Làm việc với `DataFrame` (tiếp)

23-12-2020

1 DataFrame mẫu

```
In [1]: # Như bài trước, chúng ta sẽ dùng các DataFrame sau
        import numpy as np
        import pandas as pd

        date = np.datetime64('2019-11-25', 'D') + np.random.randint(-100, 100, size =
        product = np.random.choice(['Apple', 'Banana', 'Cherry'], size = 100)
        quantity = np.random.randint(100, size = 100)

        df1 = pd.DataFrame({
            'Date' : date,
            'Product' : product,
            'Quantity' : quantity
        })
        df2 = pd.DataFrame(
            [['Apple', 10], ['Banana', 15], ['Cherry', 20]],
            columns = ['Product', 'Price']
        )

In [2]: df1.head()

Out[2]:         Date Product  Quantity
        0 2020-01-21  Cherry        13
        1 2019-10-06   Apple        92
        2 2019-10-27  Banana        24
        3 2020-02-23  Banana        44
        4 2019-11-04  Banana        36

In [3]: df2

Out[3]:   Product  Price
        0   Apple     10
        1  Banana     15
        2  Cherry     20
```

1.1 Thay đổi giá trị của index

Nhắc lại, để thay đổi giá trị của `index`, bạn thực hiện như sau:

```
<tên_DataFrame>.index = <index_mới>
```

Trong đó, `index_mới` là một `list` có độ dài bằng với độ dài `index` ban đầu.

```
In [4]: df2.index = [1, 2, 3]
        df2

Out[4]:   Product  Price
        1   Apple     10
        2  Banana     15
        3  Cherry     20
```

Lưu ý: khi sử dụng cách trên thì sẽ trực tiếp thay đổi `DataFrame` gốc ban đầu.

1.2 Đặt tên cho `index`

Để đặt tên cho `index`, bạn thực hiện như sau:

```
<tên_DataFrame>.index.name = <tên>
```

```
In [5]: df2.index.name = 'No'
        df2

Out[5]:    Product  Price
        No
        1    Apple     10
        2   Banana     15
        3   Cherry     20
```

Câu hỏi : Làm thế nào để xoá tên của `index`?

1.3 Đặt một cột làm index

Đôi khi, thay vì sử dụng cột tự sinh ra khi tạo `DataFrame`, người ta muốn sử dụng một (hoặc nhiều cột) làm `index`. Để thực hiện việc này, người ta dùng phương thức `.set_index()` như sau:

```
<tên_DataFrame>.set_index(<tên_cột_làm_index>)
```

```
In [6]: df3 = df1.set_index('Date')
        df3.head()

Out[6]:             Product  Quantity
        Date
        2020-01-21  Cherry        13
        2019-10-06   Apple        92
        2019-10-27  Banana        24
        2020-02-23  Banana        44
        2019-11-04  Banana        36
```

2 Đặt lại index

Đây là quá trình đánh số lại index, được thực hiện thông qua phương thức .reset_index() như sau:

```
<tên_DataFrame>.reset_index()
```

Nếu không muốn giữ lại index cũ, bạn thêm vào tham số drop = True.

```
In [7]: # đánh số lại df3, giữ lại index cũ (cột Date)
        df3.reset_index().head()
```

```
Out[7]:         Date Product  Quantity
        0 2020-01-21  Cherry        13
        1 2019-10-06   Apple        92
        2 2019-10-27  Banana        24
        3 2020-02-23  Banana        44
        4 2019-11-04  Banana        36
```

2.1 Sắp xếp lại index

Để sắp xếp lại thứ tự của index, ta dùng phương thức .sort_index() như sau :

```
<tên_DataFrame>.sort_index()
```

Khi đó, index sẽ được sắp xếp lại theo thứ tự tăng dần. Nếu muốn sắp xếp theo thứ tự giảm dần, bạn dùng tham số ascending = False.

```
In [8]: # sắp xếp theo index, thứ tự giảm dần
        df3.sort_index(ascending = False).head()
```

```
Out[8]:            Product  Quantity
        Date
        2020-02-29   Apple        14
        2020-02-29   Apple        78
        2020-02-26  Cherry         9
        2020-02-23  Banana        44
        2020-02-20   Apple        20
```

3 Biến đổi DataFrame bằng .reindex()

Phương thức .reindex() có cú pháp như sau :

```
<tên_DataFrame>.reindex(index = <danh_sách_tên_dòng>, columns = <danh_sách_tên_cột>)
```

Phương thức .reindex() sẽ trả về một DataFrame mới có các đặc điểm sau:

- Tên dòng theo thứ tự được chỉ ra trong danh_sách_tên_dòng.
- Tên cột theo thứ tự được chỉ ra trong danh_sách_tên_cột.
- Giá trị tại một ô là giá trị tương ứng của nó trong DataFrame gốc, sẽ dùng NaN, ... nếu không có giá trị tương ứng.

```
In [9]: df2.reindex(index = [2, 1, 3], columns = ['Price', 'Product'])

Out[9]:     Price Product
        No
        2      15  Banana
        1      10   Apple
        3      20  Cherry

In [10]: df2.reindex(columns = ['Date', 'Price', 'Product'])

Out[10]:     Date  Price Product
         No
         1    NaN     10   Apple
         2    NaN     15  Banana
         3    NaN     20  Cherry
```

4 Kết hợp `DataFrame` : `concat`

Như đã nói sơ lược trong phần trước, bạn có thể dùng `pd.concat()` để kết hợp hai hay nhiều `DataFrame` theo cú pháp sau :

```
pd.concat(<danh_sách_DataFrame>)
```

Trong phần này, chúng ta sẽ tìm hiểu thêm về một số tham số của `pd.concat()`.

4.1 Tham số `ignore_index`

Có hai giá trị là `True` và `False`, trong đó:

- `True` sẽ bỏ qua tất cả `index` của các `DataFrame` và sinh ra một `index` mới. Thường dùng khi `index` không mang nhiều ý nghĩa.
- `False` sẽ kết hợp `index` của các `DataFrame` để tạo thành `index` mới.

Giá trị mặc định của `ignore_index` là `False`.

```
In [11]: a = pd.DataFrame([1, 2, 3], columns = ['A'])
         b = pd.DataFrame([4, 5, 6], columns = ['A'])

In [12]: pd.concat([a, b], ignore_index = True)

Out[12]:    A
         0  1
         1  2
         2  3
         3  4
         4  5
         5  6
```

4.2 Tham số `axis`

Có 2 giá trị là `'index'` và `'columns'`, trong đó:

- `'index'` sẽ nối `DataFrame` theo chiều dọc.
- `'columns'` sẽ nối `DataFrame` theo chiều ngang.

Giá trị mặc định của `axis` là `'index'`.

```
In [13]: pd.concat([a, b], axis = 'columns')
```

```
Out[13]:    A  A
         0  1  4
         1  2  5
         2  3  6
```

4.3 Tham số `join`

Có 2 giá trị là `'outer'` và `'inner'`, trong đó:

- `'outer'`: lấy tất cả cột (khi `axis = 'index'`) hoặc dòng (khi `axis = 'columns'`).
- `'inner'`: chỉ lấy phần cột chung hoặc dòng chung.

Giá trị mặc định của `join` là `'outer'`.

```
In [14]: c = pd.DataFrame([['1', '2'], ['3', '4']], columns = ['A', 'B'])
         d = pd.DataFrame([['5', '6'], ['7', '8']], columns = ['B', 'C'], index = [1,
```

```
In [15]: pd.concat([c, d], join = 'inner')
```

```
Out[15]:    B
         0  2
         1  4
         1  5
         2  7
```

```
In [16]: pd.concat([c, d], axis = 'columns', join = 'inner')
```

```
Out[16]:    A  B  B  C
         1  3  4  5  6
```

5 Kết hợp `DataFrame`: `merge`

Cùng xem xét vấn đề sau: bạn muốn tính giá trị của các hàng hóa đã bán ra, nhưng thông tin về hóa đơn nằm ở `df1` còn thông tin về hàng hóa lại nằm ở `df2`. Bạn phải giải quyết vấn đề này thế nào?

Một hướng giải quyết là bạn thêm 1 cột mới vào `df1` với điều kiện nếu `Product` là `'Apple'` thì giá trị tương ứng này là `10`, nếu `Product` là `'Banana'` thì giá trị tương ứng là `15`.

Khác với `concat` là dạng kết nối 2 hay nhiều `DataFrame` theo hình thức **dán** các `DataFrame` lại với nhau thì `merge` là hình thức kết nối 2 `DataFrame` lại dựa trên đặc điểm chung nào đó giữa 2 `DataFrame` đó.

```
In [17]: pd.merge(df1, df2)
```

```
Out[17]:          Date Product  Quantity  Price
         0  2020-01-21  Cherry        13     20
         1  2020-01-21  Cherry        17     20
         2  2019-10-13  Cherry        63     20
         3  2019-10-13  Cherry        29     20
         4  2019-09-14  Cherry        37     20
         ..        …     …       …    …
         95 2019-11-14  Banana        81     15
         96 2019-12-27  Banana        80     15
         97 2019-11-09  Banana        60     15
         98 2019-10-28  Banana         6     15
         99 2019-10-05  Banana        22     15

         [100 rows x 4 columns]
```

Cú pháp của `pd.merge()` như sau :

```
pd.merge(<DataFrame_trái>, <DataFrame_phải>)
```

5.1 Một số tham số của `pd.merge`

Tham số on dùng để chỉ ra cột điều kiện để `merge`. Mặc định sẽ dùng tất cả các cột giống tên.

Tham số how dùng để chỉ ra cách thức `merge`, bao gồm 4 giá trị:

- `'inner'`: chỉ lấy các giá trị có trong cả hai bên trái, phải của `merge`.
- `'outer'`: lấy tất cả giá trị trong cả hai bên của `merge`.
- `'left'`: lấy tất cả giá trị bên trái.
- `'right'` : lấy tất cả giá trị bên phải.