

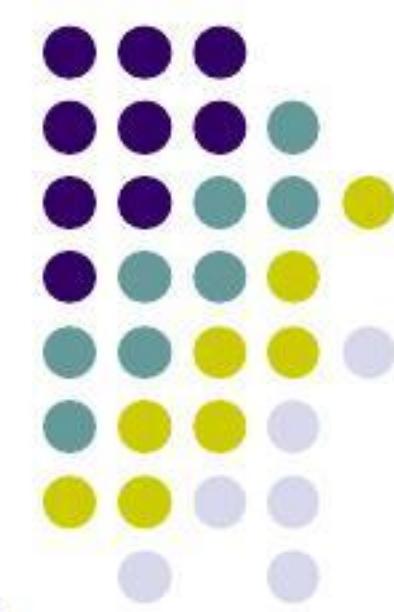


## **Bài 3: Thư viện Pandas**

*Phòng Lập Trình*



# Nội dung



- **Giới thiệu**
- **Series**
- **DataFrame**
- **Làm sạch dữ liệu**
- **Cập nhật và xử lý dữ liệu**
- **Thống kê gom nhóm**
- **Trực quan hóa dữ liệu với Pandas**

# Giới thiệu



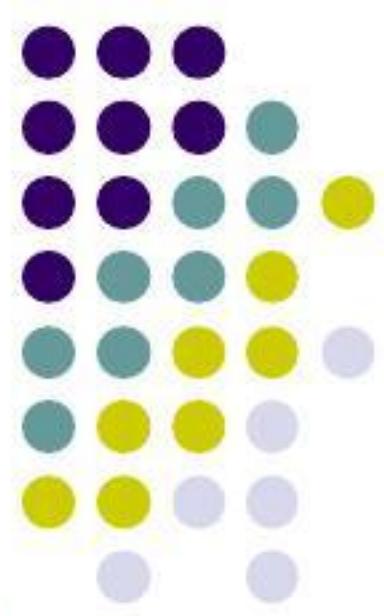
- Dữ liệu có cấu trúc là dữ liệu dạng bảng như tập tin Excel, tập tin .csv, tập tin .txt (có cấu trúc), table, view từ CSDL,...

Table

Mã sản phẩm	Tên sản phẩm	Màu sắc	Kích thước	Số lượng	Ngày nhập
AQ0001	Áo thun trơn	Đen	S	45	1/7/2020
AQ0002	Áo thun trơn	Trắng	M	65	1/7/2020
AQ0003	Áo thun trơn	Xám	L	34	1/7/2020
AQ0004	Áo thun trơn	Vàng	M	42	5/7/2020
AQ0005	Quần Short	Đen	M	27	5/7/2020
AQ0006	Quần Short	Trắng	S	43	5/7/2020
AQ0007	Quần Short	Xám	L	29	9/7/2020
AQ0008	Quần Short	Be	M	54	9/7/2020
AQ0009	Áo khoác bò	Xanh	M	37	12/7/2020
AQ0010	Áo khoác bò	Đen	L	41	12/7/2020
AQ0011	Quần jeans trơn	Xanh	L	52	12/7/2020
AQ0012	Quần jeans trơn	Đen	S	36	15/07/2020
AQ0013	Áo sơ mi	Trắng	M	24	15/07/2020
AQ0014	Áo sơ mi	Đen	L	47	15/07/2020

Rows

Columns



# Giới thiệu

## □ Pandas

- Là thư viện dùng để khám phá, làm sạch, phân tích và trực quan hóa dữ liệu.
- Có tốc độ xử lý cao, dễ thao tác, rất hữu ích cho các chuyên gia khoa học dữ liệu.
- Pandas được sử dụng trong nhiều lĩnh vực bao gồm khoa học, kinh tế, phân tích thống kê...
- Cài đặt: `pip install pandas`
- Sử dụng: `import pandas as pd`

## □ Pandas có 2 kiểu cấu trúc dữ liệu

- Series
- DataFrame

# Nội dung



- **Giới thiệu**
- **Series**
- **DataFrame**
- **Làm sạch dữ liệu**
- **Cập nhật và xử lý dữ liệu**
- **Thống kê gom nhóm**
- **Trực quan hóa dữ liệu với Pandas**

# Series



- **Series** là cấu trúc dữ liệu một chiều có gán nhãn. Mỗi phần tử trong Series gồm value (data) và index (label).
- **Đặc điểm**
  - Các phần tử trong Series được gán nhãn: index ngầm định, index tương minh (label)
  - Các phần tử trong Series phải có **cùng** kiểu dữ liệu
  - Có thể chứa **một** trong các kiểu dữ liệu như: kiểu object, int, float, bool, datetime64, ...

	First Name
0	Lois
1	Brenda
2	Joe
3	Diane
4	Benjamin
5	Patrick
6	Nancy
7	Carol
8	Frances
9	Diana

# Series



## □ Phương thức tạo series

```
pandas.Series(data [, index] [, dtype])
```

- Trong đó

- data: dữ liệu từ nhiều dạng khác nhau như constants, list, dictionary, ndarray
- index
  - Index tương minh: người dùng xác định giá trị index (label)
  - Index ngầm định: sẽ có giá trị trong np.arange(<số phần tử>)
- dtype: kiểu dữ liệu (tùy chọn)



# Series

## □ Tạo series

Tạo series từ list

```
# tạo series chứa tuổi của 5 nhân viên
age_lst = [25,27,24,28,30]
age_ser = pd.Series(age_lst)
age_ser
```

```
0    25
1    27
2    24
3    28
4    30
dtype: int64
```

# Series



## □ Tạo series

Tạo series với index tương minh

```
# Tạo series từ List với index tương minh
age_ser = pd.Series([25,27,24,28,30,27,24],
                     index = ['em1', 'em2', 'em3', 'em4', 'em5', 'em6', 'em7'])
age_ser
```

```
em1    25
em2    27
em3    24
em4    28
em5    30
em6    27
em7    24
dtype: int64
```



# Series

## □ Thuộc tính

Thuộc tính	Ý nghĩa
size	Số phần tử của series
shape	Giá trị kiểu bộ (tuple) cho biết số chiều và số phần tử của series
index	Danh sách chỉ mục của series
values	Mảng giá trị của series, kiểu ndarray
dtype	Kiểu dữ liệu của các phần tử trong series



# Series

## □ Các phương thức

Phương thức	Ý nghĩa
info()	Trả về thông tin của Series như số phần tử, kiểu dữ liệu, số lượng non-null
head(n)	Hiển thị các phần tử đầu của Series
tail(n)	Hiển thị các phần tử cuối của Series
describe()	Hiển thị thông tin thống kê chung của Series



# Series

## □ Truy xuất dữ liệu

- Truy xuất 1 phần tử

```
ser.loc[<label>]  
ser.iloc[<index>]
```

- Sử dụng iloc để truy xuất bằng index
- Sử dụng loc để truy xuất bằng label

```
# tuổi của nhân viên em3  
age_ser.loc['em3']
```

24

```
# tuổi của nhân viên ở index 3  
age_ser.iloc[3]
```

28



# Series

## □ Truy xuất dữ liệu

- Truy xuất nhiều phần tử

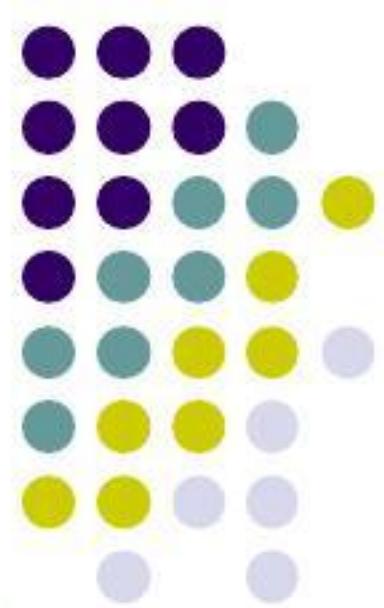
```
ser.iloc[start : stop : step]  
ser.loc[list_labels]
```

```
age_ser.iloc[1:3] # tuổi của nv ở index 1,2
```

```
em2    27  
em3    24  
dtype: int64
```

```
age_ser.loc[['em1','em3']] # tuổi của nv em1, em3
```

```
em1    25  
em3    24  
dtype: int64
```



# Series

## □ Truy xuất dữ liệu

- Truy xuất theo điều kiện

ser [biểu thức điều kiện]

```
# Tuổi của các nhân viên > 25 và < 30 tuổi
age_ser[(age_ser>25) & (age_ser<30)] # and
```

```
em2    27
em4    28
dtype: int64
```

```
# Tuổi của các nhân viên dưới 25 hoặc trên 30 tuổi
age_ser[(age_ser<25) | (age_ser>=30)] # or
```

```
em3    24
em5    30
dtype: int64
```



# Series

## □ Truy xuất dữ liệu

Thao tác	Cú pháp	Giá trị trả về
Truy xuất một phần tử	<code>series.iloc[index]</code> <code>series.loc[label]</code>	Single Value
Truy xuất nhiều phần tử	<code>series.iloc[start:stop:step]</code> <code>series.loc[start:stop:step]</code>	Series
Truy xuất theo điều kiện	<code>series[biểu thức điều kiện]</code>	Series



# Series

## □ Dữ liệu trùng (duplicate)

- Phát hiện trùng

```
series.duplicated( [keep] )
```

```
# phone_ser Lưu số điện thoại của nhân viên  
phone_ser
```

```
0 0912846759  
1 0914963258  
2 0978254361  
3 0335469512  
4 0914963258  
dtype: object
```

```
phone_ser.duplicated()
```

```
0  False  
1  False  
2  False  
3  False  
4  True  
dtype: bool
```



# Series

## □ Dữ liệu trùng (duplicate)

- Xóa dữ liệu trùng

```
series.drop_duplicates ([keep] [, inplace])
```

```
phone_ser  
0 0912846759  
1 0914963258  
2 0978254361  
3 0335469512  
4 0914963258  
dtype: object
```

```
phone_ser  
0 0912846759  
1 0914963258  
2 0978254361  
3 0335469512  
dtype: object
```





# Nội dung

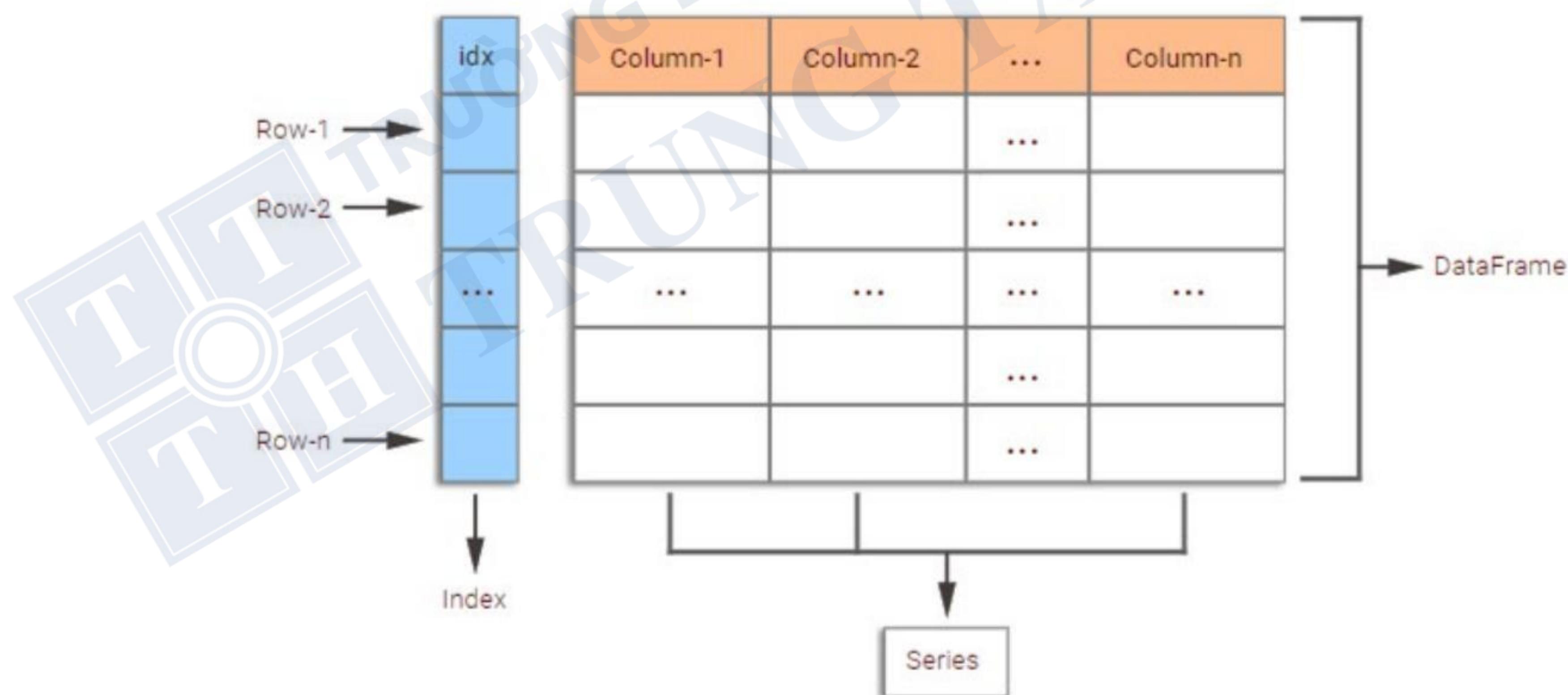
---

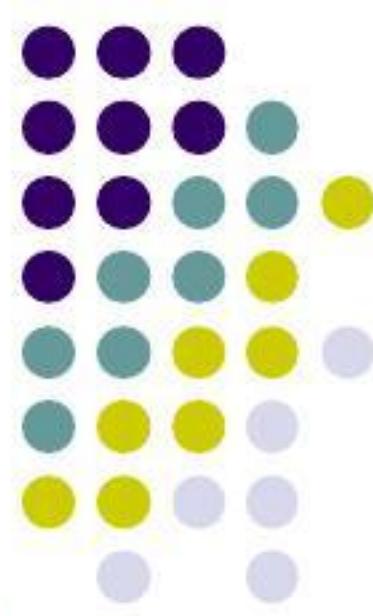
- **Giới thiệu**
- **Series**
- **DataFrame**
- **Làm sạch dữ liệu**
- **Cập nhật và xử lý dữ liệu**
- **Thống kê gom nhóm**
- **Trực quan hóa dữ liệu với Pandas**



# DataFrame

- **DataFrame** là cấu trúc dữ liệu 2D, được tổ chức theo dòng và cột.
- **Đặc điểm**
  - Mỗi cột là một kiểu dữ liệu (object, int, float, bool, datetime64, ...)
  - Mỗi cột có nhãn cột, mỗi dòng có nhãn dòng
  - Có thể thực hiện các phép tính số học theo dòng và cột





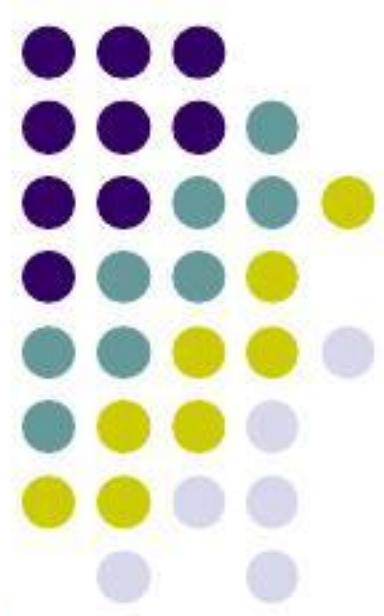
# DataFrame

## □ Phương thức tạo Data frame

```
pandas.DataFrame(data [, index] [, columns])
```

- Trong đó

- data: dữ liệu (list, dict, ndarray, series, dataframe)
- index: danh sách nhãn dòng; index = ['tên dòng 1', 'tên dòng 2', ...]
- columns: danh sách nhãn cột; columns = ['tên cột 1', 'tên cột 2', ...]

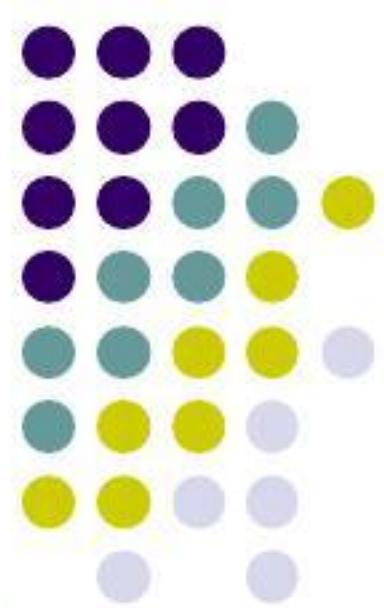


# DataFrame

## □ Tạo DataFrame

```
emp_df = pd.DataFrame({'name': ['Tom', 'Mike', 'Rose', 'Bill', 'Dick', 'John', 'Anna'],  
                      'age': [25, 27, 24, 28, 30, 27, 24],  
                      'salary': [1200, 1500, 3000, 1200, 1500, 1300, 1250]})  
emp_df
```

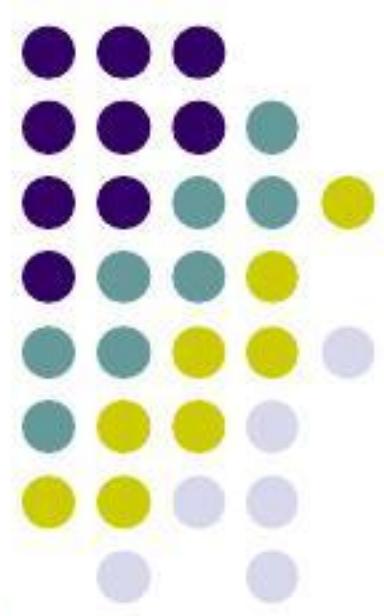
	name	age	salary
0	Tom	25	1200
1	Mike	27	1500
2	Rose	24	3000
3	Bill	28	1200
4	Dick	30	1500
5	John	27	1300
6	Anna	24	1250



# DataFrame

## □ Thuộc tính

Thuộc tính	Ý nghĩa
size	Số phần tử của dataframe
shape	Giá trị kiểu bộ (tuple) cho biết số dòng, số cột
index	Danh sách chỉ mục, nhãn dòng của dataframe
values	Mảng ndarray các giá trị của dataframe
dtypes	Kiểu dữ liệu các cột của dataframe
columns	Danh sách các cột
axes	Danh sách các nhãn dòng và các tên cột



# DataFrame

## □ Các phương thức

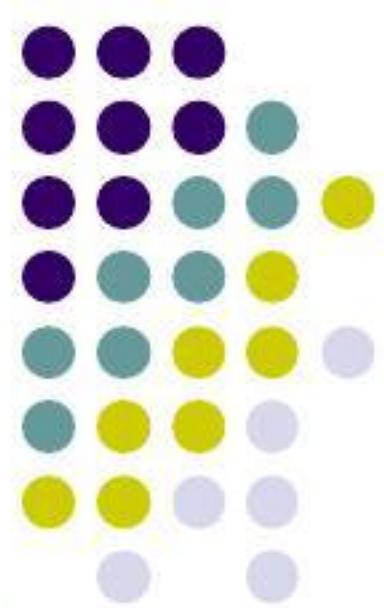
Phương thức	Ý nghĩa
info()	Trả về thông tin của DataFrame như số dòng, số cột, kiểu dữ liệu, số lượng non-null
head(n)	Hiển thị các dòng đầu của Dataframe
tail(n)	Hiển thị các dòng cuối của Dataframe
describe()	Hiển thị thông tin thống kê chung của Dataframe



# DataFrame

## □ Đọc dữ liệu từ tập tin

Data Source Type	Read Function
CSV	<code>read_csv</code>
JSON	<code>read_json</code>
HTML	<code>read_html</code>
XML	<code>read_xml</code>
Excel	<code>read_excel</code>



# DataFrame

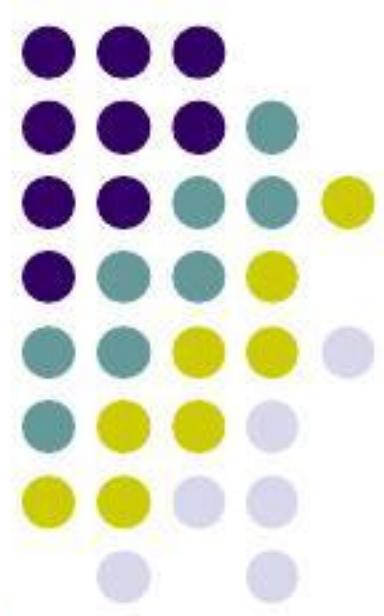
## □ Đọc dữ liệu từ tập tin

- Đọc dữ liệu từ tập tin csv

```
# Đọc dữ liệu từ employees.csv vào dataframe emp_df
emp_df = pd.read_csv('data/emp.csv')
```

```
# Xem 5 dòng đầu tiên
emp_df.head()
```

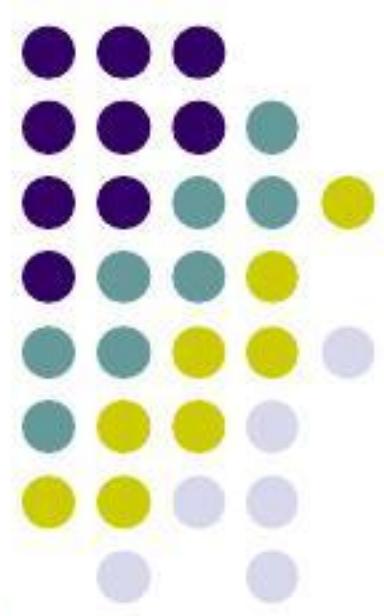
	name	age	salary
0	Tom	25	1200
1	Mike	27	1500
2	Rose	24	3000
3	Bill	28	1200
4	Dick	30	1500



# DataFrame

## □ Đọc dữ liệu từ tập tin

- `pd.read_csv`: Các tham số thông dụng
  - `filepath_or_buffer`: đường dẫn tập tin
  - `index_col = col`: đặt col làm index của DataFrame.
  - `header = k`: tên cột tương ứng với dòng thứ k sẽ làm header cho DataFrame.
  - `delimiter = None`: xác định ký tự ngăn cách giữa các trường dữ liệu trong file CSV (tương tự như `sep`)
  - `skiprows = k`: bỏ qua k dòng đầu tiên
  - `nrows = k`: chỉ lấy k dòng đầu tiên
  - `usecols = [col1, col2...]`: danh sách các cột sẽ được sử dụng cho DataFrame.



# DataFrame

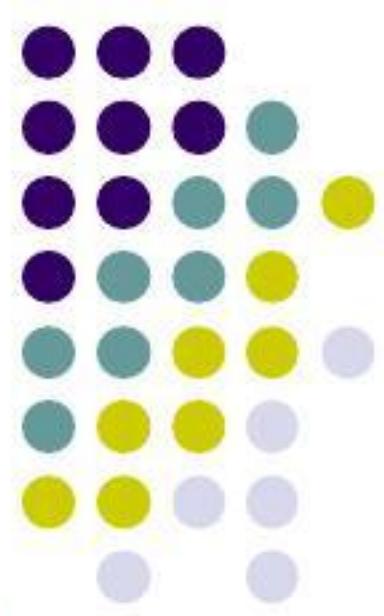
## □ Đọc dữ liệu từ tập tin

- Đọc dữ liệu từ tập tin .xlsx

```
emp_df = pd.read_excel('data/employees.xlsx')
```

```
emp_df.head()
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
0	100	Steven	King	SKING	515.123.4567	2003-06-17 00:00:00	AD_PRES	24000
1	101	Neena	Kochhar	NKOCHHAR	515.123.4568	2005-09-21 00:00:00	AD_VP	17000
2	102	Lex	De Haan	LDEHAAN	515.123.4569	2001-01-13 00:00:00	AD_VP	17000
3	103	Alexander	Hunold	AHUNOLD	590.423.4567	2006-01-03 00:00:00	IT_PROG	9000
4	104	Bruce	Ernst	BERNST	590.423.4568	2007-05-21 00:00:00	IT_PROG	6000



# DataFrame

## □ Truy xuất dữ liệu

- Truy xuất 1 phần tử

```
df.loc[row_label, col_label]  
df.iloc[row_index, col_index]
```

```
# Tên của nhân viên ở dòng đầu tiên  
emp_df.loc[0, 'FIRST_NAME'] # dòng 0, FIRST_NAME  
  
'Steven'
```

```
# họ của nhân viên ở dòng đầu tiên  
emp_df.iloc[0,2] # dòng 0, cột 2 ~ LAST_NAME  
  
'King'
```



# DataFrame

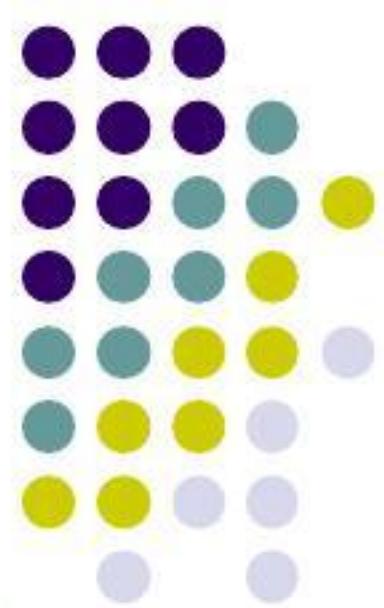
## □ Truy xuất dữ liệu

- Truy xuất nhiều phần tử

```
df.iloc[start : stop : step, start : stop : step]  
df.loc[list_row_labels, list_col_labels]
```

```
# Thông tin 'FIRST_NAME', 'LAST_NAME', 'SALARY' của các nhân viên ở dòng 1, 4  
emp_df.loc[[1,4], ['FIRST_NAME', 'LAST_NAME', 'SALARY']]
```

	FIRST_NAME	LAST_NAME	SALARY
1	Neena	Kochhar	17000
4	Bruce	Ernst	6000



# DataFrame

## □ Truy xuất dữ liệu

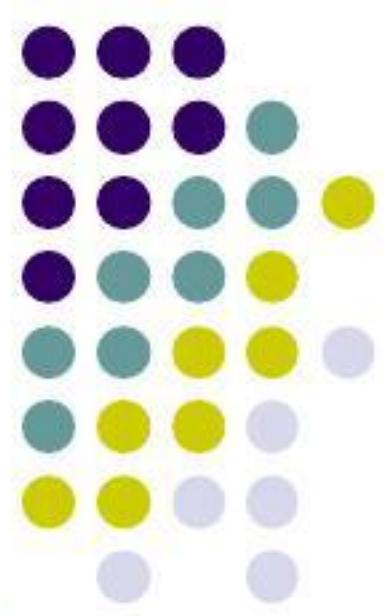
- Truy xuất 1 cột

```
df[<col_label>]
```

```
# truy xuất 1 cột: EMAIL
emp_df['EMAIL'] # Series
```

```
0      SKING
1    NKOCHHAR
2    LDEHAAN
3    AHUNOLD
4    BERNST
...
102    PFAY
103  SMAVRIS
104    HBAER
105  SHIGGINS
106    WGIETZ
Name: EMAIL, Length: 107, dtype: object
```





# DataFrame

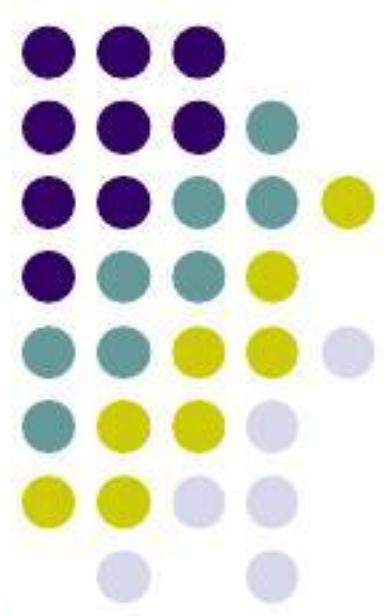
## □ Truy xuất dữ liệu

- Truy xuất nhiều cột

```
df[ [<col1_label>, <col2_label>, ... ] ]
```

```
# truy xuất các cột 'FIRST_NAME', 'LAST_NAME', 'SALARY'  
emp_df[['FIRST_NAME', 'LAST_NAME', 'SALARY']]
```

	FIRST_NAME	LAST_NAME	SALARY
0	Steven	King	24000
1	Neena	Kochhar	17000
2	Lex	De Haan	17000
3	Alexander	Hunold	9000
4	Bruce	Ernst	6000



# DataFrame

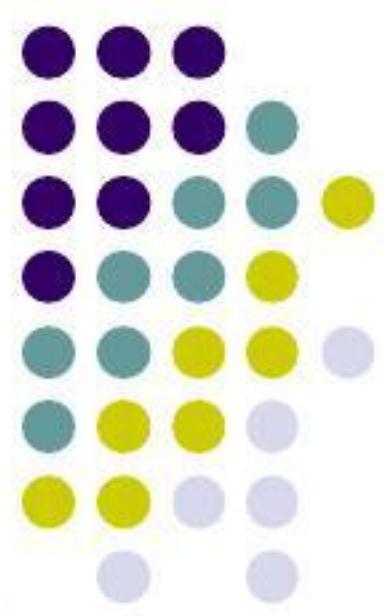
## □ Truy xuất dữ liệu

- Truy xuất nhiều dòng

```
df.iloc[list_row_index]
```

```
# Nhân viên từ dòng 2 đến dòng 5
emp_df.iloc[2:6]
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
2	102	Lex	De Haan	LDEHAAN	515.123.4569	2001-01-13 00:00:00	AD_VP
3	103	Alexander	Hunold	AHUNOLD	590.423.4567	2006-01-03 00:00:00	IT_PROG
4	104	Bruce	Ernst	BERNST	590.423.4568	2007-05-21 00:00:00	IT_PROG
5	105	David	Austin	DAUSTIN	590.423.4569	2005-06-25 00:00:00	IT_PROG



# DataFrame

## □ Truy xuất dữ liệu

- Truy xuất theo điều kiện

```
df [biểu thức điều kiện]
```

```
# Các nhân viên có vai trò là 'IT_PROG'  
emp_df[emp_df['JOB_ID']=='IT_PROG']
```

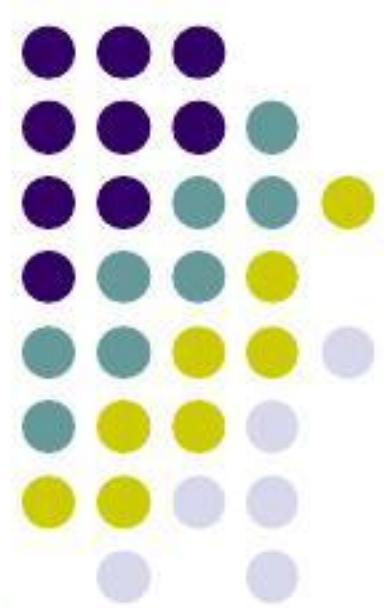
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
3	103	Alexander	Hunold	AHUNOLD	590.423.4567	2006-01-03 00:00:00	IT_PROG
4	104	Bruce	Ernst	BERNST	590.423.4568	2007-05-21 00:00:00	IT_PROG
5	105	David	Austin	DAUSTIN	590.423.4569	2005-06-25 00:00:00	IT_PROG
6	106	Valli	Pataballa	VPATABAL	590.423.4560	2006-02-05 00:00:00	IT_PROG
7	107	Diana	Lorentz	DLORENTZ	590.423.5567	2007-02-07 00:00:00	IT_PROG



# DataFrame

## □ Truy xuất dữ liệu

Thao tác	Cú pháp	Giá trị trả về
Truy xuất 1 phần tử	<code>df.loc[&lt;row_label&gt;, &lt;col_label&gt;] df.iloc[&lt;row_index&gt;, &lt;col_index&gt;]</code>	Single Value
Truy xuất nhiều phần tử	<code>df.iloc[start : stop : step,               start : stop : step] df.loc[list_row_labels, list_col_labels]</code>	Dataframe
Truy xuất 1 cột	<code>df[&lt;col_label&gt;]</code>	Series
Truy xuất nhiều cột	<code>df[[&lt;col1_label&gt;, &lt;col2_label&gt;, ...]]</code>	Dataframe
Truy xuất nhiều dòng	<code>df.iloc[start : stop : step]</code>	Dataframe
Truy xuất theo điều kiện	<code>df[&lt;biểu thức điều kiện&gt;]</code>	Dataframe



# DataFrame

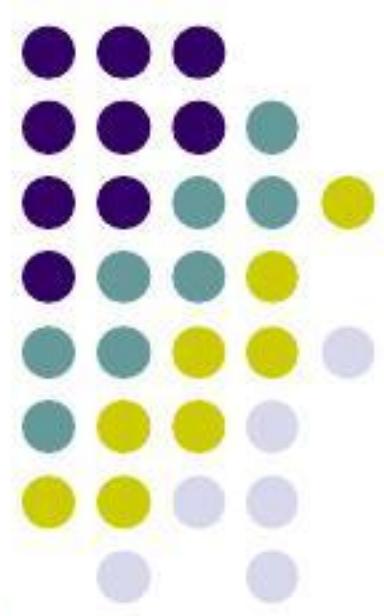
## □ Dữ liệu trùng (duplicate)

- Phát hiện trùng

```
df.duplicated( [subset] [, keep] )
```

	fullname	job_id	income	emp.duplicated()
0	John Smith	IT_PROG	1000	0 False
1	Mary Johnson	SALES_REP	1500	1 False
2	John Smith	IT_PROG	1000	2 True
3	Robert Williams	HR_REP	1200	3 False
4	Sarah Davis	IT_PROG	1000	4 False
5	Michael Wilson	AC_ACCOUNT	1100	5 False
6	John Smith	IT_PROG	1200	6 False

dtype: bool



# DataFrame

## □ Dữ liệu trùng (duplicate)

- Xóa dữ liệu trùng

```
df.drop_duplicates ([subset] [, keep] [, inplace])
```

	fullname	job_id	income
0	John Smith	IT_PROG	1000
1	Mary Johnson	SALES_REP	1500
2	John Smith	IT_PROG	1000
3	Robert Williams	HR_REP	1200
4	Sarah Davis	IT_PROG	1000
5	Michael Wilson	AC_ACCOUNT	1100
6	John Smith	IT_PROG	1200

	fullname	job_id	income
0	John Smith	IT_PROG	1000
1	Mary Johnson	SALES_REP	1500
3	Robert Williams	HR_REP	1200
4	Sarah Davis	IT_PROG	1000
5	Michael Wilson	AC_ACCOUNT	1100
6	John Smith	IT_PROG	1200

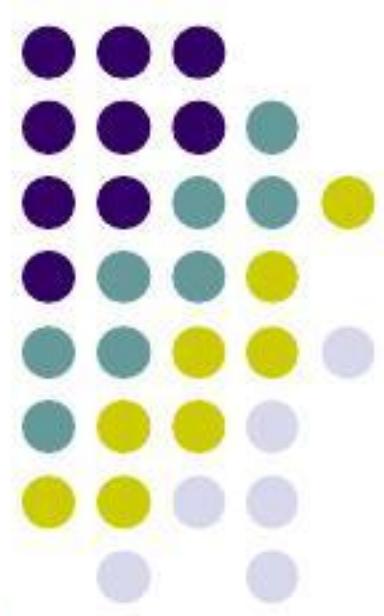




# Nội dung

---

- Giới thiệu
- Series
- DataFrame
- **Làm sạch dữ liệu**
- Cập nhật và xử lý dữ liệu
- Thống kê gom nhóm
- Trực quan hóa dữ liệu với Pandas



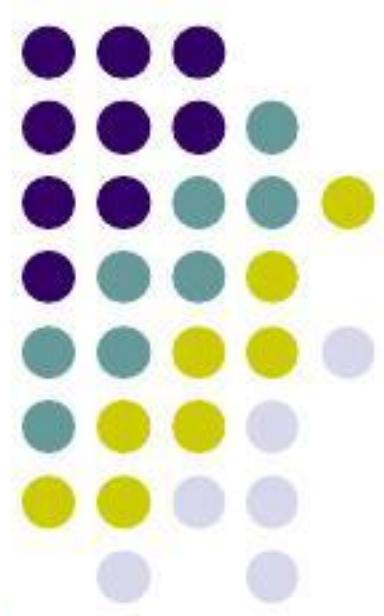
# Làm sạch dữ liệu

## □ Dữ liệu ban đầu có thể có vấn đề

- Có dữ liệu trùng
- Có các giá trị không hợp lệ (ví dụ: giá trị âm cho tuổi, cân nặng)
- Thiếu giá trị, chứa giá trị NaN (np.NaN), không có giá trị (None)

## □ Xử lý các vấn đề về dữ liệu

- Loại bỏ dữ liệu trùng
- Thay thế giá trị
- Đèn bằng dữ liệu phía trên hoặc phía dưới cho những giá trị NaN
- Loại bỏ các cột (column)
- Loại bỏ các dòng (row)
- Nội suy giá trị



# Làm sạch dữ liệu

## □ Thay thế giá trị

- Thay thế trên 1 cột

```
df['tên cột'].replace(giá_trị_cũ, giá_trị_mới)
```

```
# thay thế giá trị -1 bằng 0 trên cột y
df['y'].replace(-1, 0, inplace=True)
```

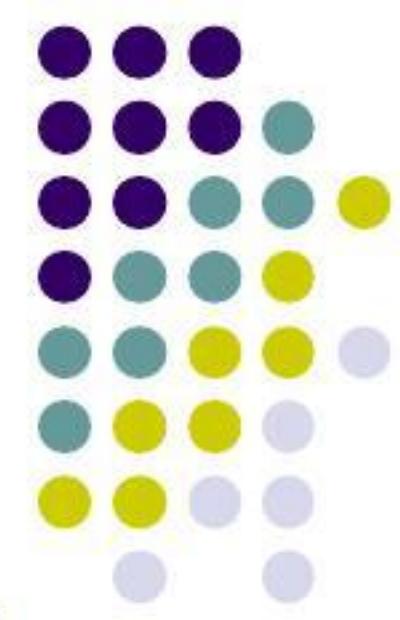
df

	x	y
0	1000	1000
1	1200	-1
2	1100	1100
3	2000	1000
4	-1	-2
5	1300	1200
6	1200	1100

df

	x	y
0	1000	1000
1	1200	0
2	1100	1100
3	2000	1000
4	-1	-2
5	1300	1200
6	1200	1100

# Làm sạch dữ liệu



## □ Thay thế giá trị

- Thay thế trên toàn bộ dataframe

```
df.replace(giá_trị_cũ, giá_trị_mới)
```

```
# thay thế giá trị -1 bằng 0 cho toàn bộ dataframe
df = df.replace(-1, 0)
```

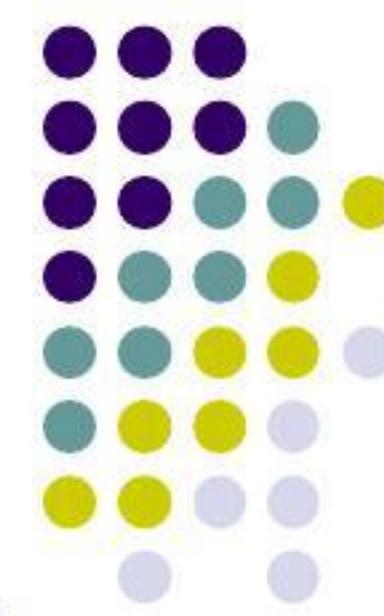
df

	x	y
0	1000	1000
1	1200	-1
2	1100	1100
3	2000	1000
4	-1	-2
5	1300	1200
6	1200	1100

df

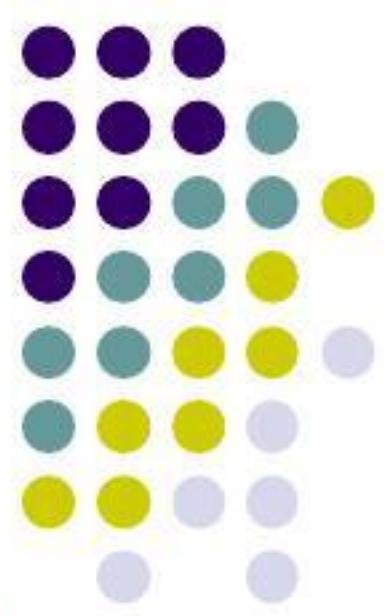
	x	y
0	1000	1000
1	1200	0
2	1100	1100
3	2000	1000
4	0	-2
5	1300	1200
6	1200	1100

# Làm sạch dữ liệu



## ☐ Xử lý dữ liệu NaN

emp_df											
PLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	
100	Steven	King	SKING	515.123.4567	2003-06-17 00:00:00	AD_PRES	24000	NaN	NaN	90.0	
101	Neena	Kochhar	NKOCHHAR	515.123.4568	2005-09-21 00:00:00	AD_VP	17000	NaN	100.0	90.0	
102	Lex	De Haan	LDEHAAN	515.123.4569	2001-01-13 00:00:00	AD_VP	17000	NaN	100.0	90.0	
103	Alexander	Hunold	AHUNOLD	590.423.4567	2006-01-03 00:00:00	IT_PROG	9000	NaN	102.0	60.0	
104	Bruce	Ernst	BERNST	590.423.4568	2007-05-21 00:00:00	IT_PROG	6000	NaN	103.0	60.0	



# Làm sạch dữ liệu

## ☐ Xử lý dữ liệu NaN

- Phát hiện NaN

```
df.info()
```

```
emp_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 107 entries, 0 to 106
Data columns (total 11 columns):
 8  COMMISSION_PCT  35 non-null    float64
 9  MANAGER_ID     106 non-null   float64
10  DEPARTMENT_ID  106 non-null   float64
```

```
df.isnull().sum()
```

```
emp_df.isnull().sum()
```

EMPLOYEE_ID	0
FIRST_NAME	0
LAST_NAME	0
EMAIL	0
PHONE_NUMBER	0
HIRE_DATE	0
JOB_ID	0
SALARY	0
COMMISSION_PCT	72
MANAGER_ID	1
DEPARTMENT_ID	1

```
dtype: int64
```

# Làm sạch dữ liệu



## ☐ Xử lý dữ liệu NaN

- Điền bằng dữ liệu phía trên hoặc phía dưới

```
df.fillna(method='ffill')
```

```
df.fillna(method='backfill')
```

df1

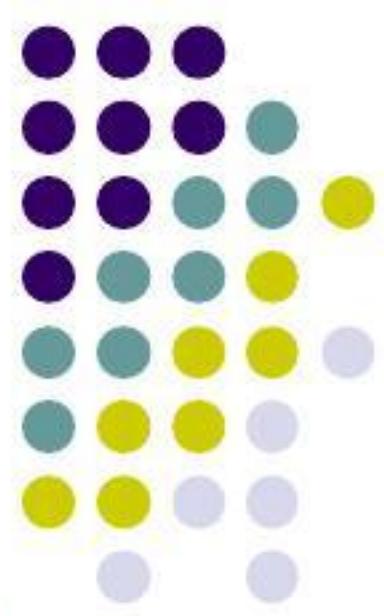
	0	1
0	-0.349596	-2.0172
1	NaN	NaN
2	1.578300	0.6374
3	NaN	NaN
4	NaN	NaN
5	NaN	NaN
6	-1.145700	0.0529

df2

	0	1
0	-0.349596	-2.0172
1	-0.349596	-2.0172
2	1.578300	0.6374
3	1.578300	0.6374
4	1.578300	0.6374
5	1.578300	0.6374
6	-1.145700	0.0529

df3

	0	1
0	-0.349596	-2.0172
1	1.578300	0.6374
2	1.578300	0.6374
3	-1.145700	0.0529
4	-1.145700	0.0529
5	-1.145700	0.0529
6	-1.145700	0.0529



# Làm sạch dữ liệu

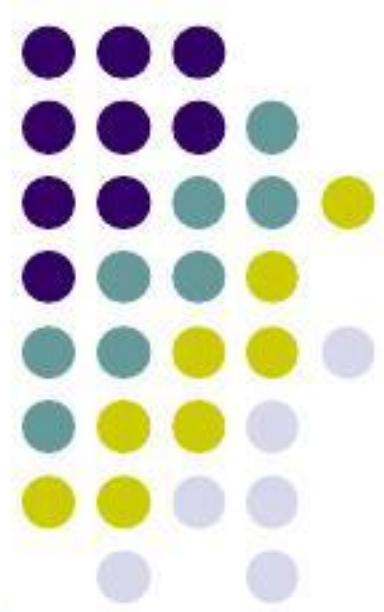
## □ Xử lý dữ liệu NaN

- Điền bằng dữ liệu khác

```
df.fillna(fill-value)
```

```
df['tên cột'].fillna(fill-value)
```

```
df.fillna(value={'tên cột': giá trị, ...})
```



# Làm sạch dữ liệu

## ☐ Xử lý dữ liệu NaN

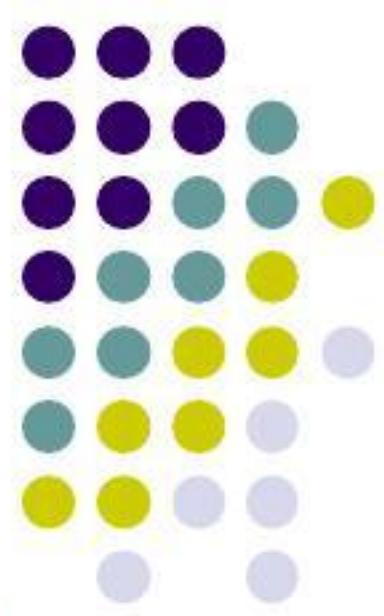
- Điền bằng dữ liệu khác

```
df6.fillna(value={'x':1300, 'y':1000}, inplace=True)
```

	x	y	z
0	1000.0	1000.0	1000.0
1	1200.0	NaN	1100.0
2	1100.0	1100.0	NaN
3	2000.0	1000.0	1000.0
4	NaN	NaN	1200.0
5	1300.0	1200.0	NaN
6	1200.0	1100.0	1100.0



	x	y	z
0	1000.0	1000.0	1000.0
1	1200.0	1000.0	1100.0
2	1100.0	1100.0	NaN
3	2000.0	1000.0	1000.0
4	1300.0	1000.0	1200.0
5	1300.0	1200.0	NaN
6	1200.0	1100.0	1100.0



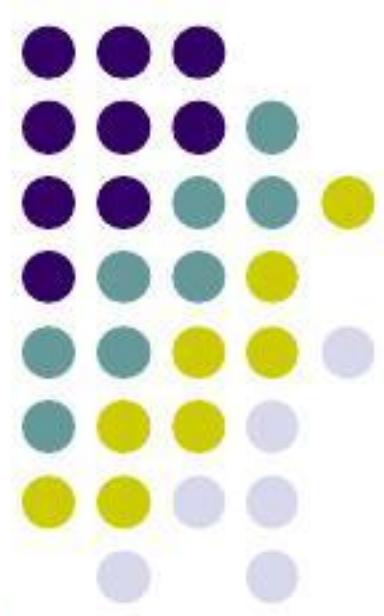
# Làm sạch dữ liệu

## ☐ Xử lý dữ liệu NaN

- Xóa bỏ dòng/cột dữ liệu có chứa NaN

```
df.dropna()           # mặc định axis=0: xóa dòng  
df.dropna(axis = 1)  # xóa cột
```

	# xóa dòng dữ liệu chứa NaN			# xóa cột
	0	1	2	2
0	-0.349596	-2.01720	-0.33450	
1	NaN	NaN	0.06800	
2	1.578300	0.63740	0.07521	
3	NaN	NaN	-0.25890	
4	NaN	NaN	0.56550	
5	NaN	NaN	-2.28520	
6	-1.145700	0.05290	-1.85410	
7	0.770500	0.08500	-0.68540	
8	0.097600	0.13705	1.25890	



# Làm sạch dữ liệu

## ☐ Xử lý dữ liệu NaN

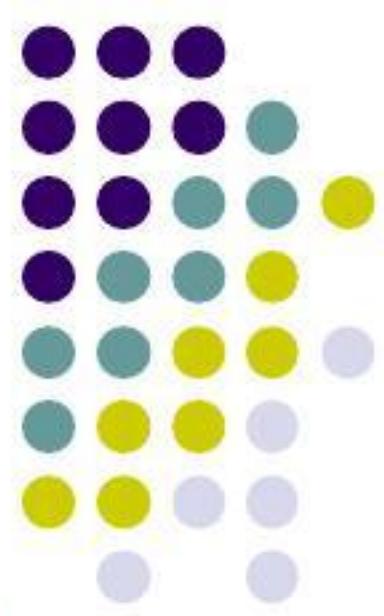
- Thực hiện nội suy tuyến tính

```
df.interpolate()
```

	0	1	2
0	-0.349596	-2.01720	-0.33450
1	NaN	NaN	0.06800
2	1.578300	0.63740	0.07521
3	NaN	NaN	-0.25890
4	NaN	NaN	0.56550
5	NaN	NaN	-2.28520
6	-1.145700	0.05290	-1.85410
7	0.770500	0.08500	-0.68540
8	0.097600	0.13705	1.25890



	0	1	2
0	-0.349596	-2.017200	-0.33450
1	0.614352	-0.689900	0.06800
2	1.578300	0.637400	0.07521
3	0.897300	0.491275	-0.25890
4	0.216300	0.345150	0.56550
5	-0.464700	0.199025	-2.28520
6	-1.145700	0.052900	-1.85410
7	0.770500	0.085000	-0.68540
8	0.097600	0.137050	1.25890

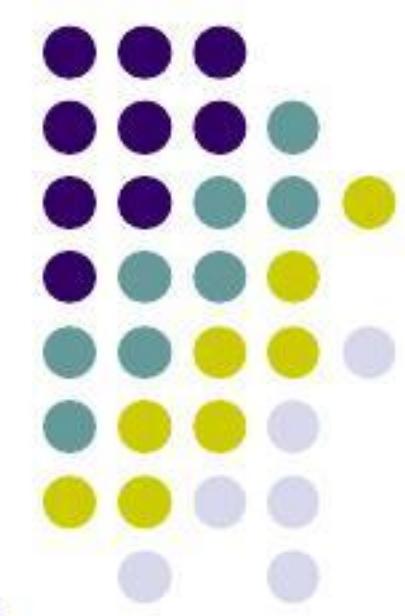


# Nội dung

---

- Giới thiệu
- Series
- DataFrame
- Làm sạch dữ liệu
- Cập nhật và xử lý dữ liệu
- Thống kê gom nhóm
- Trực quan hóa dữ liệu với Pandas

# Cập nhật và xử lý dữ liệu



## □ Thêm các dòng vào cuối DataFrame (gộp 2 DataFrame)

```
pd.concat([df_1, df_2])
```

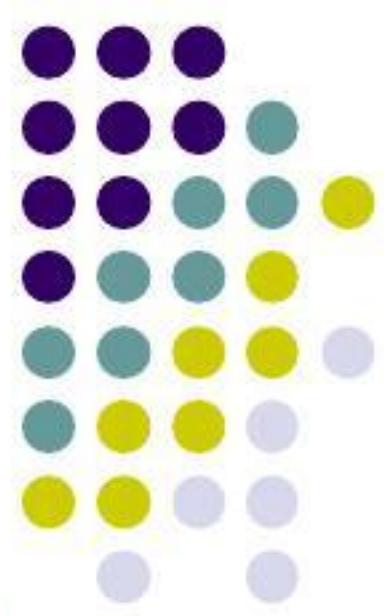
```
# Thêm new_rows vào dataframe emp
emp = pd.concat([emp, new_rows], ignore_index=True)
```

```
emp
```

	<b>id</b>	<b>fullname</b>	<b>job_id</b>	<b>income</b>
0	1	John Smith	IT_PROG	1000
1	2	Mary Johnson	SALES_REP	1500
2	3	Robert Williams	HR_REP	1200
3	4	Sarah Davis	IT_PROG	1000
4	5	Michael Wilson	AC_ACCOUNT	1100
5	6	Linda Brown	AC_MANAGER	1500



	<b>id</b>	<b>fullname</b>	<b>job_id</b>	<b>income</b>
0	1	John Smith	IT_PROG	1000
1	2	Mary Johnson	SALES_REP	1500
2	3	Robert Williams	HR_REP	1200
3	4	Sarah Davis	IT_PROG	1000
4	5	Michael Wilson	AC_ACCOUNT	1100
5	6	Linda Brown	AC_MANAGER	1500
6	7	William Jones	IT_PROG	1100
7	8	Susan Miller	SALES_REP	1600



# Cập nhật và xử lý dữ liệu

## □ Sửa giá trị trong DataFrame

Bước 1: Truy xuất

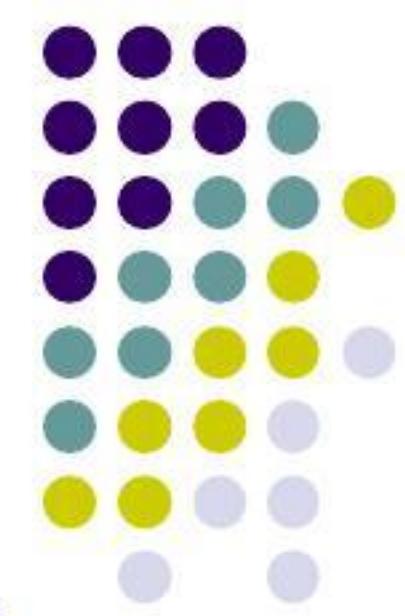
Bước 2: Gán giá trị mới

```
emp.loc[4, 'income'] = 1400
```

```
emp.loc[emp['income']==1000, 'income'] = 1300
```

```
emp.loc[(emp['job_id']=='SALES_REP') & (emp['income']<2000), 'income'] = 2000
```

# Cập nhật và xử lý dữ liệu



## ☐ Xóa dòng trong DataFrame

Xóa theo label:

```
df.drop(list_row_label)
```

Xóa theo index:

```
df.drop(df.index[list_row_index])
```

```
emp.drop([6], inplace=True)
```

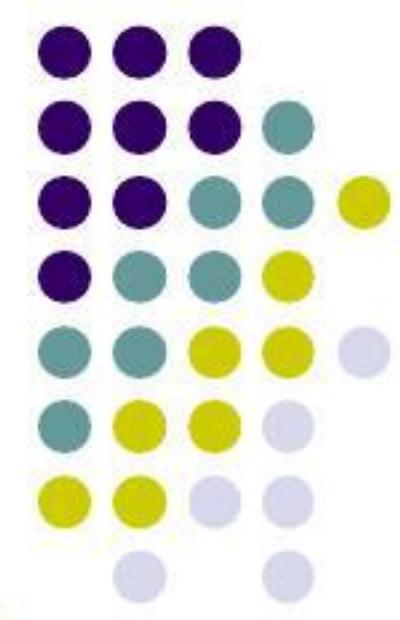
```
emp.drop(emp.index[[2,5]], inplace=True)
```

The diagram illustrates the deletion of a row from a DataFrame. On the left, the original DataFrame 'emp' is shown with 9 rows of data. On the right, the modified DataFrame 'emp' is shown with 8 rows of data, indicating that the 6th row has been removed. A red arrow points from the original DataFrame to the modified one.

emp				
	<b>id</b>	<b>fullname</b>	<b>job_id</b>	<b>income</b>
0	1	John Smith	IT_PROG	1300
1	2	Mary Johnson	SALES_REP	2000
2	3	Robert Williams	HR_REP	1200
3	4	Sarah Davis	IT_PROG	1300
4	5	Michael Wilson	AC_ACCOUNT	1400
5	6	Linda Brown	AC_MANAGER	1500
6	7	William Jones	IT_PROG	1100
7	8	Susan Miller	SALES_REP	2000

emp				
	<b>id</b>	<b>fullname</b>	<b>job_id</b>	<b>income</b>
0	1	John Smith	IT_PROG	1300
1	2	Mary Johnson	SALES_REP	2000
3	4	Sarah Davis	IT_PROG	1300
4	5	Michael Wilson	AC_ACCOUNT	1400
7	8	Susan Miller	SALES_REP	2000



# Cập nhật và xử lý dữ liệu

## □ Xóa cột trong DataFrame (axis=1)

```
df.drop(list_col_label, axis = 1, inplace = True)
```

```
# xóa cột id
emp.drop(['id'], axis=1, inplace=True)
```

emp

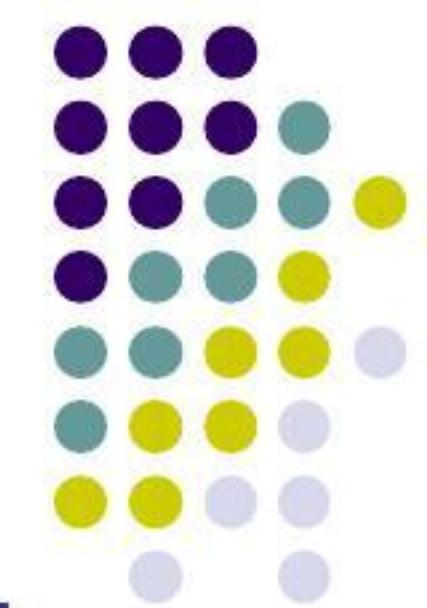
	<b>id</b>	<b>fullname</b>	<b>job_id</b>	<b>income</b>
0	1	John Smith	IT_PROG	1300
1	2	Mary Johnson	SALES_REP	2000
3	4	Sarah Davis	IT_PROG	1300
4	5	Michael Wilson	AC_ACCOUNT	1400
7	8	Susan Miller	SALES_REP	2000

emp

		<b>fullname</b>	<b>job_id</b>	<b>income</b>
0		John Smith	IT_PROG	1300
1		Mary Johnson	SALES_REP	2000
3		Sarah Davis	IT_PROG	1300
4		Michael Wilson	AC_ACCOUNT	1400
7		Susan Miller	SALES_REP	2000



# Cập nhật và xử lý dữ liệu



## □ Chuyển đổi kiểu dữ liệu

```
df['tên cột'].astype(<kiểu_dữ_liệu>)
```

# Chuyển đổi kiểu dữ liệu của cột income sang số nguyên

```
emp['income'] = emp['income'].str.replace('$', '', regex=False).astype(int)
```

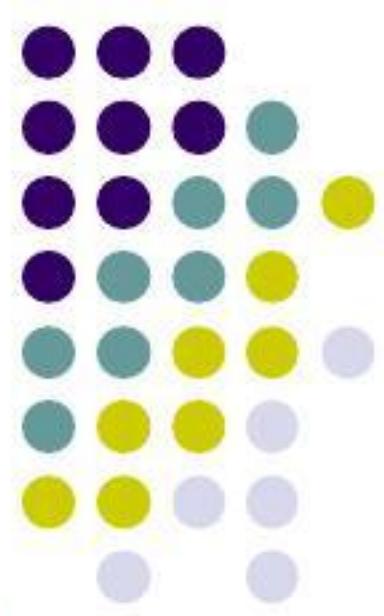
	fullname	job_id	income
0	John Smith	IT_PROG	\$1000
1	Mary Johnson	SALES_REP	\$1500
2	Robert Williams	HR_REP	\$1200
3	Sarah Davis	IT_PROG	\$1000
4	Michael Wilson	AC_ACCOUNT	\$1100

salary      object

	fullname	job_id	income
0	John Smith	IT_PROG	1000
1	Mary Johnson	SALES_REP	1500
2	Robert Williams	HR_REP	1200
3	Sarah Davis	IT_PROG	1000
4	Michael Wilson	AC_ACCOUNT	1100

salary      int32





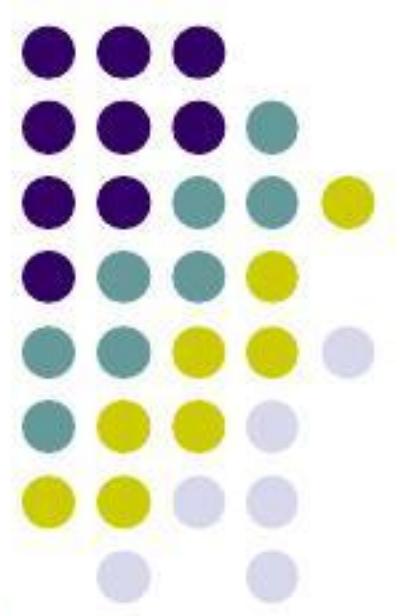
# Cập nhật và xử lý dữ liệu

## ☐ Đổi tên cột trong DataFrame

```
df.rename(columns, inplace = True)
```

```
# đổi tên cột income thành salary
emp.rename(columns={'income':'salary'}, inplace=True)
```

	fullname	job_id	income		fullname	job_id	salary	
0	John Smith	IT_PROG	1000		0	John Smith	IT_PROG	1000
1	Mary Johnson	SALES_REP	1500		1	Mary Johnson	SALES_REP	1500
2	Robert Williams	HR_REP	1200	→	2	Robert Williams	HR_REP	1200
3	Sarah Davis	IT_PROG	1000		3	Sarah Davis	IT_PROG	1000
4	Michael Wilson	AC_ACCOUNT	1100		4	Michael Wilson	AC_ACCOUNT	1100

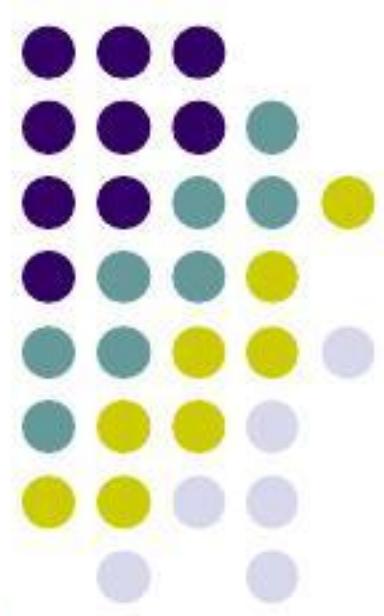


# Cập nhật và xử lý dữ liệu

## ☐ Xử lý tạo cột mới trong DataFrame

```
# tạo cột bonus Là tiền thưởng của nhân viên
emp['bonus'] = [50, 75, 0, 150, 100]
# tạo cột income Là thu nhập của nhân viên trong tháng (=salary + bonus)
emp['income'] = emp['salary'] + emp['bonus']
emp
```

	fullname	job_id	salary	bonus	income
0	John Smith	IT_PROG	1000	50	1050
1	Mary Johnson	SALES_REP	1500	75	1575
2	Robert Williams	HR_REP	1200	0	1200
3	Sarah Davis	IT_PROG	1000	150	1150
4	Michael Wilson	AC_ACCOUNT	1100	100	1200

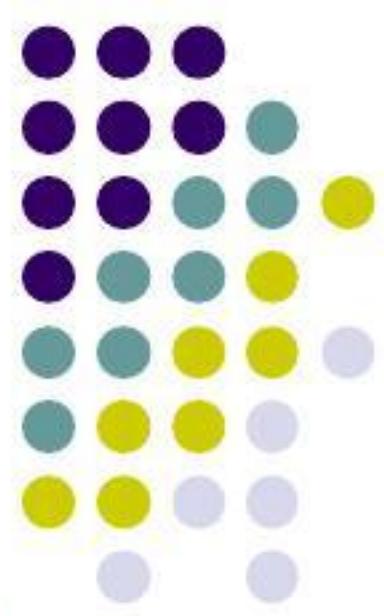


# Cập nhật và xử lý dữ liệu

## ☐ Xử lý tạo cột mới trong DataFrame

```
# tạo cột rewarded từ giá trị của cột bonus: bonus > 0 => True; bonus == 0 => False
emp['rewarded'] = emp['bonus'].map(lambda x: True if x>0 else False)
emp
```

	fullname	job_id	salary	bonus	income	rewarded
0	John Smith	IT_PROG	1000	50	1050	True
1	Mary Johnson	SALES_REP	1500	75	1575	True
2	Robert Williams	HR_REP	1200	0	1200	False
3	Sarah Davis	IT_PROG	1000	150	1150	True
4	Michael Wilson	AC_ACCOUNT	1100	100	1200	True



# Cập nhật và xử lý dữ liệu

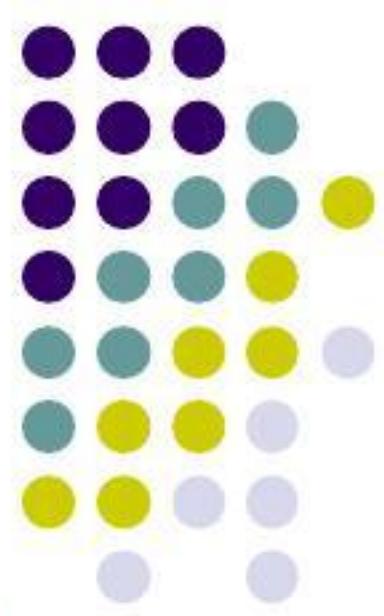
## □ Phương thức thao tác trên chuỗi

- Tìm chuỗi bắt đầu bằng chuỗi con

```
df[“Tên_cột”].str.startswith(“chuoi_con”)
```

```
# Tìm nhân viên có JOB_ID bắt đầu là “AC” (Phòng Kế toán)
emp_df[emp_df[‘JOB_ID’].str.startswith(‘AC’)]
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
105	205	Shelley	Higgins	SHIGGINS	2002-06-07 00:00:00	AC_MGR
106	206	William	Gietz	WGIETZ	2002-06-07 00:00:00	AC_ACCOUNT



# Cập nhật và xử lý dữ liệu

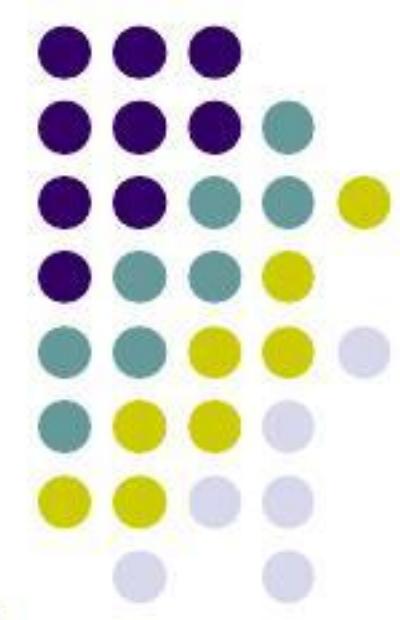
## □ Phương thức thao tác trên chuỗi

- Tìm chuỗi chứa chuỗi con

```
df[“Tên_cột”].str.contains(“chuoi_con”)
```

```
# Tìm nhân viên có JOB_ID chứa “REP” (Representative)
emp_df[emp_df[‘JOB_ID’].str.contains(‘REP’)]
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
50	150	Peter	Tucker	PTUCKER	011.44.1344.129268	2005-01-30 00:00:00
51	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24 00:00:00
52	152	Peter	Hall	PHALL	011.44.1344.478968	2005-08-20 00:00:00
53	153	Christopher	Olsen	COLSEN	011.44.1344.498718	2006-03-30 00:00:00
54	154	Nanette	Cambrault	NCAMBRAU	011.44.1344.987668	2006-12-09 00:00:00



# Cập nhật và xử lý dữ liệu

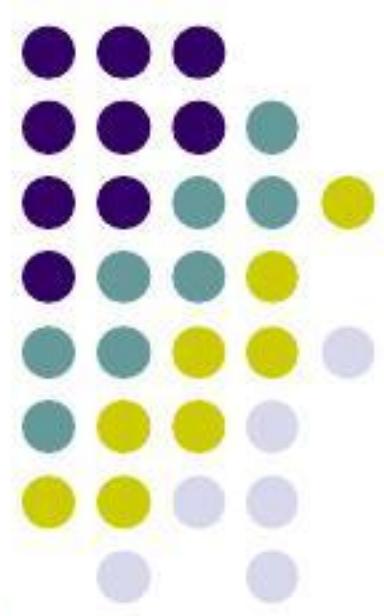
## □ Phương thức thao tác trên chuỗi

- Đổi sang chữ thường

```
df[“Tên_cột”].str.lower()
```

```
# Đổi cột EMAIL thành chữ thường
emp_df['EMAIL'] = emp_df['EMAIL'].str.lower()
emp_df.head()
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
0	100	Steven	King	sking	515.123.4567 00:00:00	AD_PRES	24000
1	101	Neena	Kochhar	nkochhar	515.123.4568 00:00:00	AD_VP	17000
2	102	Lex	De Haan	ldehaan	515.123.4569 00:00:00	AD_VP	17000
3	103	Alexander	Hunold	ahunold	590.423.4567 00:00:00	IT_PROG	9000
4	104	Bruce	Ernst	bernst	590.423.4568 00:00:00	IT_PROG	6000

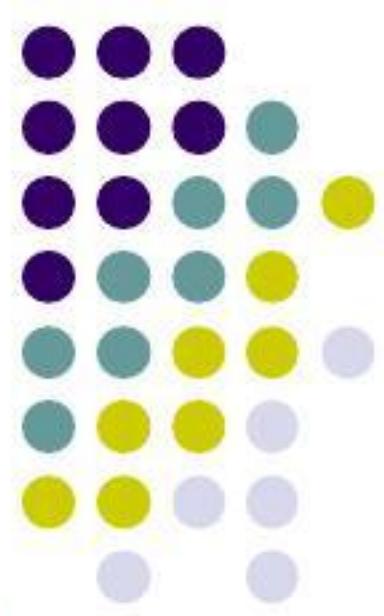


# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên chuỗi với Regular Expression

- Regular Expression là chuỗi ký tự đặc biệt được dùng để tìm kiếm và trích xuất các chuỗi theo một cấu trúc cụ thể.

Ký tự	Ý nghĩa	Ví dụ
[]	Khớp với một trong tập hợp ký tự	[xyz] => x or y or z
.	Bất kỳ ký tự nào (ngoại trừ ký tự xuống dòng)	1. => 12; 1A, ...
^	Bắt đầu của chuỗi	^a => a, account, ...
\$	Kết thúc của chuỗi	er\$ => manager, buyer, ...
*	ký tự trước đó xuất hiện 0 hoặc nhiều lần	4*6 => 6, 46, 4446, ...
+	ký tự trước đó xuất hiện 1 hoặc nhiều lần	4+6 => 46, 4446, ...
?	ký tự trước đó xuất hiện 0 hoặc 1 lần	4?6 => 6, 46
{m,n}	ký tự trước đó xuất hiện từ m đến n lần	4{2,3} => 44, 444



# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên chuỗi với Regular Expression

Ký tự	Ý nghĩa	Ví dụ
\d	Số bất kì Thay thế cho [0-9]	\d => 1, 3, ...
\D	Ký tự không phải là số Thay thế cho [^0-9]	\D => a, B, ...
\s	Kí tự khoảng trắng	\s => [space], \t, \n
\w	Ký tự thuộc a-z hoặc A-Z hoặc 0-9 hoặc _ Thay thế cho [a-zA-Z0-9_]	\w => 3, x, ...
\W	Ký tự KHÔNG thuộc a-z hoặc A-Z hoặc 0-9 hoặc _	\W => #, !, @, ...



# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên chuỗi với Regular Expression

- Ký tự [ ] : Khớp với một trong tập hợp ký tự (hoặc)

```
# Tìm các last_name có chứa chữ q hoặc x
f = open('data/last_names.txt')
for line in f:
    regex = '[qx]'
    if re.search(regex, line):
        print(line, end = '')
f.close()
```

Cox  
Alexander  
Dixon  
Fox  
Vasquez  
Rodriquez  
Maxwell  
Wilcox  
Marquez  
Vazquez  
Baxter



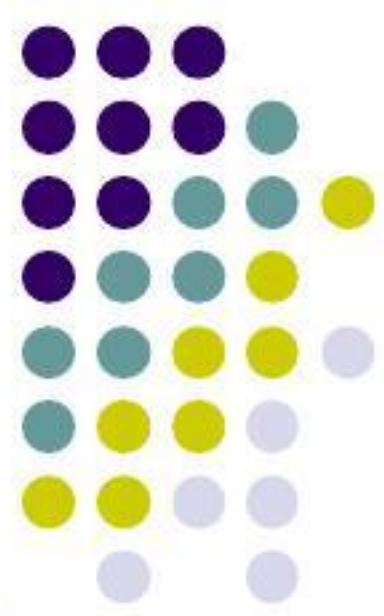
# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên chuỗi với Regular Expression

- Ký tự ^: bắt đầu của chuỗi

```
# Tìm các Last_name bắt đầu với Barn
f = open('data/last_names.txt')
for line in f:
    regex = '^Barn'
    if re.search(regex, line):
        print(line, end = '')
f.close()
```

Barnes  
Barnett  
Barnard  
Barnhart  
Barney



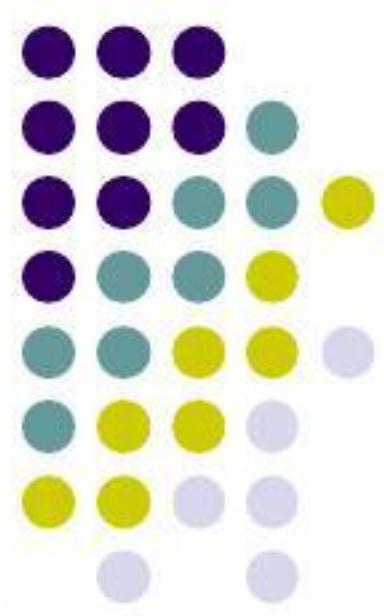
# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên chuỗi với Regular Expression

- Ký tự \$: kết thúc của chuỗi

```
# Tìm các Last_name có 5 ký tự, bắt đầu với M và kết thúc Là in
f = open('data/last_names.txt')
for line in f:
    regex = 'M..in$'
    if re.search(regex, line):
        print(line, end = '')
f.close()
```

Morin  
Marin



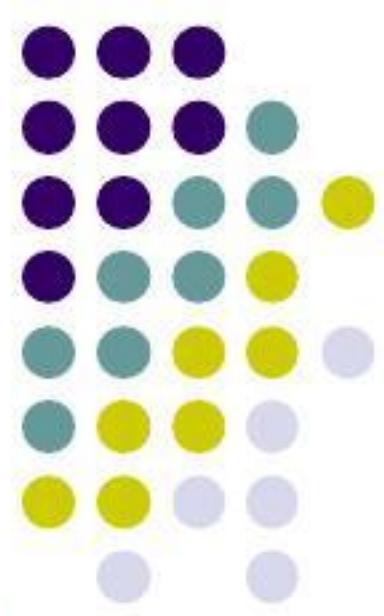
# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên chuỗi với Regular Expression

- Ký tự \* + ?: Số lần xuất hiện của ký tự trước đó

```
text = """Lương nhân viên theo đơn vị tiền tệ Euro là 800€, 1000€, 1500€, 1200€, 960€, 786€.  
Chuyển đổi sang Dollar sẽ là 848$, 1060$, 1590$, 1272.$, 1017.6$, 833.16$"""
```

```
# Lấy ra tất cả Lương theo Euro và theo Dollar (bao gồm cả ký hiệu tiền tệ)  
pattern = '\d+\.?\d*[€$]'  
  
print(re.findall(pattern, text))  
  
['800€', '1000€', '1500€', '1200€', '960€', '786€', '848$', '1060$', '1590$',  
'1272.$', '1017.6$', '833.16$']
```



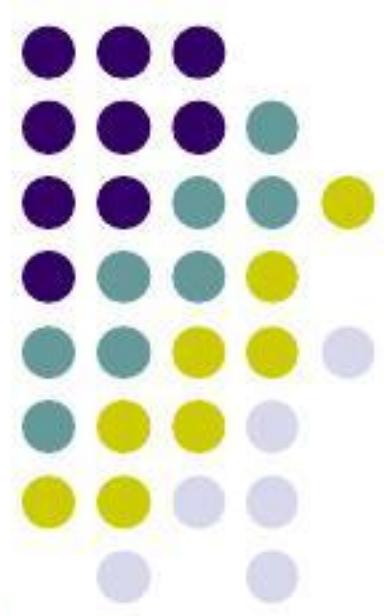
# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên chuỗi với Regular Expression

- Ký tự  $\{m, n\}$ : Ký tự trước đó xuất hiện từ  $m$  đến  $n$  lần

```
# tạo pattern cho user name từ 3 đến 16 ký tự.  
# Nó chỉ có thể bao gồm các ký tự số và chữ (không có chữ in hoa),  
# dấu gạch ngang và dấu gạch dưới
```

```
p = '^[\a-zA-Z0-9_]{3,16}$'
```



# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên chuỗi với Regular Expression



```
text = "The winners are: User9, UserN, User8. The prize is $3000."
```

```
# Tìm tất cả User có mã user Là số
```

```
re.findall("User\d", text)
```

```
['User9', 'User8']
```

```
# Tìm tất cả User có mã user không phải Là số
```

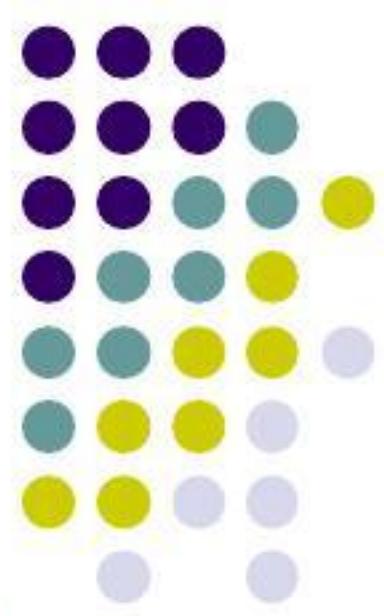
```
re.findall("User\\D", text)
```

['UserN']

```
# Tìm tất cả User có mã user là số hoặc chữ
```

```
re.findall("User\\w", text)
```

```
['User9', 'UserN', 'User8']
```



# Cập nhật và xử lý dữ liệu

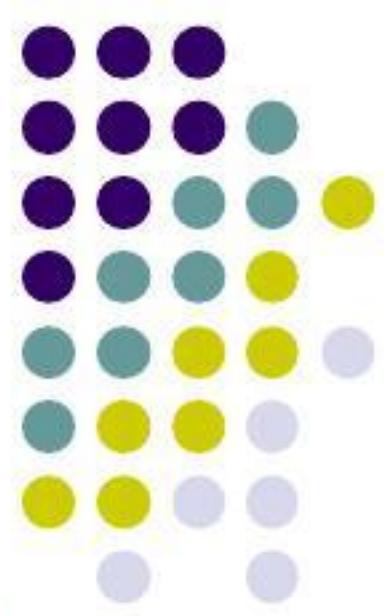
## □ Phương thức thao tác trên chuỗi với Regular Expression



```
text = "The winners are: User9,\tUserN,\tUser8.\nThe prize is $3000."  
# Tìm tiền thưởng (bao gồm cả ký hiệu tiền tệ)  
re.findall("\w\d+", text)  
['$3000']
```

```
# Tách chuỗi dựa trên dấu câu và khoảng trắng
re.split("\W\s", text)
```

['The winners are', 'User9', 'UserN', 'User8', 'The prize is \$3000.']}



# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên chuỗi với Regular Expression

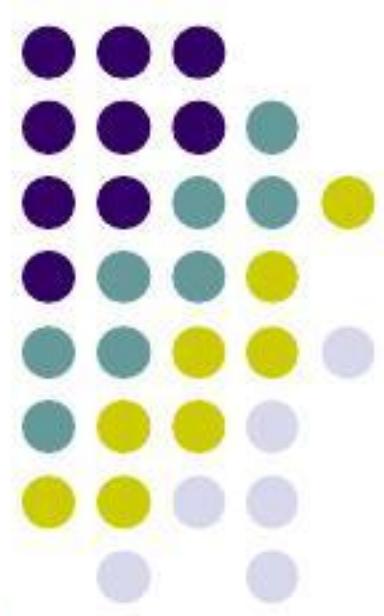
- Hàm `split()`

```
order_id = '20230209-TT*CV#0001'  
re.split('\W', order_id)
```

```
['20230209', 'TT', 'CV', '0001']
```

```
order[['order_date', 'category', 'product', 'item']] = order['order_id'].str.split('\W', expand=True)
```

	order_id	order_id	order_date	category	product	item
0	20230209-TT*CV#0001	0	20230209	TT	CV	0001
1	20230208-DT*LT#0010	1	20230208	DT	LT	0010



# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên chuỗi với Regular Expression

- Hàm `sub()`: Thay thế trên chuỗi
- Hàm `replace()`: Thay thế trên dataframe

```
price = '$40,000'  
price = int(re.sub('\D+', '', price))  
price
```

40000

```
df['Price'] = df['Price'].str.replace('\D+', '', regex=True).astype('int')
```



	Price
0	\$40,000*
1	\$40000 conditions attached



	Price
0	40000
1	40000



# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên chuỗi với Regular Expression

- Hàm.findall()

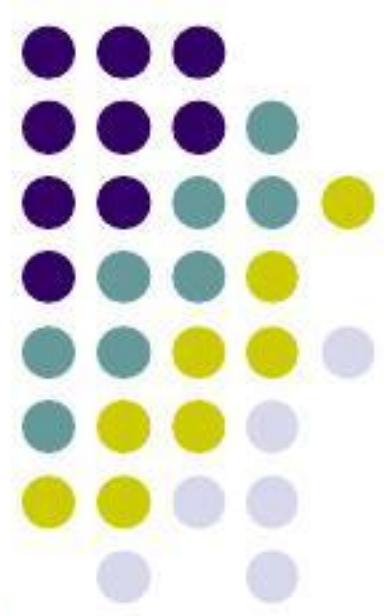
```
diem = 'toán: 7.0 văn: 6.5'  
re.findall('(\d+\.\d+)', diem)  
['7.0', '6.5']
```

```
diem_df[['Toán', 'Văn']] = diem_df['Điểm'].str.findall('(\d+\.\d+)').to_list()
```

Điểm
0 toán: 7.0 văn: 6.5
1 toán: 6.0 văn: 8.5
2 toán: 9.5 văn: 7.0



Điểm	Toán	Văn
0 toán: 7.0 văn: 6.5	7.0	6.5
1 toán: 6.0 văn: 8.5	6.0	8.5
2 toán: 9.5 văn: 7.0	9.5	7.0



# Cập nhật và xử lý dữ liệu

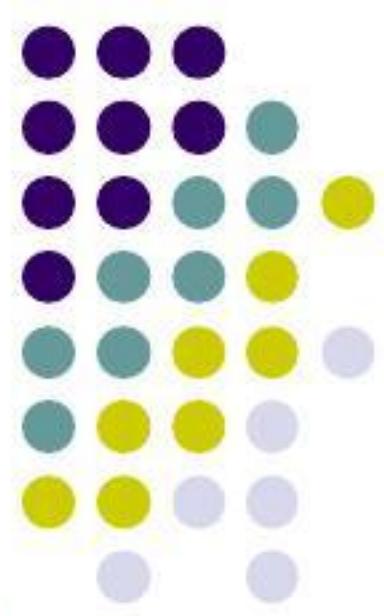
## □ Phương thức thao tác trên chuỗi với Regular Expression

- Hàm search()

```
text = "Hello, this is a sample text with some numbers 12345."  
pattern = '\d+' # Một pattern đơn giản để tìm các con số.
```

```
match = re.search(pattern, text)  
if match:  
    print("Tìm thấy kết quả:", match.group())  
else:  
    print("Không tìm thấy kết quả.")
```

Tìm thấy kết quả: 12345



# Cập nhật và xử lý dữ liệu

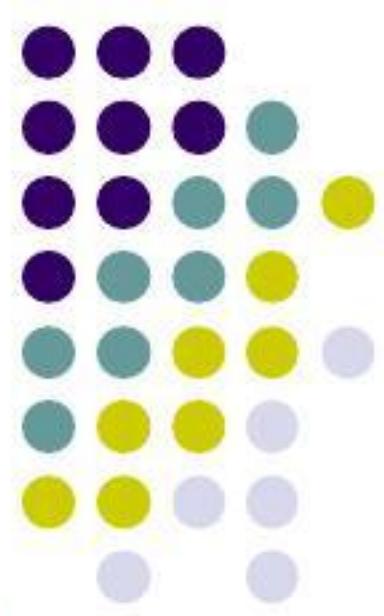
## □ Phương thức thao tác trên chuỗi với Regular Expression

- Hàm contains()

```
# Sử dụng str.contains với regular expression pattern
pattern = '\d+' # Tìm các số

matches = df[df['Text'].str.contains(pattern, regex=True)]
```

	Text		Text
0	This is a sentence with 123 numbers.	→	0 This is a sentence with 123 numbers.
1	Python is a powerful programming language.		
2	Regular expressions are useful for text proces...		
3	Data science involves data analysis.		



# Cập nhật và xử lý dữ liệu

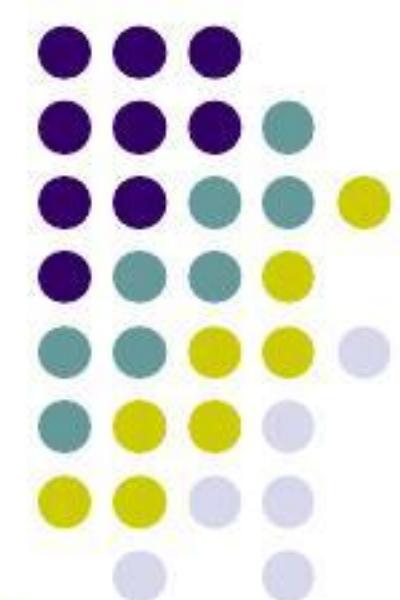
## □ Phương thức thao tác trên chuỗi với Regular Expression

- Hàm extract ()

```
# Tạo cột "AREA_CODE" Là 3 chữ số đầu tiên của "PHONE_NUMBER"
```

```
emp_df['AREA_CODE'] = emp_df['PHONE_NUMBER'].str.extract('(\d{3})')
```

	PHONE_NUMBER	AREA_CODE
0	515.123.4567	515
1	515.123.4568	515
2	515.123.4569	515
3	590.423.4567	590
4	590.423.4568	590



# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên dữ liệu thời gian

```
emp_df.head()
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
0	100	Steven	King	SKING	515.123.4567	2003-06-17 00:00:00	AD_PRES	24000
1	101	Neena	Kochhar	NKOCHHAR	515.123.4568	2005-09-21 00:00:00	AD_VP	17000
2	102	Lex	De Haan	LDEHAAN	515.123.4569	2001-01-13 00:00:00	AD_VP	17000
3	103	Alexander	Hunold	AHUNOLD	590.423.4567	2006-01-03 00:00:00	IT_PROG	9000
4	104	Bruce	Ernst	BERNST	590.423.4568	2007-05-21 00:00:00	IT_PROG	6000

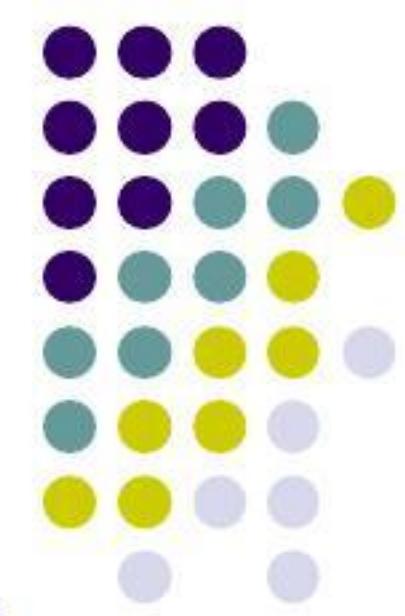
```
emp_df.dtypes
```

HIRE\_DATE

object



# Cập nhật và xử lý dữ liệu



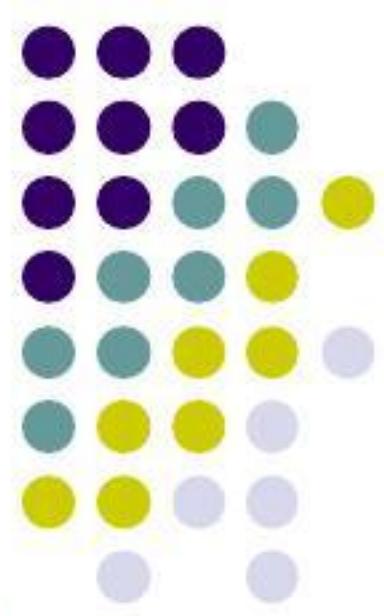
## □ Phương thức thao tác trên dữ liệu thời gian

- Chuyển dữ liệu có ý nghĩa thời gian sang dữ liệu DateTime

```
pd.to_datetime(df[<col>], [format])
```

```
emp_df['HIRE_DATE'] = pd.to_datetime(emp_df['HIRE_DATE'],  
format = '%Y-%m-%d %H:%M:%S')
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE
0	100	Steven	King	SKING	515.123.4567	2003-06-17
1	101	Neena	Kochhar	NKOCHHAR	515.123.4568	2005-09-21
2	102	Lex	De Haan	LDEHAAN	515.123.4569	2001-01-13
3	103	Alexander	Hunold	AHUNOLD	590.423.4567	2006-01-03
4	104	Bruce	Ernst	BERNST	590.423.4568	2007-05-21

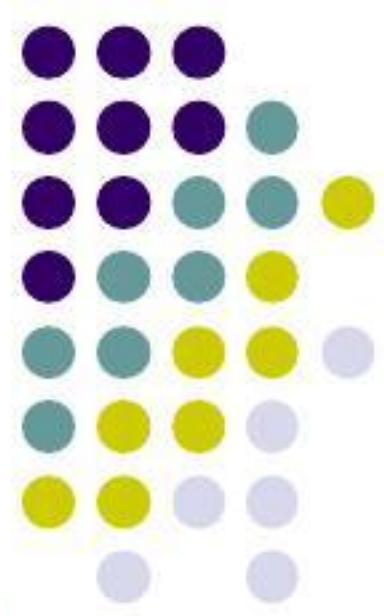


# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên dữ liệu thời gian

- Các ký hiệu thường dùng

Ký hiệu	Mô tả	Ví dụ
%d	Ngày trong tháng 01 – 31	31
%b	Tên tháng (ngắn)	Dec
%B	Tên tháng (đầy đủ)	December
%m	Tên tháng (bằng số)	11
%y	Năm (2 ký số)	23
%Y	Năm (4 ký số)	2023
%H	Giờ 00 – 23	17
%M	Phút 00 – 59	10
%S	Giây 00 – 59	32



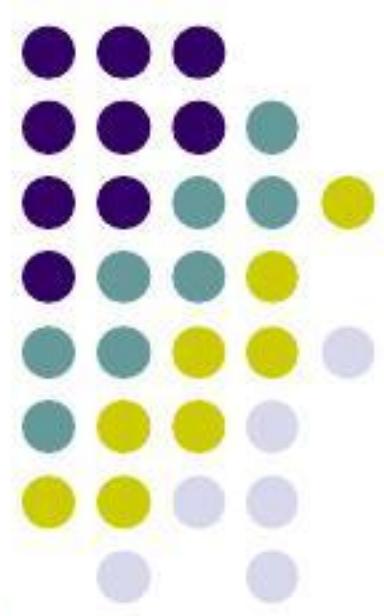
# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên dữ liệu thời gian

- Rút trích các thành phần của dữ liệu thời gian: year, month, day, isocalendar().week, dayofweek, day\_name(), kiểm tra năm nhuận is\_leap\_year

```
df[<col>].dt.func_name() hoặc  
df[<col>].dt.attributes
```

	HIRE_DATE	HIRE_YEAR	HIRE_MONTH	HIRE_DAY
0	2003-06-17	2003	6	17
1	2005-09-21	2005	9	21
2	2001-01-13	2001	1	13
3	2006-01-03	2006	1	3
4	2007-05-21	2007	5	21



# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên dữ liệu thời gian

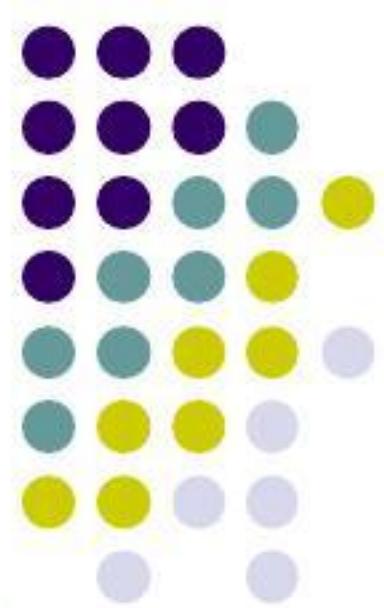
- Thay đổi định dạng hiển thị

```
df[<col>].dt.strftime(format)
```

```
# Thay đổi định dạng
ser = ser.dt.strftime("%d/%m/%Y")
ser
```

0	2016-10-16 20:30:00	0	16/10/2016
1	2016-01-27 19:45:30	1	27/01/2016
2	2013-12-10 04:05:01	2	10/12/2013

dtype: datetime64[ns]  dtype: object



# Cập nhật và xử lý dữ liệu

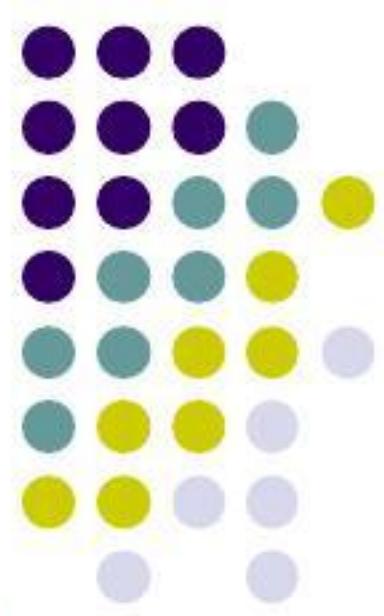
## □ Phương thức thao tác trên dữ liệu thời gian

- Khoảng cách giữa các mốc thời gian: timedelta()

```
from datetime import timedelta  
  
thu_vien['ngay_het_han'] = thu_vien['ngay_muon'] + timedelta(days = 10)
```

	ngay_muon	ngay_muon	ngay_het_han
0	2023-01-03	0	2023-01-13
1	2023-02-06	1	2023-02-16
2	2023-02-18	2	2023-02-28





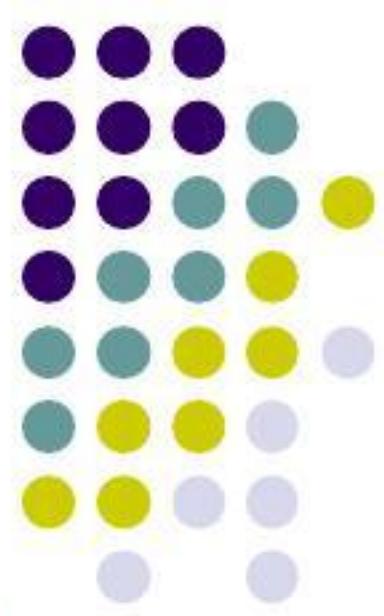
# Cập nhật và xử lý dữ liệu

## □ Phương thức thao tác trên dữ liệu thời gian

- Timestamps: Số giây trôi qua từ
  - 00:00:00
  - Giờ quốc tế - Coordinated Universal Time (UTC)
  - Thursday, 1 January 1970

```
ratings = pd.read_excel('data/movies.xlsx', 'ratings')
ratings.head()
```

	userId	movieId	rating	timestamp		
0	1	31	2.5	1260759144		
1	1	1029	3.0	1260759179	timestamp	int64
2	1	1061	3.0	1260759182		
3	1	1129	2.0	1260759185		
4	1	1172	4.0	1260759205		



# Cập nhật và xử lý dữ liệu

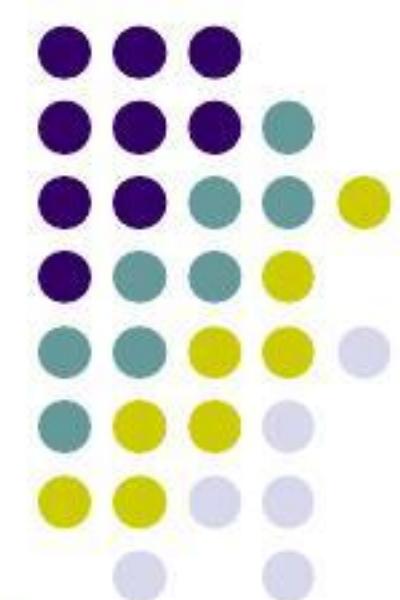
## □ Phương thức thao tác trên dữ liệu thời gian

- Chuyển Timestamps sang datetime

```
pd.to_datetime(df[<col>], unit)
```

```
ratings['parsed_time'] = pd.to_datetime(ratings['timestamp'], unit='s')
```

	userId	movieId	rating	timestamp	parsed_time	timestamp	parsed_time	int64
0	1	31	2.5	1260759144	2009-12-14 02:52:24			
1	1	1029	3.0	1260759179	2009-12-14 02:52:59			
2	1	1061	3.0	1260759182	2009-12-14 02:53:02			
3	1	1129	2.0	1260759185	2009-12-14 02:53:05			
4	1	1172	4.0	1260759205	2009-12-14 02:53:25			



# Cập nhật và xử lý dữ liệu

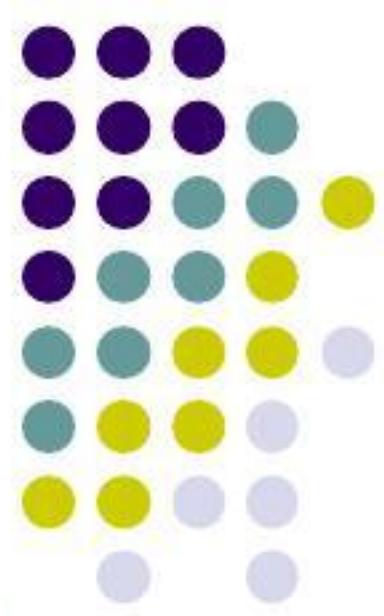
## □ Phương thức thao tác trên dữ liệu thời gian

- Lọc dữ liệu

# Những nhân viên vào làm trong 6 tháng cuối năm 2006

```
emp_df[(emp_df['HIRE_DATE'] >= '2006-07') & (emp_df['HIRE_DATE'] <= '2007')]
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE
18	118	Guy	Himuro	GHIMURO	515.127.4565
26	126	Irene	Mikkilineni	IMIKKILI	650.124.1224
34	134	Michael	Rogers	MROGERS	650.127.1834
44	144	Peter	Vargas	PVARGAS	650.121.2004
54	154	Nanette	Cambrault	NCAMBRAU	011.44.1344.987668
61	161	Sarath	Sewall	SSEWALL	011.44.1345.529268
90	190	Timothy	Gates	TGATES	650.505.3876
94	194	Samuel	McCain	SMCCAIN	650.501.3876



# Cập nhật và xử lý dữ liệu

## □ Gộp dữ liệu

```
pd.concat([df_1, df_2, ...])
```

- Gộp các dataframe có cột giống nhau

df\_1

	name	age
0	Tom	23
1	Alice	25
2	Bill	26

df\_2

	name	age
0	Jack	21
1	Mike	23

```
# mặc định axis=0  
df = pd.concat([df_1, df_2], ignore_index=True)  
df
```

	name	age
0	Tom	23
1	Alice	25
2	Bill	26
3	Jack	21
4	Mike	23



# Cập nhật và xử lý dữ liệu

## □ Gộp dữ liệu

```
pd.concat([df_1, df_2, ...])
```

- Gộp các dataframe có cột khác nhau

df\_1

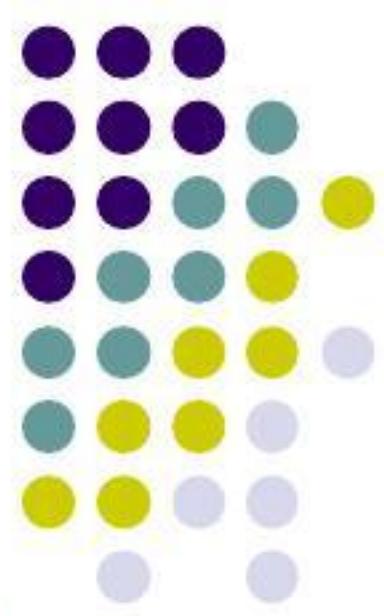
	name	age	salary
0	Tom	23	1000
1	Alice	25	1200
2	Bill	26	1500

df\_2

	name	age	sex
0	Jack	21	male
1	Mike	23	male

```
df = pd.concat([df_1, df_2], ignore_index=True)
```

	name	age	salary	sex
0	Tom	23	1000.0	NaN
1	Alice	25	1200.0	NaN
2	Bill	26	1500.0	NaN
3	Jack	21	NaN	male
4	Mike	23	NaN	male



# Cập nhật và xử lý dữ liệu

## □ Gộp dữ liệu

```
pd.concat([df_1, df_2, ...], axis=1)
```

- Gộp các dataframe theo cột

df\_1

```
name  age
```

	name	age
0	Tom	23
1	Alice	25
2	Bill	26

df\_2

```
sex  salary
```

	sex	salary
0	male	1000
1	female	1200

```
df = pd.concat([df_1, df_2], axis=1)
```

df

```
name  age   sex  salary
```

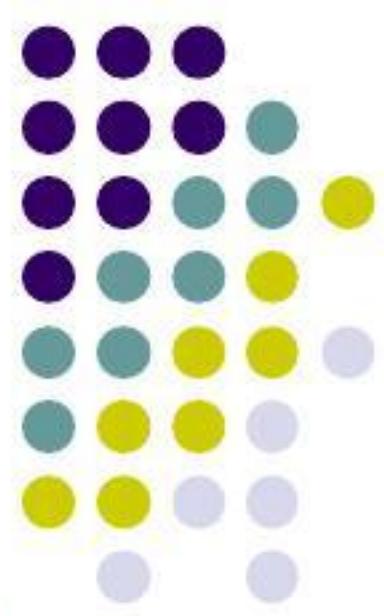
	name	age	sex	salary
0	Tom	23	male	1000.0
1	Alice	25	female	1200.0
2	Bill	26	NaN	NaN

```
df = pd.concat([df_1, df_2], axis=1, join='inner')
```

df

```
name  age   sex  salary
```

	name	age	sex	salary
0	Tom	23	male	1000
1	Alice	25	female	1200



# Cập nhật và xử lý dữ liệu

## □ Gộp dữ liệu

```
pandas.merge(df_1, df_2, [how], ...)
```

- Kết hợp 2 dataframe có cột chung

df\_1

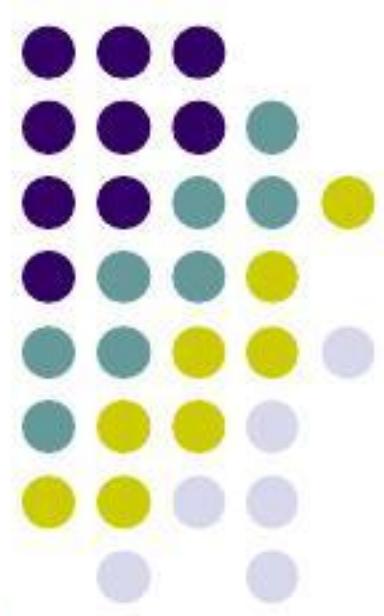
	name	age
0	Tom	23
1	Alice	25
2	Bill	26

df\_2

	name	sex	salary
0	Tom	male	1000
1	Alice	female	1200

```
df = pd.merge(df_1, df_2) # key: name  
df
```

	name	age	sex	salary
0	Tom	23	male	1000
1	Alice	25	female	1200



# Cập nhật và xử lý dữ liệu

## □ Gộp dữ liệu

```
pandas.merge(df_1, df_2, [how], ...)
```

- Kết hợp hai dataframe có tên cột khác nhau

df\_emp

	<b>id</b>	<b>name</b>	<b>department_id</b>
0	100	Mike	1.0
1	101	King	2.0
2	102	Billy	1.0
3	103	Jack	NaN

df\_dep

	<b>depid</b>	<b>depname</b>
0	1	IT
1	2	Sales
2	3	HR

```
pd.merge(df_emp, df_dep, left_on='department_id', right_on='depid')
```

	<b>id</b>	<b>name</b>	<b>department_id</b>	<b>depid</b>	<b>depname</b>
0	100	Mike	1.0	1	IT
1	102	Billy	1.0	1	IT
2	101	King	2.0	2	Sales



# Cập nhật và xử lý dữ liệu

## □ Gộp dữ liệu

```
pandas.merge(df_1, df_2, [how], ...)
```

- Kết hợp hai dataframe với Left join

```
df_dep
```

	depid	depname
0	1	IT
1	2	Sales
2	3	HR

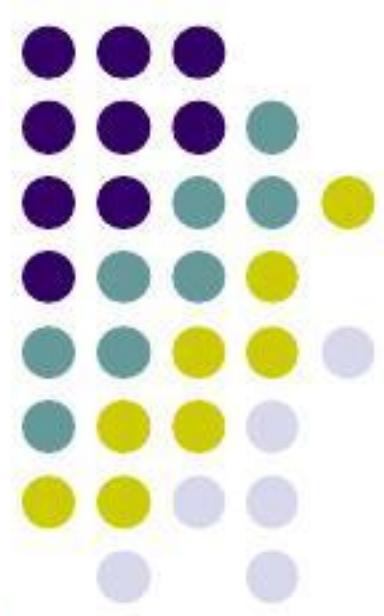
```
df_emp
```

	id	name	depid
0	100	Mike	1.0
1	101	King	2.0
2	102	Billy	1.0
3	103	Jack	NaN

```
df = pd.merge(df_emp, df_dep, how='left')  
df
```

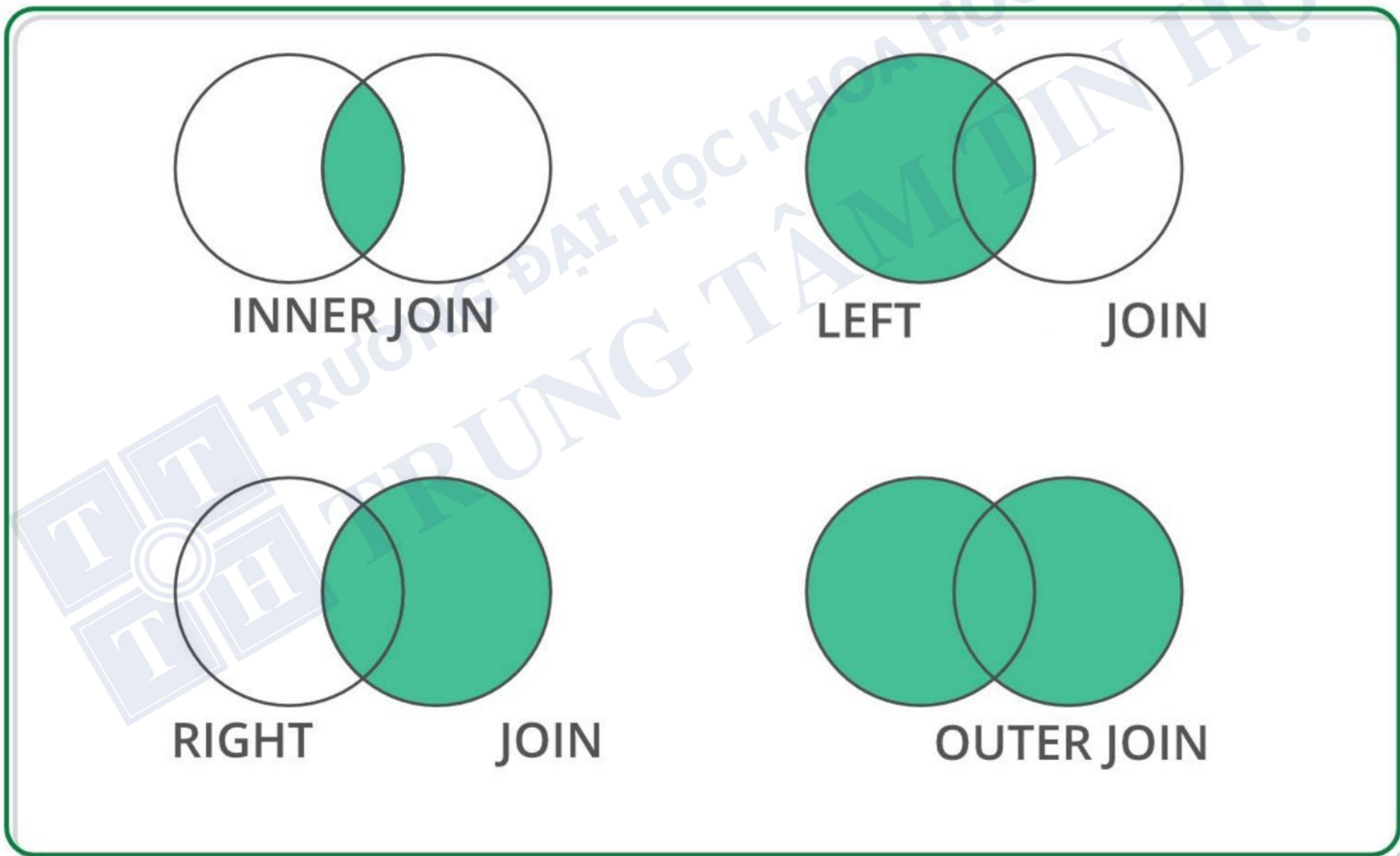
	id	name	depid	depname
0	100	Mike	1.0	IT
1	101	King	2.0	Sales
2	102	Billy	1.0	IT
3	103	Jack	NaN	NaN

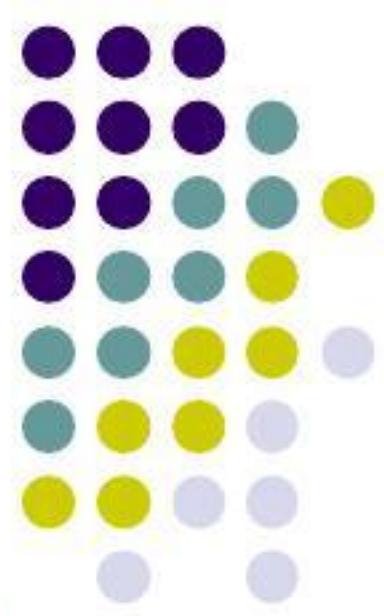
# Cập nhật và xử lý dữ liệu



## ☐ Gộp dữ liệu

```
pandas.merge(df_1, df_2, [how], ...)
```





# Cập nhật và xử lý dữ liệu

## □ Gộp dữ liệu

```
pandas.merge(df_1, df_2, [how], ...)
```

<b>id</b>	<b>size</b>
1	10
2	15
3	20
4	25
5	30
6	35
7	40

<b>id</b>	<b>color</b>
3	red
4	blue
5	green
6	yellow
7	cyan
8	magenta
9	purple

df\_1

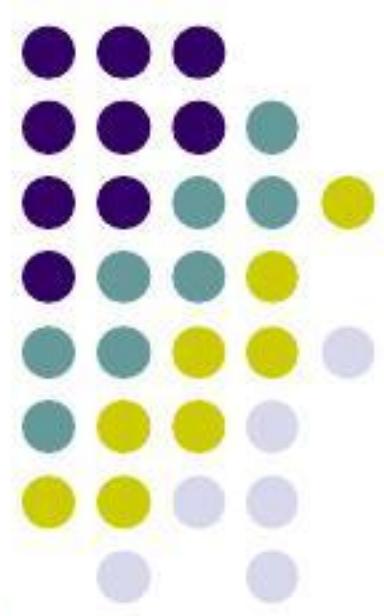
<b>id</b>	<b>size</b>	<b>color</b>
1	10.0	NaN
2	15.0	NaN
3	20.0	red
4	25.0	blue
5	30.0	green
6	35.0	yellow
7	40.0	cyan
8	NaN	magenta
9	NaN	purple

df\_2

outer

<b>id</b>	<b>size</b>	<b>color</b>
3	20	red
4	25	blue
5	30	green
6	35	yellow
7	40	cyan

inner



# Cập nhật và xử lý dữ liệu

## ☐ Gộp dữ liệu

```
pandas.merge(df_1, df_2, [how], ...)
```

<b>id</b>	<b>size</b>
1	10
2	15
3	20
4	25
5	30
6	35
7	40

<b>id</b>	<b>color</b>
3	red
4	blue
5	green
6	yellow
7	cyan
8	magenta
9	purple

df\_1

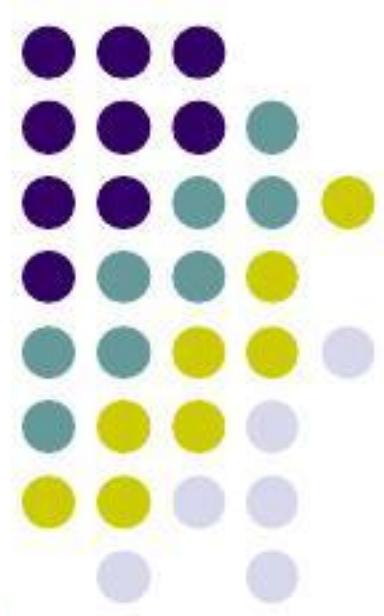
<b>id</b>	<b>size</b>	<b>color</b>
1	10	NaN
2	15	NaN
3	20	red
4	25	blue
5	30	green
6	35	yellow
7	40	cyan

df\_2

left

<b>id</b>	<b>size</b>	<b>color</b>
3	20.0	red
4	25.0	blue
5	30.0	green
6	35.0	yellow
7	40.0	cyan
8	NaN	magenta
9	NaN	purple

right



# Cập nhật và xử lý dữ liệu

## □ Gộp dữ liệu

- Kết hợp dữ liệu từ hai DataFrame

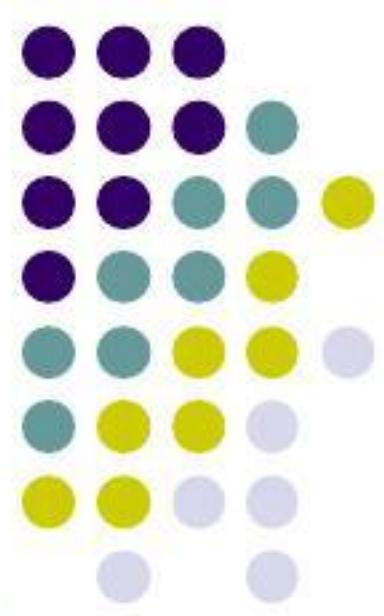
Cách kết hợp	Cú pháp	Diễn giải
Inner Join	<code>pd.merge(df_1, df_2, how='inner')</code>	Kết hợp dữ liệu có chung giá trị “key” trong 2 dataframe
Outer Join	<code>pd.merge(df_1, df_2, how='outer')</code>	Kết hợp dữ liệu 2 dataframe, tất cả giá trị “key” đều được đưa vào dataframe kết quả
Left Join	<code>pd.merge(df_1, df_2, how='left')</code>	Kết hợp dữ liệu 2 dataframe, ưu tiên tất cả giá trị “key” trong dataframe đầu tiên df_1.
Right Join	<code>pd.merge(df_1, df_2, how='right')</code>	Kết hợp dữ liệu 2 dataframe, ưu tiên tất cả giá trị “key” trong dataframe thứ hai, df_2.



# Nội dung

---

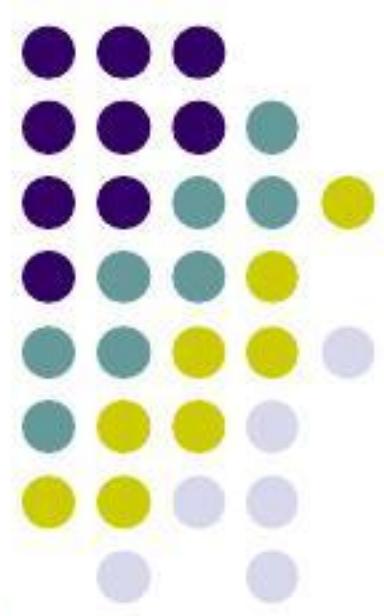
- Giới thiệu
- Series
- DataFrame
- Làm sạch dữ liệu
- Cập nhật và xử lý dữ liệu
- **Thống kê gom nhóm**
- **Trực quan hóa dữ liệu với Pandas**



# Thống kê gom nhóm

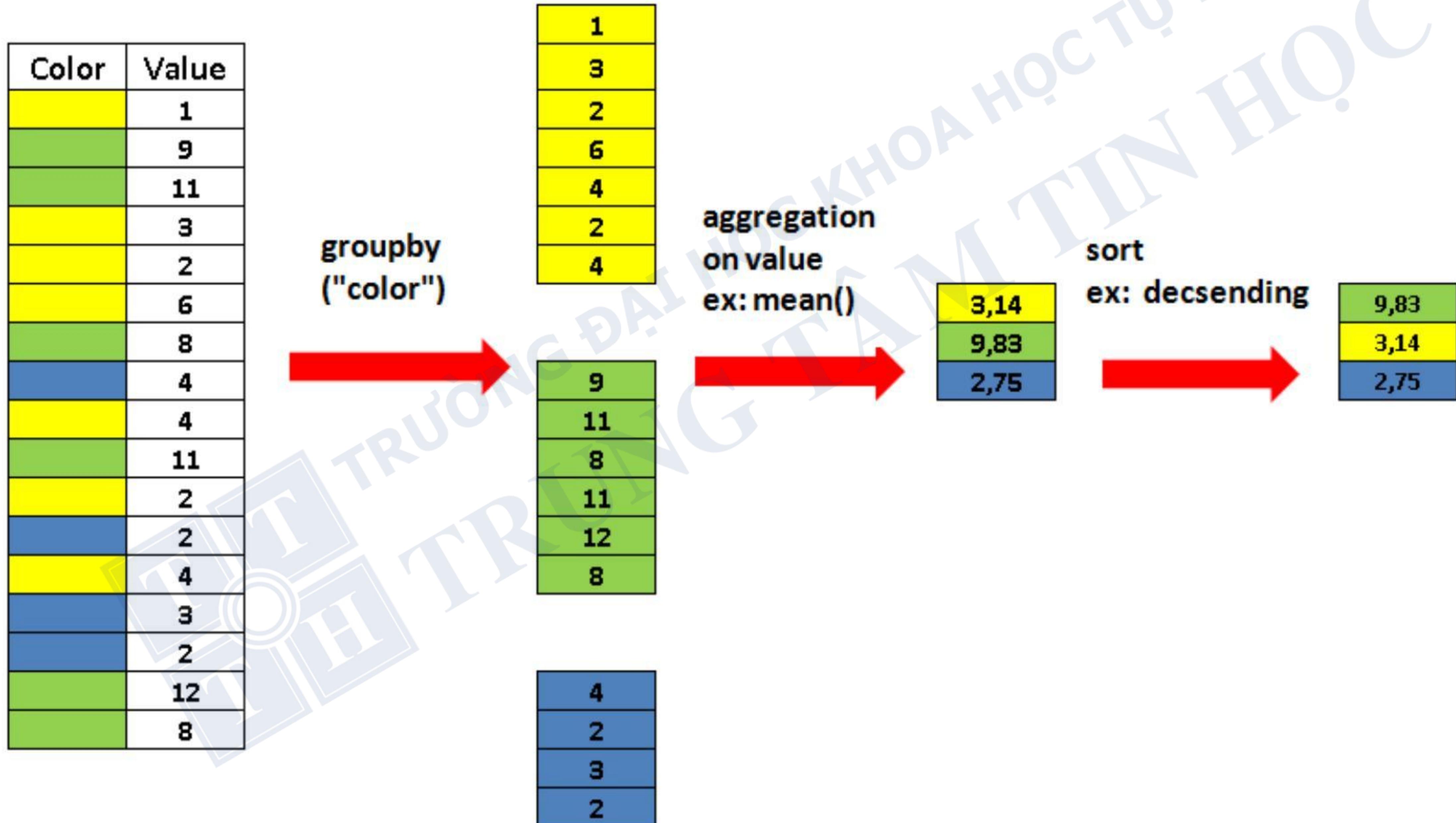
## □ Thống kê

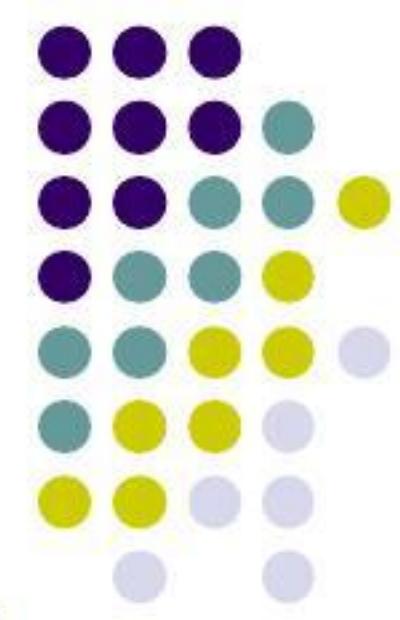
Method	Description
count ()	Số lượng các phần tử khác null của mỗi cột
sum ()	Trả về tổng các phần tử của mỗi cột
mean ()	Trả về giá trị trung bình của các phần tử mỗi cột số
median ()	Trả về giá trị ở vị trí giữa của các phần tử mỗi cột số
min () , max ()	Giá trị nhỏ nhất, lớn nhất của các phần tử mỗi cột
var () , std ()	Phương sai, Độ lệch chuẩn của các phần tử cột số
quantile ()	Phân vị của các phần tử mỗi cột số
sort_values (<col>)	Sắp xếp các dòng theo thứ tự tăng dần/giảm dần của các phần tử trong cột
value_counts (<col>)	Đếm số dòng theo giá trị của các phần tử trong cột



# Thống kê – gom nhóm

## □ Group by



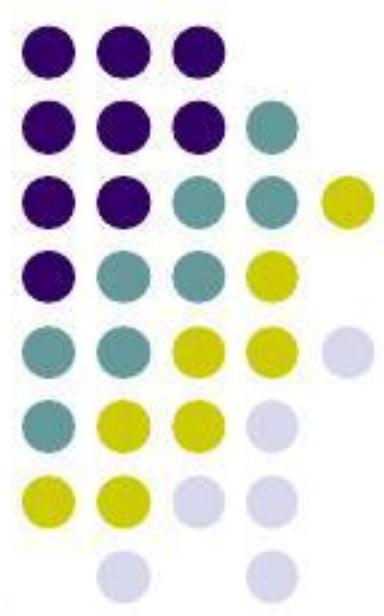


# Thống kê – gom nhóm

## □ Group by

```
df.groupby(<col(s)_group>) [<col(s)_stat>].func_name()  
emp_df.groupby('DEPARTMENT_NAME').mean(numeric_only=True)
```

DEPARTMENT_NAME	EMPLOYEE_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	HIRE_YEAR
Accounting	205.500000	10154.000000	0.000000	153.000000	110.0	2002.000000
Administration	200.000000	4400.000000	0.000000	101.000000	10.0	2003.000000
Executive	101.000000	19333.333333	0.000000	66.666667	90.0	2003.000000
Finance	110.500000	8601.333333	0.000000	106.833333	100.0	2004.500000
Human Resources	203.000000	6500.000000	0.000000	101.000000	40.0	2002.000000
IT	105.000000	5760.000000	0.000000	102.800000	60.0	2006.200000
Marketing	201.500000	9500.000000	0.000000	150.500000	20.0	2004.500000
Public Relations	204.000000	10000.000000	0.000000	101.000000	70.0	2002.000000
Purchasing	116.500000	4150.000000	0.000000	111.666667	30.0	2004.666667
Sales	162.000000	8900.000000	0.222857	140.285714	80.0	2006.000000
Shipping	157.555556	3475.555556	0.000000	119.555556	50.0	2005.733333

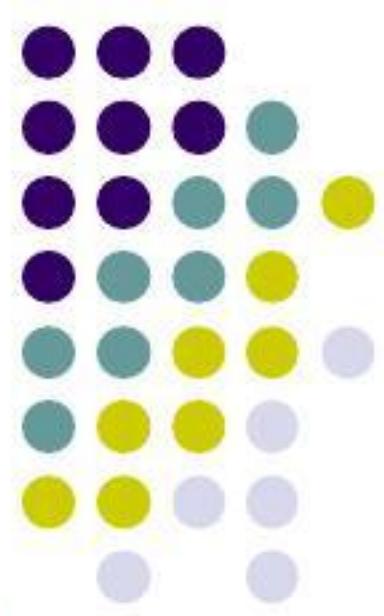


# Thống kê – gom nhóm

## □ Group by

```
# Lương nhân viên trung bình theo năm vào làm  
emp_df.groupby('HIRE_YEAR')['SALARY'].mean()
```

```
HIRE_YEAR  
2001    17000.000000  
2002    9830.857143  
2003    7750.000000  
2004    8600.000000  
2005    6824.137931  
2006    5045.833333  
2007    4994.736842  
2008    5381.818182  
Name: SALARY, dtype: float64
```

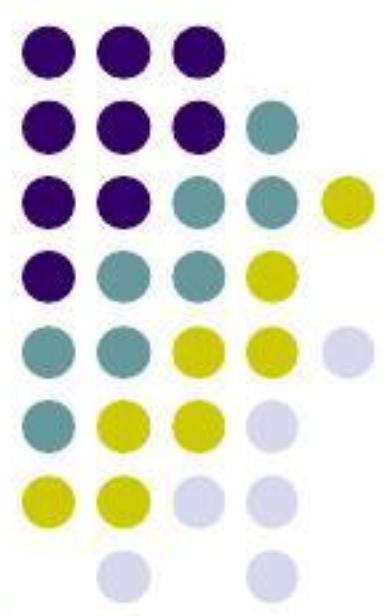


# Thống kê – gom nhóm

## □ Group by

```
# tính số Lượng NV, Lương trung bình, trung vị, Lớn nhất, nhỏ nhất theo phòng ban  
emp_df.groupby('DEPARTMENT_NAME')['SALARY'].agg(['count','min','max','mean','median'])
```

DEPARTMENT_NAME	count	min	max	mean	median
Accounting	2	8300	12008	10154.000000	10154.0
Administration	1	4400	4400	4400.000000	4400.0
Executive	3	17000	24000	19333.333333	17000.0
Finance	6	6900	12008	8601.333333	8000.0
Human Resources	1	6500	6500	6500.000000	6500.0
IT	5	4200	9000	5760.000000	4800.0
Marketing	2	6000	13000	9500.000000	9500.0
Public Relations	1	10000	10000	10000.000000	10000.0
Purchasing	6	2500	11000	4150.000000	2850.0
Sales	35	6100	14000	8900.000000	8800.0
Shipping	45	2100	8200	3475.555556	3100.0

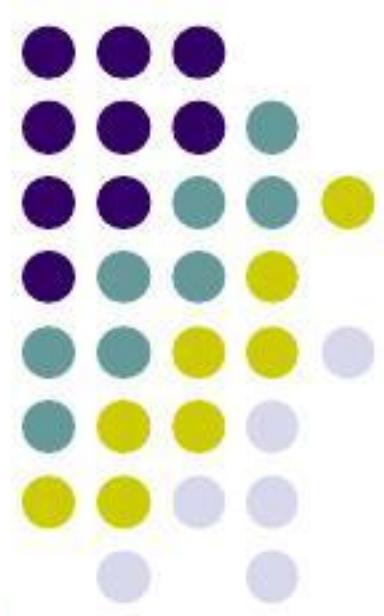


# Thống kê – gom nhóm

Area	Sales Person	Target
APAC	Lisa	1500
APAC	James	1750
APAC	Sharon	1850
EMEA	Lisa	1350
EMEA	James	950
EMEA	Sharon	2050
SA	Lisa	1800
SA	James	1200
SA	Sharon	1350



Area	Lisa	James	Sharon
APAC	1500	1750	1850
EMEA	1350	950	2050
SA	1800	1200	1350

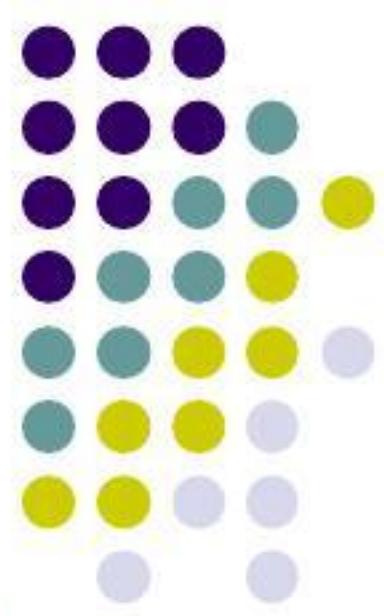


# Thống kê – gom nhóm

## □ Thống kê dùng crosstab

```
pandas.crosstab(index, columns, values, aggfunc)
```

- index: Chọn cột để nhóm dữ liệu theo dòng
- columns: Chọn cột để nhóm dữ liệu theo cột
- values: Chọn cột muốn tính toán, thống kê
- aggfunc: Chọn hàm thống kê



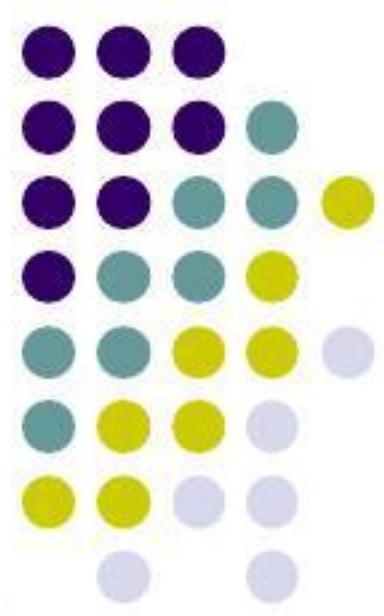
# Thống kê – gom nhóm

## □ Cross Tab

- Thống kê lương trung bình của nhân viên theo năm vào làm và phòng ban

```
pd.crosstab(index=emp_df['HIRE_YEAR'], columns=emp_df['DEPARTMENT_NAME'],  
values=emp_df['SALARY'], aggfunc='mean')
```

DEPARTMENT_NAME	Accounting	Administration	Executive	Finance	Human Resources	IT	Marketing	Public Relations	Purchasing	Sales	Shipping
HIRE_YEAR											
2001	NaN	NaN	17000.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2002	10154.0	NaN	NaN	10504.0	6500.0	NaN	NaN	10000.0	11000.0	NaN	NaN
2003	NaN	4400.0	24000.0	NaN	NaN	NaN	NaN	NaN	3100.0	NaN	5000.0
2004	NaN	NaN	NaN	NaN	NaN	NaN	NaN	13000.0	NaN	NaN	10700.0
2005	NaN	NaN	17000.0	7950.0	NaN	4800.0	6000.0	NaN	2850.0	10030.0	4017.0
2006	NaN	NaN	NaN	7800.0	NaN	6900.0	NaN	NaN	2600.0	8443.0	2908.0
2007	NaN	NaN	NaN	6900.0	NaN	5100.0	NaN	NaN	2500.0	8200.0	2900.0
2008	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	7057.0	2450.0



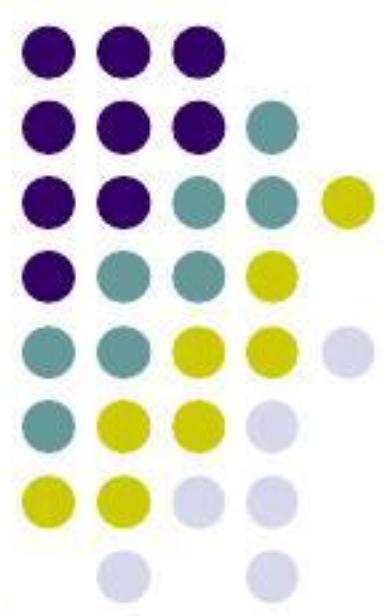
# Thống kê – gom nhóm

## □ Cross Tab

- Thống kê số lượng nhân viên theo năm vào làm và phòng ban

```
pd.crosstab(index=emp_df['HIRE_YEAR'], columns=emp_df['DEPARTMENT_NAME'], margins=True)
```

DEPARTMENT_NAME	Accounting	Administration	Executive	Finance	Human Resources	IT	Marketing	Public Relations	Purchasing	Sales	Shipping	All
HIRE_YEAR												
2001	0	0	1	0		0 0	0	0	0	0	0	1
2002	2	0	0	2		1 0	0	1	1	0	0	7
2003	0	1	1	0		0 0	0	0	1	0	3	6
2004	0	0	0	0		0 0	1	0	0	5	4	10
2005	0	0	1	2		0 1	1	0	2	10	12	29
2006	0	0	0	1		0 2	0	0	1	7	13	24
2007	0	0	0	1		0 2	0	0	1	6	9	19
2008	0	0	0	0		0 0	0	0	0	7	4	11
All	2	1	3	6		1 5	2	1	6	35	45	107

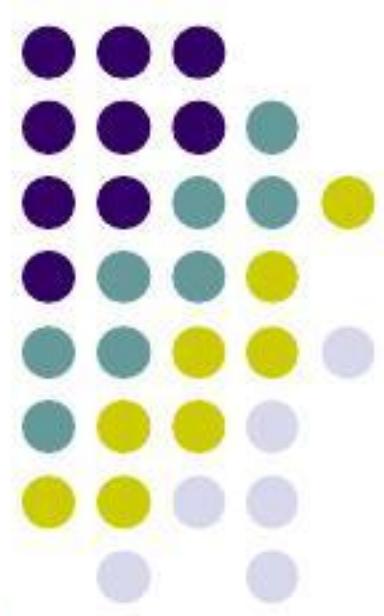


# Thống kê – gom nhóm

## □ Thống kê dùng Pivot Table

```
pivot_table(data, index, columns, values, aggfunc)
```

- data: Dataframe cần nhóm dữ liệu
- index: Nhãn cột để nhóm dữ liệu theo dòng
- columns: Nhãn cột để nhóm dữ liệu theo cột
- values: Nhãn cột muốn tính toán, thống kê
- aggfunc: Chọn hàm thống kê



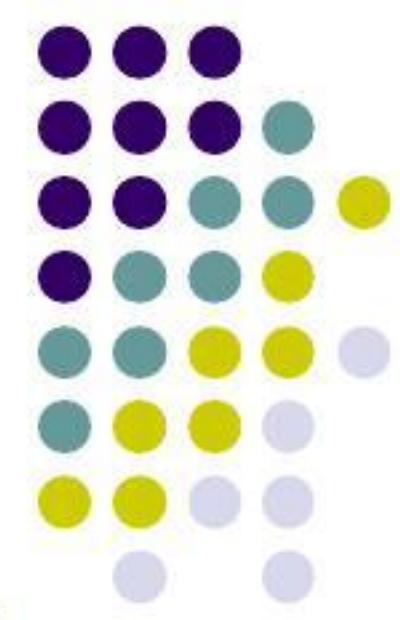
# Thống kê – gom nhóm

## □ Thống kê dùng Pivot Table

- Tổng lương của nhân viên theo năm vào làm và mã phòng ban

```
pd.pivot_table(data=emp_df, index='DEPARTMENT_NAME', columns='HIRE_YEAR',  
               values='SALARY', aggfunc='sum')
```

	HIRE_YEAR	2001	2002	2003	2004	2005	2006	2007	2008
DEPARTMENT_NAME									
Accounting		NaN	20308.0	NaN	NaN	NaN	NaN	NaN	NaN
Administration		NaN	NaN	4400.0	NaN	NaN	NaN	NaN	NaN
Executive		17000.0	NaN	24000.0	NaN	17000.0	NaN	NaN	NaN
Finance		NaN	21008.0	NaN	NaN	15900.0	7800.0	6900.0	NaN
Human Resources		NaN	6500.0	NaN	NaN	NaN	NaN	NaN	NaN
IT		NaN	NaN	NaN	NaN	4800.0	13800.0	10200.0	NaN
Marketing		NaN	NaN	NaN	13000.0	6000.0	NaN	NaN	NaN
Public Relations		NaN	10000.0	NaN	NaN	NaN	NaN	NaN	NaN
Purchasing		NaN	11000.0	3100.0	NaN	5700.0	2600.0	2500.0	NaN
Sales		NaN	NaN	NaN	53500.0	100300.0	59100.0	49200.0	49400.0
Shipping		NaN	NaN	15000.0	19500.0	48200.0	37800.0	26100.0	9800.0



# Thống kê – gom nhóm

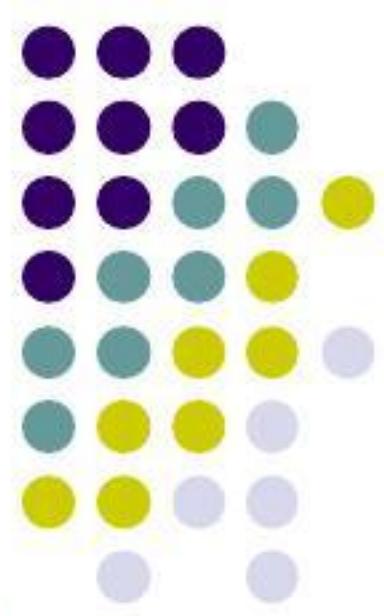
## □ Melting data (unpivoting data)

### Pandas Melt

Turn your DataFrame From Wide...

Index	Name	8/6/2020	8/7/2020
0	Liho Liho	\$234.54	\$45.32
1	Chambers	\$45.74	\$65.33
2	The Square	\$56.22	\$12.45
3	Tosca Cafe	\$32.31	\$180.34

Index	Name	Date	Bill
0	Liho Liho	8/6/2020	\$234.54
1	Liho Liho	8/7/2020	\$45.32
2	Chambers	8/6/2020	\$45.74
3	Chambers	8/7/2020	\$65.33
6	The Square	8/6/2020	\$56.22
7	The Square	8/7/2020	\$12.45
8	Tosca Cafe	8/6/2020	\$32.31
9	Tosca Cafe	8/7/2020	\$180.34

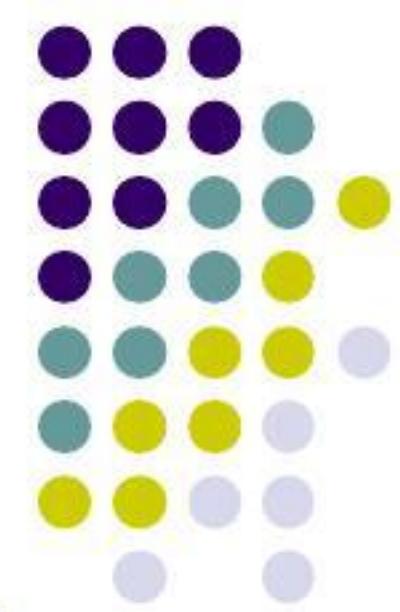


# Thống kê – gom nhóm

## □ Melting data (unpivoting data)

```
pd.melt(frame, id_vars [, value_vars] [, var_name]  
 [, value_name])
```

- pd.melt(frame, id\_vars, value\_vars, var\_name, value\_name)
  - frame: dataframe cần biến đổi
  - id\_vars: danh sách các cột muốn giữ nguyên
  - value\_vars: danh sách các cột muốn biến đổi từ cột thành dòng
  - var\_name: đặt tên của cột trong DataFrame mới để chứa tên các cột trong value\_vars
  - value\_name: đặt tên của cột trong DataFrame mới để chứa giá trị tương ứng với các cột trong value\_vars



# Thống kê – gom nhóm

## □ Melting data

```
pd.melt(pivot, id_vars=['DEPARTMENT_NAME'], var_name='HIRE_YEAR', value_name='mean_SALARY')
```

	DEPARTMENT_NAME	HIRE_YEAR	mean_SALARY
2	Executive	2001	17000.0
11	Accounting	2002	10154.0
14	Finance	2002	10504.0
15	Human Resources	2002	6500.0
18	Public Relations	2002	10000.0
19	Purchasing	2002	11000.0
23	Administration	2003	4400.0
24	Executive	2003	24000.0
30	Purchasing	2003	3100.0
32	Shipping	2003	5000.0
39	Marketing	2004	13000.0
42	Sales	2004	10700.0
43	Shipping	2004	4875.0



# Nội dung

---

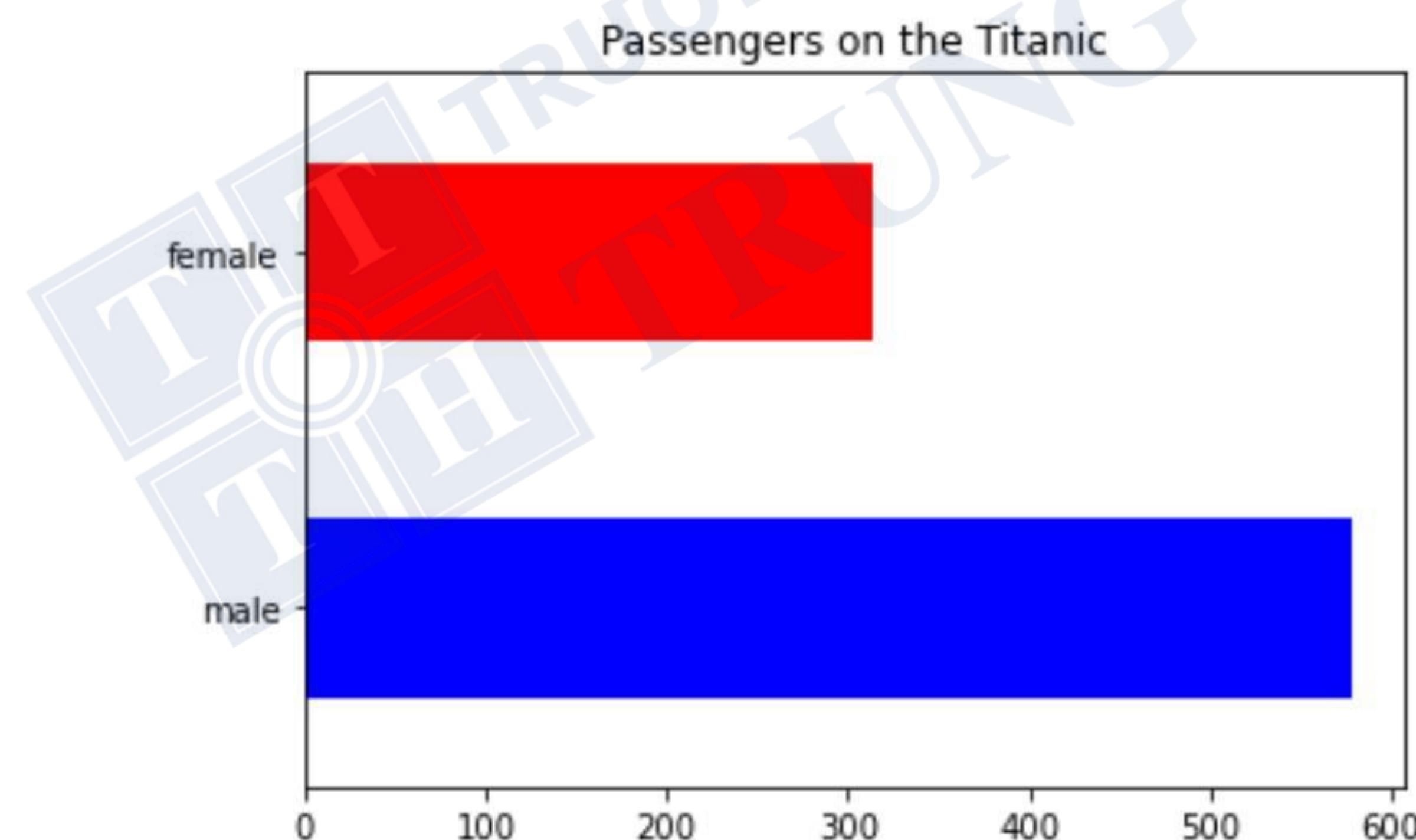
- Giới thiệu
- Series
- DataFrame
- Làm sạch dữ liệu
- Cập nhật và xử lý dữ liệu
- Thống kê gom nhóm
- **Trực quan hóa dữ liệu với Pandas**

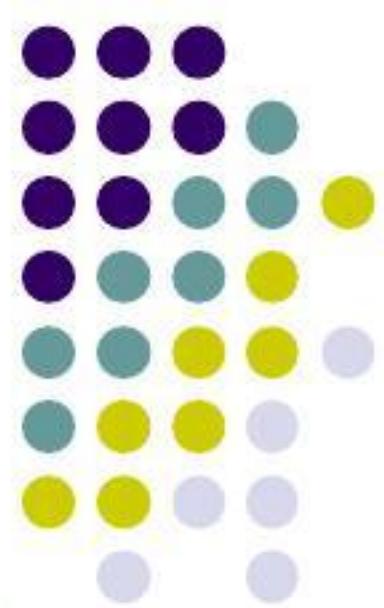


# Trực quan hóa dữ liệu

## □ Biểu đồ Bar plot

- Là dạng biểu đồ biểu diễn dữ liệu dưới dạng các khối chữ nhật, giúp so sánh giá trị tính toán giữa các biến phân loại khác nhau
- Một trục là biến phân loại, một trục là giá trị tính toán của biến phân loại tương ứng.
- Tạo barplot với Data frame columns: `df.plot.bar([x] [, y])`

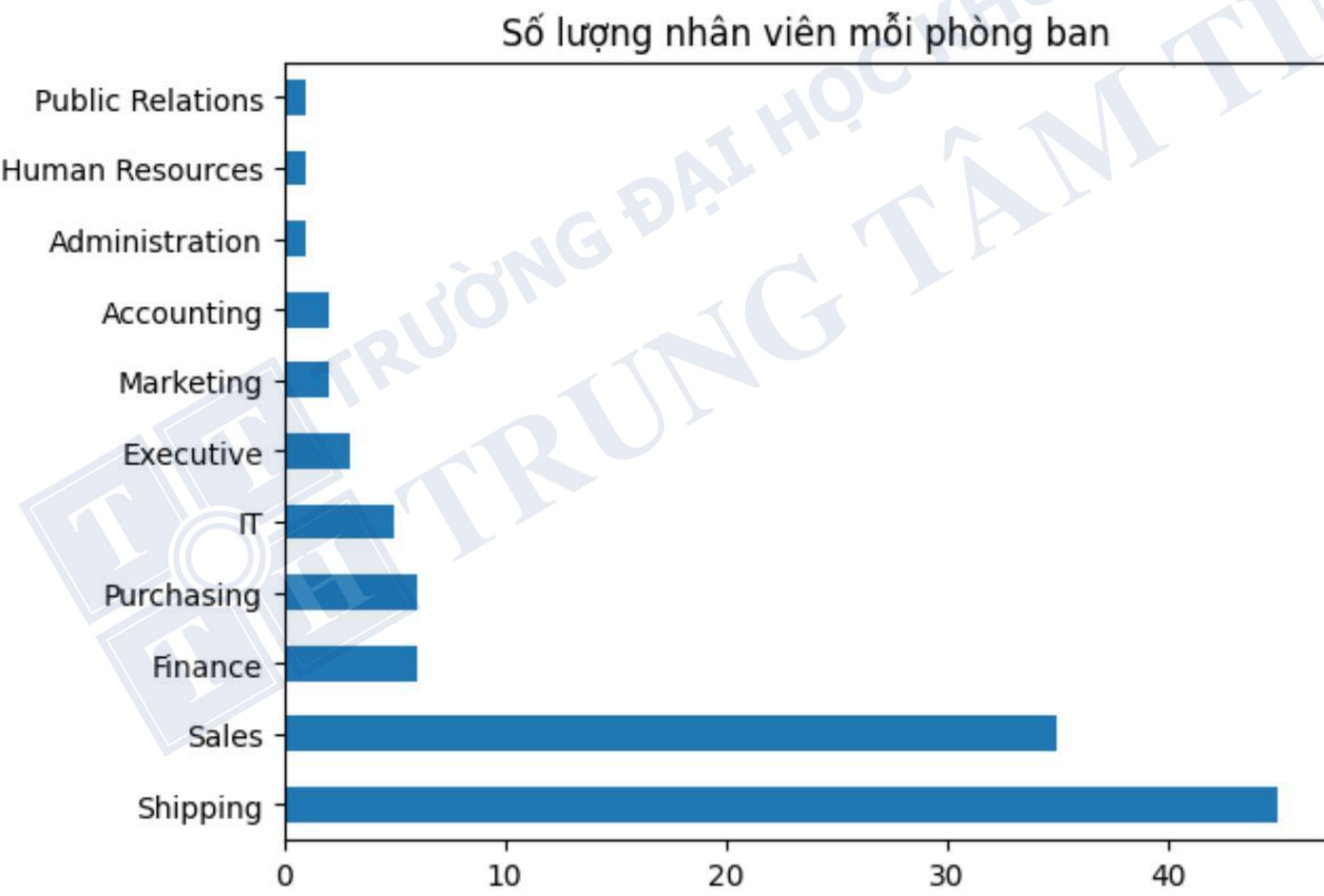


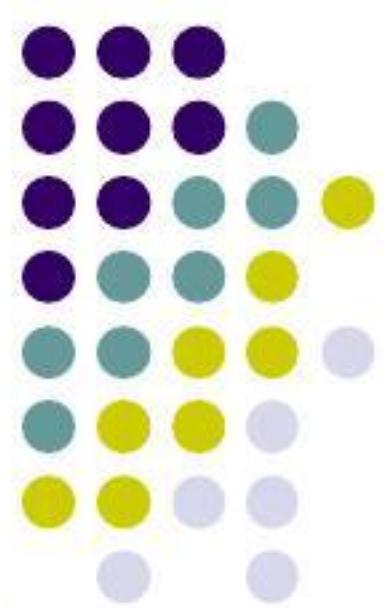


# Trực quan hóa dữ liệu

## □ Biểu đồ Bar plot

```
# đếm số NV theo phòng ban
cnt_ser = emp_df['DEPARTMENT_NAME'].value_counts()
cnt_ser.plot.bah(title = 'Số lượng nhân viên mỗi phòng ban');
```

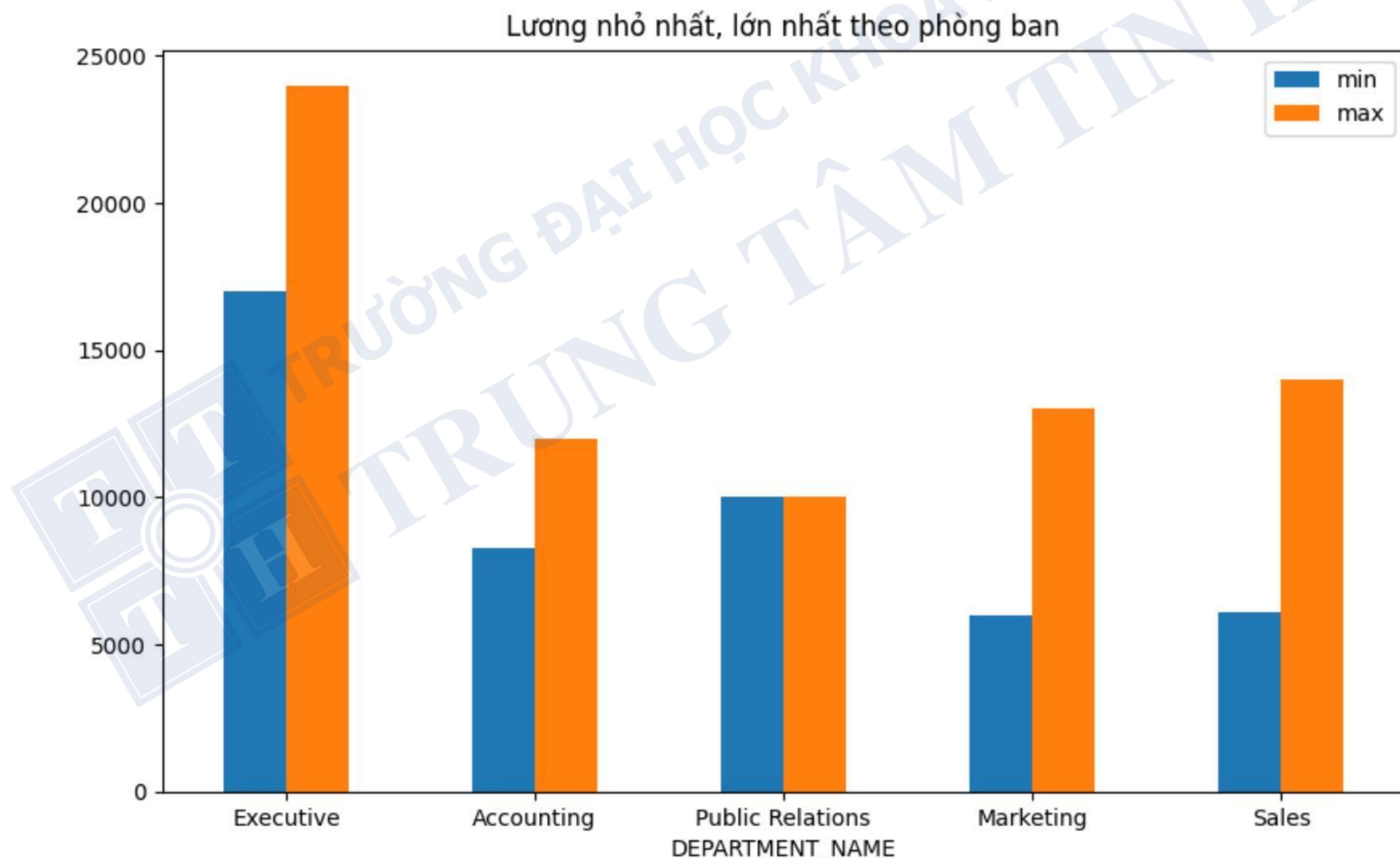




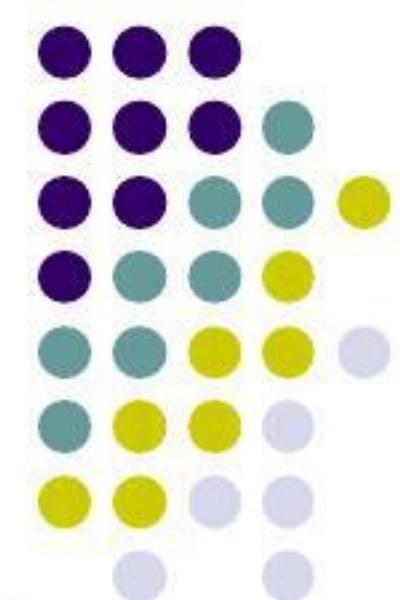
# Trực quan hóa dữ liệu

## □ Biểu đồ Bar plot

```
# Vẽ biểu đồ Lương nhỏ nhất, lớn nhất của 5 phòng ban có Lương trung bình cao nhất
salary_df[['min', 'max']].plot.bar(figsize = (10,6), rot = 0,
                                    title = 'Lương nhỏ nhất, lớn nhất theo phòng ban');
```

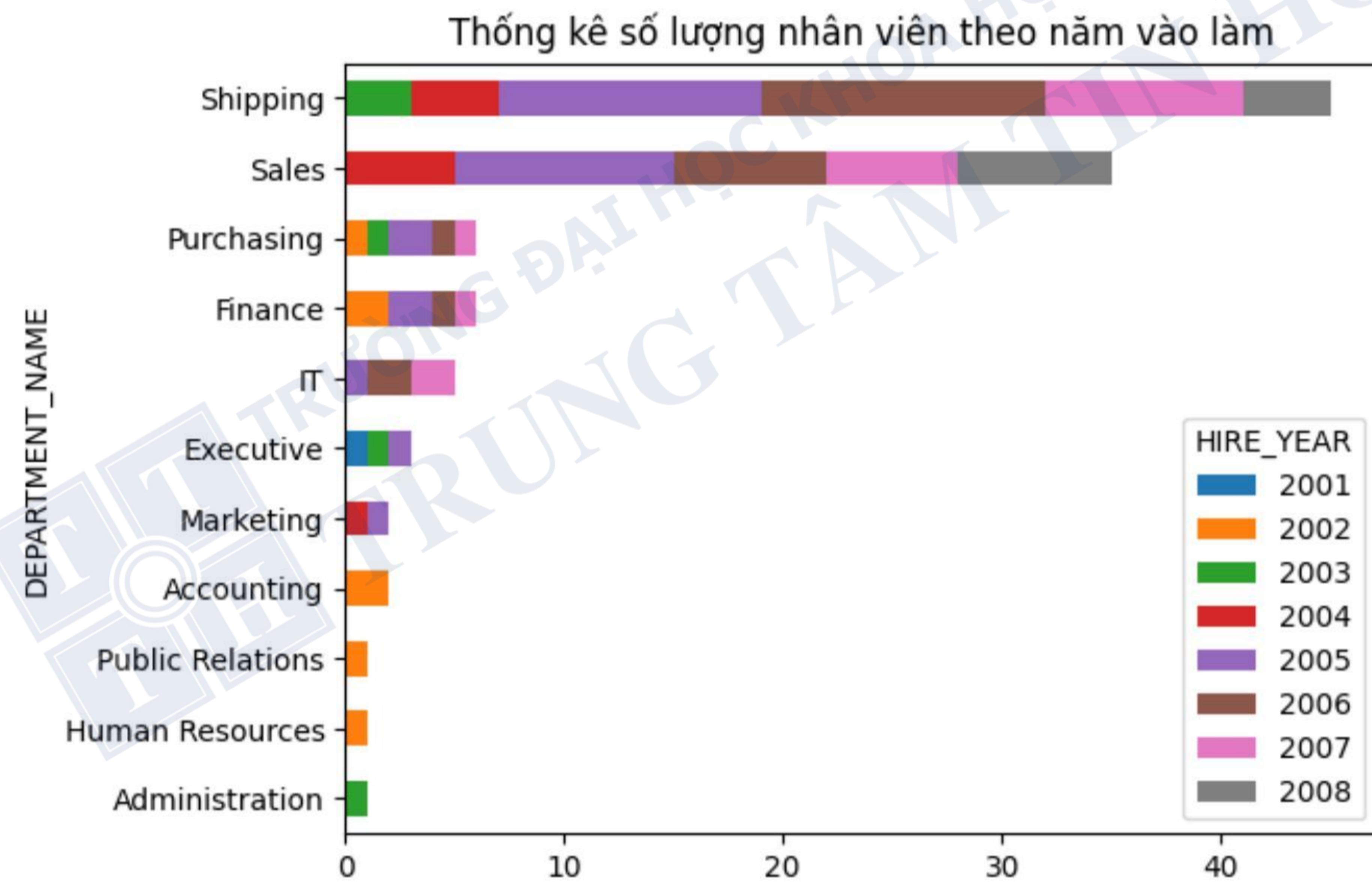


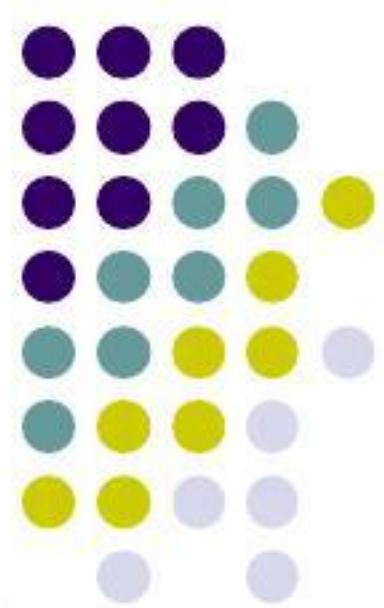
# Trực quan hóa dữ liệu



- Stacked bar chart

```
df_cnt = pd.crosstab(index=emp_df['DEPARTMENT_NAME'], columns=emp_df['HIRE_YEAR'], margins=True)
df_cnt = df_cnt.sort_values('All').iloc[:-1, :-1]
df_cnt.plot.barh(stacked = True, title = 'Thống kê số lượng nhân viên theo năm vào làm');
```

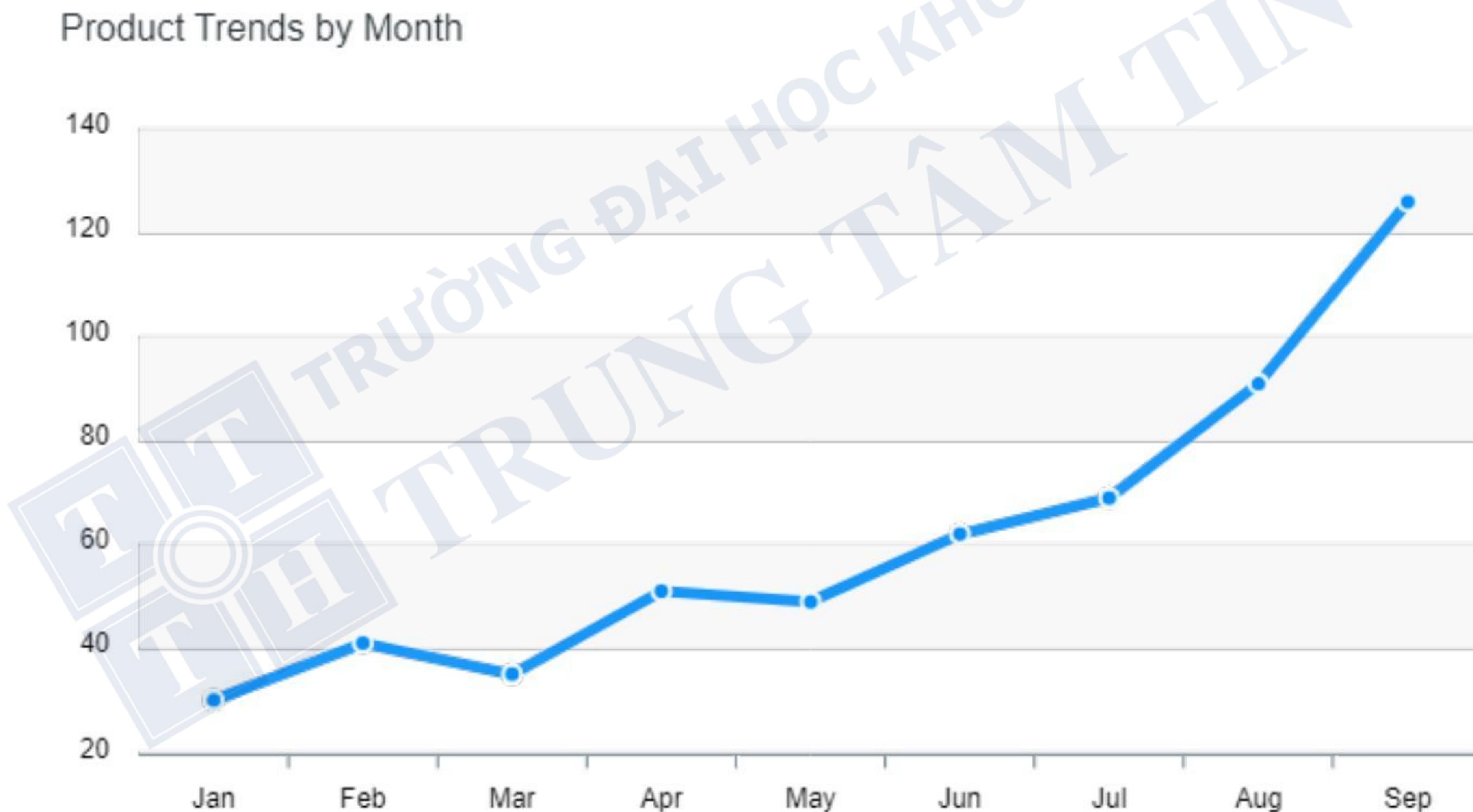


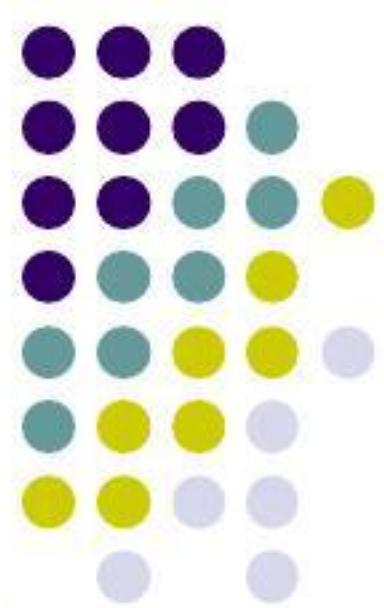


# Trực quan hóa dữ liệu

## □ Biểu đồ Line plot

- Thường được dùng để phân tích xu hướng của dữ liệu theo thời gian
- Tạo plot của Data frame columns: `df.plot([x] [, y])`





# Trực quan hóa dữ liệu

## □ Biểu đồ Line plot

```
df_cnt.plot(x = 'HIRE_YEAR', legend = False,  
            title = 'Số lượng nhân viên vào làm qua các năm');
```

