

GIỚI THIỆU VỀ PANDAS

Pandas là thư viện phân tích dữ liệu được sử dụng rộng rãi. Nhờ có Python, việc phân tích dữ liệu trở nên dễ dàng.

Cấu trúc: Trong phần này, sẽ thảo luận về các chủ đề sau:

- Pandas là gì?
- Tại sao lại sử dụng Pandas?
- Cấu trúc dữ liệu của Pandas.
- Tạo khung dữ liệu Pandas bằng cách sử dụng các công cụ trong Python.
- Nhập dữ liệu từ file bên ngoài vào khung dữ liệu Pandas:
 - Nhập dữ liệu từ file CSV.
 - Nhập dữ liệu từ file Excel.
 - Nhập dữ liệu từ file JSON.
- Khám phá dữ liệu của DataFrame.
- Chọn và lọc dữ liệu từ DataFrame.
- Xóa dữ liệu trong Pandas DataFrame.
 - Xử lý dữ liệu nhân bản.
 - Xử lý các giá trị bị thiếu mất trong dữ liệu.
- Nhóm và tổng hợp.
- Sắp xếp và thứ hạng.
- Thêm/nối dòng/cột trong DataFrame.
- Xóa dòng/cột từ DataFrame.
- Nối liên các khung dữ liệu.
- Kết hợp/tham gia các khung dữ liệu.
- Ghi nội dung khung dữ liệu vào các file bên ngoài.
 - Ghi ra file CSV.
 - Ghi ra file Excel.
 - Ghi ra file JSON.

Mục tiêu: Sau khi hoàn thành nghiên cứu nội dung này, có thể:

- Hiểu cách sử dụng Pandas.
- Làm việc với cấu trúc dữ liệu Pandas.
- Tạo khung dữ liệu từ các công cụ trong Python.
- Tải dữ liệu từ các file khác nhau (CSV, Excel, và JSON).
- Khám phá, chọn, và lọc dữ liệu bằng cách sử dụng các chức năng thiết yếu của Pandas.
- Hiểu cách xử lý dữ liệu nhân bản và các giá trị bị thiếu mất trong khung dữ liệu.
- Nhóm, tổng hợp, sắp xếp và thứ hạng dữ liệu.
- Nối và xóa dòng hoặc cột trong khung dữ liệu.
- Nối liên, kết hợp, và tham gia các khung dữ liệu.
- Ghi nội dung dữ liệu vào các file bên ngoài (CSV, Excel, và JSON).

1. Định nghĩa thư viện Pandas

Pandas là một thư viện phân tích dữ liệu mã nguồn mở dành cho Python.

Theo như tài liệu chính thức về Pandas: Pandas là một công cụ với mã nguồn mở, được cấp phép giấy chứng nhận BSD công nhận là thư viện được sử dụng với hiệu suất cao, cấu trúc dữ liệu dễ sử dụng và phân tích dữ liệu dành cho ngôn ngữ lập trình Python.

2. Tại sao chúng ta cần sử dụng thư viện Pandas?

Khi đã có 1 file nhập vào với các thông tin về nhân viên như emp_name, emp_salary, emp_department, cần tìm tổng thu nhập của nhân viên cho từng chung cư (hoặc tổng thu nhập của chung cư).

Các giải pháp để giải quyết vấn đề này theo 2 phương thức tiếp cận:

1. Cách thức 1: Sử dụng cách thông thường của ngôn ngữ lập trình Python để giải quyết mà không sử dụng thư viện Python.

2. Cách thức 2: Sử dụng thư viện Pandas để giải quyết vấn đề.

Cách thức 1 có số dòng code hơn xấp xỉ 4 lần so với cách thức 2, và cách hai sẽ cần ít thời gian hơn để làm việc, tính toán hơn.

```
df = pd.read_csv("data/emp.csv")
sal_dept_by_df = df.groupby("Department").sum("Salary_in_dollor").reset_index()
sal_dept_by_df.columns = ['Department', 'sum_of_sal']
print("Kết quả : \n")
sal_dept_by_df
```

Giải pháp dùng câu lệnh của Pandas

Output:

	Department	sum_of_sal
0	Accounting	22500
1	Advertising	45609
2	Asset Management	26885
3	Customer Relations	34939
4	Customer Service	15687
5	Finances	26259
6	Human Resources	7919
7	Legal Department	24534
8	Media Relations	22325
9	Payroll	22079
10	Public Relations	28010
11	Quality Assurance	12934
12	Research and Development	30320
13	Sales and Marketing	15919
14	Tech Support	23232

Rõ ràng, nếu so sánh với cách code thông thường của Python thì thư viện *Pandas* **hữu dụng, dễ sử dụng, và nhanh chóng hơn** rất nhiều để hoàn thành công việc phân tích dữ liệu.

3. Cấu trúc dữ liệu của Pandas

3.1. Series

Pandas series là một mảng một chiều có nhãn chỉ mục, có ý nghĩa nhiều hơn về mặt kỹ thuật, các nhãn này còn được gọi là chỉ mục trục. Vì vậy, nói cách khác Pandas series là tập hợp các đối tượng trong một chiều có chỉ số trục.

** Tạo ra chuỗi Pandas*

Phương thức **pandas.Series (data, index = index)**: tạo ra chuỗi pandas.

Ví dụ: tạo ra một chuỗi pandas từ danh sách Python

```
import pandas as pd
import numpy as np
num_list = [1,2,3,4,5]
num_series = pd.Series(num_list)
print("data type of num_series :{}".format(type(num_series)))
num_series
```

```
data type of num_series :<class 'pandas.core.series.Series'>
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

3.2. DataFrame

Pandas DataFrame là cấu trúc dữ liệu với kiểu không gian hai chiều, có kiểu chỉ mục trục (Kiểu hàng ngang). Nó có dạng một cái bảng trong SQL (có các hàng và cột). DataFrame là cấu trúc dữ liệu đơn giản nhất của pandas.

- **Tạo pandas DataFrame**

Sử dụng phương tiện trong thư viện pandas để tạo ra pandas DataFrame:

`pandas.DataFrame (data, index=index, column = [column(s)]),`

trong đó, data là danh sách và từ điển trong Python

Ví dụ 1: Tạo pandas DataFrame từ danh sách Python

```
import pandas as pd
import numpy as np
num_list = [1,2,3,4,5]
num_df = pd.DataFrame (num_list,columns = 'number')
print("data type of num_series: {}".format(type(num_df)))
numdf
```

```
data type of num_series :<class 'pandas.core.frame.DataFrame'>
```

number	
0	1
1	2
2	3
3	4
4	5

Ví dụ 2: Tạo pandas DataFrame từ từ điển Python.

```
import pandas as pd
import numpy as np
num_list_df = pd.DataFrame (numlist, columns = [column(s)])
print("data type of num_series: {}".format(type(num_list_df)))
print("printing the data frame num_list_df:")
num_list_df
```

```
data type of num_series :<class 'pandas.core.frame.DataFrame'>
```

number	
0	1
1	2
2	3
3	4
4	5

4. Đọc dữ liệu từ tập tin bên ngoài vào trong DataFrame

Thư viện pandas có nhiều chức năng để đọc dữ liệu từ các tập tin khác nhau vào DataFrame. Các dạng tập tin có thể đọc là CSV, EXCEL và JSON:

- Đọc dữ liệu từ CSV vào DataFrame:

CSV (Comma Separated Values) là một định dạng thông thường biểu thị và lưu trữ dữ liệu dạng văn bản, trong đó các giá trị được **phân tách bằng dấu phẩy**.

Ví dụ bên dưới là ảnh chụp của tập tin CSV với một tiêu đề:

```
1 Name,Department,Salary_in_dollor,city
2 Porter,Advertising,5383,Connah's Quay
3 Abraham,Media Relations,8181,Beauwelz
4 Victoria,Accounting,7921,Neerrepn
5 Hiroko,Customer Relations,7443,Sanzeno
6 Kathleen,Legal Department,9476,Minneapolis
7 Amela,Public Relations,9900,Carterton
8 Timon,Advertising,5222,Portland
9 Barclay,Accounting,8224,Sanzeno
10 Knox,Finances,6089,Alken
11 Malachi,Quality Assurance,5858,Wazirabad
12 Macy,Public Relations,9470,Richmond Hill
13 Demetria,Advertising,8282,Wanneroo
14 Lawrence,Payroll,7538,Sanzeno
15 Lavinia,Tech Support,9775,Temuka
```

Phương thức **read.csv()**: Tạo DataFrame bằng cách đọc dữ liệu từ tập CSV. Có nhiều tình huống làm việc với tập tin CSV như tập tin CSV chứa thông tin tiêu đề, tập tin CSV không chứa thông tin tiêu đề, tập tin được sử dụng dấu phân cách khác như ("|").

Ví dụ: Đọc dữ liệu từ tập tin CSV với một tiêu đề vào một DataFrame

```
import pandas as pd
import numpy as np
csv_df = pd.read_csv("data/emp.csv")
csv_df.head()
```

	Name	Department	Salary_in_dollor	city
0	Porter	Advertising	5383	Connah's Quay
1	Abraham	Media Relations	8181	Beauwelz
2	Victoria	Accounting	7921	Neerrepn
3	Hiroko	Customer Relations	7443	Sanzeno
4	Kathleen	Legal Department	9476	Minneapolis

Ví dụ: Đọc dữ liệu từ tập tin CSV với không có tiêu đề vào một DataFrame

```
import pandas as pd
import numpy as np
my_header = ['name', 'department', 'salary_in_dollor', 'city']
no_header_csv_df = pd.read_csv("data/emp_no_header.csv"), names = my_header)
no_header_csv_df.head()
```

	Name	Department	Salary_in_dollor	city
0	Porter	Advertising	5383	Connah's Quay
1	Abraham	Media Relations	8181	Beauwelz
2	Victoria	Accounting	7921	Neerrepn
3	Hiroko	Customer Relations	7443	Sanzeno
4	Kathleen	Legal Department	9476	Minneapolis

Ví dụ: Đọc dữ liệu từ tập tin chứa dấu phân cách ("|") vào một DataFrame

```
import pandas as pd
import numpy as np
text_file_df = pd.read_csv("data/emp.txt", sep="|")
text_file_df.head()
```

	Name	Department	Salary_in_dollor	city
0	Porter	Advertising	5383	Connah's Quay
1	Abraham	Media Relations	8181	Beauwelz
2	Victoria	Accounting	7921	Neerrepn
3	Hiroko	Customer Relations	7443	Sanzeno
4	Kathleen	Legal Department	9476	Minneapolis

Ví dụ: Đọc dữ liệu từ tập excel vào DataFrame

```
excel_df = pd.read_excel("data/emp.xlsx", sheet_name = 'emp')
excel_df.head()
```

	Name	Department	Salary_in_dollor	city
0	Porter	Advertising	5383	Connah's Quay
1	Abraham	Media Relations	8181	Beauwelz
2	Victoria	Accounting	7921	Neerrepen
3	Hiroko	Customer Relations	7443	Sanzeno
4	Kathleen	Legal Department	9476	Minneapolis

Ví dụ: Đọc dữ liệu từ tập tin JSON vào DataFrame

```
json_df = pd.read_json('data/emp.json')
json_df.head()
```

	Name	Department	Salary_in_dollor	city
0	Porter	Advertising	5383	Connah's Quay
1	Abraham	Media Relations	8181	Beauwelz
2	Victoria	Accounting	7921	Neerrepen
3	Hiroko	Customer Relations	7443	Sanzeno
4	Kathleen	Legal Department	9476	Minneapolis

5. Khám phá dữ liệu của một DataFrame

- **DataFrame.shape:** Trả về giá trị là một tuple bao gồm số hàng và cột của DataFrame.

```
emp_df = pd.read_csv("data/emp.csv")
shape_of_df = emp_df.shape
print("shape_of_df: ", shape_of_df)
```

shape_of_df: (50, 4)

- **DataFrame.head(n):** Trả về giá trị là một DataFrame bao gồm n hàng đầu tiên từ tập tin DataFrame được nhập vào.

```
print("Displaying the first 6 Rows from DataFrame")
emp_df.head(6)
```

Displaying the first 6 Rows from DataFrame

	Name	Department	Salary_in_dollor	city
0	Porter	Advertising	5383	Connah's Quay
1	Abraham	Media Relations	8181	Beauwelz
2	Victoria	Accounting	7921	Neerrepen
3	Hiroko	Customer Relations	7443	Sanzeno
4	Kathleen	Legal Department	9476	Minneapolis
5	Amela	Public Relations	9900	Carterton

- **DataFrame.tail(n):** Trả về giá trị một DataFrame bao gồm n hàng cuối của tệp DataFrame được nhập vào.

```
print("Displaying the last 6 Rows from DataFrame")
emp_df.tail(6)
```

Displaying the last 6 Rows from DataFrame

	Name	Department	Salary_in_dollor	city
44	Cecilia	Customer Relations	7480	Cumberland
45	Tobias	Media Relations	8619	Westlock
46	Palmer	Quality Assurance	7076	Ospedaletto d'Alpinolo
47	Hayes	Asset Management	6381	Malahide
48	Xyla	Asset Management	6531	Poole
49	Orlando	Customer Service	8524	Caramanico Terme

- **DataFrame.info()**: In ra thông tin DataFrame ví dụ như chỉ mục, cột, không có giá trị null,...

```
emp_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 4 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Name                   50 non-null    object
1   Department              50 non-null    object
2   Salary_in_dollor       50 non-null    int64
3   city                   50 non-null    object
dtypes: int64(1), object(3)
memory usage: 1.7+ KB
```

- **DataFrame.describe()**: Trả về các thống kê về cột số.

Ví dụ: describe() với cột có số liệu

```
emp_df.describe()
```

	Salary_in_dollor
count	50.000000
mean	7183.020000
std	1360.504866
min	5015.000000
25%	6099.250000
50%	7161.000000
75%	8213.250000
max	9900.000000

Ví dụ: describe() khi các cột không có số liệu:

```
emp_df[['city']].describe()
```

	city
count	50
unique	31
top	Sanzeno
freq	9

- **DataFrame.dtypes**: Trả về giá trị là kiểu dữ liệu trong DataFrame.

```
emp_df.dtypes
```

```
Name                object
Department           object
Salary_in_dollor     int64
city                 object
dtype: object
```


- **DataFrame.columns:** Trả về giá trị là các nhãn cột của DataFrame.

```
print("Columns in the DataFrame: \n", emp_df.columns)
Columns in the DataFrame :
Index(['Name', 'Department', 'Salary_in_dollor', 'city'], dtype='object')
```

6. Chọn dữ liệu từ DataFrame

Có rất nhiều cách để chọn và lọc dữ liệu từ pandas DataFrame.

- **Chọn dữ liệu cho các tập hợp con của cột**

```
print("Columns present in DataFrame:", emp_df.columns)
print("Selecting the data for Name and city column from DataFrame")
new_df = emp_df[['Name', 'city']]
new_df.head()
```

```
Columns present in DataFrame: Index(['Name', 'Department', 'Salary_in_dollor', 'city'], dtype='object')
Selecting the data for Name and city column from DataFrame
```

	Name	city
0	Porter	Connah's Quay
1	Abraham	Beauwaelz
2	Victoria	Neerpen
3	Hiroko	Sanzeno
4	Kathleen	Minneapolis

- **Chọn dữ liệu bằng cách sử dụng điều kiện đơn giản**

```
new_df = emp_df[emp_df.city == 'Sanzeno']
new_df
```

	Name	Department	Salary_in_dollor	city
3	Hiroko	Customer Relations	7443	Sanzeno
7	Barclay	Accounting	8224	Sanzeno
12	Lawrence	Payroll	7538	Sanzeno
23	Tashya	Legal Department	8826	Sanzeno
26	Hakeem	Legal Department	6232	Sanzeno
27	Ezekiel	Advertising	6848	Sanzeno
29	Charity	Research and Development	5135	Sanzeno
34	Doris	Asset Management	8242	Sanzeno
39	Kieran	Tech Support	8061	Sanzeno

- **Chọn dữ liệu bằng cách sử dụng nhiều điều kiện**

Trường hợp 1: ĐK 1 và ĐK 2:

```
out_df = emp_df[(emp_df.Salary_in_dollor <= 10000) & (emp_df.city == 'Sanzeno')]
out_df
```

	Name	Department	Salary_in_dollor	city
3	Hiroko	Customer Relations	7443	Sanzeno
7	Barclay	Accounting	8224	Sanzeno
12	Lawrence	Payroll	7538	Sanzeno
23	Tashya	Legal Department	8826	Sanzeno
26	Hakeem	Legal Department	6232	Sanzeno
27	Ezekiel	Advertising	6848	Sanzeno
29	Charity	Research and Development	5135	Sanzeno
34	Doris	Asset Management	8242	Sanzeno
39	Kieran	Tech Support	8061	Sanzeno

Trường hợp 2: ĐK 1 hoặc ĐK 2:

```
out_df = emp_df[(emp_df.Salary_in_dollor < 5000) | (emp_df.city == 'Sanzeno')]  
out_df
```

	Name	Department	Salary_in_dollor	city
3	Hiroko	Customer Relations	7443	Sanzeno
7	Barclay	Accounting	8224	Sanzeno
12	Lawrence	Payroll	7538	Sanzeno
23	Tashya	Legal Department	8826	Sanzeno
26	Hakeem	Legal Department	6232	Sanzeno
27	Ezekiel	Advertising	6848	Sanzeno
29	Charity	Research and Development	5135	Sanzeno
34	Doris	Asset Management	8242	Sanzeno
39	Kieran	Tech Support	8061	Sanzeno

7. Làm sạch dữ liệu trong pandas DataFrame

Điều cần thiết để có kết quả khách quan, chính xác và nhất quán đó là dữ liệu không bị các tạp chất như *giá trị/bản ghi trùng lặp, giá trị bị thiếu, điểm dữ liệu không liên quan*. Vì vậy, để giải quyết vấn đề về dữ liệu, có thể thao tác và sửa các tạp chất hoặc loại bỏ các bản ghi dữ liệu xấu đó.

Trong pandas, có nhiều chức năng/phương thức cho nhiệm vụ làm sạch dữ liệu như:

- Xử lý dữ liệu bị trùng lặp.
- Bản ghi bị trùng lặp.
- Một tập hợp con các cột bị trùng lặp giá trị.

Dưới đây là cái ví dụ để nhận diện và xóa các bản ghi bị trùng lặp từ một DataFrame:

* Lấy ra số đang bị trùng lặp trong dữ liệu:

```
no_of_dups = emp_df_raw.duplicated().sum()  
print("number of rows having complete row duplicate :", no_of_dups)  
  
no_of_dups = emp_df_raw.duplicated(['employee_id']).sum()  
print("number of rows having duplicate employee ids:", no_of_dups)  
number of rows having complete row duplicate : 4  
number of rows having duplicate employee ids: 8
```

* Biểu diễn bản ghi bị trùng lặp:

```
emp_df_raw[emp_df_raw.duplicated('employee_id')]
```

employee_id	employee_name	employee_salary_in_dollor	employee_city	employee_department	employee_joining_date	
11	111	Cruz	6740	NaN	Legal Department	24/06/2014
29	128	Colorado	9075	Gosselies	Media Relations	2/11/2019
35	132	Gisela	7879	Castellina in Chianti	Accounting	4/2/2016
45	NaN	Amir	9819	Püttlingen	Legal Department	18/10/2020
52	149	Leroy	8001	Meeswijk	Advertising	29/03/2021
68	NaN	Reese	5121	Orizaba	Public Relations	14/01/2021
73	168	Neil	6644	Drachten	Public Relations	22/06/2019
74	169	Rahim	9749	Nelva	Customer Service	12/9/2015

* Xóa bỏ bản ghi bị trùng lặp:

```
emp_df_raw_no_dups = emp_df_raw.drop_duplicates('employee_id')  
dups_reord_count = emp_df_raw_no_dups.duplicated(['employee_id']).sum()  
print("Duplicate records count =", dups_reord_count)  
emp_df_raw_no_dups[emp_df_raw_no_dups.employee_id == '111']
```

Duplicate recods count = 0

	employee_id	employee_name	employee_salary_in_dollor	employee_city	employee_department	employee_joining_date
10	111	Cruz	6740	Lavacherie	Legal Department	24/06/2014

Xử lý giá trị bị thiếu trong dữ liệu

Xóa dòng có giá trị bị thiếu

- Nếu tất cả các cột đều có giá trị bị thiếu thì sử dụng **dropna(how= 'all')**.
- Nếu một cột nào đó có giá trị bị thiếu thì sử dụng **dropna(how= 'any')**.
- Nếu một tập hợp con của các cột hoặc một cột cụ thể nào đó có giá trị bị thiếu thì sử dụng **dropna(subset= [col1, col2,...])**.

Điền vào các giá trị bị thiếu

```
df_fillna = df.fillna("MissingValue")
custom_display(df,titles=["| Input : Raw DataFrame|"])
custom_display(df_fillna,titles=["| Output : Applyng fillna() function|"])
```

Input : Raw DataFrame				
	Name	Department	Salary_in_dollor	city
0	Jim	Advertising	7921.0	Connah's Quay
1	Abraham	NaN	7787.0	Sanzeno
2	Porter	Accounting	7921.0	NaN
3	Victoria	HR	NaN	Neerrepn
4	Hiroko	Accounting	7443.0	Sanzeno
5	NaN	NaN	NaN	NaN
6	NA		0.0	

Output : Applyng fillna() function				
	Name	Department	Salary_in_dollor	city
0	Jim	Advertising	7921	Connah's Quay
1	Abraham	MissingValue	7787	Sanzeno
2	Porter	Accounting	7921	MissingValue
3	Victoria	HR	MissingValue	Neerrepn
4	Hiroko	Accounting	7443	Sanzeno
5	MissingValue	MissingValue	MissingValue	MissingValue
6	NA		0	

Sử dụng các câu lệnh sau để điền giá trị vào chỗ bị thiếu:

- **DataFrame.fillna (method = 'ffill')**: lấy giá trị đằng trước để điền vào chỗ bị thiếu.
- **DataFrame.fillna (method = 'bfill')**: lấy giá trị phía sau để điền vào chỗ bị thiếu.
- **mean()**: lấy giá trị trung bình của các giá trị còn lại.
- **median()**
- **mode()**
- **replace()**

8. Nhóm và tổng hợp

Để nhóm, pandas có hàm **groupby()**

Để tổng hợp, có các hàm khác nhau như min(), max(), count(), sum(),...

8.1. Nhóm

Ví dụ: Cách sử dụng hàm groupby()

Bước 1: Ở bước này, cần nhập tệp emp_agg.csv vào khung dữ liệu emp_df

Bước 2: Hàm groupby() sẽ nhận được liên kết với 'Department' và in ra kết quả được nhóm. Kết quả cho cột Department có giá trị riêng biệt sẽ biểu diễn nhóm các từ khóa.

Bước 3: In tất cả các nhóm từ khóa từ kết quả đã được nhóm.

```
emp_df = pd.read_csv("data/emp_agg.csv")
emp_dept_grouped_df = emp_df.groupby('Department')
print("Group Object:", emp_dept_grouped_df)
groups = emp_dept_grouped_df.groups.keys()
print("\nGroup_Keys : ", groups)
```

Grouped Object: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000263989FA4F0>

Group_Keys : dict_keys(['Accounting', 'Advertising', 'Customer Relations', 'Legal Department', 'Media Relations', 'Public Relations'])

Vậy nên, nếu bạn muốn nhóm 1 hoặc nhiều cột, bạn có thể sử dụng hàm groupby() như **DataFrame.groupby([col1,col2,...])**.

8.2. Tổng hợp

Các cú pháp:

DataFrame.groupby([col1,col2,...]).<aggregate_function> ().

Có thể áp dụng hàm tổng hợp cho dữ liệu được nhóm để có được kết quả tổng hợp.

Ví dụ: giả sử cần hiển thị mức lương cho từng bộ phận. Trong trường hợp đó, trước tiên cần thực hiện nhóm dữ liệu trên bộ phận cột, sau đó áp dụng hàm tổng sum() trên cột Salary.

```
emp_df.groupby(['Department'])[['Salary_in_dollor']].sum()
```

Salary_in_dollor	
Department	
Accounting	97504
Advertising	85610
Customer Relations	43819
Legal Department	41351
Media Relations	43013
Public Relations	47854

Hoặc Sử dụng hàm agg({agg_function1(), agg_function2(),...}):

```
out_df = emp_df.groupby(['Department'])[['Salary_in_dollor']].agg(['sum','mean','max','min','count'])
out_df
```

	min	count	sum	mean	max
Department					
Accounting	5396	13	97504	7500.307692	9470
Advertising	5222	13	85610	6585.384615	9064
Customer Relations	5196	6	43819	7303.166667	8640
Legal Department	5015	6	41351	6891.833333	9476
Media Relations	5740	6	43013	7168.833333	8224
Public Relations	5135	6	47854	7975.666667	9900

9. Sắp xếp và thứ hạng

9.1. Sắp xếp

Trong nhiều trường hợp cần phải sắp xếp dữ liệu theo một số thứ tự hoặc xếp hạng trong dữ liệu. Chẳng hạn, nhận tiền lương theo thứ tự tăng dần hoặc xếp hạng cho học sinh dựa trên số điểm đạt được. Vì vậy, để thực hiện các hoạt động như vậy, pandas cung cấp các phương thức sắp xếp và xếp hạng như:

- `Sort_index()`: Sắp xếp chỉ mục.
- `Sort_values()`: Sắp xếp theo giá trị.

9.2. Thứ hạng

Đối với việc xếp hạng theo thứ tự, sử dụng hàm `rank()`. Nó sẽ gán giá trị từ 1 cho tới cuối theo giá trị của cột được chỉ định từ bé tới lớn. Hoặc có thể dùng `rank(ascending=False)` để làm ngược lại.

```
df['sal_rank_asc'] = df['sal'].rank()
df
```

	emp_id	city	sal	sal_rank_asc
1	101	Sanzeno	5000	1.0
4	105	Beauwelz	8989	3.0
2	108	Minneapolis	5634	2.0
3	100	Carterton	9634	4.0

10. Thêm dòng vào DataFrame

Có thể sử dụng hàm `append()` để thêm dòng mới vào DataFrame đang được làm việc.

Để viết code trên nhiều dòng như dictionary, pandas khi sử dụng: chuỗi và `Pandas.DataFrame`. DataFrame dưới đây là khung dữ liệu đang được làm việc, và cần thêm một dòng mới trong khung dữ liệu này.

Đầu tiên là thêm 1 dòng dữ liệu bằng cách sử dụng dictionary:

```
row = {'emp_id':999, 'city':'Minneapolis', 'sal':6789}
print(type(row))
df=df.append(row,ignore_index=True)
df
```

	emp_id	city	sal
0	101	Sanzeno	5000
1	105	Beauwelz	8989
2	108	Minneapolis	5634
3	100	Carterton	9634
4	888	Minneapolis	6789
5	777	Minneapolis	6789
6	999	Minneapolis	6789

Tiếp theo, thêm 1 dòng dữ liệu bằng cách sử dụng series trong thư viện pandas:

```
row = pd.Series(data = {'emp_id':888, 'city':'Minneapolis', 'sal':6789}, name=6)
print(type(row))
df = df.append(row,ignore_index=False)
df
```

11. Thêm cột vào DataFrame

Có thể thêm một cột mới trong một DataFrame bằng những cách sau đây:

- Thêm một cột mới với một giá trị không đổi

```
df['constant_value_col'] = 'New_value'
df
```

	emp_id	city	sal	constant_value_col
0	101	Sanzeno	5000	New_value
1	105	Beauwelz	8989	New_value
2	108	Minneapolis	5634	New_value
3	100	Carterton	9634	New_value
4	999	Minneapolis	6789	New_value
6	888	Minneapolis	6789	New_value
0	777	Minneapolis	6789	New_value

- Thêm một cột mới với một danh sách các giá trị

```
df['new_col'] = [1,2,3,4]
df
```

	emp_id	city	sal	constant_value_col	new_col
0	101	Sanzeno	5000	New_value	1
1	105	Beauwelz	8989	New_value	2
2	108	Minneapolis	5634	New_value	3
3	100	Carterton	9634	New_value	4

- Thêm một cột mới bằng cách áp dụng chuyển đổi logic

```
emp_dict = {'emp_id':[101,105,108,100],
            'city':['Sanzeno', 'Beauwelz', 'Minneapolis', 'Carterton'],
            'sal':[5000, 8989, 5634, 9634]}
df = pd.DataFrame(emp_dict)
custom_display(df,titles=["Input DataFrame"])

def sal_with_10_percent_hike(old_sal):
```

```
""" Function will new salary by adding 10% with current salary"""
return int(old_sal)+int(old_sal)*.1
```

```
df['updated_salary'] = df['sal'].apply(sal_with_10_percent_hike)
df
custom_display(df,titles=["Output DataFrame with New Added column"])
```

Input Data Frame			
	emp_id	city	sal
0	101	Sanzeno	5000
1	105	Beauwelz	8989
2	108	Minneapolis	5634
3	100	Carterton	9634

Output DataFrame with New Added column				
	emp_id	city	sal	updated_salary
0	101	Sanzeno	5000	5500.0
1	105	Beauwelz	8989	9887.9
2	108	Minneapolis	5634	6197.4
3	100	Carterton	9634	10597.4

• Thêm cột mới bằng cách sử dụng hàm lambda

```
emp_dict = {'emp_id':[101,105,108,100],
            'city':['Sanzeno', 'Beauwelz', 'Minneapolis', 'Carterton'],
            'sal':[5000, 8989, 5634, 9634]}
df = pd.DataFrame(emp_dict)
custom_display(df,titles=["Input DataFrame"])
df['updated_salary'] = df['sal'].apply(lambda sal:int(sal)+int(sal)*.1)
custom_display(df,titles=["Output DataFrame with New Added column"])
```

Input Data Frame			
	emp_id	city	sal
0	101	Sanzeno	5000
1	105	Beauwelz	8989
2	108	Minneapolis	5634
3	100	Carterton	9634

Output DataFrame with New Added column				
	emp_id	city	sal	updated_salary
0	101	Sanzeno	5000	5500.0
1	105	Beauwelz	8989	9887.9
2	108	Minneapolis	5634	6197.4
3	100	Carterton	9634	10597.4

12. Xóa bỏ hàng hoặc cột khỏi DataFrame

```
emp_dict = {'emp_id':[101,105,108,100],
            'city':['Sanzeno', 'Beauwelz', 'Minneapolis', 'Carterton'],
            'sal':[5000, 8989, 5634, 9634]}
```

```

custom_display(df,titles=["Input DataFrame"])

new_df = df.drop([2], axis=0)
new_df
custom_display(new_df,titles=["Output DataFrame"])

```

Input Data Frame

	emp_id	city
0	101	Sanzeno
1	105	Beauwelz
2	108	Minneapolis
3	100	Carterton

Output Data Frame

	emp_id	city
0	101	Sanzeno
1	105	Beauwelz
3	100	Carterton

13. Nối liền các khung dữ liệu

Để nối liền các DataFrame, sử dụng hàm concat() hoặc append().

```

emp_dict1 = {'emp_id':[101,105,108,100],
             'city':['Sanzeno', 'Beauwelz', 'Minneapolis', 'Carterton']}
emp_dict2 = {'emp_id':[111,112,113,114],
             'city':['Delhi', 'Banglore', 'Panjab', 'Chennai']}
emp_dict3 = {'emp_id':[101,105,108,100],
             'sal':['5555', '6666', '7777', '8888']}
df1 = pd.DataFrame(emp_dict1)
df2 = pd.DataFrame(emp_dict2)
df3 = pd.DataFrame(emp_dict3)

out_df = pd.concat([df1, df2], keys=['df1','df2'],ignore_index=True)
out_df
custom_display(df1, df2, out_df, titles=['Input : df1', 'Input : df2',
                                         'O/P : pd.concat([df1, df2], ignore_index=True)'])

df_concat_axis1 = pd.concat([df1, df3], axis=1)
df_concat_axis1
custom_display(df1, df3, df_concat_axis1=['Input : df1', 'Input : df3',
                                         'O/P : pd.concat([df1, df3], axis=1)'])

```


df1			df2			df3		
emp_id		city	emp_id		city	id		sal
0	101	Sanzeno	0	111	Delhi	0	101	5555
1	105	Beauwelz	1	112	Banglore	1	105	6666
2	108	Minneapolis	2	113	Panjab	2	108	7777
3	100	Carterton	3	114	Chennai	3	100	8888

O/P : `pd.concat([df1,df2],ignore_index=True)`

emp_id		city
0	101	Sanzeno
1	105	Beauwelz
2	108	Minneapolis
3	100	Carterton
4	111	Delhi
5	112	Banglore
6	113	Panjab
7	114	Chennai

O/P : `pd.concat([df1,df3],axis=1)`

emp_id		city	id	sal
0	101	Sanzeno	101	5555
1	105	Beauwelz	105	6666
2	108	Minneapolis	108	7777
3	100	Carterton	100	8888

14. Kết hợp/Tham gia các khung dữ liệu

Sử dụng cả hàm `merge()` và `join()` để kết nối các khung dữ liệu với nhau.

15. Ghi DataFrame vào tập tin bên ngoài

Pandas hỗ trợ ghi lại DataFrame vào trong nhiều dạng tập tin CSV, EXCEL, JSON, Parquet,...

Có các phương thức để lưu DataFrame như sau:

• Ghi vào tập tin CSV

```
df.to_csv("data/out.csv")
```

```
out.csv
1 ,emp_id,city
2 0,101,Sanzeno
3 1,105,Beauwelz
4 2,108,Minneapolis
5 3,111,Carterton
6
```

```
df.to_csv("data/out_wo_index.csv", index=False)
```

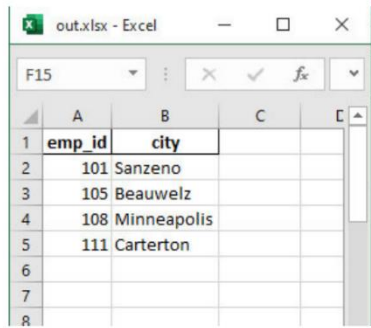
```
out_wo_index.csv
1 emp_id,city
2 101,Sanzeno
3 105,Beauwelz
4 108,Minneapolis
5 111,Carterton
6
```

```
df.to_csv("data/out_pipe.txt", index=False, sep="|")
```

```
out_pipe.txt
1 emp_id|city
2 101|Sanzeno
3 105|Beauwelz
4 108|Minneapolis
5 111|Carterton
```

- Ghi vào tệp Excel

```
df.to_excel("data_out.xlsx", index=False)
```



	A	B	C	D
1	emp_id	city		
2	101	Sanzeno		
3	105	Beauwelz		
4	108	Minneapolis		
5	111	Carterton		
6				
7				
8				

- Ghi vào tệp JSON

```
df.to_json("data/out.json")
```



```
1 [{"emp_id": {"0": 101, "1": 105, "2": 108, "3": 111},  
2  "city": {"0": "Sanzeno", "1": "Beauwelz", "2": "Minneapolis", "3": "Carterton"}}]
```

16. Kết luận

Trong phần này này, tìm hiểu về các tính năng và chức năng khác nhau của thư viện pandas. Đã khám phá tính hữu ích của pandas trong các hoạt động phân tích dữ liệu như tạo DataFrame, nhập dữ liệu vào DataFrame, dọn dẹp và tiền xử lý, phân tích, tóm tắt kết quả và xuất chúng vào các tệp tin bên ngoài để báo cáo.