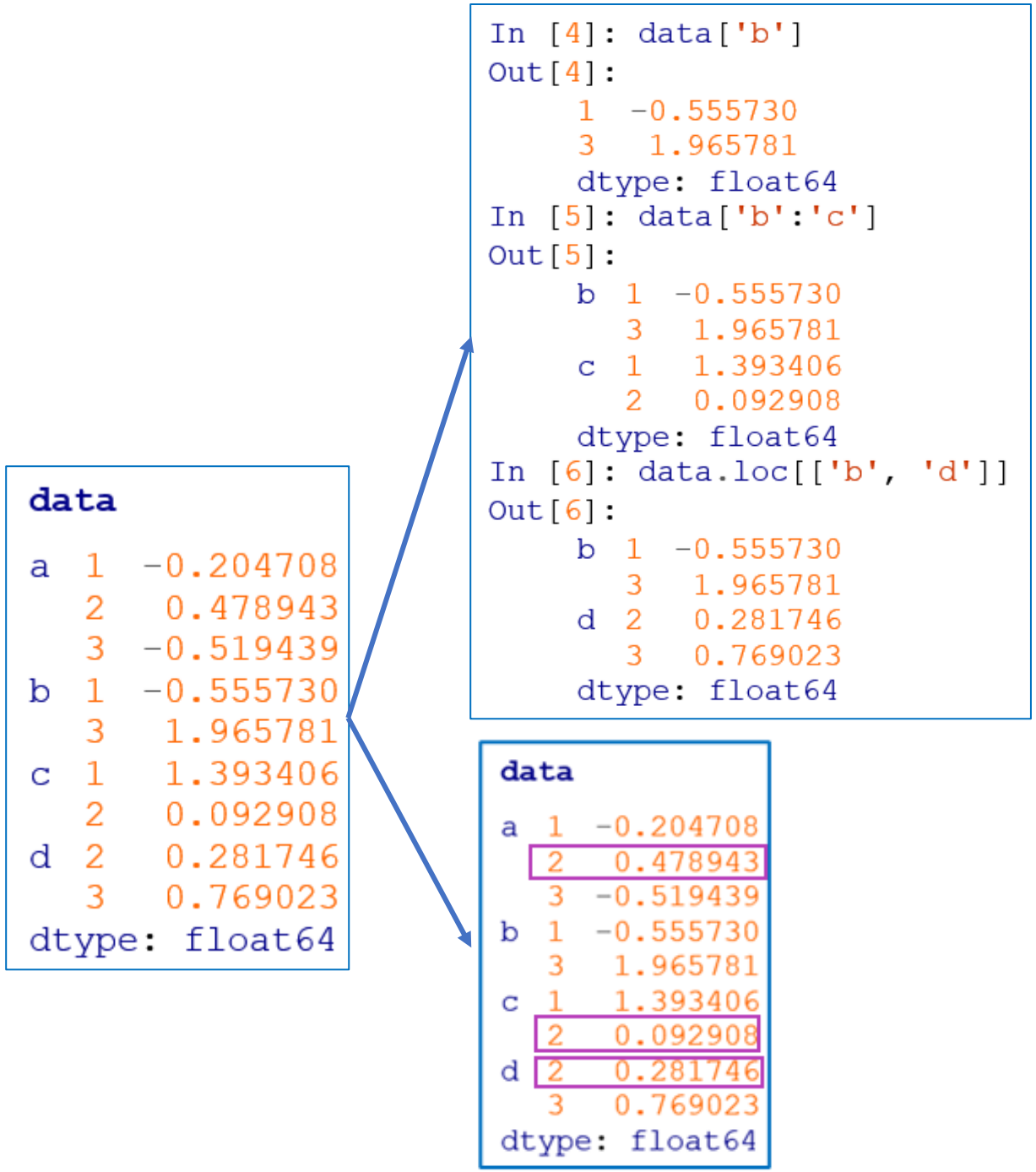
```
In [1]: data = pd.Series(np.random.randn(9),
 ...: index =[['a', 'a', 'a', 'b', 'b', 'c', 'c', 'd', 'd'],
 ...:         [1, 2, 3, 1, 3, 1, 2, 2, 3]])
In [2]: data
Out[2]:
a  1   -0.204708
   2    0.478943
   3   -0.519439
b  1   -0.555730
   3    1.965781
c  1    1.393406
   2    0.092908
```

```
In [3]: data.index
Out[3]:
    MultiIndex(levels=[['a', 'b', 'c', 'd'], [1, 2, 3]],
               labels=[[0, 0, 0, 1, 1, 2, 2, 3, 3], [0, 1, 2, 0,
            2, 0, 1, 1, 2]])
```

1.1. Lập chỉ mục phân cấp (Hierarchical Indexing)

- Với một đối tượng đã được lập chỉ mục phân cấp, có thể gọi là lập chỉ mục một phần (*partial indexing*), cho phép chọn chính xác các tập hợp con của dữ liệu:

- Thậm chí có thể lựa chọn mức "bên trong" (*"inner" level*):



1.1. Lập chỉ mục phân cấp

- Lập chỉ mục phân cấp đóng vai trò quan trọng trong việc định hình lại dữ liệu và các tác vụ dựa trên nhóm như hình thành bảng tổng hợp (*pivot table*).

```
data
a  1   -0.204708
   2    0.478943
   3   -0.519439
b  1   -0.555730
   3    1.965781
c  1    1.393406
   2    0.092908
d  2    0.281746
   3    0.769023
dtype: float64
```

```
In [8]: data.unstack()
Out[8]:
                1         2         3
a       -0.204708  0.478943 -0.519439
b       -0.555730       NaN  1.965781
c        1.393406  0.092908       NaN
d             NaN  0.281746  0.769023
```

```
In [9]: data.unstack().stack()
Out[9]:
     a  1   -0.204708
        2    0.478943
        3   -0.519439
     b  1   -0.555730
        3    1.965781
     c  1    1.393406
        2    0.092908
     d  2    0.281746
        3    0.769023
     dtype: float64
```

1.1. Lập chỉ mục phân cấp

- Với *DataFrame*,
  - Có thể có chỉ mục phân cấp trên cả hai trục:

```
In [10]: frame = pd.DataFrame(np.arange(12).reshape((4, 3))
    ....:       index=[['a', 'a', 'b', 'b'], [1, 2, 1, 2]],
    ....:       columns=[['Ohio', 'Ohio', 'Colorado'],
    ....:                ['Green', 'Red', 'Green']])
In [11]: frame
Out[11]:
             Ohio      Colorado
            Green   Red     Green
a   1           0     1         2
    2           3     4         5
b   1           6     7         8
    2           9    10        11
```

  - Các cấp độ phân cấp có thể có tên (dưới dạng chuỗi hoặc bất kỳ đối tượng *Python* nào). Khi đó, chúng sẽ hiển thị trong đầu ra của console:

```
In [12]: frame.index.names = ['key1', 'key2']
In [13]: frame.columns.names = ['state', 'color']
In [14]: frame
Out[14]:
state           Ohio      Colorado
color          Green   Red     Green
key1 key2
a    1             0     1         2
     2             3     4         5
b    1             6     7         8
     2             9    10        11
```

1.1. Lập chỉ mục phân cấp

- Với *DataFrame*,
  - Với việc lập chỉ mục một phần cột, có thể chọn các nhóm cột có index tương tự nhau:

```
frame
state          Ohio     Colorado
color         Green  Red    Green
key1 key2
a    1            0    1        2
     2            3    4        5
b    1            6    7        8
     2            9   10       11
```

```
In [15]: frame['Ohio']
Out[15]:
     color         Green  Red
     key1 key2
     a    1            0    1
          2            3    4
     b    1            6    7
          2            9   10
```

  - `MultiIndex` có thể được tự tạo và sau đó được sử dụng lại

```
frame
state          Ohio     Colorado
color         Green  Red    Green
key1 key2
a    1            0    1        2
     2            3    4        5
b    1            6    7        8
     2            9   10       11
```

```
MultiIndex.from_arrays([['Ohio', 'Ohio', 'Colorado'],
                        ['Green', 'Red', 'Green']],
                        names=['state', 'color'])
```

1.2. Sắp xếp lại và sắp xếp theo mức *(Reordering and Sorting Levels)*

- **`Swaplevel`**: lấy hai số (hoặc tên) mức và trả về một đối tượng mới với các mức được hoán đổi cho nhau (nhưng dữ liệu không bị thay đổi):

frame

state		Ohio		Colorado
color		Green	Red	Green
key1	key2			
a	1	0	1	2
	2	3	4	5
b	1	6	7	8
	2	9	10	11

```
In [16]: frame.swaplevel('key1', 'key2')
Out[16]:
```

state		Ohio		Colorado
color		Green	Red	Green
key1	key2			
1	a	0	1	2
2	a	3	4	5
1	b	6	7	8
2	b	9	10	11

- **`sort_index`**: sắp xếp dữ liệu chỉ bằng cách sử dụng các giá trị ở một mức duy nhất. Khi hoán đổi giữa các mức, không có gì lạ khi sử dụng **`sort_index`** để kết quả được sắp xếp theo thứ tự từ điển với mức được chỉ định:

```
In [17]: frame.sort_index(level=1)
Out[17]:
```

state		Ohio		Colorado
color		Green	Red	Green
key1	key2			
a	1	0	1	2
b	1	6	7	8
a	2	3	4	5
b	2	9	10	11

frame

state		Ohio		Colorado
color		Green	Red	Green
key1	key2			
a	1	0	1	2
	2	3	4	5
b	1	6	7	8
	2	9	10	11

```
In [18]: frame.swaplevel(0, 1).sort_index(level=0)
Out[18]:
```

state		Ohio		Colorado
color		Green	Red	Green
key2	key1			
1	a	0	1	2
	b	6	7	8
2	a	3	4	5
	b	9	10	11

1.2. Sắp xếp lại và sắp xếp theo mức *(Reordering and Sorting Levels)*

- **`Swaplevel`**: lấy hai số (hoặc tên) mức và trả về một đối tượng mới với các mức được hoán đổi cho nhau (nhưng dữ liệu không bị thay đổi):

frame

state		Ohio		Colorado
color		Green	Red	Green
key1	key2			
a	1	0	1	2
	2	3	4	5
b	1	6	7	8
	2	9	10	11

```
In [16]: frame.swaplevel('key1', 'key2')
Out[16]:
```

state		Ohio		Colorado
color		Green	Red	Green
key1	key2			
1	a	0	1	2
2	a	3	4	5
1	b	6	7	8
2	b	9	10	11

- **`sort_index`**: sắp xếp dữ liệu chỉ bằng cách sử dụng các giá trị ở một mức duy nhất. Khi hoán đổi giữa các mức, không có gì lạ khi sử dụng **`sort_index`** để kết quả được sắp xếp theo thứ tự từ điển với mức được chỉ định:

```
In [17]: frame.sort_index(level=1)
Out[17]:
```

state		Ohio		Colorado
color		Green	Red	Green
key1	key2			
a	1	0	1	2
b	1	6	7	8
a	2	3	4	5
b	2	9	10	11

frame

state		Ohio		Colorado
color		Green	Red	Green
key1	key2			
a	1	0	1	2
	2	3	4	5
b	1	6	7	8
	2	9	10	11

```
In [18]: frame.swaplevel(0, 1).sort_index(level=0)
Out[18]:
```

state		Ohio		Colorado
color		Green	Red	Green
key2	key1			
1	a	0	1	2
	b	6	7	8
2	a	3	4	5
	b	9	10	11

1.3. Thống kê tóm tắt theo mức *(Summary Statistics by Level)*

- Nhiều số liệu thống kê mô tả và tóm tắt trên *DataFrame* và *Series* có tùy chọn **`level`** giúp chỉ định mức mà ta muốn tổng hợp theo một trục cụ thể.

```
In [19]: frame.groupby(level=1).sum()
Out[19]:
      State     Ohio         Colorado
      color     Green Red       Green
      key2
      1             6   8           10
      2            12  14           16
In [20]: frame.groupby(level=0).sum()
Out[20]:
      State     Ohio         Colorado
      color     Green Red       Green
      key1
      a             3   5           7
      b            15  17          19
```

```
frame
      State  Ohio          Colorado
      color  Green     Red     Green
key2 key1
1    a             0     1         2
     b             6     7         8
2    a             3     4         5
     b             9    10        11
```

1.4. Lập chỉ mục với các cột của *DataFrame* *(Indexing with a DataFrame's columns)*

- Hàm **`set_index`** của *DataFrame* sẽ tạo một *DataFrame* mới bằng cách sử dụng một hoặc nhiều cột của nó làm chỉ mục. Theo mặc định, các cột được sử dụng làm chỉ mục sẽ bị xóa khỏi *DataFrame*, tuy nhiên vẫn có thể giữ lại với đối số **`drop=False`**:

frame

	a	b	c	d
0	0	7	one	0
1	1	6	one	1
2	2	5	one	2
3	3	4	two	0
4	4	3	two	1
5	5	2	two	2
6	6	1	two	3

```
In [23]: frame2 = frame.set_index(['c', 'd'])
In [23]: frame2
Out[24]:
```

		a	b
c	d		
one	0	0	7
	1	1	6
	2	2	5
two	0	3	4
	1	4	3
	2	5	2
	3	6	1

```
In [25]: frame.set_index(['c', 'd'], drop=False)
Out[25]:
```

		a	b	c	d
c	d				
one	0	0	7	one	0
	1	1	6	one	1
	2	2	5	one	2
two	0	3	4	two	0
	1	4	3	two	1
	2	5	2	two	2
	3	6	1	two	3

1.4. Lập chỉ mục với các cột của *DataFrame* *(Indexing with a DataFrame's columns)*

- ***reset_index*** lại làm ngược lại với ***set_index***; các mức chỉ mục phân cấp được chuyển vào các cột:

frame2

		a	b
c	d		
one	0	0	7
	1	1	6
	2	2	5
two	0	3	4
	1	4	3
	2	5	2
	3	6	1

```
In [26]: frame2.reset_index()
Out[26]:
     c  d  a  b
0  one  0  0  7
1  one  1  1  6
2  one  2  2  5
3  two  0  3  4
4  two  1  4  3
5  two  2  5  2
6  two  3  6  1
```

2. KẾT HỢP VÀ HỢP NHẤT CÁC BỘ DỮ LIỆU

(Combining and Merging Datasets)

Dữ liệu chứa trong các đối tượng *pandas* có thể được kết hợp với nhau theo một số cách:

- **`pandas.merge`**: kết nối các hàng trong *DataFrame*s dựa trên một hoặc nhiều khóa (*keys*). Điều này quen thuộc với người dùng SQL hoặc các cơ sở dữ liệu quan hệ khác vì nó thực hiện tác vụ **`join`** trong cơ sở dữ liệu.
- **`pandas.concat`**: nối (*concatenates*) hoặc "xếp chồng" (*"stacks"*) các đối tượng lại với nhau dọc theo một trục.
- Phương thức của đối tượng **`Combine_first`**: cho phép ghép các dữ liệu chồng chéo (*overlapping data*) lại với nhau để điền vào các giá trị còn thiếu trong một đối tượng bằng các giá trị từ một đối tượng khác.

Các thao tác này là trung tâm của cơ sở dữ liệu quan hệ (ví dụ: dựa trên SQL).

2.1. Nối kết DataFrame theo kiểu cơ sở dữ liệu quan hệ *(Database-Style DataFrame Joins)*

Hàm `merge` là hàm chính trong `pandas` trong việc nối kết theo kiểu CSDL quan hệ

Các đối số khi thực hiện hợp nhất bằng hàm *merge*

Argument	*Description*
`left`	*DataFrame* được hợp nhất ở phía bên trái.
`right`	*DataFrame* được hợp nhất ở phía bên phải.
`how`	Chỉ định cách nối kết 2 *DataFrame*. Là 1 trong các giá trị `'inner'` (mặc định), `'outer'`, `'left'`, `'right'`;
`on`	Tên cột tham gia nối kết. Phải được tìm thấy trong cả hai đối tượng *DataFrame*. Nếu không được chỉ định và không có khóa nối nào khác được cung cấp, sẽ sử dụng tên những cột trùng nhau ở bên trái và bên phải làm khóa nối (*join keys*).
`left_on`	Tên các cột trong *DataFrame* bên ***trái*** sẽ sử dụng làm khóa nối.
`right_on`	Tên các cột trong *DataFrame* bên ***phải*** sẽ sử dụng làm khóa nối.
`left_index`	Sử dụng chỉ mục hàng ở bên ***trái*** làm khóa nối (hoặc khóa - *keys*, nếu là *MultiIndex*).
`right_index`	Sử dụng chỉ mục hàng ở bên ***phải*** làm khóa nối (hoặc khóa - *keys*, nếu là *MultiIndex*).
`sort`	Sắp xếp dữ liệu đã hợp nhất theo từ điển bằng các khóa nối; `True` theo mặc định (để có hiệu suất tốt hơn trong một số trường hợp trên tập dữ liệu lớn).
`suffixes`	Chỉ định 1 tuple chứa các giá trị kiểu string để thêm vào tên cột trong trường hợp các tên cột của 2 bên trái và phải trùng nhau. Mặc định là (`'_x'`, `'_y'`) (ví dụ: nếu `'data'` trong cả hai đối tượng *DataFrame*, kết quả sẽ xuất hiện dưới dạng `'data_x'` và `'data_y'`).
`copy`	Nếu là `False`, tránh sao chép dữ liệu vào cấu trúc dữ liệu thu được trong một số trường hợp ngoại lệ; theo mặc định luôn thực hiện sao chép (`True`).
`indicator`	Thêm một cột đặc biệt `_merge` cho biết nguồn của mỗi hàng; các giá trị sẽ là `'left_only'`, `'right_only'` hoặc `'both'` dựa trên nguồn gốc của dữ liệu đã nối trong mỗi hàng.

2.1. Nối kết DataFrame theo kiểu cơ sở dữ liệu quan hệ *(Database-Style DataFrame Joins)*

Hàm `merge` là hàm chính trong `pandas` trong việc nối kết theo kiểu CSDL quan hệ

Các đối số khi thực hiện hợp nhất bằng hàm *merge*

Argument	*Description*
`left`	*DataFrame* được hợp nhất ở phía bên trái.
`right`	*DataFrame* được hợp nhất ở phía bên phải.
`how`	Chỉ định cách nối kết 2 *DataFrame*. Là 1 trong các giá trị `'inner'` (mặc định), `'outer'`, `'left'`, `'right'`;
`on`	Tên cột tham gia nối kết. Phải được tìm thấy trong cả hai đối tượng *DataFrame*. Nếu không được chỉ định và không có khóa nối nào khác được cung cấp, sẽ sử dụng tên những cột trùng nhau ở bên trái và bên phải làm khóa nối (*join keys*).
`left_on`	Tên các cột trong *DataFrame* bên ***trái*** sẽ sử dụng làm khóa nối.

Các kiểu nối kết với đối số how

Tùy chọn	*Ý nghĩa*
`'inner'`	Tương tự `inner join` trong SQL
`'left'`	Tương tự `left join` trong SQL
`'right'`	Tương tự `right join` trong SQL
`'outter'`	Tương tự `outer join` trong SQL. cho kết quả là sự kết hợp của việc áp dụng cả phép nối `'left'` và `'right'`

2.1. Nối kết DataFrame theo kiểu cơ sở dữ liệu quan hệ *(Database-Style DataFrame Joins)*

2.1.1. Nối kết ***many-to-one***

- Dữ liệu trong `df1` có nhiều hàng được gắn nhãn `a` và `b`, trong khi `df2` chỉ có một hàng cho mỗi giá trị trong cột `key`. Theo mặc định, việc ***merge*** dữ liệu sẽ sử dụng tên cột trùng nhau làm khóa (*keys*). Tuy nhiên, người dùng nên xác định rõ ràng thuộc tính tham gia nối kết thông qua đối số **on**

```
In [27]: df1 = pd.DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'a', 'b'],
    ....: 'data1': range(7)})
In [28]: df2 = pd.DataFrame({'key': ['a', 'b', 'd'],
    ....: 'data2': range(3)})
In [29]: df1
Out[29]:
   data1  key
0      0    b
1      1    b
2      2    a
3      3    c
4      4    a
5      5    a
6      6    b
```

```
In [30]: df2
Out[30]:
   data2  key
0      0    a
1      1    b
2      2    d
```

```
In [31]: pd.merge(df1, df2)
''' + hoặc dùng pd.merge(df1, df2, how='inner')
    + hoặc dùng pd.merge(df1, df2, on='key', how='inner')
    + Khi tên field dùng nối kết khác nhau, cần dùng thêm các đối
số left_on và right_on:
            pd.merge(df1, df2, left_on='key', right_on='key')'''
Out[31]:
   data1  key  data2
0      0    b      1
1      1    b      1
2      6    b      1
3      2    a      0
4      4    a      0
5      5    a      0
```

2.1. Nối kết DataFrame theo kiểu cơ sở dữ liệu quan hệ *(Database-Style DataFrame Joins)*

2.1.1. Nối kết ***many-to-one***

- Theo mặc định, việc ***`merge`*** sẽ thực hiện phép nối ***`'join'`***. Do đó trong minh họa sau, có thể nhận thấy rằng giá trị 'c' và 'd' cũng như dữ liệu liên quan bị thiếu trong kết quả do các khóa trong kết quả là phần giao nhau hoặc tập hợp chung được tìm thấy trong cả hai bảng.

```
In [35]: pd.merge(df3, df4, left_on='lkey', right_on='rkey')
Out[35]:
   data1 lkey  data2 rkey
0      0    b      1    b
1      1    b      1    b
2      6    b      1    b
3      2    a      0    a
4      4    a      0    a
5      5    a      0    a
```

df3

	lkey	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

df4

	rkey	data2
0	a	0
1	b	1
2	d	2

2.1. Nối kết DataFrame theo kiểu cơ sở dữ liệu quan hệ *(Database-Style DataFrame Joins)*

2.1.1. Nối kết ***many-to-one***

- Phép nối ngoài (*outer join*) lấy sự kết hợp của các keys, cho kết quả là sự kết hợp của việc áp dụng cả phép nối **`'left'`** và **`'right'`**:

df1

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

df2

	data2	key
0	0	a
1	1	b
2	2	d

```
In [36]: pd.merge(df1, df2, how='outer')
Out[36]:
    data1  key  data2
0     0.0    b    1.0
1     1.0    b    1.0
2     6.0    b    1.0
3     2.0    a    0.0
4     4.0    a    0.0
5     5.0    a    0.0
6     3.0    c    NaN
7     NaN    d    2.0
```

2.1.2. Nối kết ***many-to-many***

df1

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

JOIN

df2

	data2	key
0	0	a
1	1	b
2	2	a
3	3	b
4	4	d

INNER

```
In [50]: pd.merge(df1, df2, how='inner')
Out[50]:
   data1 key  data2
0      0   b      1
1      0   b      3
2      1   b      1
3      1   b      3
4      5   b      1
5      5   b      3
6      2   a      0
7      2   a      2
8      4   a      0
9      4   a      2
```

LEFT

```
In [41]: pd.merge(df1, df2, on='key', how='left')
Out[41]:
   data1 key  data2
0      0   b    1.0
1      0   b    3.0
2      1   b    1.0
3      1   b    3.0
4      2   a    0.0
5      2   a    2.0
6      3   c    NaN
7      4   a    0.0
8      4   a    2.0
9      5   b    1.0
10     5   b    3.0
```

2.1. Nối kết DataFrame theo kiểu cơ sở dữ liệu quan hệ *(Database-Style DataFrame Joins)*

2.1.2. Nối kết ***many-to-many***

- Phép nối ngoài (*outer join*) lấy sự kết hợp của các keys, cho kết quả là sự kết hợp của việc áp dụng cả phép nối ***'left'*** và ***'right'***:

df1

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

JOIN

df2

	data2	key
0	0	a
1	1	b
2	2	a
3	3	b
4	4	d

```
In [41]: pd.merge(df1, df2, on='key', how='outer')
Out[41]:
```

	key	data1	data2
0	b	0	1
1	b	0	3
2	b	1	1
3	b	1	3
4	b	5	1
5	b	5	3
6	a	2	0
7	a	2	2
8	a	4	0
9	a	4	2
10	c	3	**NaN**
11	d	**NaN**	4

```
In [42]: pd.merge(df1, df2, on='key', how='left')
Out[42]:
```

	key	data1	data2
0	b	0	1
1	b	0	3
2	b	1	1
3	b	1	3
4	a	2	0
5	a	2	2
6	c	3	**NaN**
7	a	4	0
8	a	4	2
9	b	5	1
10	b	5	3

```
In [43]: pd.merge(df1, df2, on='key', how='right')
Out[43]:
```

	key	data1	data2
0	a	2	0
1	a	4	0
2	b	0	1
3	b	1	1
4	b	5	1
5	a	2	2
6	a	4	2
7	b	0	3
8	b	1	3
9	b	5	3
10	d	**NaN**	4

2.1. Nối kết DataFrame theo kiểu cơ sở dữ liệu quan hệ *(Database-Style DataFrame Joins)*

2.1.2. Nối kết ***many-to-many***

- Để hợp nhất với nhiều khóa, hãy chuyển list chứa tên các cột của cả 2 table:

```
In [44]: leftDF = pd.DataFrame({'key1': ['foo', 'foo', 'bar'],
   ....: 'key2': ['one', 'two', 'one'],
   ....: 'lval': [1, 2, 3]})
   ....: leftDF
Out[44]:
   key1 key2  lval
0   foo  one     1
1   foo  two     2
2   bar  one     3

In [45]: rightDF = pd.DataFrame({'key1': ['foo', 'foo', 'bar', 'bar'],
   ....: 'key2': ['one', 'one', 'one', 'two'],
   ....: 'rval': [4, 5, 6, 7]})
   ....: rightDF
Out[45]:
   key1 key2  rval
0   foo  one     4
1   foo  one     5
2   bar  one     6
3   bar  two     7

In [46]: pd.merge(leftDF, rightDF, on=['key1', 'key2'], how='outer')
Out[46]:
   key1 key2  lval  rval
0   foo  one   1.0   4.0
1   foo  one   1.0   5.0
2   foo  two   2.0   NaN
3   bar  one   3.0   6.0
4   bar  two   NaN   7.0
```

Thận trọng: Khi thực hiện join trên nhiều cột, các chỉ mục trên đối tượng *DataFrame* đã truyền sẽ bị loại bỏ.

2.1.2. Nối kết ***many-to-many***

- Xử lý các tên cột trùng nhau (*overlapping column names*). Mặc dù có thể giải quyết phần trùng tên theo cách thủ công (xem lại các chương trước về đổi tên nhãn trục), việc hợp nhất có tùy chọn **`suffixes`** để chỉ định các chuỗi để nối vào các tên trùng nhau trong các đối tượng *DataFrame* bên trái và bên phải.
  - Theo mặc định, tên cột trùng nhau ở bên trái sẽ thêm hậu tố **`_x`** và bên phải sẽ thêm **`_y`**:

leftDF

	key1	key2	lval
0	foo	one	1
1	foo	two	2
2	bar	one	3

rightDF

	key1	key2	rval
0	foo	one	4
1	foo	one	5
2	bar	one	6
3	bar	two	7

```
In [47]: pd.merge(leftDF, rightDF, on='key1')
Out[47]:
```

	key1	key2_x	lval	key2_y	rval
0	foo	one	1	one	4
1	foo	one	1	one	5
2	foo	two	2	one	4
3	foo	two	2	one	5
4	bar	one	3	one	6
5	bar	one	3	two	7

2.1. Nối kết DataFrame theo kiểu cơ sở dữ liệu quan hệ *(Database-Style DataFrame Joins)*

2.1.2. Nối kết ***many-to-many***

- Xử lý các tên cột trùng nhau (*overlapping column names*).
  - Người dùng có thể chỉ định chuỗi sẽ làm hậu tố cho cả 2 phía bằng tùy chọn ***suffixes***:

leftDF

	key1	key2	lval
0	foo	one	1
1	foo	two	2
2	bar	one	3

rightDF

	key1	key2	rval
0	foo	one	4
1	foo	one	5
2	bar	one	6
3	bar	two	7

```
In [48]: pd.merge(leftDF, rightDF, on='key1', suffixes=('_left', '_right'))
Out[48]:
```

	key1	key2_left	lval	key2_right	rval
0	foo	one	1	one	4
1	foo	one	1	one	5
2	foo	two	2	one	4
3	foo	two	2	one	5
4	bar	one	3	one	6
5	bar	one	3	two	7

2.2. Hợp nhất dữ liệu dựa trên chỉ mục *(Merging on Index)*

2.2.1. Hợp nhất các DataFrame dựa trên chỉ mục

- *Sử dụng phương thức* `merge` *của Pandas*: có thể chuyển ***`left_index=True`*** hoặc ***`right_index=True`*** (hoặc cả hai) để cho biết chỉ mục được sử dụng làm khóa hợp nhất:

```
In [61]: left2 = pd.DataFrame([[1., 2.], [3., 4.], [5., 6.]],
    ....:                     index=['a', 'c', 'e'],
    ....:                     columns=['Ohio', 'Nevada'])
In [62]: right2 = pd.DataFrame([[7., 8.], [9., 10.], [11., 12.],
                                [13, 14]],
    ....:                     index=['b', 'c', 'd', 'e'],
    ....:                     columns=['Missouri', 'Alabama'])
In [63]: left2
Out[63]:
   Ohio  Nevada
a   1.0     2.0
c   3.0     4.0
e   5.0     6.0
In [64]: right2
Out[64]:
   Missouri  Alabama
b       7.0      8.0
c       9.0     10.0
d      11.0     12.0
e      13.0     14.0
```

```
In [65]: pd.merge(left2, right2, how='outer', left_index=True,
    ....:                                right_index=True)
Out[65]:
   Ohio  Nevada  Missouri  Alabama
a   1.0     2.0       NaN      NaN
b   NaN     NaN       7.0      8.0
c   3.0     4.0       9.0     10.0
d   NaN     NaN      11.0     12.0
e   5.0     6.0      13.0     14.0
```

2.2. Hợp nhất dữ liệu dựa trên chỉ mục *(Merging on Index)*

2.2.1. Hợp nhất các DataFrame dựa trên chỉ mục

- Sử dụng phương thức `join` của đối tượng *DataFrame*:
  - Hợp nhất 1 *DataFrame* với 1 *DataFrame* khác (có cùng chỉ mục hoặc tương tự nhau nhưng các cột không trùng nhau). Có thể viết ngắn gọn lại lệnh của ví dụ trước để có cùng kết quả:

left2

	Ohio	Nevada
a	1.0	2.0
c	3.0	4.0
e	5.0	6.0

right2

	Missouri	Alabama
b	7.0	8.0
c	9.0	10.0
d	11.0	12.0
e	13.0	14.0

```
In [66]: left2.join(right2, how='outer')
Out[66]:
```

	Ohio	Nevada	Missouri	Alabama
a	1.0	2.0	NaN	NaN
b	NaN	NaN	7.0	8.0
c	3.0	4.0	9.0	10.0
d	NaN	NaN	11.0	12.0
e	5.0	6.0	13.0	14.0

left1

	key	value
0	a	0
1	b	1
2	a	2
3	a	3
4	b	4
5	c	5

right1

	group_val
a	3.5
b	7.0

```
In [67]: left1.join(right1, on='key')
Out[67]:
```

	key	value	group_val
0	a	0	3.5
1	b	1	7.0
2	a	2	3.5
3	a	3	3.5
4	b	4	7.0
5	c	5	NaN

2.2. Hợp nhất dữ liệu dựa trên chỉ mục *(Merging on Index)*

2.2.1. Hợp nhất các DataFrame dựa trên chỉ mục

- Sử dụng phương thức `join` của đối tượng *DataFrame*:
  - Hợp nhất 1 *DataFrame*s với 2 (hay nhiều) *DataFrame*s khác, có thể chuyển list chứa tên các *DataFrame*s sẽ tham gia:
    - *Sử dụng inner join*

```
In [68]: another = pd.DataFrame([[7., 8.], [9., 10.],
                                 [11., 12.], [16., 17.]],
    ....: index=['a', 'c', 'e', 'f'],
    ....: columns=['New York', 'Oregon'])
In [69]: another
Out[69]:
   New York  Oregon
a       7.0     8.0
c       9.0    10.0
e      11.0    12.0
f      16.0    17.0
In [70]: left2.join([right2, another])
Out[70]:
   Ohio  Nevada  Missouri  Alabama  New York  Oregon
a   1.0     2.0       NaN      NaN       7.0     8.0
c   3.0     4.0       9.0     10.0       9.0    10.0
e   5.0     6.0      13.0     14.0      11.0    12.0
```

left2

	Ohio	Nevada
a	1.0	2.0
c	3.0	4.0
e	5.0	6.0

right2

	Missouri	Alabama
b	7.0	8.0
c	9.0	10.0
d	11.0	12.0
e	13.0	14.0

2.2. Hợp nhất dữ liệu dựa trên chỉ mục *(Merging on Index)*

2.2.1. Hợp nhất các DataFrame dựa trên chỉ mục

- Sử dụng phương thức `join` của đối tượng *DataFrame*:
  - Hợp nhất 1 *DataFrames* với 2 (hay nhiều) *DataFrames* khác, có thể chuyển list chứa tên các *DataFrames* sẽ tham gia:
    - *Sử dụng outer join*

right2

	Missouri	Alabama
b	7.0	8.0
c	9.0	10.0
d	11.0	12.0
e	13.0	14.0

another

	New York	Oregon
a	7.0	8.0
c	9.0	10.0
e	11.0	12.0
f	16.0	17.0

```
In [71]: left2.join([right2, another], how='outer')
Out[71]:
```

	Ohio	Nevada	Missouri	Alabama	New York	Oregon
a	1.0	2.0	**NaN**	**NaN**	7.0	8.0
b	**NaN**	**NaN**	7.0	8.0	**NaN**	**NaN**
c	3.0	4.0	9.0	10.0	9.0	10.0
d	**NaN**	**NaN**	11.0	12.0	**NaN**	**NaN**
e	5.0	6.0	13.0	14.0	11.0	12.0
f	**NaN**	**NaN**	**NaN**	**NaN**	16.0	17.0

2.2. Hợp nhất dữ liệu dựa trên chỉ mục *(Merging on Index)*

2.2.2. Hợp nhất các DataFrame dựa trên chỉ mục và khóa (key)

- *Hợp nhất dựa trên chỉ mục và khóa chính:*
  - *Hợp nhất với inner join (mặc định):*

```
In [53]: pd.merge(left1, right1, left_on='key', right_index=True)
Out[53]:
  key  value  group_val
0   a      0        3.5
2   a      2        3.5
3   a      3        3.5
1   b      1        7.0
4   b      4        7.0
```

left1

	key	value
0	a	0
1	b	1
2	a	2
3	a	3
4	b	4
5	c	5

right1

	group_val
a	3.5
b	7.0

  - *Hợp nhất với outer join*

```
In [54]: pd.merge(left1, right1, left_on='key', right_index=True,
   .... : how='outer')
Out[54]:
  key  value  group_val
0   a      0        3.5
2   a      2        3.5
3   a      3        3.5
1   b      1        7.0
4   b      4        7.0
5   c      5        NaN
```

2.2. Hợp nhất dữ liệu dựa trên chỉ mục *(Merging on Index)*

2.2.2. Hợp nhất các DataFrame dựa trên chỉ mục và khóa (key)

- *Hợp nhất dựa trên chỉ mục có thứ bậc và khóa chính:*
  - Mọi thứ lúc này phức tạp hơn vì việc tham gia vào chỉ mục hoàn toàn là sự hợp nhất nhiều khóa. do đó phải chỉ ra nhiều cột để hợp nhất thành một danh sách (lưu ý việc xử lý các giá trị chỉ mục trùng lặp với `how='outer'`)

```
In [55]: lefth = pd.DataFrame({'key1': ['Ohio', 'Ohio', 'Ohio','Nevada', 'Nevada'],
   ....:                                'key2': [2000, 2001, 2002, 2001, 2002],
   ....:                                'data': np.arange(5.)})
In [56]: righth = pd.DataFrame(np.arange(12).reshape((6, 2)),
   ....: index=[['Nevada','Nevada', 'Ohio','Ohio','Ohio','Ohio'],
   ....:           [2001, 2000, 2000, 2000, 2001, 2002]],
                columns=['event1', 'event2'])
```

lefth

	data	key1	key2
0	0.0	Ohio	2000
1	1.0	Ohio	2001
2	2.0	Ohio	2002
3	3.0	Nevada	2001
4	4.0	Nevada	2002

righth

		event1	event2
Nevada	2001	0	1
	2000	2	3
Ohio	2000	4	5
	2000	6	7
	2001	8	9
	2002	10	11

```
In [59]: pd.merge(lefth, righth, left_on=['key1', 'key2'],
                                   right_index=True) # how='inner'
```

Out[59]:

	data	key1	key2	event1	event2
0	0.0	Ohio	2000	4	5
0	0.0	Ohio	2000	6	7
1	1.0	Ohio	2001	8	9
2	2.0	Ohio	2002	10	11
3	3.0	Nevada	2001	0	1

2.2.2. Hợp nhất các DataFrame dựa trên chỉ mục và khóa (key)

- *Hợp nhất dựa trên chỉ mục có thứ bậc và khóa chính:*

lefth

	data	key1	key2
0	0.0	Ohio	2000
1	1.0	Ohio	2001
2	2.0	Ohio	2002
3	3.0	Nevada	2001
4	4.0	Nevada	2002

righth

		event1	event2
Nevada	2001	0	1
	2000	2	3
Ohio	2000	4	5
	2000	6	7
	2001	8	9
	2002	10	11

```
In [59]: pd.merge(lefth, righth, left_on=['key1', 'key2'],
                                    right_index=True) # how='inner'
Out[59]:
   data   key1   key2   event1   event2
0   0.0   Ohio   2000        4        5
0   0.0   Ohio   2000        6        7
1   1.0   Ohio   2001        8        9
2   2.0   Ohio   2002       10       11
3   3.0 Nevada   2001        0        1
```

```
In [60]: pd.merge(lefth, righth, left_on=['key1', 'key2'],
    ....:                          right_index=True, how='outer')
Out[60]:
   data   key1   key2   event1   event2
0   0.0   Ohio   2000      4.0      5.0
0   0.0   Ohio   2000      6.0      7.0
1   1.0   Ohio   2001      8.0      9.0
2   2.0   Ohio   2002     10.0     11.0
3   3.0 Nevada   2001      0.0      1.0
4   4.0 Nevada   2002      NaN      NaN
```

2.3. Nối dữ liệu dọc theo một trục *(Concatenating Along an Axis)*

Một loại thao tác kết hợp dữ liệu khác là nối dữ liệu dọc theo 1 trục. Tùy theo cách nhìn của từng người mà thao tác này có thể được gọi bằng nhiều tên khác nhau như:

- Nối (*concatenation*)
- Liên kết (*binding*)
- Xếp chồng (*stacking*).

2.3.1. Đối với mảng trong Numpy

```
In [72]: arr = np.arange(12).reshape((3, 4))
In [73]: arr
Out[73]:
     array([[ 0, 1, 2, 3],
            [ 4, 5, 6, 7],
            [ 8, 9, 10, 11]])
In [74]: np.concatenate([arr, arr], axis=1)
Out[74]:
     array([[ 0, 1, 2, 3, 0, 1, 2, 3],
            [ 4, 5, 6, 7, 4, 5, 6, 7],
            [ 8, 9, 10, 11, 8, 9, 10, 11]])
```

2.3.2. Đối với Series và DataFrame trong Pandas

- Việc có các trục được gắn nhãn cho phép khái quát hóa hơn nữa việc nối mảng. Đặc biệt, có thêm một số điều cần suy nghĩ:
  - Nếu các đối tượng được lập chỉ mục khác nhau trên các trục khác, nên kết hợp các phần tử riêng biệt trong các trục này hay chỉ sử dụng các giá trị chung (điểm giao)?
  - Các khối dữ liệu được nối có cần được nhận dạng trong đối tượng kết quả không?
  - "Trục nối" (*"concatenation axis"*) có chứa dữ liệu cần được bảo toàn không? Trong nhiều trường hợp, tốt nhất nên loại bỏ các nhãn số nguyên mặc định trong *DataFrame* trong quá trình nối.

Hàm **`concat`** trong *pandas* cung cấp một cách nhất quán để giải quyết từng vấn đề này.

2.3. Nối dữ liệu dọc theo một trục

2.3.2. Đối với Series và DataFram

Các đối số của hàm concat

Đối số	*Description*
`objs`	List hoặc dictionary của các đối tượng pandas sẽ được nối; đây là đối số bắt buộc duy nhất
`axis`	Trục mà dữ liệu sẽ được nối; mặc định là 0 (nối dọc theo hàng)
`join`	`'inner'` hoặc `'outer'` (`'outer'` theo mặc định); cho biết các chỉ mục sẽ thực hiện phép giao (*intersection* - khi dùng `inner`) hay thực hiện phép hội (*union* - khi dùng `outer`) với nhau dọc theo các trục
`join_axes`	Các chỉ mục cụ thể để sử dụng cho các trục quan hệ *n–1* khác thay vì thực hiện các phép hợp/giao logic
`keys`	Các giá trị để liên kết với các đối tượng được nối (*concatenated*), tạo thành chỉ mục phân cấp dọc theo trục nối (*concatenation axis*); có thể là một list hoặc array các giá trị tùy ý, một array các tuple hoặc một list các array (nếu array nhiều cấp được truyền theo đối số `levels`)
`levels`	Các chỉ mục cụ thể để sử dụng làm mức (*level*) hoặc các mức của chỉ mục phân cấp (hierarchical index level) nếu đối số keys được truyền.
`names`	Tên cho các cấp độ phân cấp được tạo nếu đối số `keys` và/hoặc `levels` được truyền
`verify_integrity`	Kiểm tra trục mới trong đối tượng được nối xem có trùng lặp không và phát sinh ngoại lệ (`exception`) nếu có; theo mặc định (`False`) cho phép trùng lặp
`ignore_index`	Không bảo toàn các chỉ mục dọc theo trục nối (*concatenation axis*), thay vào đó hãy tạo mới chỉ mục với `range(total_length)`

2.3.2. Đối với Series và DataFrame trong Pandas

- *Series*
  - Giả sử có ba *Series* không có chỉ mục trùng lặp:

```
In [75]: s1 = pd.Series([0, 1], index=['a', 'b'])
In [76]: s2 = pd.Series([2, 3, 4], index=['c', 'd', 'e'])
In [77]: s3 = pd.Series([5, 6], index=['f', 'g'])
```

  - Gọi **concat** với các đối tượng này trong danh sách sẽ gắn kết các giá trị và chỉ mục lại với nhau

```
In [78]: pd.concat([s1, s2, s3])
Out[78]:
     a 0
     b 1
     c 2
     d 3
     e 4
     f 5
     g 6
     dtype: int64
```

2.3. Nối dữ liệu dọc theo một trục

2.3.2. Đối với Series và DataFrame trong Pandas

- *Series*

  - Theo mặc định, concat hoạt động dọc theo axis=0, tạo ra một *Series* khác. Nếu có thêm đối số `axis=1`, thay vào đó, kết quả sẽ là *DataFrame* (`axis=1` là các cột):

```
s1
a    0
b    1
dtype: int64
```

```
s2
c    2
d    3
e    4
dtype: int64
```

```
s3
f    5
g    6
dtype: int64
```

```
In [78]: pd.concat([s1, s2, s3])
Out[78]:
     a 0
     b 1
     c 2
     d 3
     e 4
     f 5
     g 6
     dtype: int64
```

```
In [79]: pd.concat([s1, s2, s3], axis=1)
Out[79]:
         0    1    2
     a 0.0  NaN  NaN
     b 1.0  NaN  NaN
     c NaN  2.0  NaN
     d NaN  3.0  NaN
     e NaN  4.0  NaN
     f NaN  NaN  5.0
     g NaN  NaN  6.0
```

2.3.2. Đối với Series và DataFrame trong Pandas

- *Series*
  - `join` trong `concat`:
    - Mặc định dữ liệu liên kết được sắp xếp chồng lên nhau (`join='outer'`) của các chỉ mục:

```
In [80]: s4 = pd.concat([s1, s3])# join='outer'
In [81]: s4
Out[81]:
     a 0
     b 1
     f 5
     g 6
     dtype: int64
In [82]: pd.concat([s1, s4], axis=1) # join='outer'
Out[82]:
         0   1
     a 0.0 0
     b 1.0 1
     f NaN 5
     g NaN 6
```

    - Thay vào đó, có thể giao nhau bằng cách sử dụng `join='inner'`:

```
In [83]: pd.concat([s1, s4], axis=1, join='inner')
Out[83]:
        0   1
     a  0   0
     b  1   1
```

Trong ví dụ trên, nhãn `'f'` và `'g'` biến mất do tùy chọn `join='inner'`.

2.3. Nối dữ liệu dọc theo một trục

2.3.2. Đối với Series và DataFrame trong Pandas

- *Series*
  - Có thể chỉ định các trục (*axes*) sẽ được sử dụng trên các trục khác bằng đối số `join_axes`:

```
In [84]: pd.concat([s1, s4], axis=1, join_axes=[['a','c','b','e']])
Out[84]:
     0    1
a  0.0  0.0
c  NaN  NaN
b  1.0  1.0
e  NaN  NaN
```

s1

```
a    0
b    1
dtype: int64
```

s4

```
a    0
b    1
f    5
g    6
dtype: int64
```

2.3. Nối dữ liệu dọc theo một trục

2.3.2. Đối với Series và DataFrame trong Pandas

- *Series*

  - Tạo chỉ mục phân cấp trên trục nối bằng cách sử dụng đối số `keys`:

```
In [85]: result = pd.concat([s1, s2, s3], keys=['one','two','three'])
In [86]: result
Out[86]:
one    a    0
       b    1
two    c    2
       d    3
       e    4
three  f    5
       g    6
dtype: int64
In [87]: result.unstack()
Out[87]:
         a    b    c    d    e    f    g
one    0.0  1.0  NaN  NaN  NaN  NaN  NaN
two    NaN  NaN  2.0  3.0  4.0  NaN  NaN
three  NaN  NaN  NaN  NaN  NaN  5.0  6.0
```

```
s1
a    0
b    1
dtype: int64
```

```
s2
c    2
d    3
e    4
dtype: int64
```

```
s3
f    5
g    6
dtype: int64
```

2.3.2. Đối với Series và DataFrame trong Pandas

- *Series*
  - Trong trường hợp kết hợp *Series* dọc theo `axis=1`, các khóa sẽ trở thành tiêu đề cột *DataFrame*:

```
In [88]: pd.concat([s1, s2, s3], axis=1,
                   keys=['one', 'two', 'three'])
Out[88]:
   one  two  three
a  0.0  NaN    NaN
b  1.0  NaN    NaN
c  NaN  2.0    NaN
d  NaN  3.0    NaN
e  NaN  4.0    NaN
f  NaN  NaN    5.0
g  NaN  NaN    6.0
```

2.3. Nối dữ liệu dọc theo một trục

2.3.2. Đối với Series và DataFrame

- *DataFrame*

df1

	one	two
a	0	1
b	2	3
c	4	5

df2

	three	four
a	5	6
c	7	8

2 cách dùng đều cho cùng 1 kết quả

```
In [92]: pd.concat(df1, df2)
Out[92]:
     one  two  three  four
a 0.0  1.0    NaN   NaN
b 2.0  3.0    NaN   NaN
c 4.0  5.0    NaN   NaN
a NaN  NaN    5.0   6.0
c NaN  NaN    7.0   8.0
In [93]: pd.concat([df1, df2], axis=1)
Out[93]:
     one  two  three  four
a 0.0  1.0    5.0   6.0
b 2.0  3.0    NaN   NaN
c 4.0  5.0    7.0   8.0
In [94]: pd.concat([df1, df2], axis=1, keys=['level1', 'level2'])
Out[94]:
     level1     level2
      one  two  three  four
a   0    1    5.0   6.0
b   2    3    NaN   NaN
c   4    5    7.0   8.0
```

```
In [95]: pd.concat({'level1': df1, 'level2': df2}, axis=1)
Out[95]:
     level1     level2
      one  two  three  four
a   0    1    5.0   6.0
b   2    3    NaN   NaN
c   4    5    7.0   8.0
```

2.3. Nối dữ liệu dọc theo một trục

2.3.2. Đối với Series và DataFrame trong Pandas

- *DataFrame*
  - Đặt tên cho các cấp độ trục được tạo bằng đối số `names`:

df1

	one	two
a	0	1
b	2	3
c	4	5

df2

	three	four
a	5	6
c	7	8

```
In [95]: pd.concat({'level1': df1, 'level2': df2}, axis=1)
Out[95]:
      level1     level2
       one  two   three  four
    a    0    1     5.0   6.0
    b    2    3     NaN   NaN
    c    4    5     7.0   8.0
```

```
In [96]: pd.concat([df1, df2], axis=1, keys=['level1', 'level2'],
   .....:                                 names=['upper', 'lower'])
Out[96]:
    upper level1    level2
    lower  one  two three  four
    a        0    1   5.0   6.0
    b        2    3   NaN   NaN
    c        4    5   7.0   8.0
```

2.4. Kết hợp dữ liệu với sự trùng lắp của chỉ mục *(Combining Data with Overlap)*

Trong thực tế, có thể có hai tập dữ liệu có chỉ mục trùng nhau toàn bộ hoặc một phần.

- Đối với mảng của *NumPy*: hàm `where` thực hiện định hướng mảng (*array-oriented*) tương đương với biểu thức `if-else`:

```
X
f    NaN
e    2.5
d    NaN
c    3.5
b    4.5
a    NaN
dtype: float64
```

```
Y
f    0.0
e    1.0
d    2.0
c    3.0
b    4.0
a    NaN
dtype: float64
```

```
In [106]: np.where(pd.isnull(X), Y, X)
Out[106]: array([ 0. , 2.5, 2. , 3.5, 4.5, nan])
```

- Với *Series:* có phương thức ***Combine_first***, thực hiện tương đương với ***where*** của mảng, cộng thêm tính năng căn chỉnh dữ liệu thông thường của *pandas*

```
In [107]: Y[:-2].combine_first(X[2:])
Out[107]:
a    NaN
b    4.5
c    3.0
d    2.0
e    1.0
f    0.0
dtype: float64
```

```
Y
f    0.0
e    1.0
d    2.0
c    3.0
b    4.0
a    NaN
dtype: float64
```

```
X
f    NaN
e    2.5
d    NaN
c    3.5
b    4.5
a    NaN
dtype: float64
```

2.4. Kết hợp dữ liệu với sự trùng lắp của chỉ mục *(Combining Data with Overlap)*

- Với *DataFrames*: **`Combine_first`** thực hiện điều tương tự theo từng cột, vì vậy có thể coi đó là "vá" (“patching”) dữ liệu bị thiếu trong đối tượng gọi với dữ liệu từ đối tượng được truyền:

df1

	a	b	c
0	1.0	**NaN**	2
1	**NaN**	2.0	6
2	5.0	**NaN**	10
3	**NaN**	6.0	14

df2

	a	b
0	5.0	**NaN**
1	4.0	3.0
2	**NaN**	4.0
3	3.0	6.0
4	7.0	8.0

```
In [112]: df1.combine_first(df2)
Out[112]:
```

	a	b	c
0	1.0	**NaN**	2.0
1	**4.0**	2.0	6.0
2	5.0	**4.0**	10.0
3	**3.0**	6.0	14.0
4	7.0	**8.0**	**NaN**

3. ĐỊNH HÌNH LẠI VÀ XOAY TRỤC DỮ LIỆU *(Reshaping and Pivoting)*

3.1. Định hình lại dữ liệu bằng lập chỉ mục phân cấp *(Reshaping with Hierarchical Indexing)*

Lập chỉ mục phân cấp cung cấp một cách nhất quán để sắp xếp lại dữ liệu trong *DataFrame*. Có hai hành động chính:

- **`stack`**: Thao tác này sẽ "xoay" (*"rotates"*) hoặc xoay (*pivot*) từ các cột trong dữ liệu sang các hàng.
- **`unstack`**: thực hiện ngược lại với stack là xoay từ các hàng trong dữ liệu sang các cột.

3.1. Định hình lại dữ liệu bằng lập chỉ mục phân cấp *(Reshaping with Hierarchical Indexing)*

3.1.1. stack

- Phương thức ***stack*** sẽ xoay các cột thành các hàng, tạo ra một *Series*:

```
In [113]: data = pd.DataFrame(np.arange(6).reshape((2, 3)),
     .....: index=pd.Index(['Ohio', 'Colorado'], name='state'),
     .....: columns=pd.Index(['one', 'two', 'three'],
     .....: name='number'))
In [114]: data
Out[114]:
number    one   two  three
state
Ohio        0     1      2
Colorado    3     4      5
In [115]: result = data.stack()
In [116]: result
Out[116]:
state     number
Ohio        one     0
            two     1
            three   2
Colorado    one     3
            two     4
            three   5
dtype: int64
```

3.1. Định hình lại dữ liệu bằng lập chỉ mục phân cấp *(Reshaping with Hierarchical Indexing)*

3.1.2. unstack

- Từ *Series* được lập chỉ mục theo thứ bậc, có thể sắp xếp lại dữ liệu trở lại *DataFrame* với ***unstack***:

```
result
state     number
Ohio      one       0
          two       1
          three     2
Colorado  one       3
          two       4
          three     5
dtype: int64
```

```
In [117]: result.unstack()
Out[117]:
      number    one    two  three
      state
      Ohio        0      1      2
      Colorado    3      4      5
```

- Có thể `unstack` một cấp độ khác bằng cách chuyển số hoặc tên của mức:

```
In [118]: result.unstack(0)
Out[118]:
      state   Ohio  Colorado
      number
      one        0         3
      two        1         4
      three      2         5
```

*sử dụng **số** của mức*

```
In [119]: result.unstack('state')
Out[119]:
      state   Ohio  Colorado
      number
      one        0         3
      two        1         4
      three      2         5
```

*sử dụng **tên** của mức*

3.1. Định hình lại dữ liệu bằng lập chỉ mục phân cấp *(Reshaping with Hierarchical Indexing)*

3.1.2. unstack

- Việc ***unstack*** có thể tạo ra dữ liệu bị thiếu (NaN) nếu không tìm thấy tất cả các giá trị ở cấp độ trong mỗi nhóm con:

```
In [120]: s1 = pd.Series([0, 1, 2, 3], index=['a', 'b', 'c', 'd'])
In [121]: s2 = pd.Series([4, 5, 6], index=['c', 'd', 'e'])
In [122]: data2 = pd.concat([s1, s2], keys=['one', 'two'])
In [123]: data2
Out[123]:
     one   a    0
           b    1
           c    2
           d    3
     two   c    4
           d    5
           e    6
     dtype: int64
In [124]: data2.unstack()
Out[124]:
             a    b    c    d    e
     one   0.0  1.0  2.0  3.0  NaN
     two   NaN  NaN  4.0  5.0  6.0
```

3.1. Định hình lại dữ liệu bằng lập chỉ mục phân cấp *(Reshaping with Hierarchical Indexing)*

3.1.2. unstack

- Việc ***unstack*** có thể tạo ra dữ liệu bị thiếu (NaN) nếu không tìm thấy tất cả các giá trị ở cấp độ trong mỗi nhóm con:

```
In [125]: data2.unstack()
Out[125]:
          a    b    c    d    e
one     0.0  1.0  2.0  3.0  NaN
two     NaN  NaN  4.0  5.0  6.0
In [126]: data2.unstack().stack()
Out[126]:
one  a    0
     b    1
     c    2
     d    3
two  c    4
     d    5
     e    6
dtype: float64
```

```
In [127]: data2.unstack().stack(dropna=False)
Out[127]:
one  a    0
     b    1
     c    2
     d    3
     e  NaN
two  a  NaN
     b  NaN
     c    4
     d    5
     e    6
dtype: int64
```

3.1. Định hình lại dữ liệu bằng lập chỉ mục phân cấp

3.1.2. unstack

- Khi ***unstack*** trong *DataFrame*, mức ***unstack*** sẽ trở thành mức thấp nhất trong kết quả:

```
In [128]: df = pd.DataFrame({'left': result, 'right': result + 5},
     .....:                  columns=pd.Index(['left', 'right'], name='side'))
In [128]: df
Out[129]:
```

side		left	right
state	number		
Ohio	one	0	5
	two	1	6
	three	2	7
Colorado	one	3	8
	two	4	9
	three	5	10

```
In [130]: df.unstack('state')
Out[130]:
```

side	left		right	
state	Ohio	Colorado	Ohio	Colorado
number				
one	0	3	5	8
two	1	4	6	9
three	2	5	7	10

3.1. Định hình lại dữ liệu bằng lập chỉ mục phân cấp

3.1.2. unstack

- Khi gọi ***stack***, có thể chỉ ra tên của trục cần ***stack***:

```
df
side                 left   right
state    number
Ohio        one         0       5
            two         1       6
          three         2       7
Colorado    one         3       8
            two         4       9
          three         5      10
```

```
df.unstack('state')
side       left                  right
state      Ohio  Colorado         Ohio  Colorado
number
one           0         3            5         8
two           1         4            6         9
three         2         5            7        10
```

```
In [131]: df.unstack('state').stack('side')
Out[131]:
      state             Colorado      Ohio
      number     side
      one         left         3         0
                 right         8         5
      two         left         4         1
                 right         9         6
      three       left         5         2
                 right        10         7
```

3.2. Xoay trục dữ liệu từ định dạng "dài" sang định dạng "rộng" *(Pivoting "Long" to "Wide" Format)*

- Phương thức ***PeriodIndex*** thực hiện kết hợp các cột `year` và `quarter` để tạo ra một kiểu *time interval* (khoảng thời gian).

```
In [132]: data = pd.read_csv('D:/macrodata.csv')
In [133]: data.head()
Out[133]:
     year quarter  realgdp realcons  realinv  realgovt  realdpi   cpi \
0  1959.0    1.0  2710.349   1707.4  286.898   470.045   1886.9  28.98
1  1959.0    2.0  2778.801   1733.7  310.859   481.301   1919.7  29.15
2  1959.0    3.0  2775.488   1751.8  289.226   491.260   1916.4  29.35
3  1959.0    4.0  2785.204   1753.7  299.356   484.052   1931.3  29.37
4  1960.0    1.0  2847.699   1770.5  331.722   462.199   1955.5  29.54
      m1  tbilrate  unemp      pop  infl  realint
0  139.7      2.82    5.8  177.146  0.00     0.00
1  141.7      3.08    5.1  177.830  2.34     0.74
2  140.5      3.82    5.3  178.657  2.74     1.09
3  140.0      4.33    5.6  179.386  0.27     4.06
4  139.6      3.50    5.2  180.007  2.31     1.19
In [134]: periods = pd.PeriodIndex(year = data.year,
   .....:                          quarter = data.quarter, name = 'date')
In [135]: columns = pd.Index(['realgdp', 'infl', 'unemp'], name='item')
In [136]: data = data.reindex(columns=columns)
In [137]: data.index = periods.to_timestamp('D', 'end')
In [138]: ldata=data.stack().reset_index().rename(columns={0: 'value'})
```

3.2. Xoay trục dữ liệu từ định dạng "dài" sang định dạng "rộng" *(Pivoting "Long" to "Wide" Format)*

`ldata` có dạng như hình sau. Đây được gọi là định dạng dài (*long* format) cho nhiều *Series* thời gian (*multiple time Series*) hoặc dữ liệu quan sát khác có hai khóa (*keys*) trở lên (ở đây, khóa là **`date`** và **`item`**). Mỗi hàng trong bảng đại diện cho một quan sát duy nhất.

Dữ liệu dạng này thường được lưu trữ trong cơ sở dữ liệu quan hệ (như MySQL), dưới dạng lược đồ cố định (tên cột và kiểu dữ liệu) cho phép số lượng giá trị riêng biệt trong cột **`item`** thay đổi khi dữ liệu được thêm vào bảng.

Trong ví dụ trước này, **`date`** và **`item`** là khóa chính (theo cách nói của CSDL quan hệ).

```
In [139]: ldata[:10]
Out[139]:
         date     item     value
0  1959-03-31  realgdp  2710.349
1  1959-03-31     infl     0.000
2  1959-03-31    unemp     5.800
3  1959-06-30  realgdp  2778.801
4  1959-06-30     infl     2.340
5  1959-06-30    unemp     5.100
6  1959-09-30  realgdp  2775.488
7  1959-09-30     infl     2.740
8  1959-09-30    unemp     5.300
9  1959-12-31  realgdp  2785.204
```

3.2. Xoay trục dữ liệu từ định dạng "dài" sang định dạng "rộng"

Trong một số trường hợp, dữ liệu có thể khó xử lý hơn ở định dạng này; có thể ta muốn có một *DataFrame* chứa một cột cho mỗi giá trị mục riêng biệt được lập chỉ mục theo dấu thời gian trong cột `date`. Phương thức xoay của *DataFrame* thực hiện chính xác phép chuyển đổi này:

Hai giá trị đầu tiên được truyền là các cột được sử dụng tương ứng làm chỉ mục hàng và cột, sau đó cuối cùng là cột giá trị tùy chọn để điền vào *DataFrame*.

```
In [140]: pivoted = ldata.pivot('date',
                                'item', 'value')
In [141]: pivoted
Out[141]:
            item     infl     realgdp  unemp
date
1959-03-31   0.00  2710.349      5.8
1959-06-30   2.34  2778.801      5.1
1959-09-30   2.74  2775.488      5.3
1959-12-31   0.27  2785.204      5.6
1960-03-31   2.31  2847.699      5.2
1960-06-30   0.14  2834.390      5.2
1960-09-30   2.70  2839.022      5.6
1960-12-31   1.21  2802.616      6.3
1961-03-31  -0.40  2819.264      6.8
1961-06-30   1.47  2872.005      7.0
...           ...       ...      ...
2007-06-30   2.75  13203.977     4.5
2007-09-30   3.45  13321.109     4.7
2007-12-31   6.38  13391.249     4.8
2008-03-31   2.82  13366.865     4.9
2008-06-30   8.53  13415.266     5.4
2008-09-30  -3.16  13324.600     6.0
2008-12-31  -8.79  13141.920     6.9
2009-03-31   0.94  12925.410     8.1
2009-06-30   3.37  12901.504     9.2
2009-09-30   3.56  12990.341     9.6
[203 rows x 3 columns]
```

3.2. Xoay trục dữ liệu từ định dạng "dài" sang định dạng "rộng"

Giả sử ta muốn định hình lại đồng thời giá trị của hai cột:

Bằng cách bỏ qua đối số cuối cùng, sẽ có được *DataFrame* với các cột phân cấp:

```
In [142]: ldata['value2'] = np.random.randn(len(ldata))
In [143]: ldata[:10]
Out[143]:
         date       item       value      value2
0  1959-03-31    realgdp    2710.349    0.523772
1  1959-03-31       infl       0.000    0.000940
2  1959-03-31      unemp       5.800    1.343810
3  1959-06-30    realgdp    2778.801   -0.713544
4  1959-06-30       infl       2.340   -0.831154
5  1959-06-30      unemp       5.100   -2.370232
6  1959-09-30    realgdp    2775.488   -1.860761
7  1959-09-30       infl       2.740   -0.860757
8  1959-09-30      unemp       5.300    0.560145
9  1959-12-31    realgdp    2785.204   -1.265934
In [144]: pivoted = ldata.pivot('date', 'item')
In [145]: pivoted[:5]
Out[145]:
            value                          value2
item         infl    realgdp  unemp        infl     realgdp     unemp
date
1959-03-31   0.00   2710.349    5.8    0.000940    0.523772  1.343810
1959-06-30   2.34   2778.801    5.1   -0.831154   -0.713544 -2.370232
1959-09-30   2.74   2775.488    5.3   -0.860757   -1.860761  0.560145
1959-12-31   0.27   2785.204    5.6    0.119827   -1.265934 -1.063512
1960-03-31   2.31   2847.699    5.2   -2.359419    0.332883 -0.199543

In [146]: pivoted['value'][:5]
Out[146]:
item           infl     realgdp  unemp
date
1959-03-31     0.00    2710.349    5.8
1959-06-30     2.34    2778.801    5.1
1959-09-30     2.74    2775.488    5.3
1959-12-31     0.27    2785.204    5.6
1960-03-31     2.31    2847.699    5.2
```

3.2. Xoay trục dữ liệu từ định dạng "dài" sang định dạng "rộng"

Lưu ý rằng `pivot` tương đương với việc tạo chỉ mục phân cấp bằng cách sử dụng ***`set_index`***, sau đó là lệnh gọi để ***`unstack`***:

```
In [147]: unstacked = ldata.set_index(['date', 'item']).unstack('item')
In [148]: unstacked[:7]
Out[148]:
```

	value			value2		
item	infl	realgdp	unemp	infl	realgdp	unemp
date						
1959-03-31	0.00	2710.349	5.8	0.000940	0.523772	1.343810
1959-06-30	2.34	2778.801	5.1	-0.831154	-0.713544	-2.370232
1959-09-30	2.74	2775.488	5.3	-0.860757	-1.860761	0.560145
1959-12-31	0.27	2785.204	5.6	0.119827	-1.265934	-1.063512
1960-03-31	2.31	2847.699	5.2	-2.359419	0.332883	-0.199543
1960-06-30	0.14	2834.390	5.2	-0.970736	-1.541996	-1.307030
1960-09-30	2.70	2839.022	5.6	0.377984	0.286350	-0.753887

3.3. Xoay trục dữ liệu từ định dạng "rộng" sang định dạng "dài" *(Pivoting "Wide" to "Long" Format)*

Một thao tác nghịch đảo để ***pivot*** cho *DataFrames* là ***pandas.melt***. Thay vì chuyển đổi một cột thành nhiều cột trong *DataFrame* mới, ***pandas.melt*** hợp nhất nhiều cột thành một, tạo ra *DataFrame* dài hơn dữ liệu đầu vào.

```
In [149]: df = pd.DataFrame(
     .....:{'key': ['foo', 'bar', 'baz'],
     .....: 'A': [1, 2, 3],
     .....: 'B': [4, 5, 6],
     .....: 'C': [7, 8, 9]})
In [150]: df
Out[150]:
          A   B   C   key
     0    1   4   7   foo
     1    2   5   8   bar
     2    3   6   9   baz
```

Cột ***'key'*** có thể là chỉ báo nhóm và các cột khác là giá trị dữ liệu. Khi sử dụng ***pandas.melt***, phải chỉ ra cột nào (nếu có) là chỉ báo nhóm (*group indicators*). Hãy sử dụng ***'key'***làm chỉ báo nhóm duy nhất ở đây:

```
In [151]: melted = pd.melt(df, ['key'])
In [152]: melted
Out[152]:
          key   variable   value
     0    foo          A       1
     1    bar          A       2
     2    baz          A       3
     3    foo          B       4
     4    bar          B       5
     5    baz          B       6
     6    foo          C       7
     7    bar          C       8
     8    baz          C       9
```

3.3. Xoay trục dữ liệu từ định dạng "rộng" sang định dạng "dài" *(Pivoting "Wide" to "Long" Format)*

- Sử dụng ***pivot***, có thể định hình lại bố cục ban đầu:

df

	A	B	C	key
0	1	4	7	foo
1	2	5	8	bar
2	3	6	9	baz

```
melted = pd.melt(df, ['key'])
```

melted

	key	variable	value
0	foo	A	1
1	bar	A	2
2	baz	A	3
3	foo	B	4
4	bar	B	5
5	baz	B	6
6	foo	C	7
7	bar	C	8
8	baz	C	9

```
In [153]: reshaped = melted.pivot('key', 'variable', 'value')
In [154]: reshaped
Out[154]:
variable   A    B    C
key
bar        2    5    8
baz        3    6    9
foo        1    4    7
```

3.3. Xoay trục dữ liệu từ định dạng "rộng" sang định dạng "dài" *(Pivoting "Wide" to "Long" Format)*

- Vì kết quả của ***pivot*** tạo chỉ mục từ cột được sử dụng làm nhãn của hàng nên có thể người dùng muốn sử dụng ***reset_index*** để di chuyển dữ liệu trở lại cột:

```
In [155]: reshaped.reset_index()
Out[155]:
  variable key    A    B    C
0        bar   2    5    8
1        baz   3    6    9
2        foo   1    4    7
```

- Cũng có thể chỉ định một tập hợp con các cột để sử dụng làm cột giá trị:

```
In [156]: pd.melt(df, id_vars=['key'],
                  value_vars=['A', 'B'])
Out[156]:
   key variable  value
0  foo        A      1
1  bar        A      2
2  baz        A      3
3  foo        B      4
4  bar        B      5
5  baz        B      6
```

3.3. Xoay trục dữ liệu từ định dạng "rộng" sang định dạng "dài"

- ***pandas.melt*** cũng có thể được sử dụng mà không cần bất kỳ mã định danh nhóm nào:

```
In [157]: pd.melt(df, value_vars=['A', 'B', 'C'])
Out[157]:
   variable  value
0         A      1
1         A      2
2         A      3
3         B      4
4         B      5
5         B      6
6         C      7
7         C      8
8         C      9
In [158]: pd.melt(df, value_vars=['key', 'A', 'B'])
Out[158]:
   variable value
0       key   foo
1       key   bar
2       key   baz
3         A     1
4         A     2
5         A     3
6         B     4
7         B     5
8         B     6
```


Q&A
How
Why
When
Where
What
Who