

Bài 7:

LẬP TRÌNH PYTHON CƠ BẢN

(Phân tích và xử lý dữ liệu với Pandas - 02)

AI Academy Vietnam

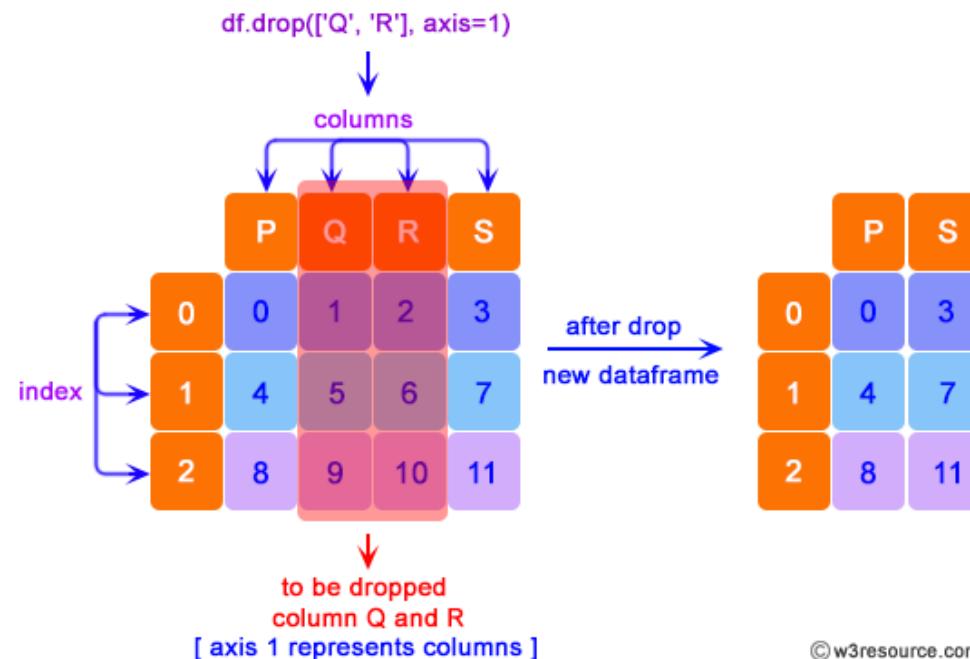
1. Xóa cột/hàng trong Dataframe
2. Xử lý hàng trùng lặp (Duplicate rows)
3. Sắp xếp trong DataFrame (Sort)
4. Nhóm các hàng trong DataFrame dựa vào giá trị
5. Áp dụng hàm cho các phần tử trong DataFrame (Apply)
6. Trộn các DataFrame (Merging)
7. Ghép nối các DataFrame (Concatenating)
8. Phát hiện và xử lý giá trị khuyết thiếu (Missing)
9. Phát hiện và xử lý giá trị ngoại biên (Outliers)
10. Phân tích dữ liệu bán hàng (Tiếp cận từ bài toán thực tế)



1. Xóa cột/hàng trong DataFrame

1.1 Xóa cột

- `df.drop([column_name], axis=1 | 'columns', inplace=True | Flase)`: Để xóa 1 cột hoặc nhiều cột trong một DataFrame.
- **Lưu ý khi sử dụng tham số `inplace = True` → Áp dụng thay đổi cho chính DataFrame hiện tại**



```

1 #Xử dụng .drop(axis=1/columns) để xóa cột
2 #Xóa một số cột trong df_loan
3 df_loan1 = df_loan.drop(['annual_inc',
4                           'dti','delinq_2yrs',
5                           'revol_util',
6                           'longest_credit_length',
7                           'verification_status'], axis=1)
8 df_loan1.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 163987 entries, 0 to 163986
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   loan_amnt        163987 non-null  int64  
 1   term              163987 non-null  object 
 2   int_rate          163987 non-null  float64 
 3   emp_length        158183 non-null  float64 
 4   home_ownership    163987 non-null  object 
 5   purpose            163987 non-null  object 
 6   addr_state         163987 non-null  object 
 7   total_acc          163958 non-null  float64 
 8   bad_loan           163987 non-null  int64  
dtypes: float64(3), int64(2), object(4)
memory usage: 11.3+ MB

```

1.1 Xóa cột

- df.drop(df.columns[index], axis=1 | columns): Để xóa 1 cột hoặc nhiều cột trong một DataFrame trong trường hợp DataFrame không có tên cột, sử dụng chỉ số cột.**

```

1 #Xóa cột trong một DataFrame sử dụng chỉ số cột
2 df_loan2 = df_loan.drop(df_loan.columns[[5,8,9,10,13,14]],
3                         axis='columns')
4 df_loan2.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 163987 entries, 0 to 163986
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   loan_amnt        163987 non-null   int64  
 1   term              163987 non-null   object  
 2   int_rate          163987 non-null   float64 
 3   emp_length        158183 non-null   float64 
 4   home_ownership    163987 non-null   object  
 5   purpose            163987 non-null   object  
 6   addr_state         163987 non-null   object  
 7   total_acc          163958 non-null   float64 
 8   bad_loan           163987 non-null   int64  
dtypes: float64(3), int64(2), object(4)
memory usage: 11.3+ MB

```

1.2 Xóa hàng

Xóa hàng theo index:

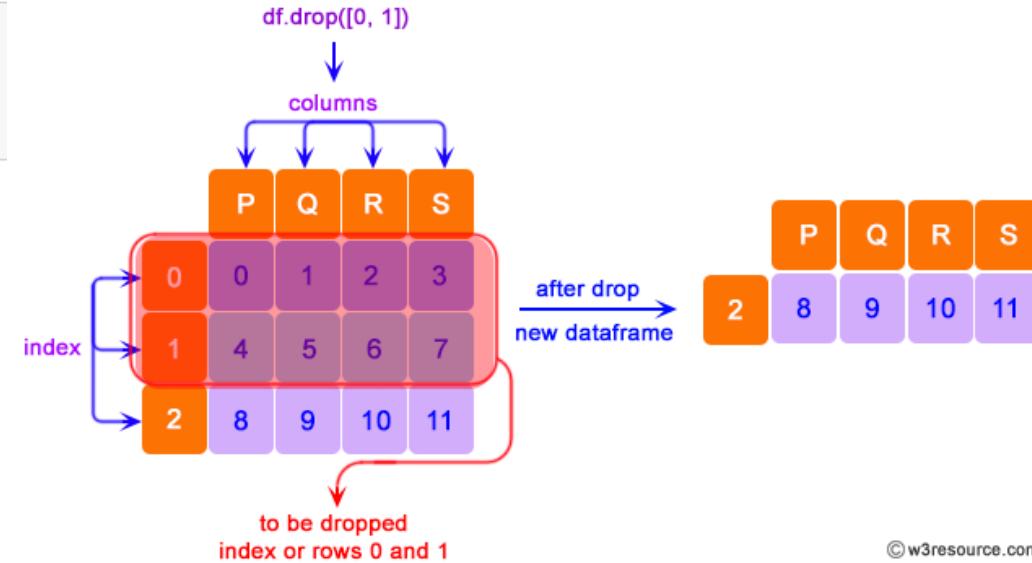
- **df.drop([index rows], axis=0):** Để xóa 1 hàng hoặc nhiều hàng trong một DataFrame theo index của hàng.
- Tham số axis = 0 (Default)

```

1 #Xóa hàng trong một DataFrame
2 #Xóa hàng có index: 3,9
3 df_loan2.drop([3,9], inplace=True)
4 df_loan2.head(10)

```

	loan_amnt	term	int_rate	emp_length	home_ownership	purpose	addr_state	total_acc	bad_loan
0	5000	36 months	10.65	10.0	RENT	credit_card	AZ	9.0	0
1	2500	60 months	15.27	0.0	RENT	car	GA	4.0	1
2	2400	36 months	15.96	10.0	RENT	small_business	IL	10.0	0
4	5000	36 months	7.90	3.0	RENT	wedding	AZ	12.0	0
5	3000	36 months	18.64	9.0	RENT	car	CA	4.0	0
6	5600	60 months	21.28	4.0	OWN	small_business	CA	13.0	1
7	5375	60 months	12.69	0.0	RENT	other	TX	3.0	1
8	6500	60 months	14.65	5.0	OWN	debt_consolidation	AZ	23.0	0
10	9000	36 months	13.49	0.0	RENT	debt_consolidation	VA	9.0	1
11	3000	36 months	9.91	3.0	RENT	credit_card	IL	11.0	0



©w3resource.com

1.2 Xóa hàng

Xóa hàng theo điều kiện (filter data):

```

1 #Loại bỏ tất cả các dòng dữ liệu có addr_state = CA
2 df_loan3 = df_loan1[df_loan1.addr_state != 'CA']
3 df_loan3

```

	loan_amnt	term	int_rate	emp_length	home_ownership	purpose	addr_state	total_acc	bad_loan
0	5000	36 months	10.65	10.0	RENT	credit_card	AZ	9.0	0
1	2500	60 months	15.27	0.0	RENT	car	GA	4.0	1
2	2400	36 months	15.96	10.0	RENT	small_business	IL	10.0	0
4	5000	36 months	7.90	3.0	RENT	wedding	AZ	12.0	0
7	5375	60 months	12.69	0.0	RENT	other	TX	3.0	1
...
163982	15000	60 months	12.39	3.0	MORTGAGE	credit_card	OK	34.0	0
163983	20000	36 months	14.99	10.0	OWN	home_improvement	VA	18.0	0
163984	12825	36 months	17.14	6.0	MORTGAGE	debt_consolidation	TX	24.0	0
163985	27650	60 months	21.99	0.0	RENT	credit_card	NY	20.0	0
163986	17000	60 months	15.99	10.0	MORTGAGE	debt_consolidation	PA	28.0	0

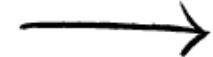
135285 rows × 9 columns

2. Xử lý hàng trùng lặp

2. Xử lý các hàng trùng lặp



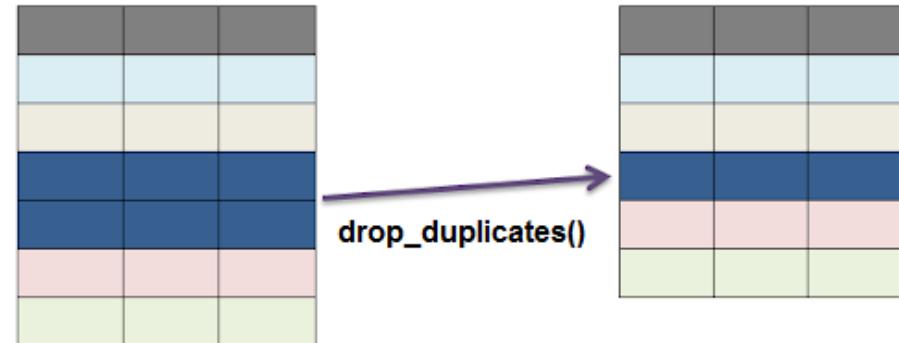
	Pet	Color	Eyes
0	Cat	Brown	Black
1	Dog	Golden	Black
2	Dog	Golden	Black
3	Dog	Golden	Brown
4	Cat	Black	Green



	Pet	Color	Eyes
0	Cat	Brown	Black
1	Dog	Golden	Black
2	Dog	Golden	Black
3	Dog	Golden	Brown
4	Cat	Black	Green

Drop duplicates

Drop Duplicate Pandas



- `df.duplicated()`: để tìm kiếm các hàng trùng lặp

2. Xử lý các hàng trùng lặp

- `df.drop_duplicates()`: với các tham số mặc định sẽ thực hiện
 - Xóa hết các hàng dữ liệu trùng lặp nhau ở tất cả các cột
 - Giữ lại hàng đầu tiên trùng lặp

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
3	jodha	23	55.0
4	raghu	23	31.0
5	Cathrine	24	77.0
6	Alisa	26	85.0
7	Bobby	24	63.0
8	Bobby	22	42.0
9	Alisa	26	85.0
10	raghu	23	31.0
11	Cathrine	24	NaN

```
1 #Trường hợp 1:  
2 #Sử dụng df.drop_duplicates() với các tham số mặc định  
3 #--> giữ lại hàng trùng lặp đầu tiên  
4 df1 = df.drop_duplicates()  
5 df1
```

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
5	Cathrine	24	77.0
7	Bobby	24	63.0
8	Bobby	22	42.0
11	Cathrine	24	NaN

2. Xử lý các hàng trùng lặp

- **df.drop_duplicates(keep='last'):**

- Xóa hết các hàng dữ liệu trùng lặp nhau ở tất cả các cột
- Giữ lại hàng trùng lặp cuối cùng

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
3	jodha	23	55.0
4	raghu	23	31.0
5	Cathrine	24	77.0
6	Alisa	26	85.0
7	Bobby	24	63.0
8	Bobby	22	42.0
9	Alisa	26	85.0
10	raghu	23	31.0
11	Cathrine	24	Nan

```
1 #Trường hợp 2:  
2 #Sử dụng df.drop_duplicates()  
3 #với các tham số keep='Last'  
4 #Giữ Lại các hàng trùng Lặp cuối cùng  
5 df2=df.drop_duplicates(keep='last')  
6 df2
```

	Name	Age	Score
3	jodha	23	55.0
5	Cathrine	24	77.0
7	Bobby	24	63.0
8	Bobby	22	42.0
9	Alisa	26	85.0
10	raghu	23	31.0
11	Cathrine	24	NaN

2. Xử lý các hàng trùng lặp

- **df.drop_duplicates(keep=False):**

- Xóa hết các hàng dữ liệu trùng lặp nhau ở tất cả các cột, chỉ giữ lại các hàng dữ liệu không trùng lặp

	Name	Age	Score	
0	Alisa	26	85.0	3
1	raghu	23	31.0	2
2	jodha	23	55.0	1
3	jodha	23	55.0	
4	raghu	23	31.0	2
5	Cathrine	24	77.0	
6	Alisa	26	85.0	3
7	Bobby	24	63.0	
8	Bobby	22	42.0	
9	Alisa	26	85.0	3
10	raghu	23	31.0	2
11	Cathrine	24	NaN	

```
1 #Trường hợp 3:  
2 #Sử dụng df.drop_duplicates()  
3 #với các tham số keep=False  
4 #Xóa hết các hàng trùng Lặp khỏi df  
5 df3=df.drop_duplicates(keep=False)  
6 df3
```

	Name	Age	Score
5	Cathrine	24	77.0
7	Bobby	24	63.0
8	Bobby	22	42.0
11	Cathrine	24	NaN

2. Xử lý các hàng trùng lặp

- df.drop_duplicates([name columns], keep='first' | 'last' | False):
 - Xóa các hàng dữ liệu trùng lặp nhau ở các cột được chỉ định

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
3	jodha	23	55.0
4	raghu	23	31.0
5	Cathrine	24	77.0
6	Alisa	26	85.0
7	Bobby	24	63.0
8	Bobby	22	42.0
9	Alisa	26	85.0
10	raghu	23	31.0
11	Cathrine	24	NaN

01

```

1 #Trường hợp 4:
2 #Sử dụng df.drop_duplicates()
3 #Loại bỏ các hàng trùng nhau theo cột Name
4 df4=df.drop_duplicates(['Name'],keep='first')
5 df4

```

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
5	Cathrine	24	77.0
7	Bobby	24	63.0

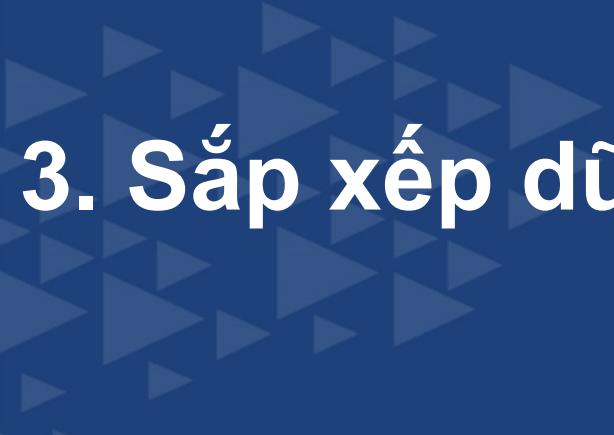
```

1 #Trường hợp 5:
2 #Sử dụng df.drop_duplicates()
3 #Loại bỏ các hàng trùng nhau theo cột Name, Age
4 df5=df.drop_duplicates(['Name','Age'],
5 keep='first')
6 df5

```

	Name	Age	Score
0	Alisa	26	85.0
1	raghu	23	31.0
2	jodha	23	55.0
5	Cathrine	24	77.0
7	Bobby	24	63.0
8	Bobby	22	42.0

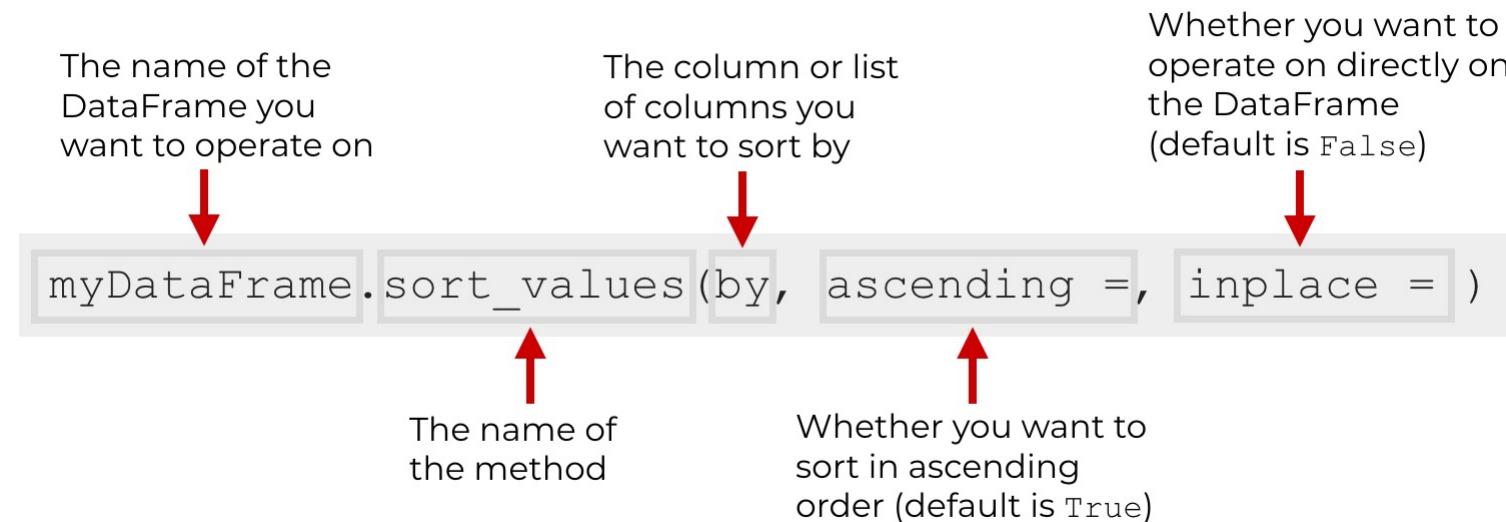
02



3. Sắp xếp dữ liệu trong DataFrame

3. Sắp xếp dữ liệu

- **df.sort_values():** sắp xếp dữ liệu trong DataFrame theo giá trị của các cột, tăng dần (**ascending = True**)-default hoặc giảm dần (**ascending=False**)
- Lưu ý khi sử dụng tham số **inplace = True**



- **df.sort_index():** sắp xếp dữ liệu trong DataFrame theo index

3. Sắp xếp dữ liệu

- Ví dụ:

	Name	Age	Score
0	Alisa	26	89
1	Bobby	27	87
2	Cathrine	25	67
3	Madonna	24	55
4	Rocky	31	47
5	Sebastian	27	72
6	Jaqluine	25	76
7	Rahul	33	79
8	David	42	44
9	Andrew	32	92
10	Ajay	51	99
11	Teresa	47	69
12	Madonna	38	73

```
1 #Trường hợp 1:  
2 #Sắp xếp dữ liệu Dataframe theo cột Score  
3 #Mặc định là sắp xếp tăng dần  
4 df.sort_values(by='Name')
```

	Name	Age	Score
10	Ajay	51	99
0	Alisa	26	89
9	Andrew	32	92
1	Bobby	27	87
2	Cathrine	25	67
8	David	42	44
6	Jaqluine	25	76
3	Madonna	24	55
12	Madonna	38	73

01

3. Sắp xếp dữ liệu

- Ví dụ:

	Name	Age	Score
0	Alisa	26	89
1	Bobby	27	87
2	Cathrine	25	67
3	Madonna	24	55
4	Rocky	31	47
5	Sebastian	27	72
6	Jaqluine	25	76
7	Rahul	33	79
8	David	42	44
9	Andrew	32	92
10	Ajay	51	99
11	Teresa	47	69
12	Madonna	38	73

1 #Trường hợp 2:
2 #Sắp xếp dữ liệu Dataframe theo cột Score
3 #Giá trị giảm dần
4 df.sort_values(by='Score', ascending=False)

	Name	Age	Score
10	Ajay	51	99
9	Andrew	32	92
0	Alisa	26	89
1	Bobby	27	87
7	Rahul	33	79
6	Jaqluine	25	76
12	Madonna	38	73
5	Sebastian	27	72

02

3. Sắp xếp dữ liệu

- Trường hợp sắp xếp nhiều cột, sẽ thực hiện sắp xếp theo thứ tự các cột từ trái sang phải:

	Name	Age	Score
0	Alisa	26	89
1	Bobby	27	87
2	Cathrine	25	67
3	Madonna	24	55
4	Rocky	31	47
5	Sebastian	27	72
6	Jaqluine	25	76
7	Rahul	33	79
8	David	42	44
9	Andrew	32	92
10	Ajay	51	99
11	Teresa	47	69
12	Madonna	38	73

```
1 #Trường hợp 3:  
2 #Sắp xếp dữ Liệu Dataframe theo cột Name, Score  
3 #Giá trị tăng dần  
4 df.sort_values(by=['Name', 'Score'])
```

	Name	Age	Score
10	Ajay	51	99
0	Alisa	26	89
9	Andrew	32	92
1	Bobby	27	87
2	Cathrine	25	67
8	David	42	44
6	Jaqluine	25	76
3	Madonna	24	55
12	Madonna	38	73

03

3. Sắp xếp dữ liệu

- Ví dụ:

	Name	Age	Score
0	Alisa	26	89
1	Bobby	27	87
2	Cathrine	25	67
3	Madonna	24	55
4	Rocky	31	47
5	Sebastian	27	72
6	Jaqluine	25	76
7	Rahul	33	79
8	David	42	44
9	Andrew	32	92
10	Ajay	51	99
11	Teresa	47	69
12	Madonna	38	73

```
1 #Trường hợp 4:  
2 #Sắp xếp dữ liệu Dataframe theo cột Name, Score  
3 #Giá trị cột Name tăng dần  
4 #Giá trị cột Score giảm dần  
5 df.sort_values(by=['Name','Score'],  
6 ascending=[True,False])
```

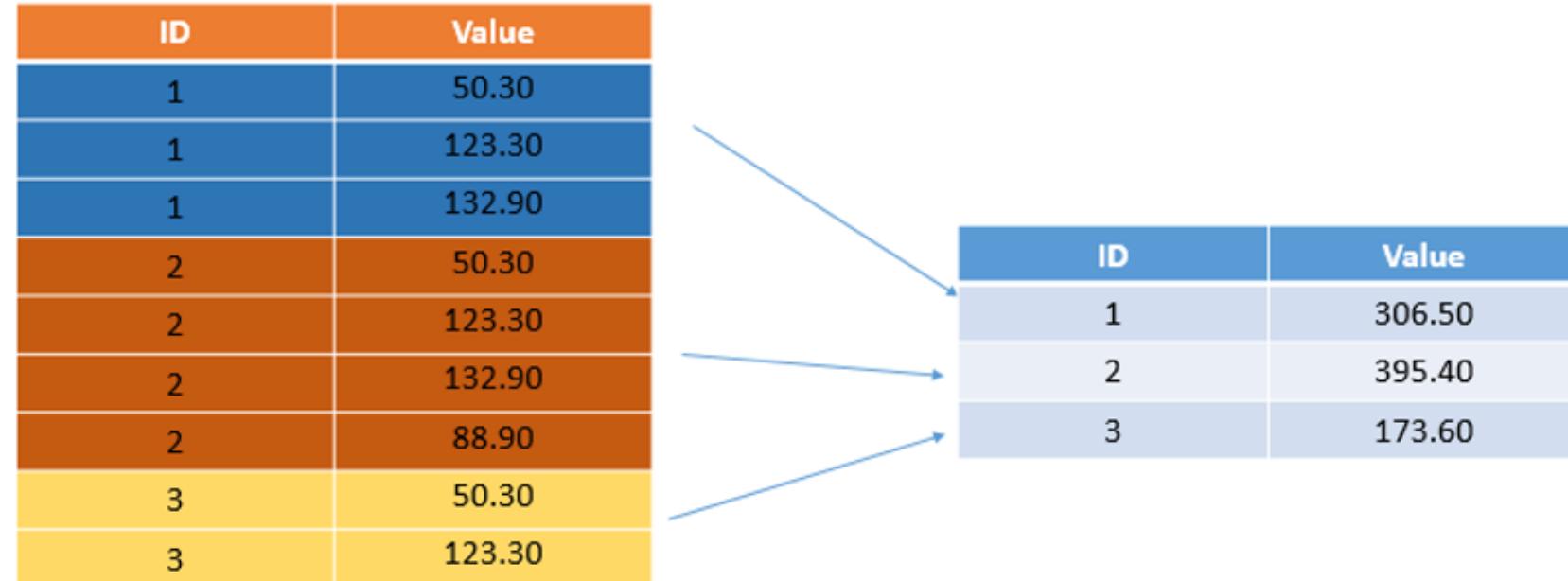
	Name	Age	Score
10	Ajay	51	99
0	Alisa	26	89
9	Andrew	32	92
1	Bobby	27	87
2	Cathrine	25	67
8	David	42	44
6	Jaqluine	25	76
12	Madonna	38	73
3	Madonna	24	55
7	Rahul	33	79
4	Rocky	31	47

04

4. Nhóm dữ liệu (groupby)

Groupby values

- df.groupby(): Gom nhóm các giá trị trong một DataFrame bởi các cột được chỉ định.
- Kết hợp với sum(), mean(), max(), min() để xác định các thông số theo từng nhóm.



The diagram illustrates the process of grouping data. On the left, there is a large table with two columns: 'ID' and 'Value'. The data is as follows:

ID	Value
1	50.30
1	123.30
1	132.90
2	50.30
2	123.30
2	132.90
2	88.90
3	50.30
3	123.30

Three blue arrows point from this large table to three smaller tables on the right, each representing a group:

ID	Value
1	306.50
2	395.40
3	173.60

Groupby values

- Sử dụng phương thức groupby():

	Name	Exam	Subject	Score
0	Alisa	Semester 1	Mathematics	62
1	Bobby	Semester 1	Mathematics	47
2	Cathrine	Semester 1	Mathematics	55
3	Alisa	Semester 1	Science	74
4	Bobby	Semester 1	Science	31
5	Cathrine	Semester 1	Science	77
6	Alisa	Semester 2	Mathematics	85
7	Bobby	Semester 2	Mathematics	63
8	Cathrine	Semester 2	Mathematics	42
9	Alisa	Semester 2	Science	67
10	Bobby	Semester 2	Science	89
11	Cathrine	Semester 2	Science	81

```

1 #Trường hợp 1:
2 #Nhóm theo tên sinh viên (Name)
3 #Thực hiện tính điểm trung bình Score
4 df['Score'].groupby([df['Name']]).mean()

```

```

Name
Alisa      72.00
Bobby      57.50
Cathrine   63.75
Name: Score, dtype: float64

```

01

```

1 #Trường hợp 2:
2 #Nhóm dữ liệu theo tên sinh viên (Name)
3 #và Bài kiểm tra (Exam)
4 #sau đó thực hiện tính tổng
5 df['Score'].groupby([df['Name'],
6                      df['Exam']]).sum()

```

```

Name      Exam
Alisa    Semester 1  136
          Semester 2  152
Bobby    Semester 1  78
          Semester 2  152
Cathrine Semester 1  132
          Semester 2  123
Name: Score, dtype: int64

```

02

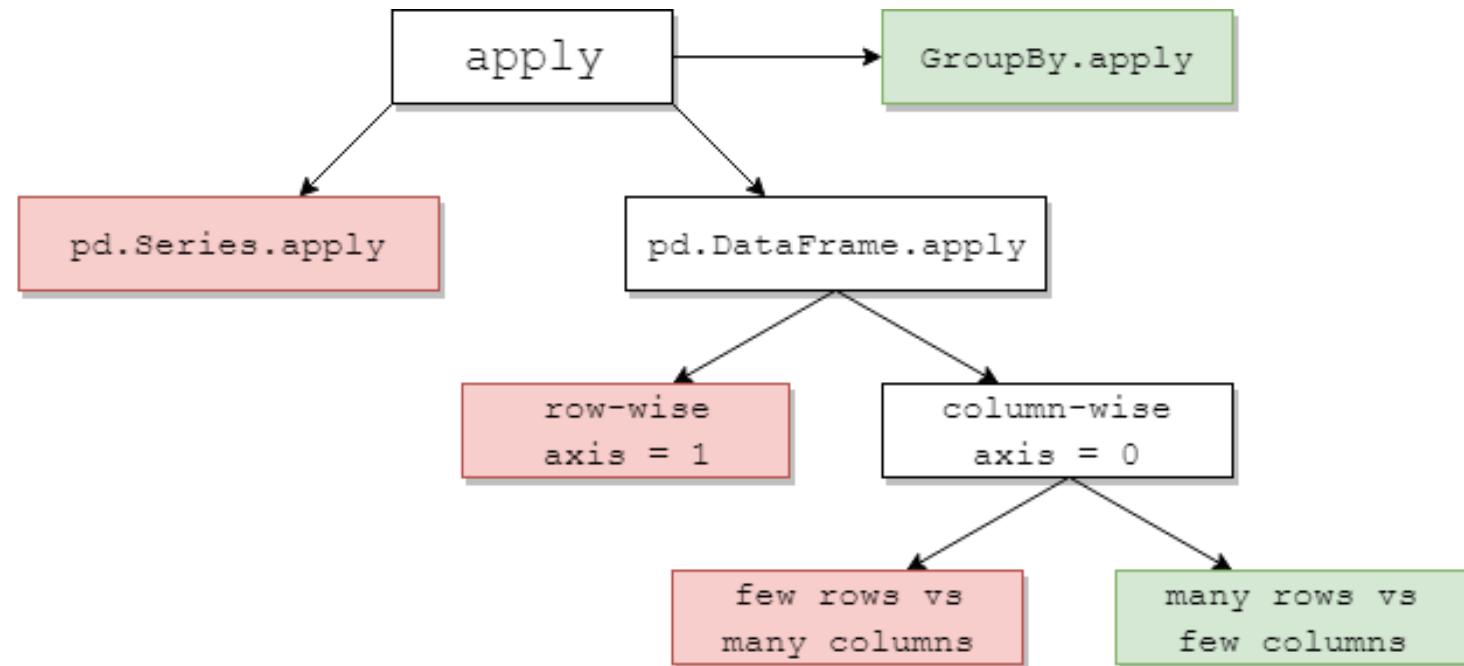
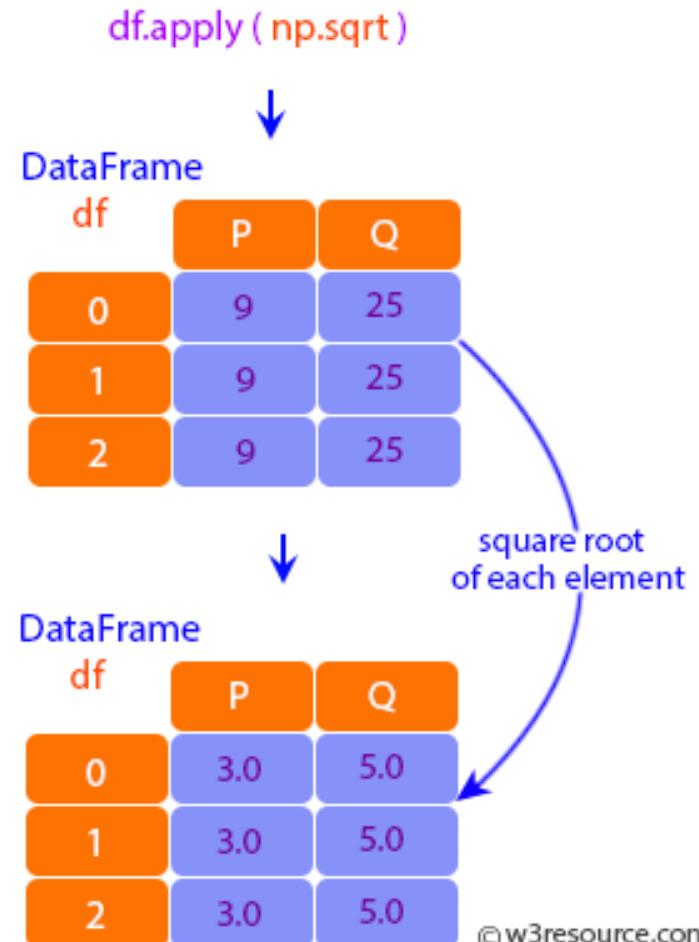
Thực hành 1

5. `apply(function)`



5 .apply(func)

- df.apply(func): Thực hiện thao tác func áp dụng cho từng cột riêng lẻ trong DataFrame, hoặc cho nhiều cột



5 .apply(func)

- Áp dụng hàm cho các phần tử trong một cột dữ liệu của DataFrame: Viết hoa các giá trị trong cột Name (3 Cách thực hiện)

	Name	Score_Math	Score_Science
0	william	66	89
1	Mason	57	87
2	ella	75	67
3	jackson	44	55
4	lincoln	31	47
5	aubrey	67	72
6	Hudson	85	76
7	christian	33	79
8	Sawyer	42	44
9	silas	62	92
10	Bennett	51	93
11	kingston	47	69

```

1 #Thực hiện: viết hoa tên học sinh
2 #Cách 1:
3 def uppercase(x):
4     return x.upper()
5
6 df[ 'Name' ] = df[ 'Name' ].apply(uppercase)
7 df

```

01

	Name	Score_Math	Score_Science
0	WILLIAM	66	89
1	MASON	57	87
2	ELLA	75	67

```

1 #Cách 2:
2 df[ 'Name' ] = df[ 'Name' ].apply(lambda x:x.upper())
3 df

```

02

	Name	Score_Math	Score_Science
0	WILLIAM	66	89
1	MASON	57	87
2	ELLA	75	67

```

1 #Cách 3:
2 df[ 'Name' ] = df[ 'Name' ].str.upper()
3 df

```

03

	Name	Score_Math	Score_Science
0	WILLIAM	66	89
1	MASON	57	87
2	ELLA	75	67

5 .apply(func)

- Áp dụng hàm cho các phần tử trong nhiều cột dữ liệu của DataFrame:
- Thực hiện tính điểm cho từng học sinh theo công thức:
 - **Point = (Score_Math*2 + Score_Science)/3**

	Name	Score_Math	Score_Science
0	william	66	89
1	Mason	57	87
2	ella	75	67
3	jackson	44	55
4	lincoln	31	47
5	aubrey	67	72
6	Hudson	85	76
7	christian	33	79
8	Sawyer	42	44
9	silas	62	92
10	Bennett	51	93
11	kingston	47	69

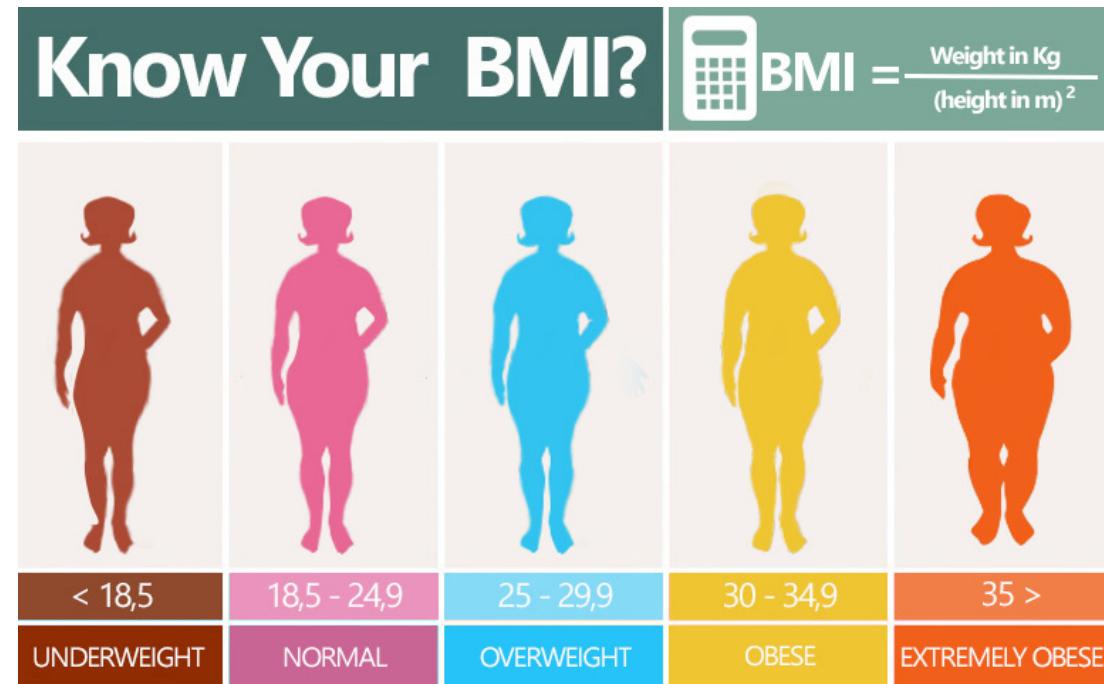
```
1 #Điểm trung bình = (score_math*2 + score_science)/3
2 #Viết hàm tính điểm trung bình
3 def mean_point(point1,point2):
4     return round((point1*2+point2)/3,1)

1 #Tạo một cột Point tính điểm của từng học sinh
2 df['Point'] = df.apply(lambda row: mean_point(row['Score_Math'],
3                                                 row['Score_Science']),
4                                                 axis=1)
5 df
```

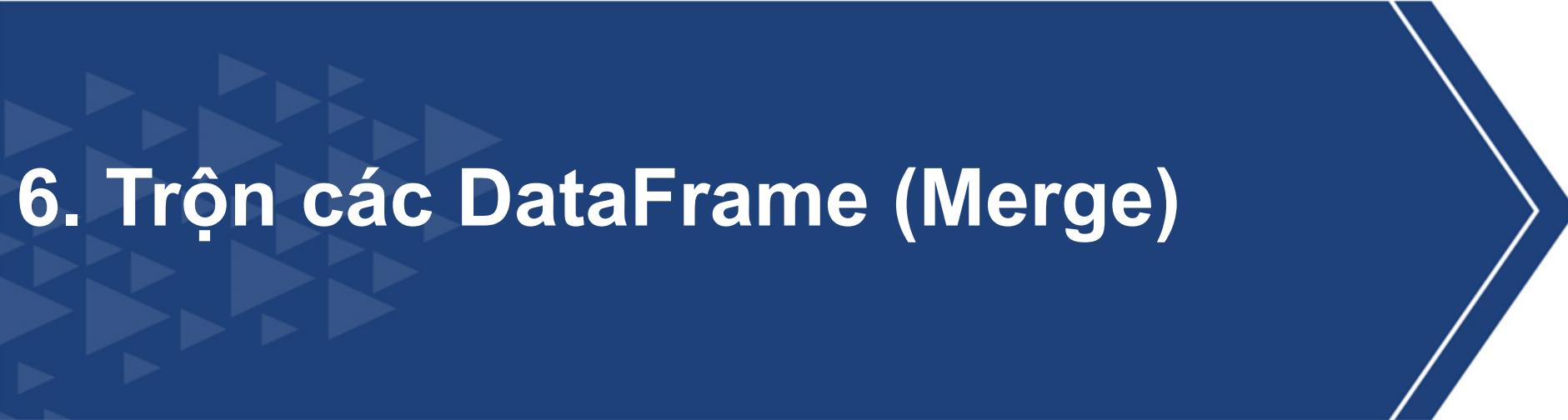
	Name	Score_Math	Score_Science	Point
0	WILLIAM	66	89	73.7
1	MASON	57	87	67.0
2	ELLA	75	67	72.3
3	JACKSON	44	55	47.7
4	LINCOLN	31	47	36.3

5 .apply(func)

- Áp dụng viết các hàm để tính chỉ số BMI, và phân loại dựa theo chỉ số tính được trên tập dữ liệu csv_Data_BMI.csv



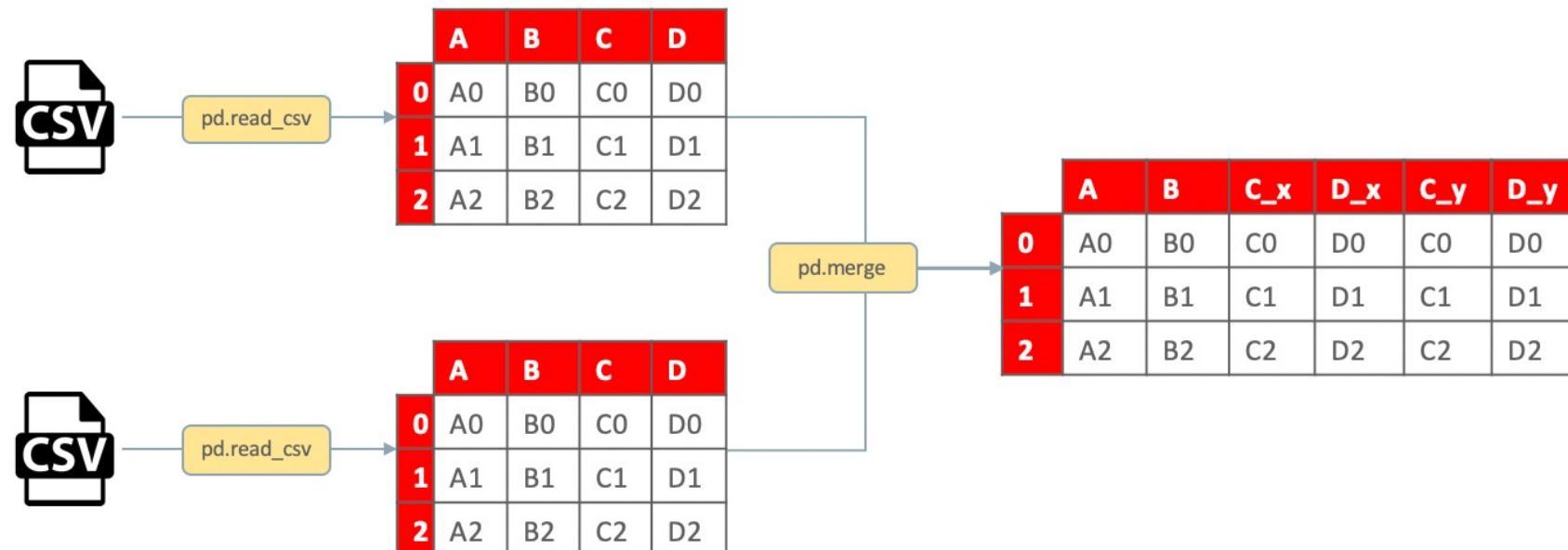
Gender	Height_cm	Weight_kg	BMI
Personal			
P1	Male	174	96 31.7
P2	Male	189	87 24.4
P3	Female	185	110 32.1
P4	Female	195	104 27.4
P5	Male	149	61 27.5



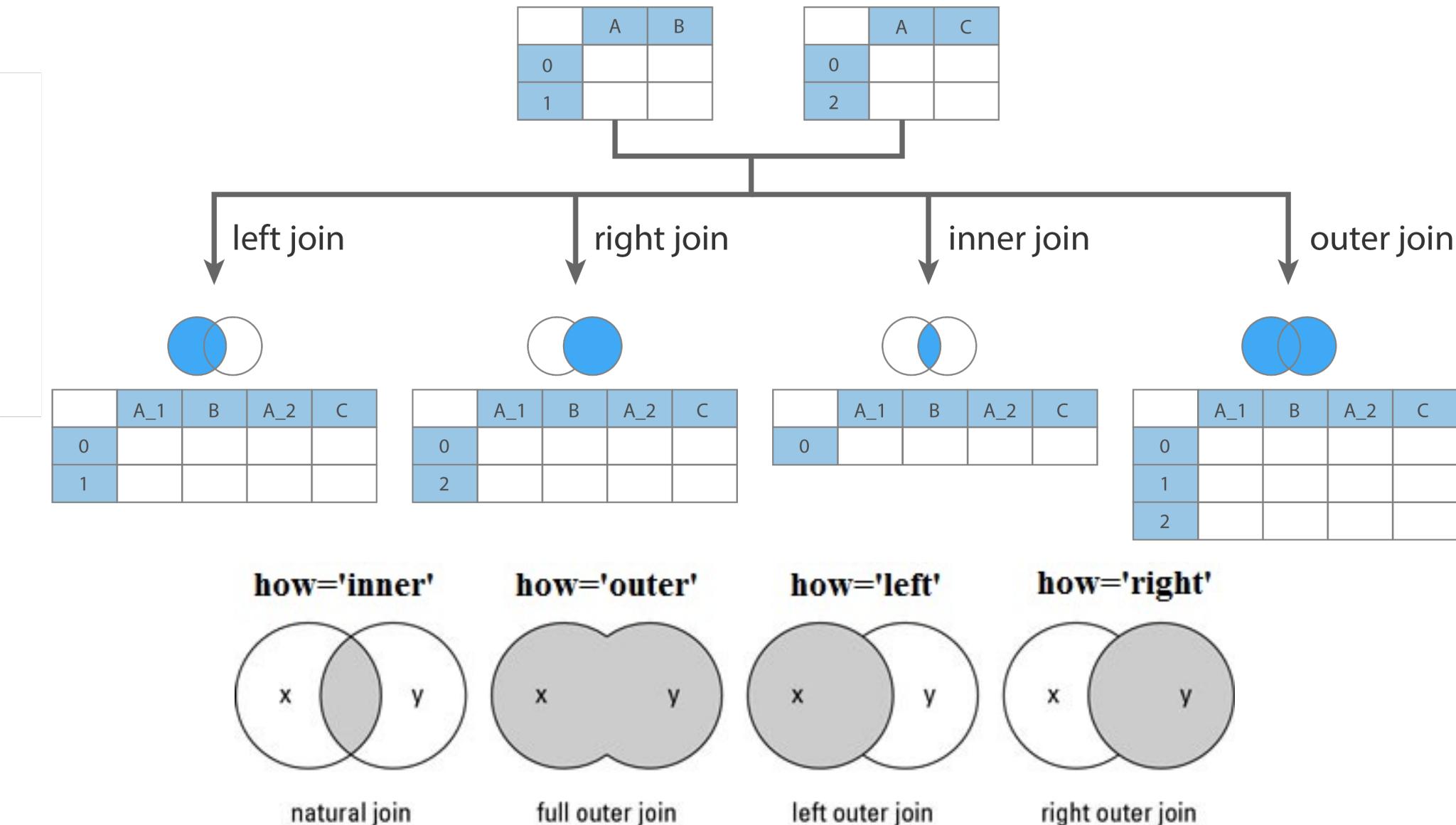
6. Trộn các DataFrame (Merge)

Trộn các DataFrame

- **pd.merge(left_df,right_df, on='key', how='left' | 'right' | 'inner' | 'outer')**: Thực hiện trộn 2 DataFrame lại với nhau.
 - Left_df: DataFrame 1
 - Right_df: DataFrame2
 - On: Tên cột dùng để nối dữ liệu giữa 2 DataFrame (tên cột phải có ở trong cả 2 DataFrame 1, 2)
 - How: Cách thức trộn dữ liệu [left, right, outer, inner (default)]



Trộn các DataFrame



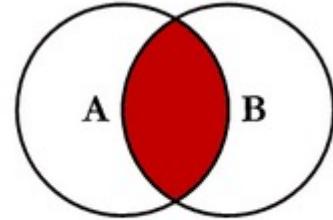
Trộn các DataFrame (inner)

Customer_id	Product
0	1 Oven
1	2 Oven
2	3 Oven
3	4 Television
4	5 Television
5	6 Television

df1

Customer_id	State
0	2 California
1	4 California
2	6 Texas
3	7 New York
4	8 Indiana

df2



Inner Join

```
1 #Trường hợp 1:  
2 #Inner join DataFrame  
3 inner_join_df= pd.merge(df1, df2,  
4                           on='Customer_id',  
5                           how='inner')  
6 inner_join_df
```

Customer_id	Product	State
0	2 Oven	California
1	4 Television	California
2	6 Television	Texas

Trộn các DataFrame (outer)



VINBIGDATA VINGROUP

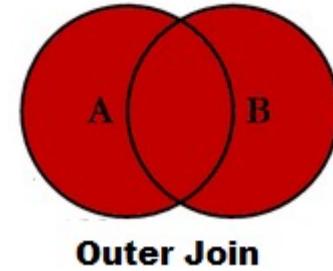
Academy Vietnam

Customer_id	Product
0	1 Oven
1	2 Oven
2	3 Oven
3	4 Television
4	5 Television
5	6 Television

df1

Customer_id	State
0	2 California
1	4 California
2	6 Texas
3	7 New York
4	8 Indiana

df2



Outer Join

```
1 #Trường hợp 2:  
2 #Outer join DataFrame  
3 inner_join_df = pd.merge(df1, df2,  
4                           on='Customer_id',  
5                           how='outer')  
6 inner_join_df
```

Customer_id	Product	State
0	1 Oven	NaN
1	2 Oven	California
2	3 Oven	NaN
3	4 Television	California
4	5 Television	NaN
5	6 Television	Texas
6	7 NaN	New York
7	8 NaN	Indiana

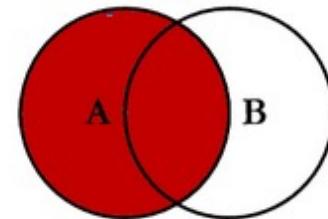
Trộn các DataFrame (left)

	Customer_id	Product
0	1	Oven
1	2	Oven
2	3	Oven
3	4	Television
4	5	Television
5	6	Television

df1

	Customer_id	State
0	2	California
1	4	California
2	6	Texas
3	7	New York
4	8	Indiana

df2



Left join

```
1 #Trường hợp 3:  
2 #Left join DataFrame  
3 inner_join_df= pd.merge(df1, df2,  
4                           on='Customer_id',  
5                           how='left')  
6 inner_join_df
```

	Customer_id	Product	State
0	1	Oven	NaN
1	2	Oven	California
2	3	Oven	NaN
3	4	Television	California
4	5	Television	NaN
5	6	Television	Texas

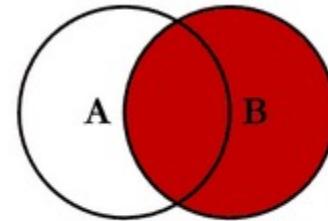
Trộn các DataFrame (right)

Customer_id	Product
0	1 Oven
1	2 Oven
2	3 Oven
3	4 Television
4	5 Television
5	6 Television

df1

Customer_id	State
0	2 California
1	4 California
2	6 Texas
3	7 New York
4	8 Indiana

df2



Right Join

```
1 #Trường hợp 4:  
2 #Right join DataFrame  
3 inner_join_df= pd.merge(df1, df2,  
4                           on='Customer_id',  
5                           how='right')  
6 inner_join_df
```

Customer_id	Product	State
0	2 Oven	California
1	4 Television	California
2	6 Television	Texas
3	7 NaN	New York
4	8 NaN	Indiana

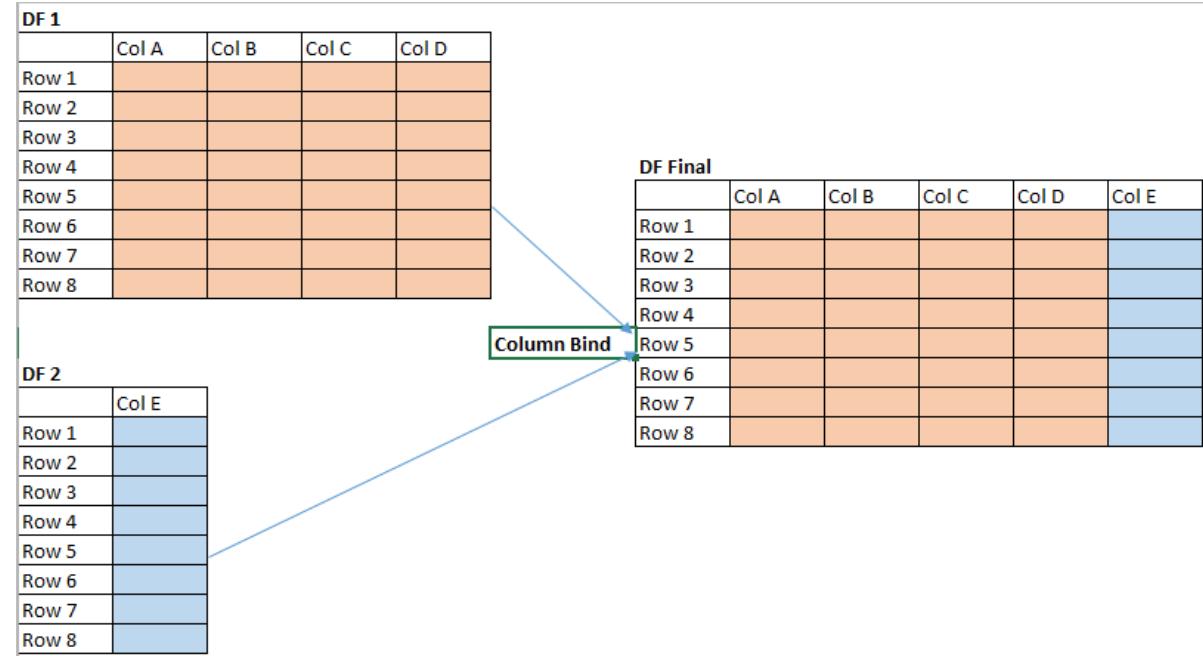


7. Nối các DataFrame (concat, append)

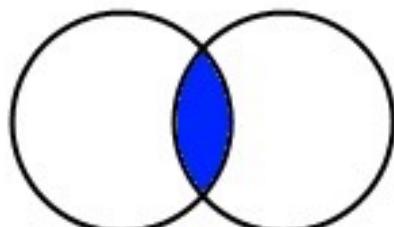


Nối các DataFrame theo cột

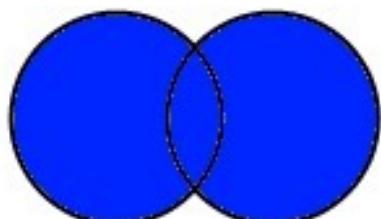
- `pd.concat([df1,df2], axis=1, join='inner'|'outer')`: Thực hiện ghép nối các DataFrame lại với nhau theo cột



INNER JOIN



FULL OUTER JOIN



Nối các DataFrame theo cột

	Name	Score1	Score2
0	Alisa	62	89
1	Bobby	47	87
2	Cathrine	55	67
3	Madonna	74	55
4	Rocky	31	47
5	Sebastian	77	72
6	Jaqluine	85	76
7	Rahul	63	79
8	David	42	44

01

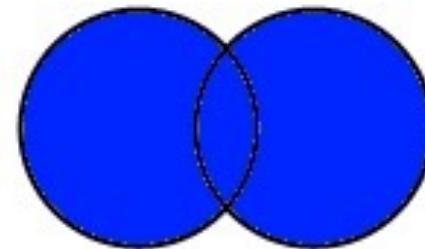
- **pd.concat([df1,df2], axis=1)**: sử dụng các tham số mặc định

```
1 #Trường hợp 1:  
2 #Mặc định join='outer'  
3 df_concat1 = pd.concat([df1, df2], axis=1)  
4 df_concat1
```

	Name	Score3
0	Alisa	56
1	Bobby	86
2	Cathrine	77
3	Madonna	45
4	Rocky	73
5	Sebastian	62
6	Jaqluine	74

02

FULL OUTER JOIN



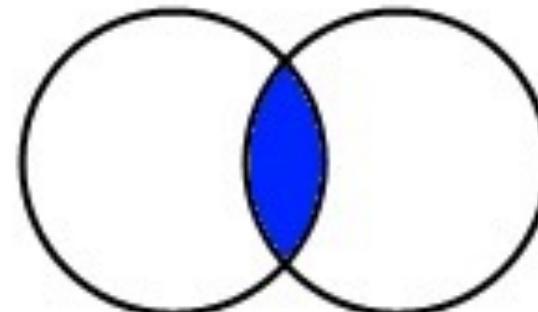
	Name	Score1	Score2	Name	Score3
0	Alisa	62	89	Alisa	56.0
1	Bobby	47	87	Bobby	86.0
2	Cathrine	55	67	Cathrine	77.0
3	Madonna	74	55	Madonna	45.0
4	Rocky	31	47	Rocky	73.0
5	Sebastian	77	72	Sebastian	62.0
6	Jaqluine	85	76	Jaqluine	74.0
7	Rahul	63	79	NaN	NaN
8	David	42	44	NaN	NaN

Nối các DataFrame theo cột

	Name	Score1	Score2
0	Alisa	62	89
1	Bobby	47	87
2	Cathrine	55	67
3	Madonna	74	55
4	Rocky	31	47
5	Sebastian	77	72
6	Jaqluine	85	76
7	Rahul	63	79
8	David	42	44

01

INNER JOIN



	Name	Score3
0	Alisa	56
1	Bobby	86
2	Cathrine	77
3	Madonna	45
4	Rocky	73
5	Sebastian	62
6	Jaqluine	74

02

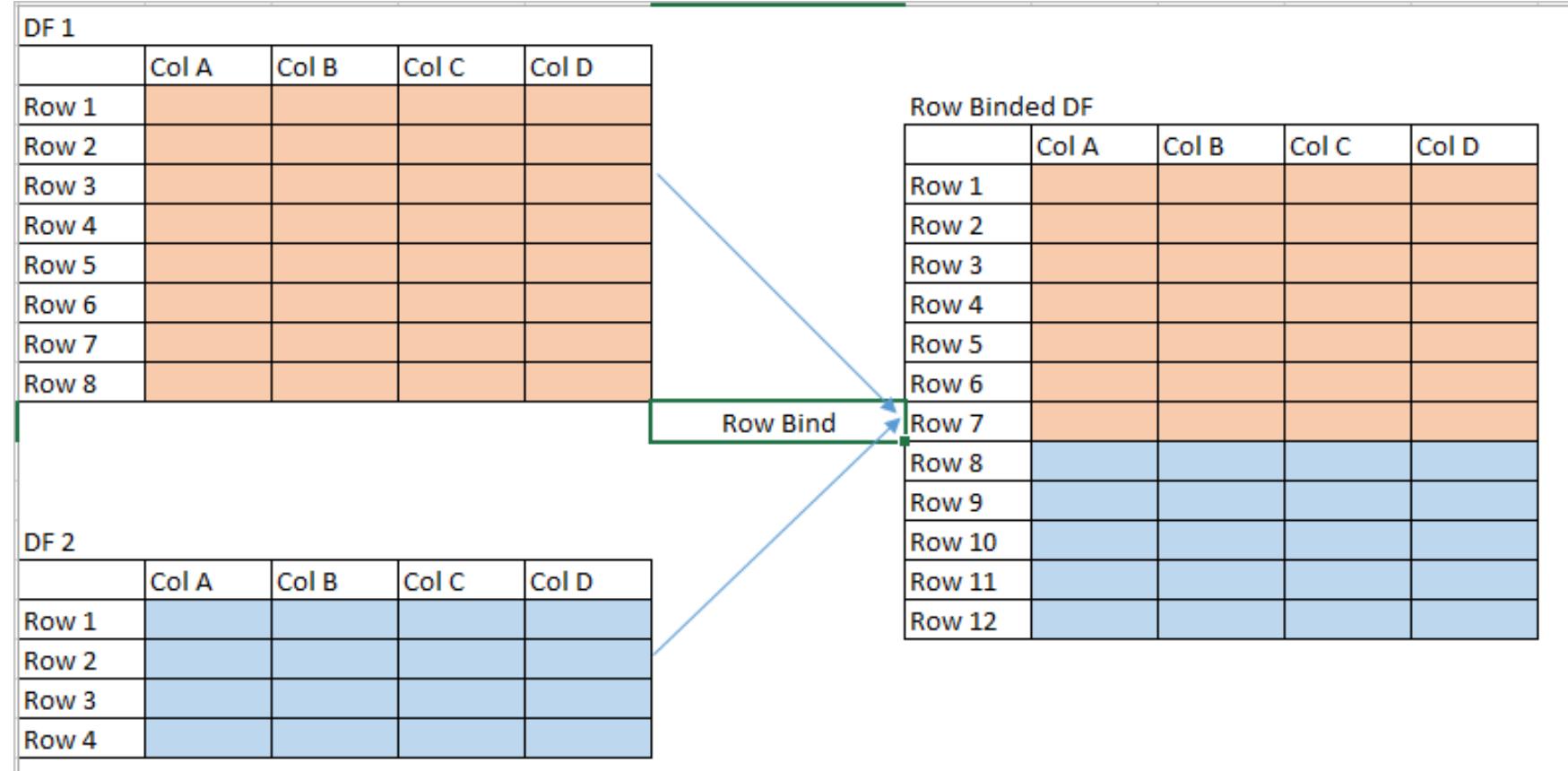
- **pd.concat([df1,df2], axis=1, join='inner'):**
Sử dụng tham số join

```
1 #Trường hợp 2:  
2 #Tham số join='inner'  
3 df_concat2 = pd.concat([df1, df2],  
4                         axis=1,  
5                         join='inner')  
6 df_concat2
```

	Name	Score1	Score2	Name	Score3
0	Alisa	62	89	Alisa	56
1	Bobby	47	87	Bobby	86
2	Cathrine	55	67	Cathrine	77
3	Madonna	74	55	Madonna	45
4	Rocky	31	47	Rocky	73
5	Sebastian	77	72	Sebastian	62
6	Jaqluine	85	76	Jaqluine	74

Nối các DataFrame theo hàng

- **pd.concat([df1,df2], axis=0, join='inner'|'outer', ignore_index=True|False)** hoặc
df1.append(df2): Thực hiện ghép nối các DataFrame lại với nhau theo hàng



Nối các DataFrame theo hàng



VINBIGDATA VINGROUP

Academy
Vietnam

01

	Name	Score1	Score2	Score3
0	Alisa	62	89	56
1	Bobby	47	87	86
2	Cathrine	55	67	77
3	Madonna	74	55	45
4	Rocky	31	47	73

- **pd.concat([df1,df2]):**

```
1 #Trường hợp 1: sử dụng concat
2 df_row = pd.concat([df1,df2])
3 df_row
```

	Name	Score1	Score2	Score3
0	Alisa	62	89	56
1	Bobby	47	87	86
2	Cathrine	55	67	77
3	Madonna	74	55	45
4	Rocky	31	47	73
0	Andrew	32	92	67
1	Ajay	71	99	97
2	Teresa	57	69	68

Trường hợp các cột cùng tên:

	Name	Score1	Score2	Score3
0	Andrew	32	92	67
1	Ajay	71	99	97
2	Teresa	57	69	68

- **df1.append(df2):**

```
1 #Trường hợp 1: sử dụng append()
2 df_row2 = df1.append(df2)
3 df_row2
```

	Name	Score1	Score2	Score3
0	Alisa	62	89	56
1	Bobby	47	87	86
2	Cathrine	55	67	77
3	Madonna	74	55	45
4	Rocky	31	47	73
0	Andrew	32	92	67
1	Ajay	71	99	97
2	Teresa	57	69	68

Nối các DataFrame theo hàng



VINBIGDATA VINGROUP

Academy Vietnam

01

	Name	Score1	Score2	Score3
0	Alisa	62	89	56
1	Bobby	47	87	86
2	Cathrine	55	67	77
3	Madonna	74	55	45
4	Rocky	31	47	73

Trường hợp các cột khác tên:

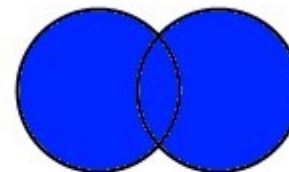
	Name	Score1	Score4	Score5
0	Jack	32	72	57
1	danny	71	91	72
2	vishwa	70	89	78

- pd.concat([df1,df3]) | df1.append(df3):

```
1 #Trường hợp các cột khác tên  
2 pd.concat([df1,df3])
```

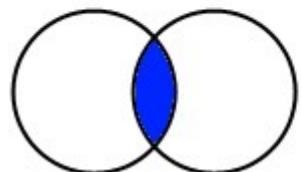
	Name	Score1	Score2	Score3	Score4	Score5
0	Alisa	62	89.0	56.0	NaN	NaN
1	Bobby	47	87.0	86.0	NaN	NaN
2	Cathrine	55	67.0	77.0	NaN	NaN
3	Madonna	74	55.0	45.0	NaN	NaN
4	Rocky	31	47.0	73.0	NaN	NaN
0	Jack	32	NaN	NaN	72.0	57.0
1	danny	71	NaN	NaN	91.0	72.0
2	vishwa	70	NaN	NaN	89.0	78.0

FULL OUTER JOIN



	Name	Score1
0	Alisa	62
1	Bobby	47
2	Cathrine	55
3	Madonna	74
4	Rocky	31
0	Jack	32
1	danny	71
2	vishwa	70

INNER JOIN



Thực hành 2



8. Phát hiện và xử lý giá trị khuyết thiếu

Phát hiện và xử lý missing data

- Vì nhiều lý do, dữ liệu có thể bị lỗi hoặc bị thiếu ở một số vị trí của một số feature.
- Hầu hết các thuật toán học máy không xử lý với dữ liệu bị thiếu, do đó cần phải được xử lý ở bước tiền xử lý dữ liệu.

Các nguyên nhân dẫn đến missing data:

- Khuyết ngẫu nhiên (Missing at Random – MAR):
- Khuyết hoàn toàn ngẫu nhiên (Missing Completely at Random – MCAR):
- Khuyết không ngẫu nhiên (Missing not at Random – MNAR):



	A	B	C	D	E	F	G
1	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
2	00 15-9-2019	25.65	24.79	24.01	25.06	25.48	24.97
3	01 15-9-2019		24.21	24.02	24.93	25.16	24.83
4	02 15-9-2019	25.05	23.73	23.89	24.79	24.8	24.55
5	03 15-9-2019	24.79	23.36	23.83		24.74	24.48
6	04 15-9-2019	24.59	23.05	23.69	24.82	24.8	24.38
7	05 15-9-2019	24.4		23.52	24.79	24.87	24.4
8	06 15-9-2019	24.38	22.79	23.68	25.1	24.71	24.41
9	07 15-9-2019	26.72	25.61	24.92	26.56	25.03	24.91
10	08 15-9-2019	28.84	26.93	26.51	26.53	25.75	25.85
11	09 15-9-2019	30.29	28.72	27.48	26.95	26.64	26.79
12	10 15-9-2019		29.97			27.68	27.53
13	11 15-9-2019	32.05	28.93	26.86	27.38	28.43	28.98
14	12 15-9-2019	31.31	28.94	26.65	27.47	28.29	29.24
15	13 15-9-2019	30.95		27.83	27.44	28	30.66
16	14 15-9-2019	30.56	30.62	26.49	27.16	27.67	30.97
17	15 15-9-2019	31.13	30.58	26.29	26.68	27.29	30.59
18	16 15-9-2019	30.8	30.2		26.45	27.29	29.13
19	17 15-9-2019	29.94	29.36	25.8	26.67	26.69	28.72
20	18 15-9-2019	28.53	27.48	24.82	25.92	25.81	27.46
21	19 15-9-2019	28.89	27.03	24.93	25.88	25.93	27.07
22	20 15-9-2019	28.06	26.41	24.7		25.97	26.75
23	21 15-9-2019	27.43	26.2	24.41	25.62	25.94	26.32
24	22 15-9-2019	26.98	25.79	24.17	25.6	25.9	26.29
25	23 15-9-2019	26.68	25.31	23.81	25.53	25.8	26.36
26							
27							

a.Phát hiện missing data

- Thống kê dữ liệu missing trong dataframe:
 - `df.isnull().sum()`

```

1 #Thống kê số lượng missing trong Data frame
2 #Theo từng cột
3 print('Số lượng missing data trong file dữ liệu:')
4 print(data_temp.isnull().sum())

```

Số lượng missing data trong file dữ liệu:

time	0
Ha Noi	2
Vinh	2
Da Nang	2
Nha Trang	3
Ho Chi Minh	0
Ca Mau	0
dtype: int64	

- Xây dựng hàm thống kê `missing_values()`

```

1 #Xây dựng hàm thống kê dữ liệu missing trong dataframe:
2 -----
3 #Đầu vào của hàm là 1 biến Dataframe
4 #Đầu ra bao gồm các thông số:
5 #Tổng số cột của file dữ liệu
6 #Tổng số cột có chứa dữ liệu missing
7 #Danh sách các cột chứa dữ liệu missing với 2 thông số:
8 #Tổng số giá trị missing tương ứng với cột đó
9 #Tỷ lệ % dữ liệu missing trên tổng số dữ liệu của cột
10 def missing_values(df):
11     mis_val = df.isnull().sum()
12     mis_val_percent = 100 * df.isnull().sum() / len(df)
13     mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
14     mis_val_table_ren_columns = mis_val_table.rename(
15         columns = {0 : 'Số giá trị Missing', 1 : 'Tỷ lệ % missing'})
16     mis_val_table_ren_columns = mis_val_table_ren_columns[
17         mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
18         'Tỷ lệ % missing', ascending=False).round(1)
19     print ("File dữ liệu bao gồm có: " + str(df.shape[1]) + " cột.\n"
20           "Có " + str(mis_val_table_ren_columns.shape[0]) +
21           " cột chứa missing values.")
22     return mis_val_table_ren_columns

```

a.Phát hiện missing data

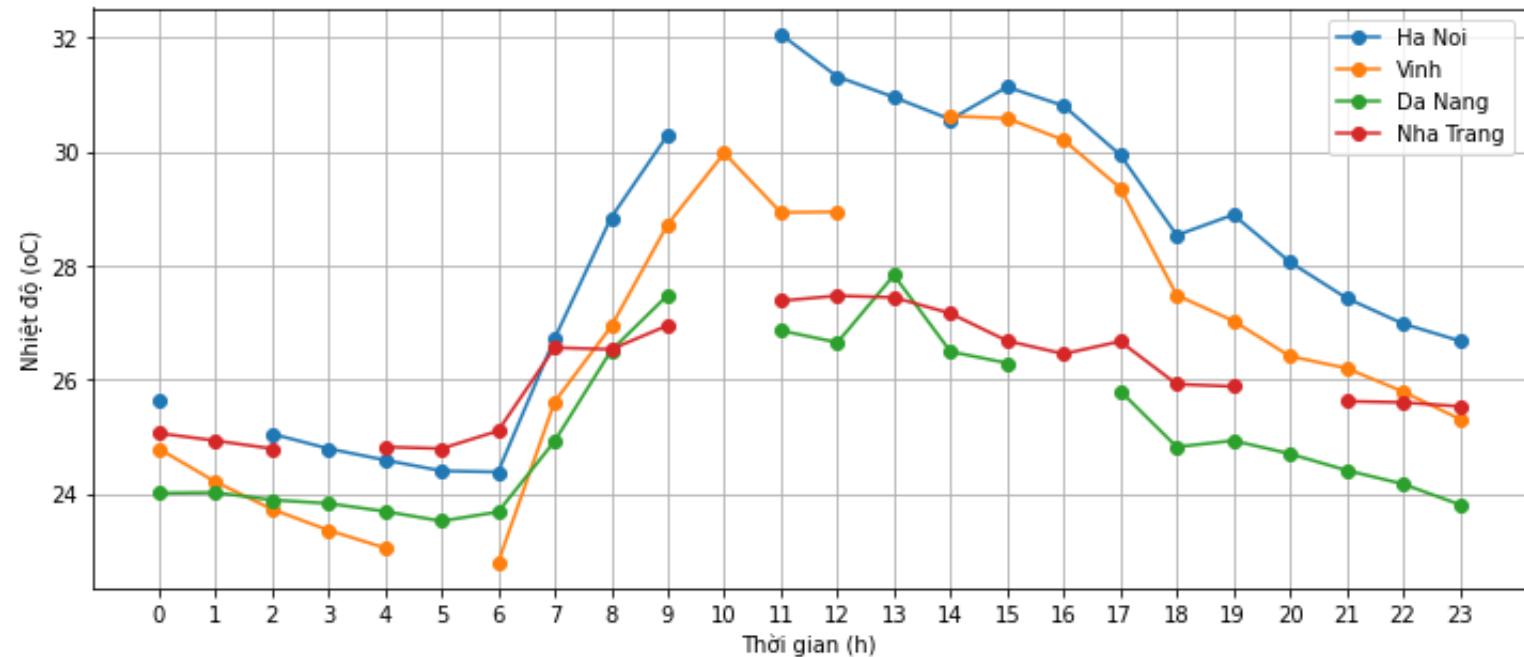
- Xây dựng hàm thống kê **missing_values()**

```
1 missing_values(data_temp)
```

File dữ liệu bao gồm có: 7 cột.

Có 4 cột chứa missing values.

Số giá trị Missing	Tỷ lệ % missing
Nha Trang	3
Ha Noi	2
Vinh	2
Da Nang	2



a. Phát hiện missing data

- Liệt kê các hàng chứa missing trong Dataframe:

```

1 #Liệt kê các dòng dữ liệu missing
2 #1. Liệt kê theo từng features:
3 data_temp[pd.isnull(data_temp['Ha Noi'])]

```

	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
1	01 15-9-2019	NaN	24.21	24.02	24.93	25.16	24.83
10	10 15-9-2019	NaN	29.97	NaN	NaN	27.68	27.53

```
1 data_temp[pd.isnull(data_temp['Nha Trang'])]
```

	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
3	03 15-9-2019	24.79	23.36	23.83	NaN	24.74	24.48
10	10 15-9-2019	NaN	29.97	NaN	NaN	27.68	27.53
20	20 15-9-2019	28.06	26.41	24.70	NaN	25.97	26.75

```

1 #2. Liệt kê các dòng missing của tất cả các features:
2 data_temp[data_temp.isnull().any(axis=1)]

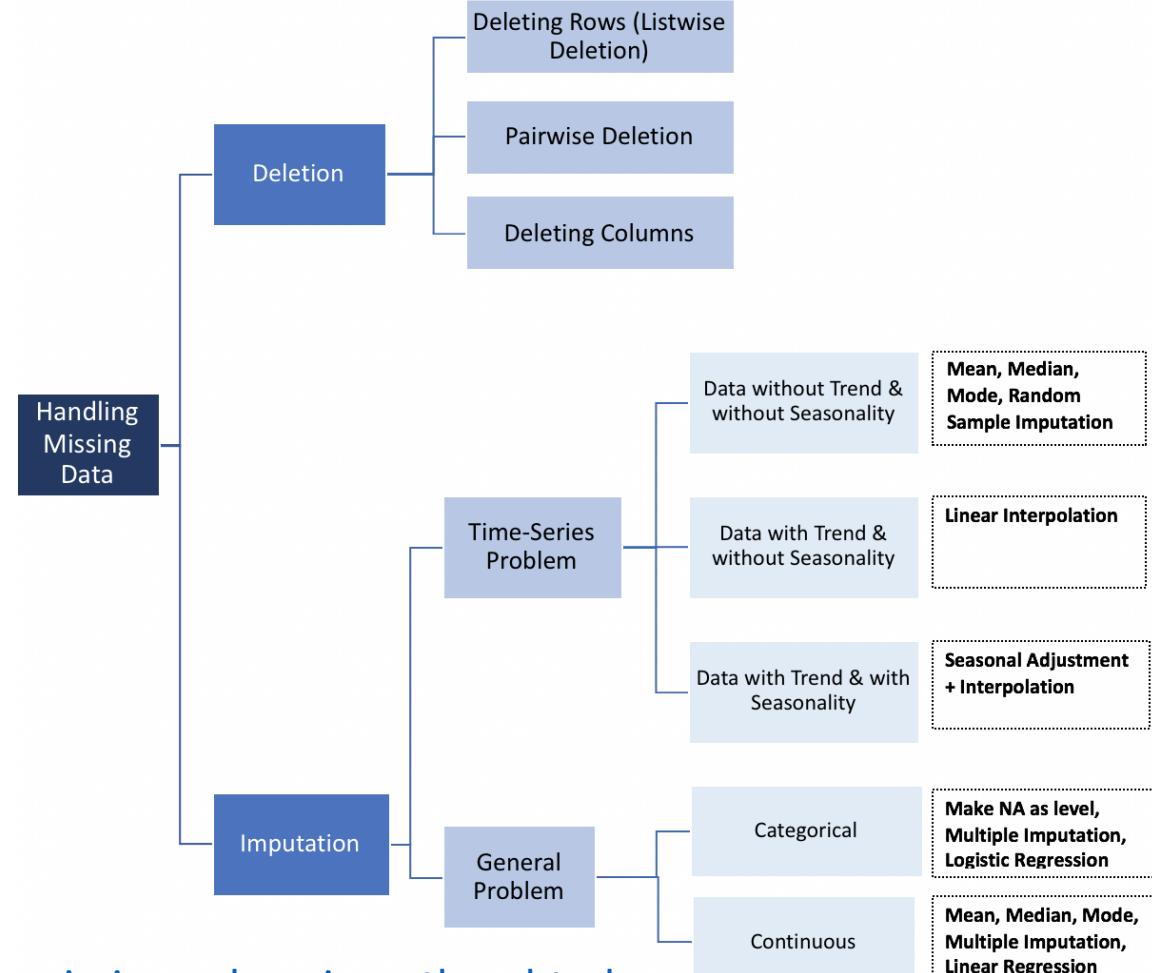
```

	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
1	01 15-9-2019	NaN	24.21	24.02	24.93	25.16	24.83
3	03 15-9-2019	24.79	23.36	23.83	NaN	24.74	24.48
5	05 15-9-2019	24.40	NaN	23.52	24.79	24.87	24.40
10	10 15-9-2019	NaN	29.97	NaN	NaN	27.68	27.53
13	13 15-9-2019	30.95	NaN	27.83	27.44	28.00	30.66
16	16 15-9-2019	30.80	30.20	NaN	26.45	27.29	29.13
20	20 15-9-2019	28.06	26.41	24.70	NaN	25.97	26.75

b.Xử lý missing data

- Để xử lý dữ liệu missing cần phải hiểu sâu sắc tập dữ liệu, việc lựa chọn phương pháp nào phụ thuộc vào từng bài toán cụ thể, một số phương pháp xử lý dữ liệu missing cơ bản:

1) Loại bỏ các missing (Deletion)



2) Thay thế các missing(Imputation)

b.Xử lý missing data

**df.dropna(axis=0) →
loại bỏ hàng**

```

1 #1) Phương pháp 1: Loại bỏ các dữ liệu missing (Deletion)
2
3 #Xóa toàn bộ các hàng chứa missing data: axis=0 -> xóa hàng
4 data_new = data_temp.dropna(axis=0, how='any')
5 #Kết quả sau khi loại bỏ các row chứa missing
6 print(data_new)

```

	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
0	00 15-9-2019	25.65	24.79	24.01	25.06	25.48	24.97
2	02 15-9-2019	25.05	23.73	23.89	24.79	24.80	24.55
4	04 15-9-2019	24.59	23.05	23.69	24.82	24.80	24.38
6	06 15-9-2019	24.38	22.79	23.68	25.10	24.71	24.41
7	07 15-9-2019	26.72	25.61	24.92	26.56	25.03	24.91
8	08 15-9-2019	28.84	26.93	26.51	26.53	25.75	25.85
9	09 15-9-2019	30.29	28.72	27.48	26.95	26.64	26.79
11	11 15-9-2019	32.05	28.93	26.86	27.38	28.43	28.98
12	12 15-9-2019	31.31	28.94	26.65	27.47	28.29	29.24
14	14 15-9-2019	30.56	30.62	26.49	27.16	27.67	30.97
15	15 15-9-2019	31.13	30.58	26.29	26.68	27.29	30.59
17	17 15-9-2019	29.94	29.36	25.80	26.67	26.69	28.72
18	18 15-9-2019	28.53	27.48	24.82	25.92	25.81	27.46
19	19 15-9-2019	28.89	27.03	24.93	25.88	25.93	27.07
21	21 15-9-2019	27.43	26.20	24.41	25.62	25.94	26.32
22	22 15-9-2019	26.98	25.79	24.17	25.60	25.90	26.29
23	23 15-9-2019	26.68	25.31	23.81	25.53	25.80	26.36



b.Xử lý missing data

df.dropna(axis=1) →
loại bỏ cột

```
1 #1) Phương pháp 1: Loại bỏ các dữ liệu missing (Deletion)
2
3 #Xóa toàn bộ các cột chứa missing data: axis=1 -> xóa cột
4 data_new = data_temp.dropna(axis=1, how='any')
5 #Kết quả sau khi loại bỏ các cột chứa missing
6 print(data_new)
```

	time	Ho Chi Minh	Ca Mau
0	00 15-9-2019	25.48	24.97
1	01 15-9-2019	25.16	24.83
2	02 15-9-2019	24.80	24.55
3	03 15-9-2019	24.74	24.48
4	04 15-9-2019	24.80	24.38
5	05 15-9-2019	24.87	24.40
6	06 15-9-2019	24.71	24.41
7	07 15-9-2019	25.03	24.91
8	08 15-9-2019	25.75	25.85
9	09 15-9-2019	26.64	26.79
10	10 15-9-2019	27.68	27.53
11	11 15-9-2019	28.43	28.98
12	12 15-9-2019	28.29	29.24
13	13 15-9-2019	28.00	30.66
14	14 15-9-2019	27.67	30.97
15	15 15-9-2019	27.29	30.59
16	16 15-9-2019	27.29	29.13
17	17 15-9-2019	26.69	28.72

Các cột **Hà Nội, Vinh, Đà Nẵng, Nha Trang** có chứa dữ liệu missing đã bị loại bỏ

b.Xử lý missing data

df.fillna(value) → thay thế bằng một giá trị cố định

```

1 #PHƯƠNG PHÁP 2: Thay thế (Imputation)
2 #2.1) Thay thế các dữ liệu mất mát bằng một hằng số cố định
3 value = 25.0
4 #thay thế các giá trị missing bằng một giá trị cố định Value
5 data_new = data_temp.fillna(value)
6 print(data_new)

```

		time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
0	00	15-9-2019	25.65	24.79	24.01	25.06	25.48	24.97
1	01	15-9-2019	25.00	24.21	24.02	24.93	25.16	24.83
2	02	15-9-2019	25.05	23.73	23.89	24.79	24.80	24.55
3	03	15-9-2019	24.79	23.36	23.83	25.00	24.74	24.48
4	04	15-9-2019	24.59	23.05	23.69	24.82	24.80	24.38
5	05	15-9-2019	24.40	25.00	23.52	24.79	24.87	24.40
6	06	15-9-2019	24.38	22.79	23.68	25.10	24.71	24.41
7	07	15-9-2019	26.72	25.61	24.92	26.56	25.03	24.91
8	08	15-9-2019	28.84	26.93	26.51	26.53	25.75	25.85
9	09	15-9-2019	30.29	28.72	27.48	26.95	26.64	26.79
10	10	15-9-2019	25.00	29.97	25.00	25.00	27.68	27.53
11	11	15-9-2019	32.05	28.93	26.86	27.38	28.43	28.98
12	12	15-9-2019	31.31	28.94	26.65	27.47	28.29	29.24
13	13	15-9-2019	30.95	25.00	27.83	27.44	28.00	30.66
14	14	15-9-2019	30.56	30.62	26.49	27.16	27.67	30.97
15	15	15-9-2019	31.13	30.58	26.29	26.68	27.29	30.59
16	16	15-9-2019	30.80	30.20	25.00	26.45	27.29	29.13

b.Xử lý missing data

df.fillna(method='pad') →
thay thế bằng giá trị liền trước

```

1 #PHƯƠNG PHÁP 2: Thay thế (Imputation)
2 #2.2)Thay thế các dữ liệu mất mát bằng giá trị liền trước của nó
3 data_new2 = data_temp.fillna(method='pad')
4 print(data_new2)

```

		time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
0	00	15-9-2019	25.65	24.79	24.01	25.06	25.48	24.97
1	01	15-9-2019	25.65	24.21	24.02	24.93	25.16	24.83
2	02	15-9-2019	25.05	23.73	23.89	24.79	24.80	24.55
3	03	15-9-2019	24.79	23.36	23.83	24.79	24.74	24.48
4	04	15-9-2019	24.59	23.05	23.69	24.82	24.80	24.38
5	05	15-9-2019	24.40	23.05	23.52	24.79	24.87	24.40
6	06	15-9-2019	24.38	22.79	23.68	25.10	24.71	24.41
7	07	15-9-2019	26.72	25.61	24.92	26.56	25.03	24.91
8	08	15-9-2019	28.84	26.93	26.51	26.53	25.75	25.85
9	09	15-9-2019	30.29	28.72	27.48	26.95	26.64	26.79
10	10	15-9-2019	30.29	29.97	27.48	26.95	27.68	27.53
11	11	15-9-2019	32.05	28.93	26.86	27.38	28.43	28.98
12	12	15-9-2019	31.31	28.94	26.65	27.47	28.29	29.24
13	13	15-9-2019	30.95	28.94	27.83	27.44	28.00	30.66
14	14	15-9-2019	30.56	30.62	26.49	27.16	27.67	30.97
15	15	15-9-2019	31.13	30.58	26.29	26.68	27.29	30.59
16	16	15-9-2019	30.80	30.20	26.29	26.45	27.29	29.13



b.Xử lý missing data

df.fillna(method='bfill')

→ thay thế bằng giá trị
liền sau

```
1 #PHƯƠNG PHÁP 2: Thay thế (Imputation)
2 #2.3)Thay thế các dữ liệu mất mát bằng giá trị liền sau của nó
3 data_new3 = data_temp.fillna(method='bfill')
4 print(data_new3)
```

	time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
0	00 15-9-2019	25.65	24.79	24.01	25.06	25.48	24.97
1	01 15-9-2019	25.05	24.21	24.02	24.93	25.16	24.83
2	02 15-9-2019	25.05	23.73	23.89	24.79	24.80	24.55
3	03 15-9-2019	24.79	23.36	23.83	24.82	24.74	24.48
4	04 15-9-2019	24.59	23.05	23.69	24.82	24.80	24.38
5	05 15-9-2019	24.40	22.79	23.52	24.79	24.87	24.40
6	06 15-9-2019	24.38	22.79	23.68	25.10	24.71	24.41
7	07 15-9-2019	26.72	25.61	24.92	26.56	25.03	24.91
8	08 15-9-2019	28.84	26.93	26.51	26.53	25.75	25.85
9	09 15-9-2019	30.29	28.72	27.48	26.95	26.64	26.79
10	10 15-9-2019	32.05	29.97	26.86	27.38	27.68	27.53
11	11 15-9-2019	32.05	28.93	26.86	27.38	28.43	28.98
12	12 15-9-2019	31.31	28.94	26.65	27.47	28.29	29.24
13	13 15-9-2019	30.95	30.62	27.83	27.44	28.00	30.66
14	14 15-9-2019	30.56	30.62	26.49	27.16	27.67	30.97
15	15 15-9-2019	31.13	30.58	26.29	26.68	27.29	30.59
16	16 15-9-2019	30.80	30.20	25.80	26.45	27.29	29.13



b.Xử lý missing data

df.interpolate() → thay thế giá trị bằng giá trị trung bình của giá trị liền trước và liền sau

```
1 #PHƯƠNG PHÁP 2: Thay thế (Imputation)
2 #2.4)Xử Lý các giá trị missing theo phương pháp nội suy
3 #Sử dụng hàm interpolate để thay thế giá trị missing với tham số:
4 #Thuật toán nội suy: Tuyến tính (linear)
5 #Hướng nội suy: Tiến lên (forward)
6 data_new4 = data_temp.interpolate(method='linear', limit_direction ='forward')
7 print(data_new4)
```

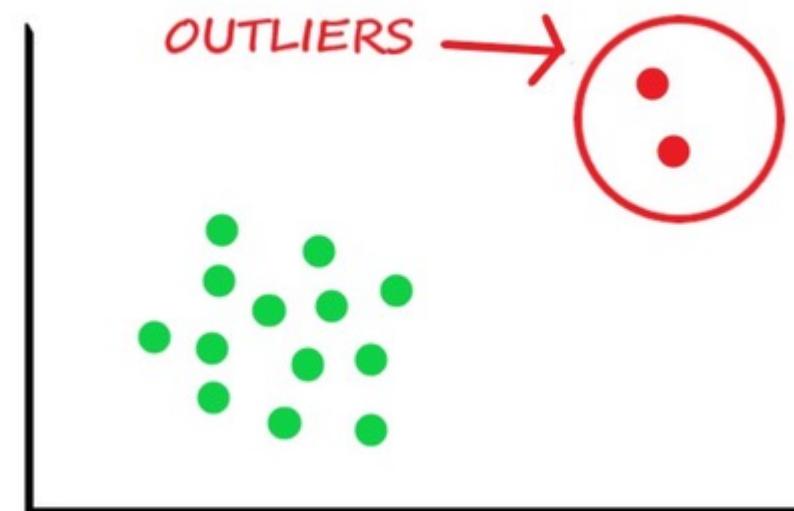
		time	Ha Noi	Vinh	Da Nang	Nha Trang	Ho Chi Minh	Ca Mau
0	00	15-9-2019	25.65	24.79	24.010	25.060	25.48	24.97
1	01	15-9-2019	25.35	24.21	24.020	24.930	25.16	24.83
2	02	15-9-2019	25.05	23.73	23.890	24.790	24.80	24.55
3	03	15-9-2019	24.79	23.36	23.830	24.805	24.74	24.48
4	04	15-9-2019	24.59	23.05	23.690	24.820	24.80	24.38
5	05	15-9-2019	24.40	22.92	23.520	24.790	24.87	24.40
6	06	15-9-2019	24.38	22.79	23.680	25.100	24.71	24.41
7	07	15-9-2019	26.72	25.61	24.920	26.560	25.03	24.91
8	08	15-9-2019	28.84	26.93	26.510	26.530	25.75	25.85
9	09	15-9-2019	30.29	28.72	27.480	26.950	26.64	26.79
10	10	15-9-2019	31.17	29.97	27.170	27.165	27.68	27.53
11	11	15-9-2019	32.05	28.93	26.860	27.380	28.43	28.98
12	12	15-9-2019	31.31	28.94	26.650	27.470	28.29	29.24
13	13	15-9-2019	30.95	29.78	27.830	27.440	28.00	30.66
14	14	15-9-2019	30.56	30.62	26.490	27.160	27.67	30.97
15	15	15-9-2019	31.13	30.58	26.290	26.680	27.29	30.59
16	16	15-9-2019	30.80	30.20	26.045	26.450	27.29	29.13



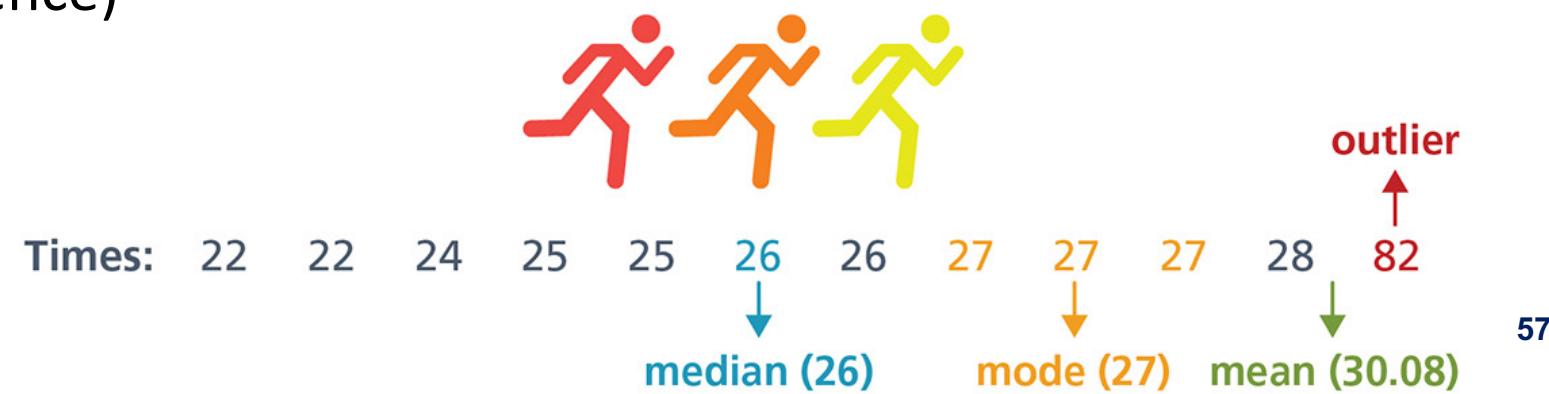
9. Phát hiện và xử lý ngoại lai

Giá trị ngoại lai (Outliers)

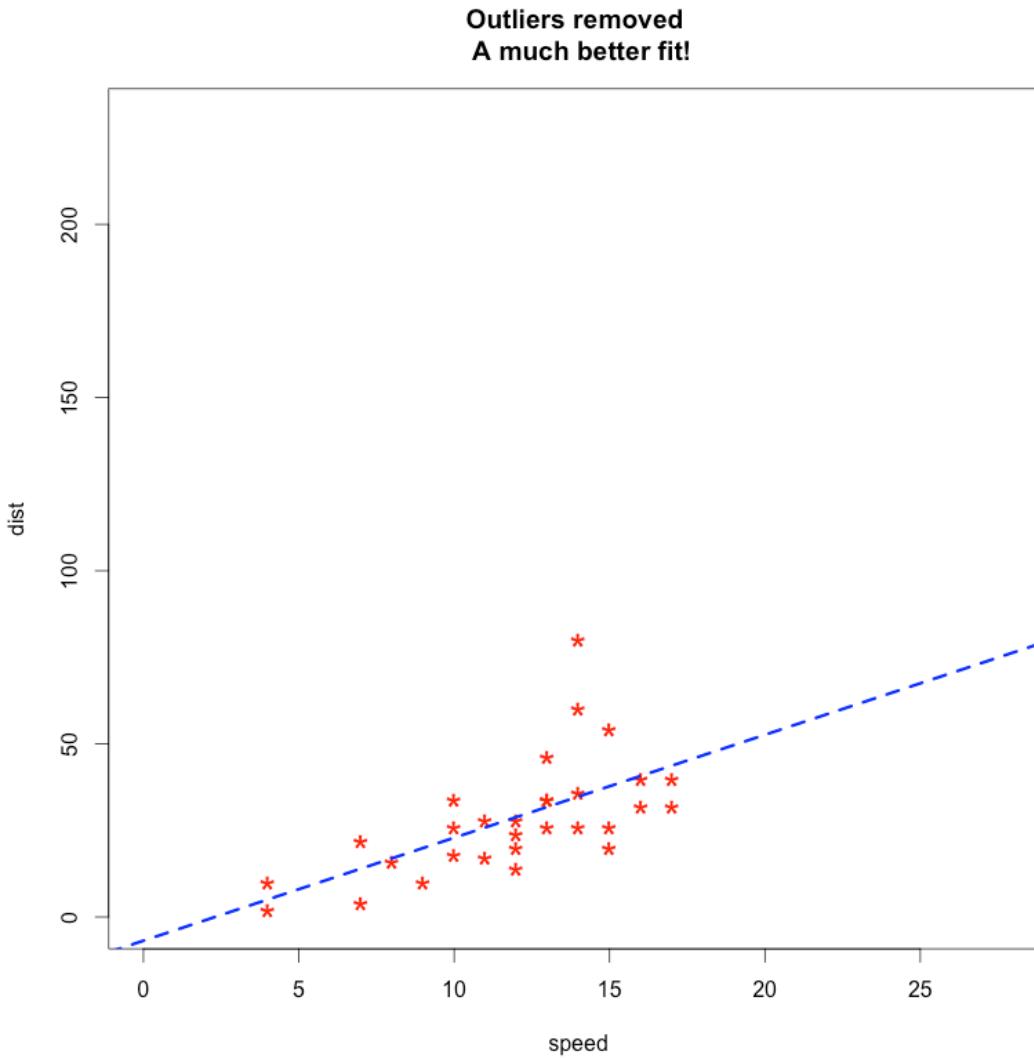
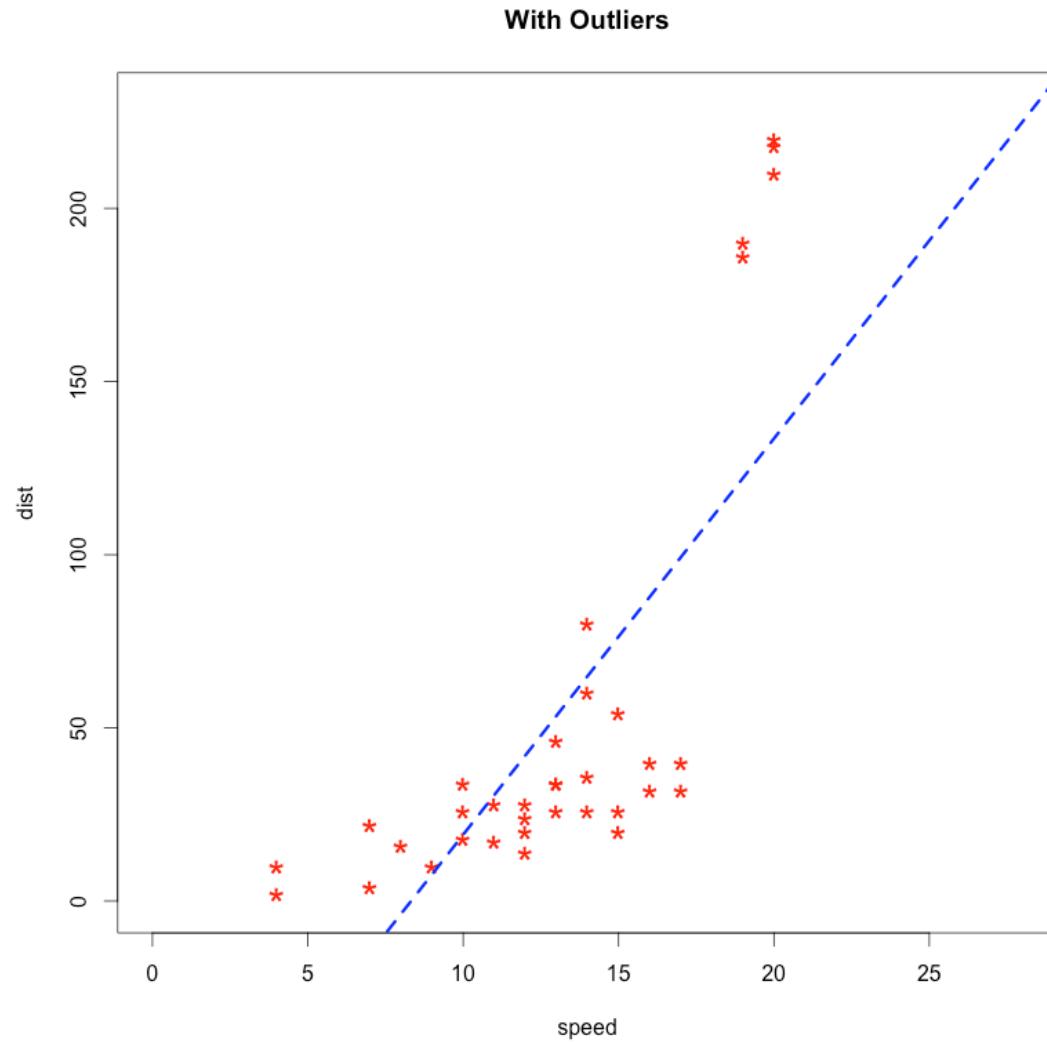
Một điểm ngoại lai là một điểm dữ liệu khác biệt đáng kể so với phần còn lại của tập dữ liệu. Ta thường xem các giá trị ngoại lai như là các mẫu dữ liệu đặc biệt, cách xa khỏi phần lớn dữ liệu khác trong tập dữ liệu



- Hệ thống phát hiện xâm nhập (Intrusion detection systems)
- Phát hiện gian lận tín dụng (Credit card fraud)
- Trong chuẩn đoán y tế (Medical diagnosis)
- Trong thực thi pháp luật (Law enforcement)
- Trong khoa học trái đất (Earth science)



Giá trị ngoại lai (Outliers)



Outliers

How to
identify?

How to
handle?

Box plot

Interquartile Range (IQR) based method

Standard Deviation based method

Histogram

Doing nothing

Deleting/Trimming

Winsorizing

Transformation

Binning

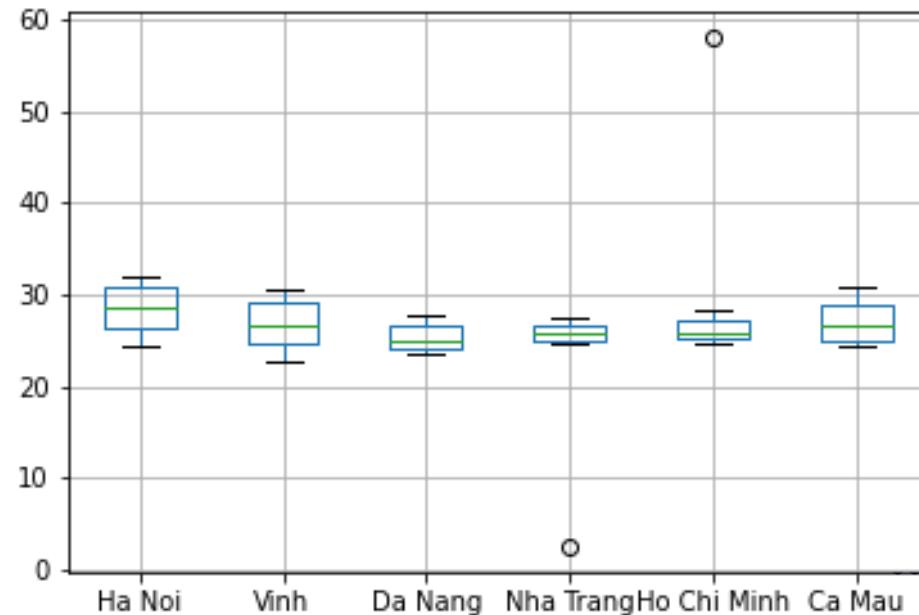
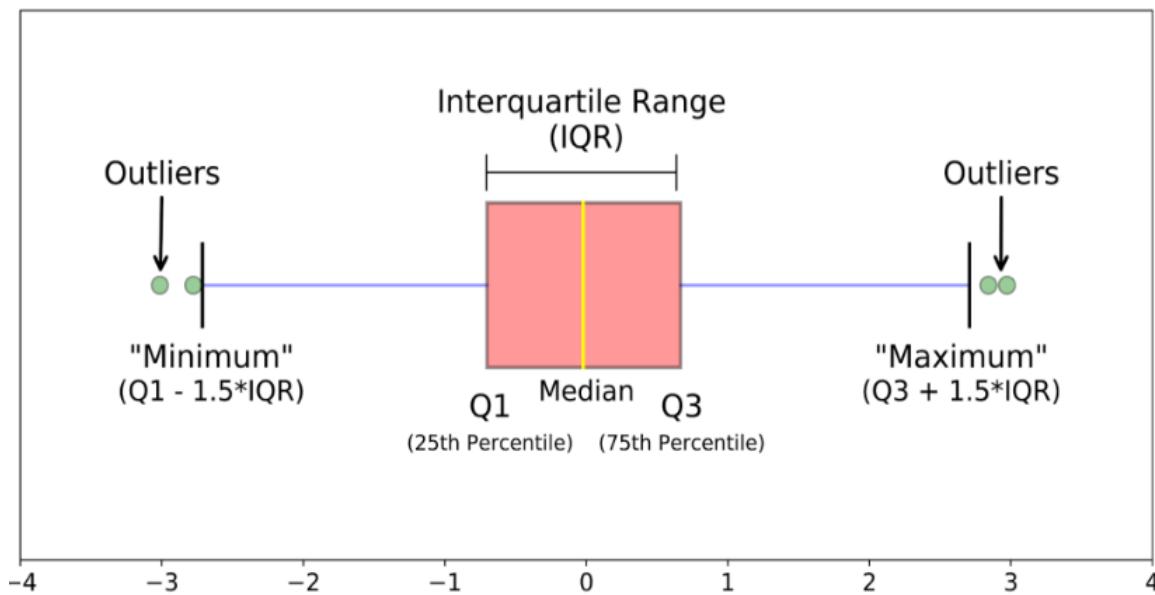
Use robust estimators

Imputing

Phát hiện Outliers

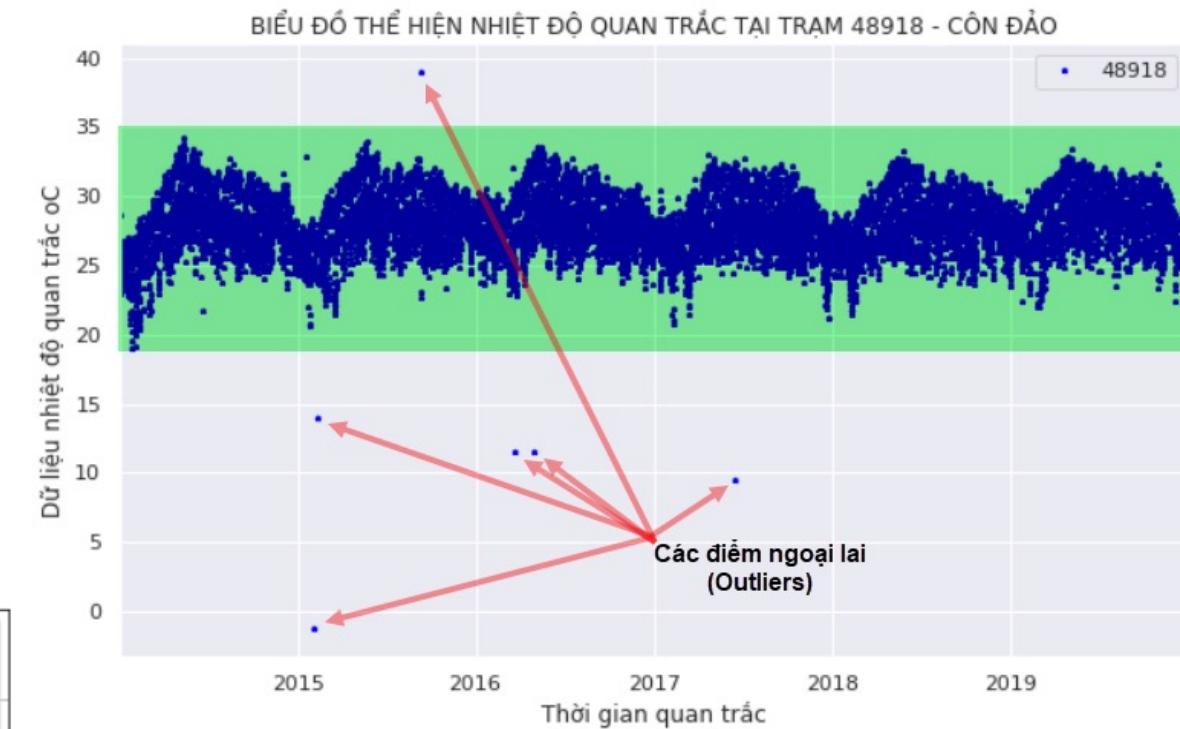
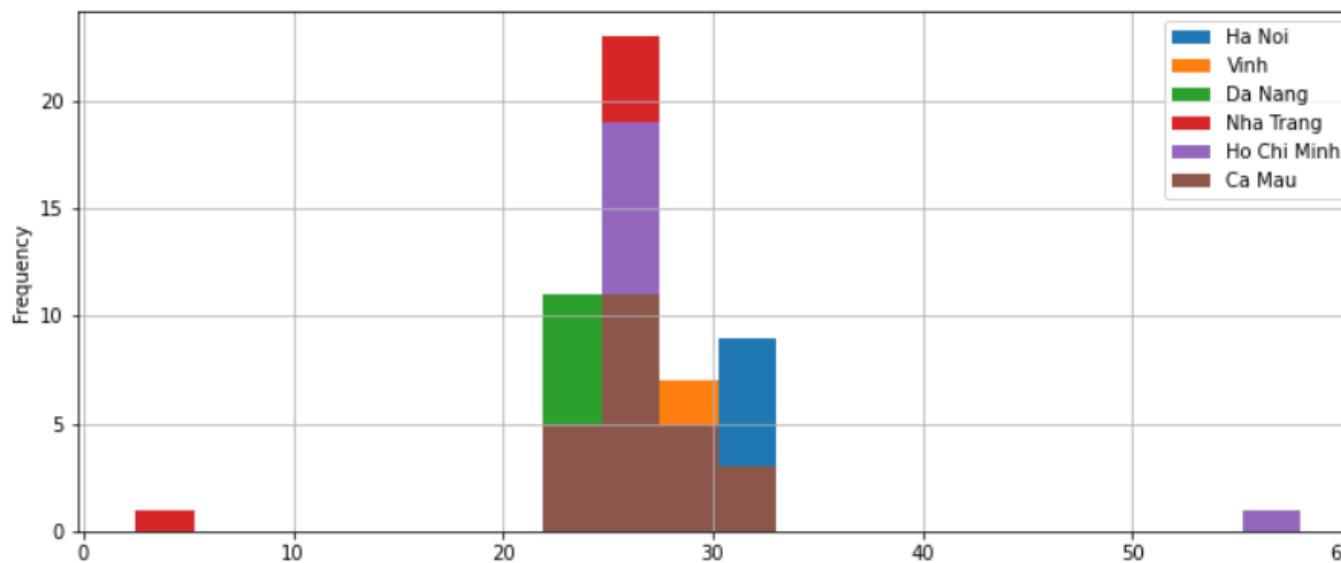
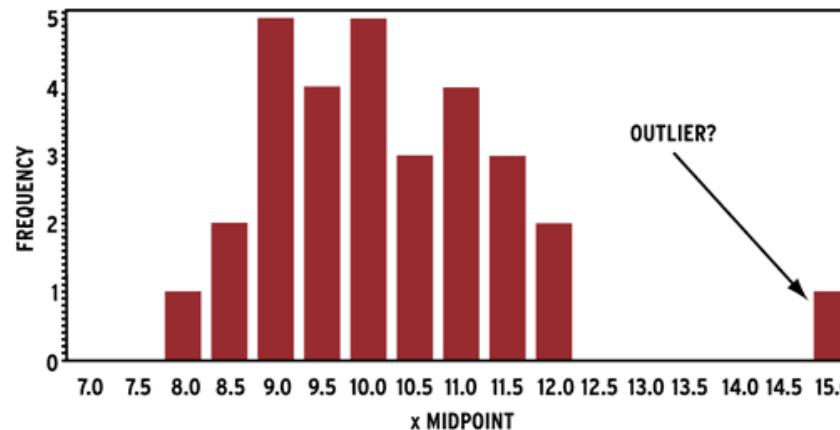
Biểu đồ Box-plot được sử dụng để đo khuynh hướng phân tán và xác định các giá trị ngoại lai của tập dữ liệu

- Giá trị bé nhất (**Minimum**) của tập dữ liệu được xác định bằng $Q1 - 1.5 * IQR$;
- Tứ phân vị thứ nhất (**Q1**) của tập dữ liệu
- Tứ phân vị thứ hai (**Q2**) chính là giá trị trung vị (Median) của tập dữ liệu
- Tứ phân vị thứ ba (**Q3**) của tập dữ liệu
- Giá trị lớn nhất (**Maximum**) của tập dữ liệu có giá trị bằng $Q3 + 1.5 * IQR$



Phát hiện Outliers

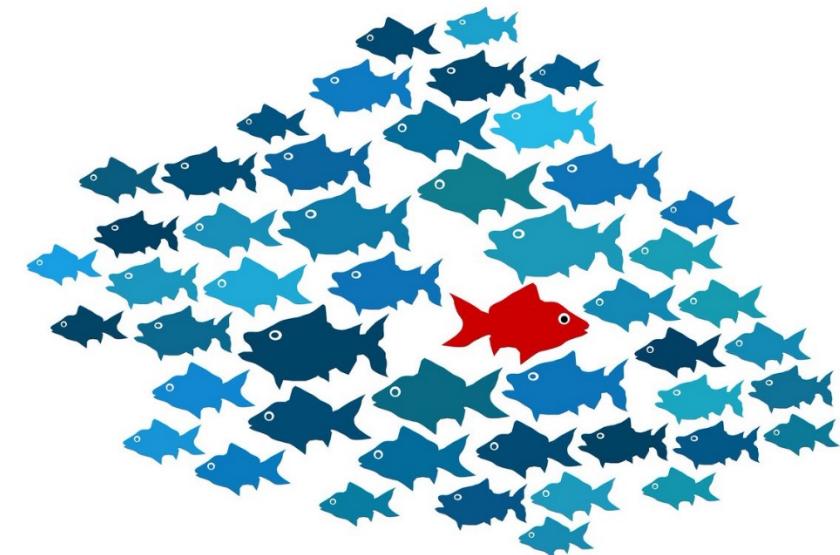
Sử dụng biểu đồ histogram;
Hoặc scatter (nhiều hơn 2 biến).



Xử lý Outliers

Không có một phương pháp, cách thức xử lý ngoại lai chung nào áp dụng cho tất cả các bài toán, các kiểu dữ liệu khác nhau.

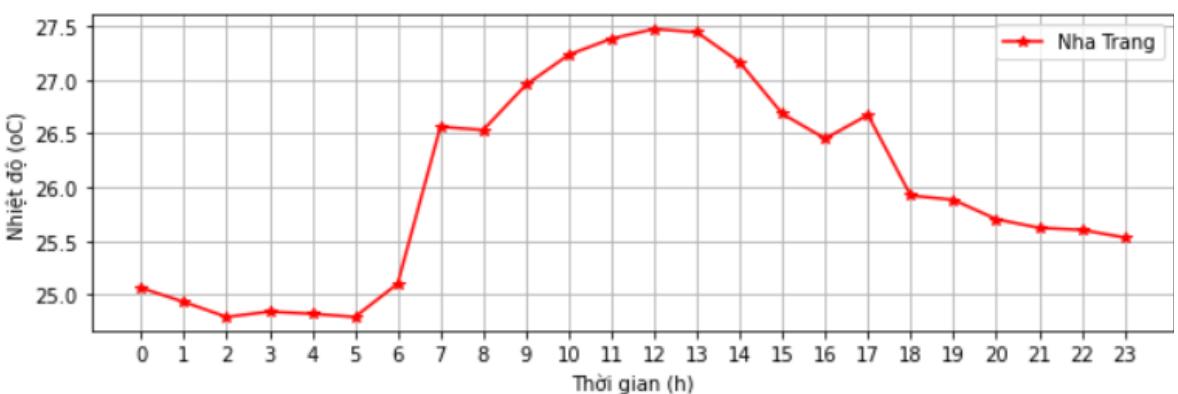
- Loại bỏ dòng dữ liệu chứa ngoại lai.
- Thay thế bằng một giá trị khác
- Thay thế giá trị ngoại lai về NAN (null) coi như là một điểm dữ liệu missing và xử lý như với dữ liệu missing



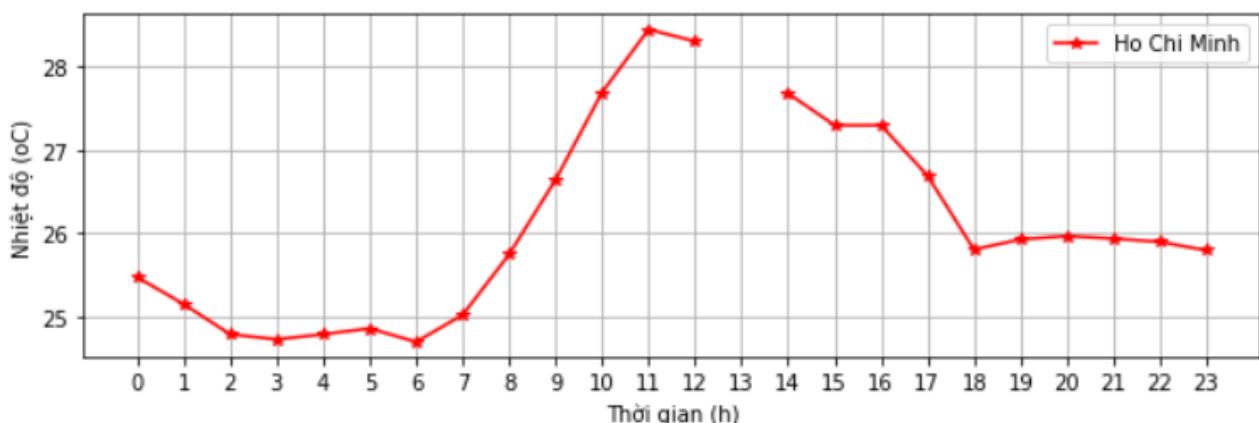
Để lựa chọn được phương pháp phù hợp cần có những hiểu biết sâu sắc về tập dữ liệu, về bài toán đang giải quyết, có thể sử dụng chỉ một phương pháp xử lý ngoại lai và/hoặc kết hợp cả 3 nhóm phương pháp đã chỉ ra ở trên để xử lý ngoại lai cho cùng một tập dữ liệu.

Xử lý Outliers

```
1 #Với giá trị ngoại lai trạm Nha Trang
2 #Xử lý bằng cách thay thế giá trị: 2.51 --> 25.1
3 x = np.arange(0,24)
4 plt.rcParams["figure.figsize"] = (10,3)
5 data_handling_outlier.loc[6,'Nha Trang'] = 25.10
6 data_handling_outlier[['Nha Trang']].plot(style='-*', color='red')
7 plt.xticks(x)
8 plt.xlabel('Thời gian (h)')
9 plt.ylabel('Nhiệt độ (oC)')
10 plt.grid(True)
11 plt.show()
```



```
1 #Với giá trị ngoại lai trạm Hồ Chí Minh
2 #Xử lý bằng cách chuyển về giá trị Null - coi như giá trị missing
3 data_handling_outlier.loc[13,'Ho Chi Minh'] = np.NaN
4 data_handling_outlier[['Ho Chi Minh']].plot(style='-*', color='red')
5 plt.xticks(x)
6 plt.xlabel('Thời gian (h)')
7 plt.ylabel('Nhiệt độ (oC)')
8 plt.grid(True)
9 plt.show()
```



Thực hành 3

10. Phân tích dữ liệu bán hàng

(Tiếp cận từ bài toán thực tế)

Phân tích dữ liệu bán hàng

Dữ liệu bao gồm 12 file của một chuỗi cửa hàng kinh doanh thiết bị điện tử; Mỗi file dữ liệu lưu lại toàn bộ các đơn hàng đã bán được theo từng tháng tương ứng của năm 2019.

- * Dữ liệu tháng 1: Sales_January_2019.csv
- * Dữ liệu tháng 2: Sales_February_2019.csv
- * Dữ liệu tháng 3: Sales_March_2019.csv
- * Dữ liệu tháng 4: Sales_April_2019.csv
- * Dữ liệu tháng 5: Sales_May_2019.csv
- * Dữ liệu tháng 6: Sales_June_2019.csv
- * Dữ liệu tháng 7: Sales_July_2019.csv
- * Dữ liệu tháng 8: Sales_August_2019.csv
- * Dữ liệu tháng 9: Sales_September_2019.csv
- * Dữ liệu tháng 10: Sales_October_2019.csv
- * Dữ liệu tháng 11: Sales_November_2019.csv
- * Dữ liệu tháng 12: Sales_December_2019.csv



Phân tích dữ liệu bán hàng

Mỗi file dữ liệu bao gồm 6 thông tin:

1. Order ID: Mã đơn hàng
2. Product: Tên sản phẩm đã bán theo từng đơn hàng
3. Quantity Ordered: Số lượng sản phẩm bán
4. Price Each: Giá của mỗi sản phẩm
5. Order Date: Thời gian bán hàng
6. Purchase Address: Địa chỉ mua hàng

Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
141234	iPhone	1	700	01/22/19 21:25	944 Walnut St, Boston, MA 02215
141235	Lightning Charging Cable	1	14.95	01/28/19 14:15	185 Maple St, Portland, OR 97035
141236	Wired Headphones	2	11.99	01/17/19 13:33	538 Adams St, San Francisco, CA 94016
141237	27in FHD Monitor	1	149.99	01/05/19 20:33	738 10th St, Los Angeles, CA 90001
141238	Wired Headphones	1	11.99	01/25/19 11:59	387 10th St, Austin, TX 73301

Mục tiêu:

- Đọc dữ liệu từ nhiều file, Sử dụng các kỹ thuật làm sạch và chuẩn bị dữ liệu để phân tích
- Thực hiện phân tích tìm ra các thông tin có ích (insights) từ dữ liệu



Kết quả

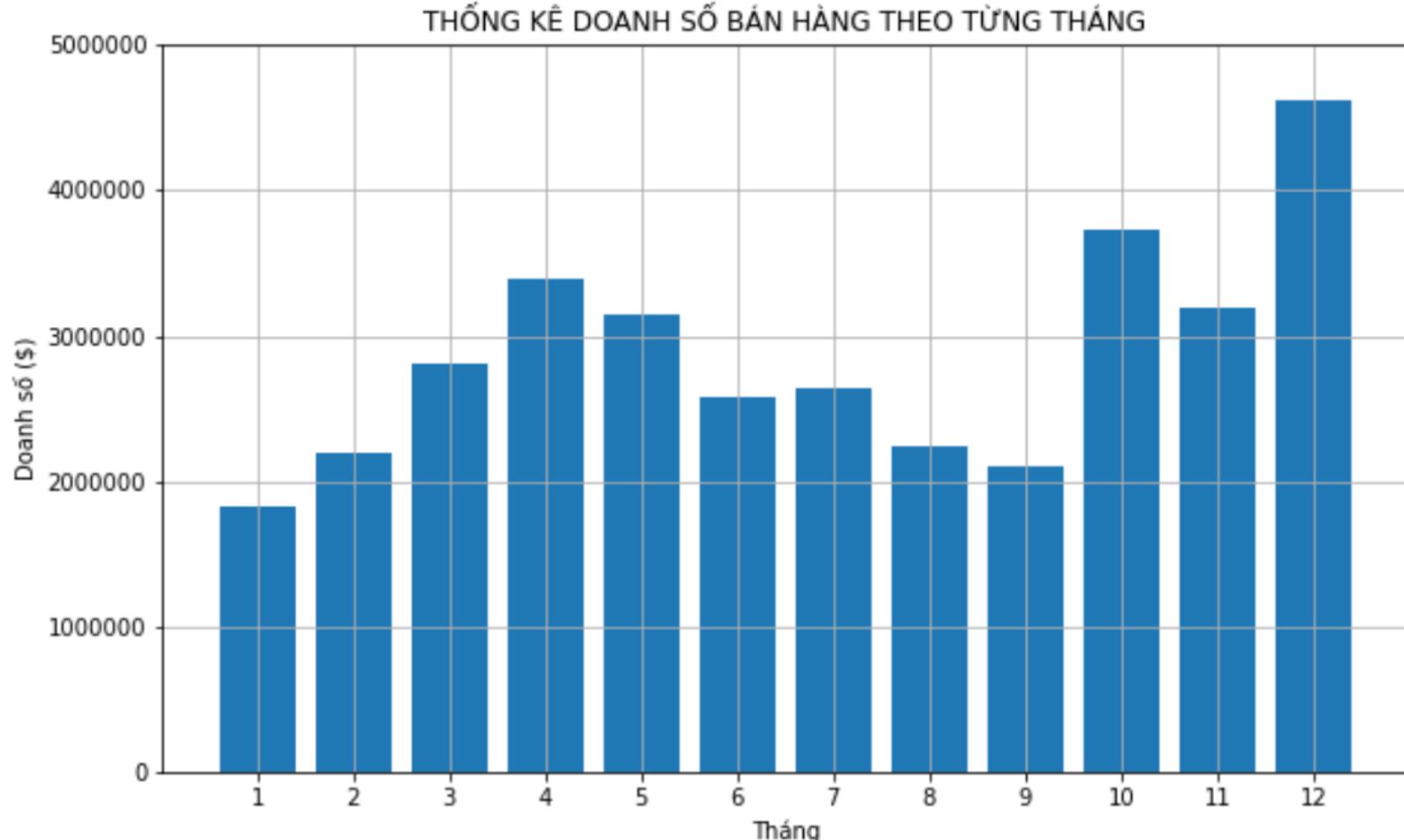
Tập dữ liệu sau khi được tiền chuẩn hóa và làm sạch:

Order ID	Product	Quantity	Price Each	Order Date	Purchase Address	Month	Hour	City	Sales
141234	iPhone	1	700	2019-01-22 21:25	944 Walnut St, Boston, MA 02215	1	21	Boston (MA)	700
141235	Lightning Charging Cable	1	14.95	2019-01-28 14:15	185 Maple St, Portland, OR 97035	1	14	Portland (OR)	14.95
141236	Wired Headphones	2	11.99	2019-01-17 13:33	538 Adams St, San Francisco, CA 94016	1	13	San Francisco (CA)	23.98
141237	27in FHD Monitor	1	149.99	2019-01-05 20:33	738 10th St, Los Angeles, CA 90001	1	20	Los Angeles (CA)	149.99
141238	Wired Headphones	1	11.99	2019-01-25 11:59	387 10th St, Austin, TX 73301	1	11	Austin (TX)	11.99
141239	AAA Batteries (4-pack)	1	2.99	2019-01-29 20:22	775 Willow St, San Francisco, CA 94016	1	20	San Francisco (CA)	2.99
141240	27in 4K Gaming Monitor	1	389.99	2019-01-26 12:16	979 Park St, Los Angeles, CA 90001	1	12	Los Angeles (CA)	389.99
141241	USB-C Charging Cable	1	11.95	2019-01-05 12:04	181 6th St, San Francisco, CA 94016	1	12	San Francisco (CA)	11.95
141242	Bose SoundSport Headphones	1	99.99	2019-01-01 10:30	867 Willow St, Los Angeles, CA 90001	1	10	Los Angeles (CA)	99.99
141243	Apple Airpods Headphones	1	150	2019-01-22 21:20	657 Johnson St, San Francisco, CA 94016	1	21	San Francisco (CA)	150
141244	Apple Airpods Headphones	1	150	2019-01-07 11:29	492 Walnut St, San Francisco, CA 94016	1	11	San Francisco (CA)	150
141245	Macbook Pro Laptop	1	1700	2019-01-31 10:12	322 6th St, San Francisco, CA 94016	1	10	San Francisco (CA)	1700
141246	AAA Batteries (4-pack)	3	2.99	2019-01-09 18:57	618 7th St, Los Angeles, CA 90001	1	18	Los Angeles (CA)	8.97
141247	27in FHD Monitor	1	149.99	2019-01-25 19:19	512 Wilson St, San Francisco, CA 94016	1	19	San Francisco (CA)	149.99
141248	Flatscreen TV	1	300	2019-01-03 21:54	363 Spruce St, Austin, TX 73301	1	21	Austin (TX)	300
141249	27in FHD Monitor	1	149.99	2019-01-05 17:20	440 Cedar St, Portland, OR 97035	1	17	Portland (OR)	149.99
141250	Vareebadd Phone	1	400	2019-01-10 11:20	471 Center St, Los Angeles, CA 90001	1	11	Los Angeles (CA)	400
141251	Apple Airpods Headphones	1	150	2019-01-24 8:13	414 Walnut St, Boston, MA 02215	1	8	Boston (MA)	150
141252	USB-C Charging Cable	1	11.95	2019-01-30 9:28	220 9th St, Los Angeles, CA 90001	1	9	Los Angeles (CA)	11.95
141253	AA Batteries (4-pack)	1	3.84	2019-01-17 0:09	385 11th St, Atlanta, GA 30301	1	0	Atlanta (GA)	3.84
141254	AAA Batteries (4-pack)	1	2.99	2019-01-08 11:51	238 Sunset St, Seattle, WA 98101	1	11	Seattle (WA)	2.99
141255	USB-C Charging Cable	1	11.95	2019-01-09 20:55	764 11th St, Los Angeles, CA 90001	1	20	Los Angeles (CA)	11.95
141256	Google Phone	1	600	2019-01-29 10:40	675 Washington St, Portland, OR 97035	1	10	Portland (OR)	600
141257	Apple Airpods Headphones	1	150	2019-01-12 18:51	338 Highland St, San Francisco, CA 94016	1	18	San Francisco (CA)	150

Kết quả

Câu hỏi 1: Tháng nào trong năm có doanh số bán hàng cao nhất - thấp nhất?

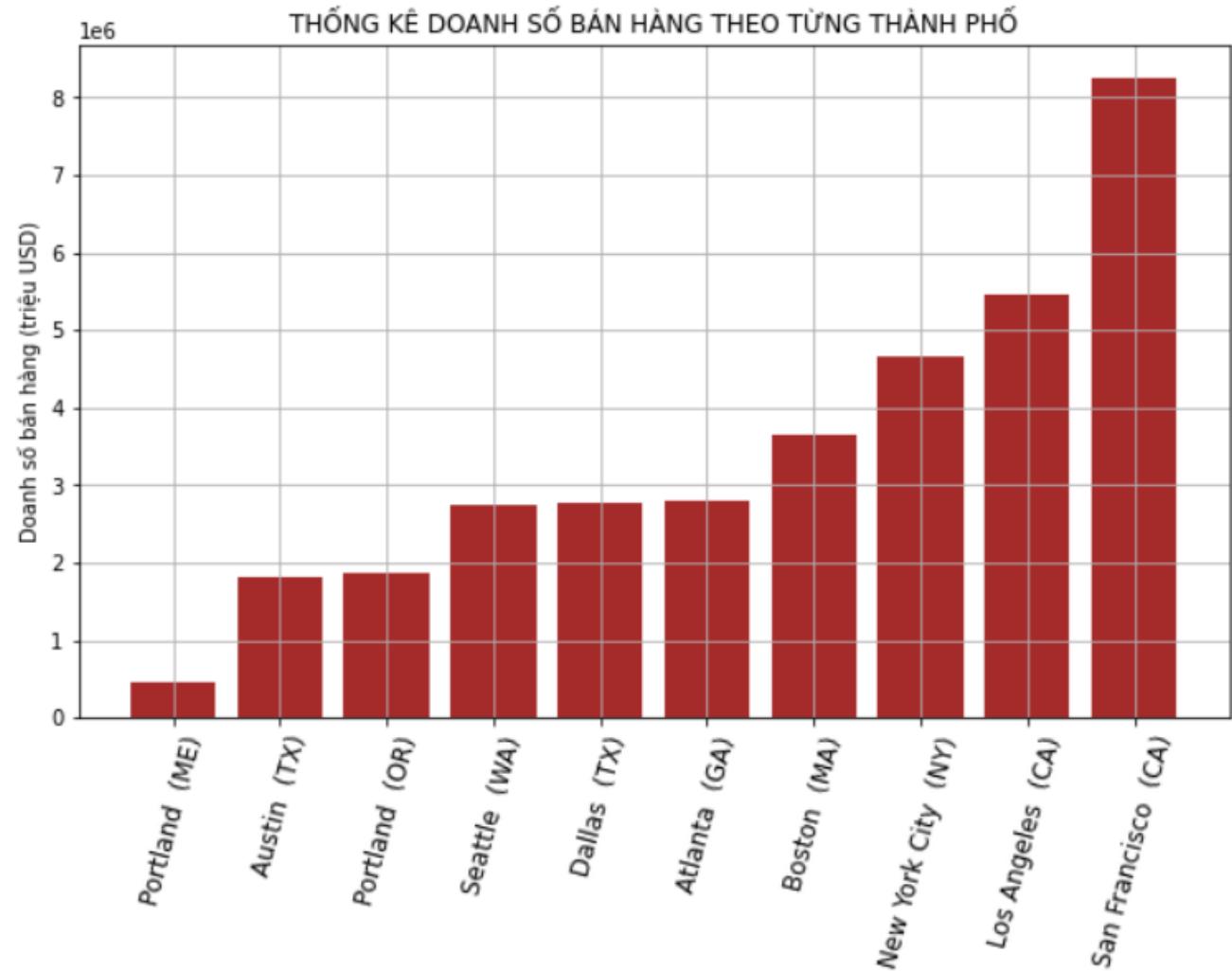
W
Insight!



Kết quả

Câu hỏi 2: Cửa hàng ở thành phố nào bán được hàng nhiều nhất?

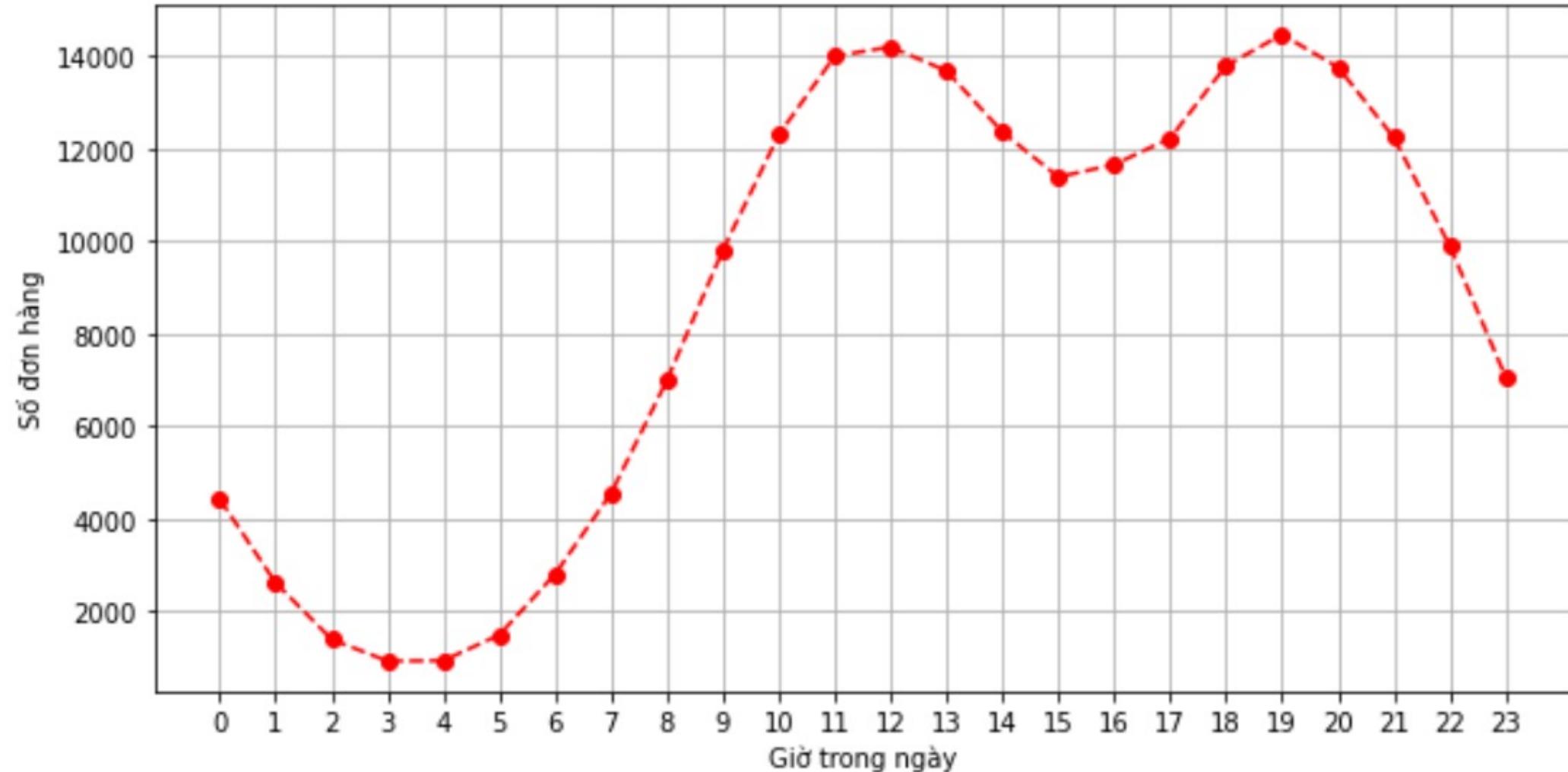
City	Sales
Portland (ME)	4.497583e+05
Austin (TX)	1.819582e+06
Portland (OR)	1.870732e+06
Seattle (WA)	2.747755e+06
Dallas (TX)	2.767975e+06
Atlanta (GA)	2.795499e+06
Boston (MA)	3.661642e+06
New York City (NY)	4.664317e+06
Los Angeles (CA)	5.452571e+06
San Francisco (CA)	8.262204e+06



W
Insight!

Kết quả

Câu hỏi 3: Khách hàng mua sản phẩm thường tập trung vào khung giờ nào trong ngày?



Insight!

Câu hỏi 4: Trong các hóa đơn của Khách hàng, Sản phẩm nào thường được mua cùng với nhau?

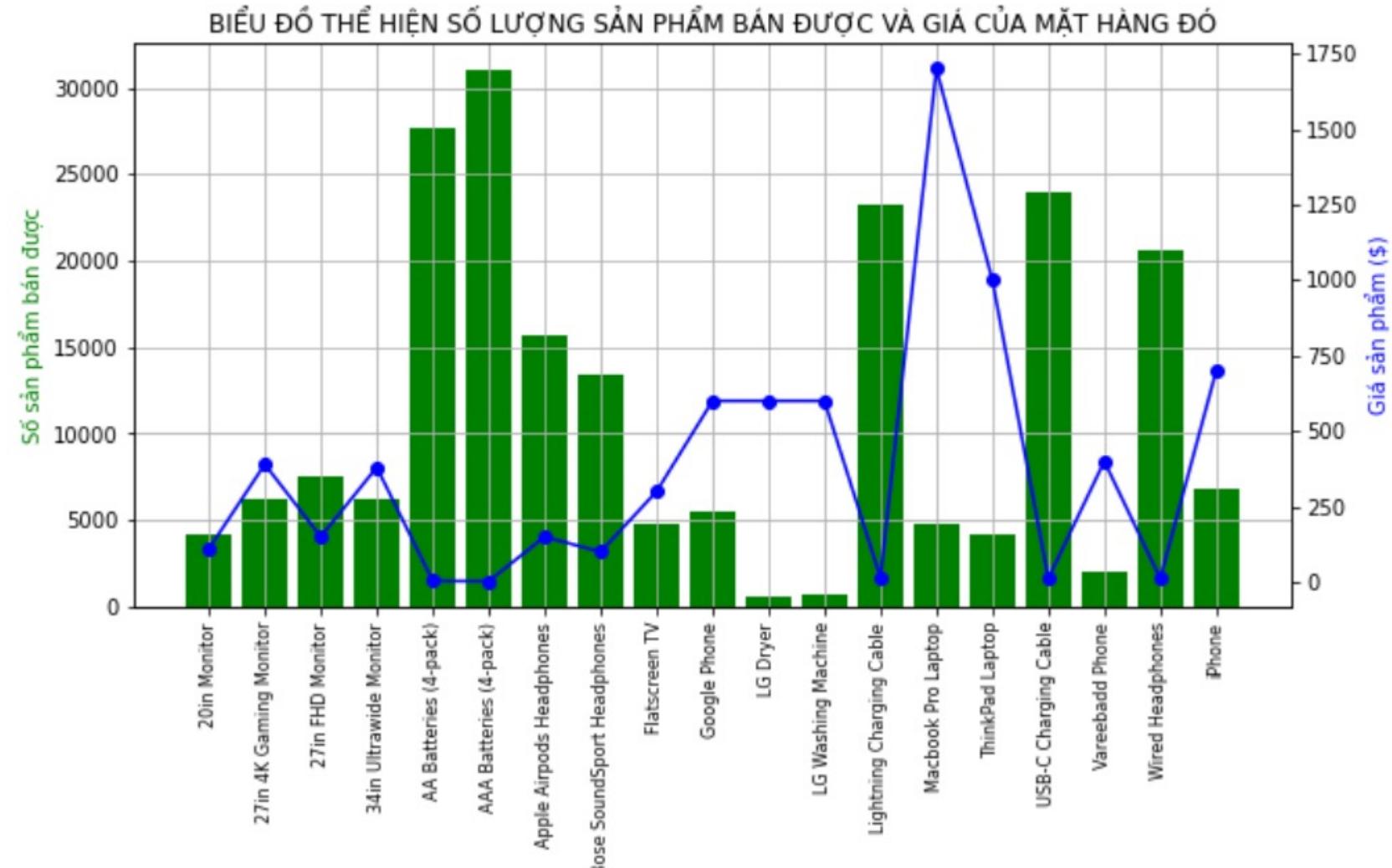
('iPhone', 'Lightning Charging Cable') 1005
('Google Phone', 'USB-C Charging Cable') 987
('iPhone', 'Wired Headphones') 447
('Google Phone', 'Wired Headphones') 414
('Vareebadd Phone', 'USB-C Charging Cable') 361
('iPhone', 'Apple Airpods Headphones') 360
('Google Phone', 'Bose SoundSport Headphones') 220
('USB-C Charging Cable', 'Wired Headphones') 160
('Vareebadd Phone', 'Wired Headphones') 143
('Lightning Charging Cable', 'Wired Headphones') 92
('Lightning Charging Cable', 'Apple Airpods Headphones') 81
('Vareebadd Phone', 'Bose SoundSport Headphones') 80
('USB-C Charging Cable', 'Bose SoundSport Headphones') 77
('Apple Airpods Headphones', 'Wired Headphones') 69
('Lightning Charging Cable', 'USB-C Charging Cable') 58



Kết quả

Câu hỏi 5: Sản phẩm nào của chuỗi cửa hàng bán được số lượng nhiều nhất?
Tại sao?

W
Insight!





Q & A
Thank you!