

Bài 6:

LẬP TRÌNH PYTHON CƠ BẢN

(Phân tích và xử lý dữ liệu với Pandas - 01)

AI Academy Vietnam

Nội dung bài 6

1. Giới thiệu Pandas
2. Tạo đối tượng cơ bản trong Pandas
 - Series
 - Dataframe
3. Đọc dữ liệu từ các nguồn khác nhau
4. Quan sát và truy xuất dữ liệu trong DataFrame
5. Replacing Values, Rename Columns
6. Lọc dữ liệu trong DataFrame
7. Xác định các tham số thống kê: Sum, Cumsum, Min, Max, Mean, Median, Std
8. Giá trị duy nhất (Unique)
9. Phân tích Time series data (Tiếp cận từ bài toán thực tế)

Pandas



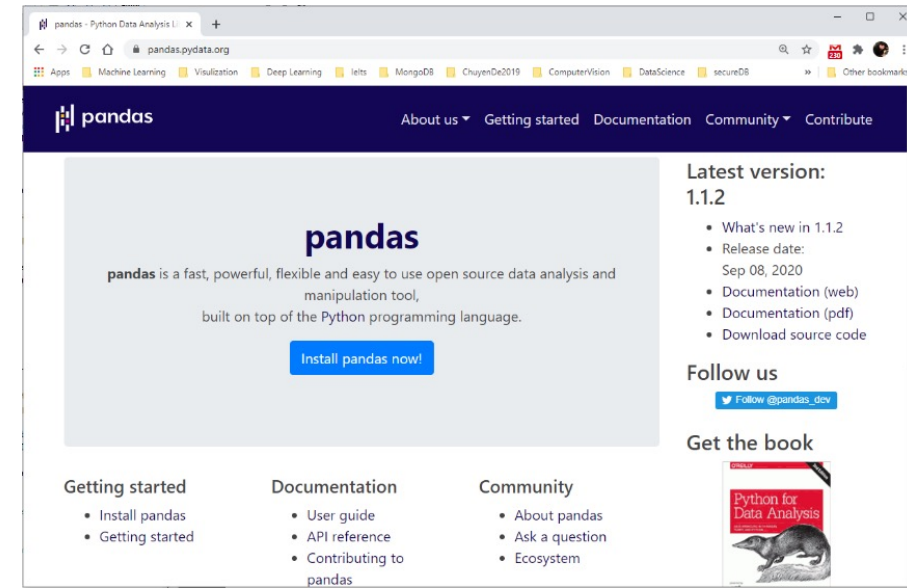
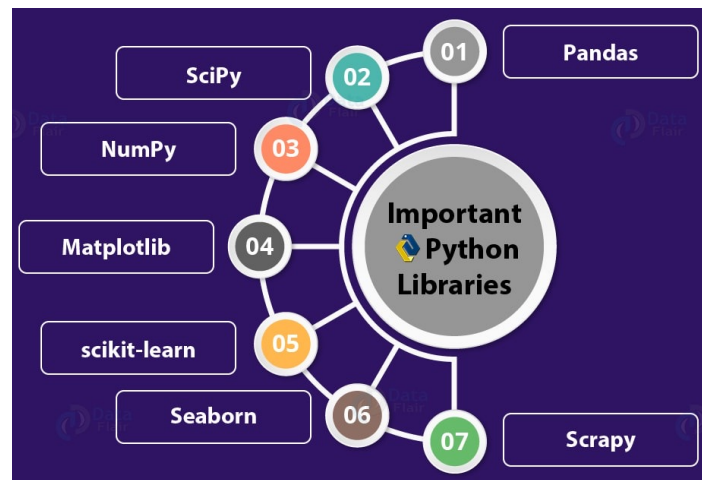
1. Giới thiệu Pandas

1. Giới thiệu

Pandas là một thư viện mã nguồn mở được xây dựng dựa trên NumPy, sử dụng để thao tác và phân tích dữ liệu. Với Pandas chúng ta có thể:

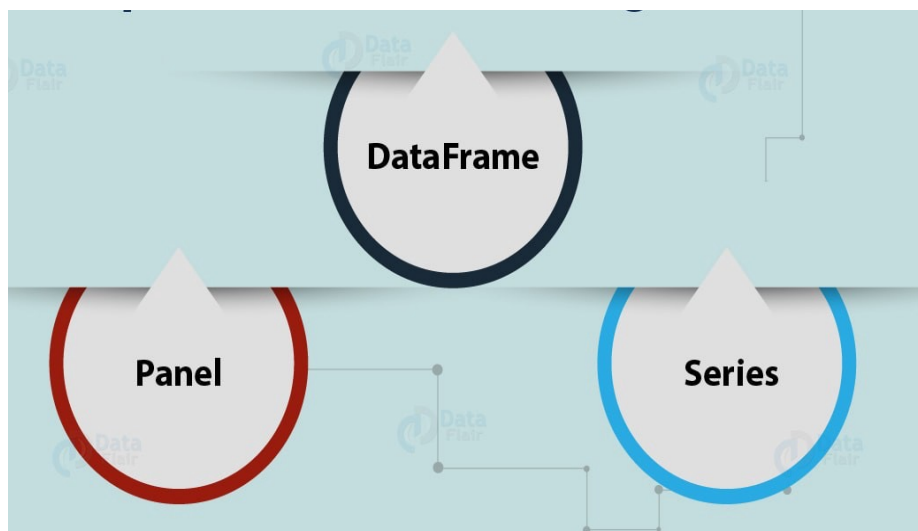
- Xử lý tập dữ liệu khác nhau về định dạng: chuỗi thời gian, bảng không đồng nhất, ma trận dữ liệu
- Import dữ liệu từ nhiều nguồn khác nhau như CSV, DB/SQL...
- Xử lý vô số phép toán cho tập dữ liệu: subsetting, slicing, filtering, merging, groupBy, re-ordering, and re-shaping,...
- Xử lý dữ liệu mất mát theo mong muốn.
- Xử lý, phân tích dữ liệu tốt như mô hình hoá và thống kê.
- Tích hợp tốt với các thư viện khác của python.

<https://pandas.pydata.org/>



1. Giới thiệu Pandas

Pandas làm việc thông qua 3 đối tượng Series, **DataFrame**, Panel



Trong ba kiểu dữ liệu, DataFrame là kiểu dữ liệu được sử dụng rộng rãi nhất.

```
1 #Kiểm tra phiên bản của thư viện Pandas
2 import pandas as pd
3 print('Version Pandas: ',pd.__version__)
```

Version Pandas: 1.1.1

SERIES

7	2	9	10
---	---	---	----

axis 0 →

DATA FRAME

axis 0 ↓	5.2	3.0	4.5
	9.1	0.1	0.3

axis 1 →

PANEL

axis 0 ↓	1	2	3	4
	4	7	7	5
	1	3	0	2
	9	6	9	8

axis 1 → axis 2 →

2. Series, DataFrame trong Pandas

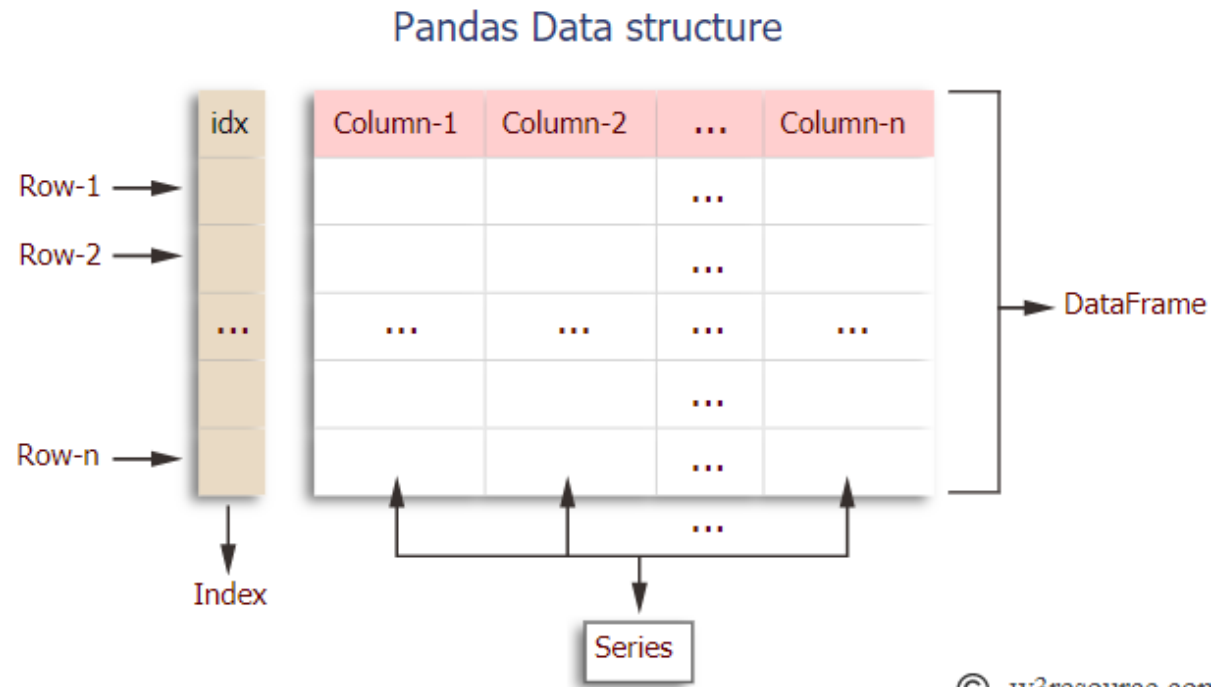
2.1 Series

- Series** là mảng một chiều (1D) giống như kiểu vector trong Numpy, hay như một cột của một bảng, nhưng nó bao gồm thêm một bảng đánh index.

		Data
Index	0	2.80
	1	3.00
	2	4.44
	3	5.00

dtype: float64

Series



2.1 Series

- Tạo Series sử dụng phương thức;
 - `pd.Series(data, index, dtype, name)`

```
1 #Tạo một đối tượng series
2 #index mặc định đánh số từ 0
3 data = pd.Series([2.8, 3, 4.44, 5])
4 data
```

```
0    2.80
1    3.00
2    4.44
3    5.00
dtype: float64
```

```
1 #Mỗi một đối tượng series bao gồm 2 thành phần
2 #1. Values
3 #2. index
4
5 print('Values:', data.values)
6 print('Indices:', data.index)
```

```
Values: [2.8  3.   4.44 5. ]
Indices: RangeIndex(start=0, stop=4, step=1)
```

```
1 #Tạo một đối tượng series với index thiết lập
2 data = pd.Series([1.25, 2, 3.5, 4.75, 8.0],
3                  index=['a', 'b', 'c', 'd', 'k'])
4 data
```

```
a    1.25
b    2.00
c    3.50
d    4.75
k    8.00
dtype: float64
```

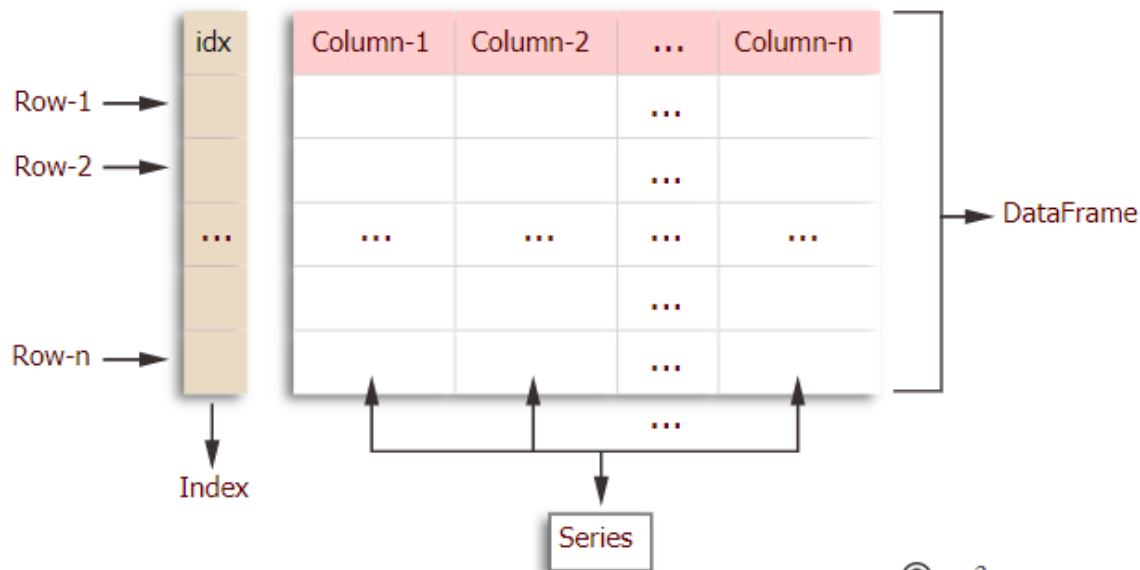
```
1 print('Values:', data.values)
2 print('Indices:', data.index)
```

```
Values: [0.25 0.5  0.75 1. ]
Indices: Index(['a', 'b', 'c', 'd'], dtype='object')
```


2.2 DataFrame

DataFrame: Cấu trúc dạng bảng 2D, kích thước có thể thay đổi được. Dữ liệu một cột là đồng nhất nhưng có thể không đồng nhất giữa các cột

Pandas Data structure



The diagram shows a table with columns labeled 'Name', 'Team', 'Number', 'Position', and 'Age'. The rows are indexed from 0 to 6. Arrows point from the label 'Columns' to the column headers. Arrows point from the label 'Rows' to the row indices. A purple box labeled 'Data' highlights the data cells in the table.

	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

2.2 DataFrame

- Tạo DataFrame sử dụng phương thức;

- `pd.DataFrame(data, index, columns, dtype)`

```
1 #Tạo một DataFrame từ một biến Dict
2 #Chỉ số được tạo mặc định từ 0
3 data_dict = {
4     'apples': [3, 2, 0, 1],
5     'oranges': [0, 3, 7, 2]}
6
7 purchases = pd.DataFrame(data_dict)
8 purchases
```

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

```
1 #Tạo DataFrame với index thiết lập
2 purchases = pd.DataFrame(data_dict,
3                           index=['June', 'Robert', 'Lily', 'David'])
4 purchases
```

	apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2

3. Đọc dữ liệu từ các nguồn khác nhau (CSV, Text, Excel)

3.1 Đọc file CSV, Text

- CSV là một định dạng dữ liệu văn bản đơn giản có tên đầy đủ là Comma Separated Values. Với định dạng CSV này, các giá trị được chia tách với nhau bởi các dấu phẩy. Định dạng CSV phổ biến bởi vì chúng có tính tương thích cao, dễ dàng di chuyển từ phần mềm này sang phần mềm khác để sử dụng mà không lo gặp các xung đột.
- Tài liệu CSV cũng làm một trong những tài liệu phổ biến trên thế giới với khả năng lưu trữ nhỏ nhẹ.



What Is A CSV File?

Example CSV

File Edit View Insert Format Data Tools Add-ons Help [All changes saved in Drive](#)

100% \$ % .0 .00 123 Arial 10 B I S A

	A	B	C	D	E	F
1	Email	First Name	Last Name	Company	Snippet 1	
2	example1@domain.com	John	Smith	Company 1	Snippet Sentence1	
3	example2@gmail.com	Mary	Blake	Company 2	Snippet Sentence 2	
4	example3@outlook.com	James	Joyce	Company 3	Snippet Sentence 3	
5						
6						
7						
8						

CSV file

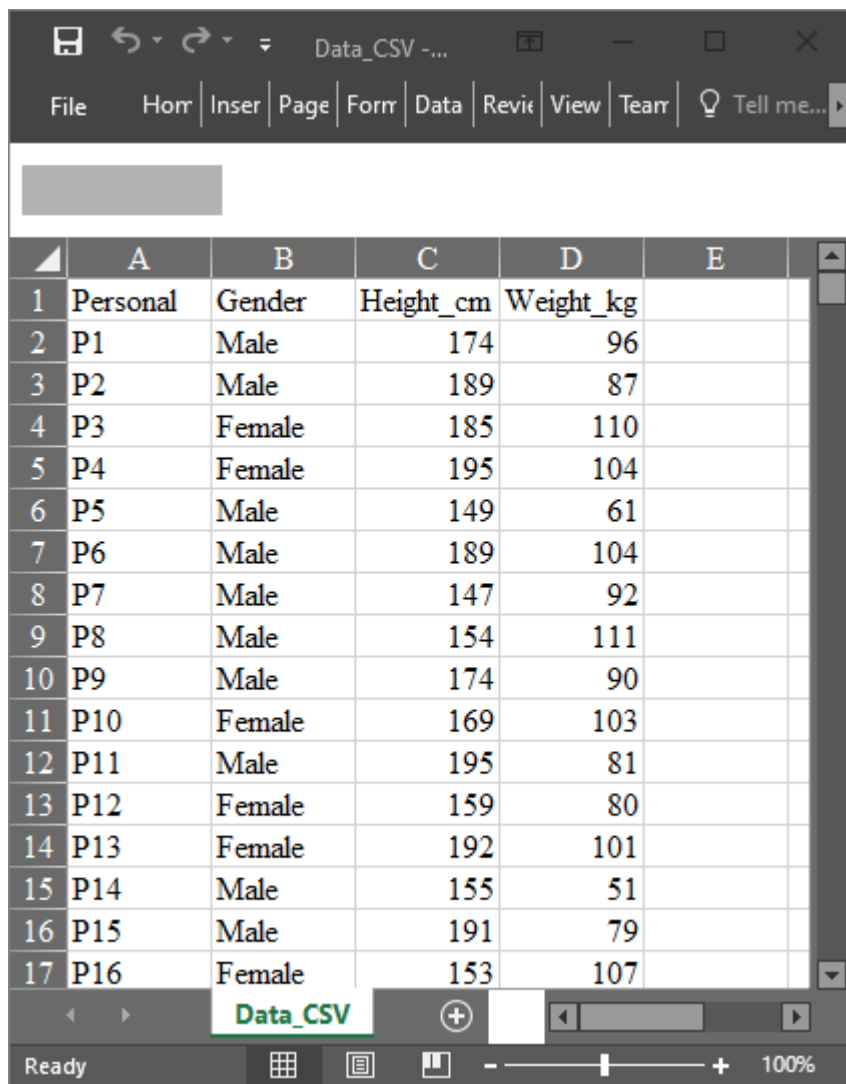
Example CSV - Sheet1 — Notatnik

Plik Edycja Format Widok Pomoc

Email,First Name,Last Name,Company,Snippet 1
example1@domain.com,John,Smith,Company 1,Snippet Sentence1
example2@gmail.com,Mary,Blake,Company 2,Snippet Sentence 2
example3@outlook.com,James,Joyce,Company 3,Snippet Sentence 3

3.1 Đọc file CSV, Text

Sử dụng phương thức read_csv() đọc dữ liệu từ file .CSV



	A	B	C	D	E
1	Personal	Gender	Height_cm	Weight_kg	
2	P1	Male	174	96	
3	P2	Male	189	87	
4	P3	Female	185	110	
5	P4	Female	195	104	
6	P5	Male	149	61	
7	P6	Male	189	104	
8	P7	Male	147	92	
9	P8	Male	154	111	
10	P9	Male	174	90	
11	P10	Female	169	103	
12	P11	Male	195	81	
13	P12	Female	159	80	
14	P13	Female	192	101	
15	P14	Male	155	51	
16	P15	Male	191	79	
17	P16	Female	153	107	

```
1 import pandas as pd
2 path = 'Data_Excercise\CSV\Data_CSV.csv'
3 #Sử dụng phương thức read_csv
4 data = pd.read_csv(path)
5 #Hiển thị thông tin biến Data
6 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Personal    500 non-null    object
1   Gender      500 non-null    object
2   Height_cm   500 non-null    int64
3   Weight_kg   500 non-null    int64
dtypes: int64(2), object(2)
memory usage: 15.8+ KB
```

3.1 Đọc file CSV, Text

Sử dụng phương thức `read_csv()` có rất nhiều tham số khác nhau để thiết lập cách thức đọc file .csv

```
1 import numpy as np
2 import pandas as pd
3
4
5 df_loan = pd.read_csv("loan.csv", sep=";",
6 encoding = "ISO-8859-1",
7 index_col=None,
8 low_memory=False,
9 dtype={'id':np.int32}, nrows=16, skiprows=0)
10 df_loan.head(3)
```

Print top 3 rows of dataframe

	funding_round_type	funded_at	raised_amount_usd	name	category_list	status	country_code
ID							
1	VENTURE	1/8/2014	0	DERON	NAN	OPERATING	NaN
2	SEED	1/1/2015	18192	ASYS	CONSUMER ELECTRONICS	OPERATING	USA
3	GRANT	1/10/2013	14851	Â HIZMETLERI TIC	CONSUMER GOODS	OPERATING	NaN

Arguments

```
read_csv(filepath_or_buffer, sep=',',
delimiter=None, header='infer',
names=None, index_col=None, usecols=None,
squeeze=False, prefix=None,
mangle_dupe_cols=True, dtype=None,
engine=None, converters=None,
true_values=None, false_values=None,
skipinitialspace=False, skiprows=None,
nrows=None, na_values=None,
keep_default_na=True, na_filter=True,
verbose=False, skip_blank_lines=True,
parse_dates=False,
infer_datetime_format=False,
keep_date_col=False, date_parser=None,
dayfirst=False, iterator=False,
chunksize=None, compression='infer',
thousands=None, decimal='.',
lineterminator=None, quotechar='"',
quoting=0, escapechar=None, comment=None,
encoding=None, dialect=None,
tupleize_cols=False,
error_bad_lines=True,
warn_bad_lines=True, skipfooter=0,
skip_footer=0, doublequote=True,
delim_whitespace=False,
as_recarray=False, compact_ints=False,
use_unsigned=False, low_memory=True,
buffer_lines=None, memory_map=False,
float_precision=None)
```

3.1 Đọc file CSV, Text

Vd1: sử dụng tham số `index_col` để thiết lập cột index khi đọc file csv

```
1 #Sử dụng phương thức read_csv()
2 #Tham số: Thiết lập cột index là cột Personal
3 data1 = pd.read_csv(path,
4                     index_col=0)
5 data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 500 entries, P1 to P500
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Gender      500 non-null    object
1   Height_cm   500 non-null    int64
2   Weight_kg   500 non-null    int64
dtypes: int64(2), object(1)
memory usage: 15.6+ KB
```

```
1 #Hiển thị dữ liệu 5 dòng đầu tiên
2 data1.head()
```

	Gender	Height_cm	Weight_kg
Personal			
P1	Male	174	96
P2	Male	189	87
P3	Female	185	110
P4	Female	195	104
P5	Male	149	61

3.1 Đọc file CSV, Text

Vd2: Thiết lập tham số chỉ đọc 100 dòng đầu tiên và dữ liệu trong 2 cột Height_cm, Weight_kg

```
1 #Sử dụng phương thức read_csv()
2 #Thiết lập số hàng, cột muốn đọc dữ liệu
3 data2 = pd.read_csv(path,
4                       nrows=100,
5                       usecols=['Height_cm', 'Weight_kg'])
6 data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Height_cm    100 non-null    int64
1   Weight_kg    100 non-null    int64
dtypes: int64(2)
memory usage: 1.7 KB
```

```
1 #Hiển thị dữ liệu 5 dòng đầu tiên
2 data2.head()
```

	Height_cm	Weight_kg
0	174	96
1	189	87
2	185	110
3	195	104
4	149	61

3.1 Đọc file CSV, Text

Vd3: Thiết lập tham số đọc dữ liệu từ dòng thứ 5 trở đi, và đặt lại tên của từng cột dữ liệu thành ['ID','Sex','H(cm)','W(kg)']

```
1 #Thiết lập tham số đọc dữ liệu từ dòng thứ 5 trong file
2 #và đặt lại tên của các cột dữ liệu
3 data3 = pd.read_csv(path,
4                       names=['ID','Sex','H(cm)','W(kg)'],
5                       skiprows=5)
6 data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 496 entries, 0 to 495
Data columns (total 4 columns):
#   Column    Non-Null Count  Dtype
---  -
0    ID        496 non-null    object
1    Sex        496 non-null    object
2    H(cm)     496 non-null    int64
3    W(kg)     496 non-null    int64
dtypes: int64(2), object(2)
memory usage: 15.6+ KB
```

```
1 #Hiển thị 5 dòng dữ liệu đầu tiên
2 data3.head()
```

	ID	Sex	H(cm)	W(kg)
0	P5	Male	149	61
1	P6	Male	189	104
2	P7	Male	147	92
3	P8	Male	154	111
4	P9	Male	174	90

3.1 Đọc file CSV, Text

Vd4: Đọc dữ liệu lưu trữ trong file Text vào biến DataFrame cũng sử dụng phương thức `read_csv()`

```
1 #Đọc dữ liệu trong file txt_Data_Diamonds.txt:
2 df_Diamonds = pd.read_csv('Data_Excercise/txt_Data_Diamonds.txt',
3                             names=['Weight(carat)', 'Price(USD)'],
4                             sep='\t', #mặc định sep=', '
5                             header=None)
6
7 df_Diamonds
```

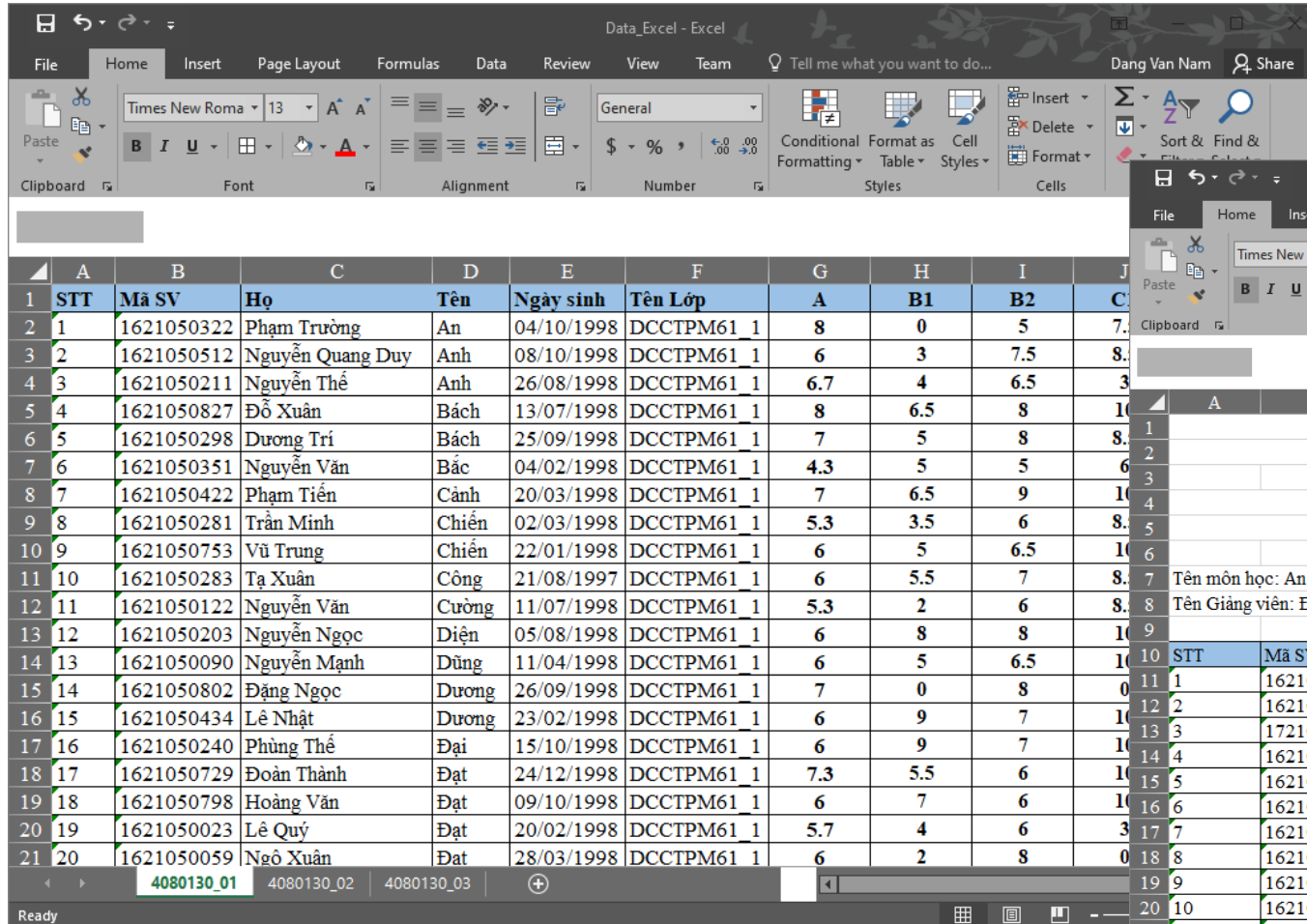
	Weight(carat)	Price(USD)
0	0.23	484
1	0.31	942
2	0.20	345
3	1.02	4459

Thực hành 1

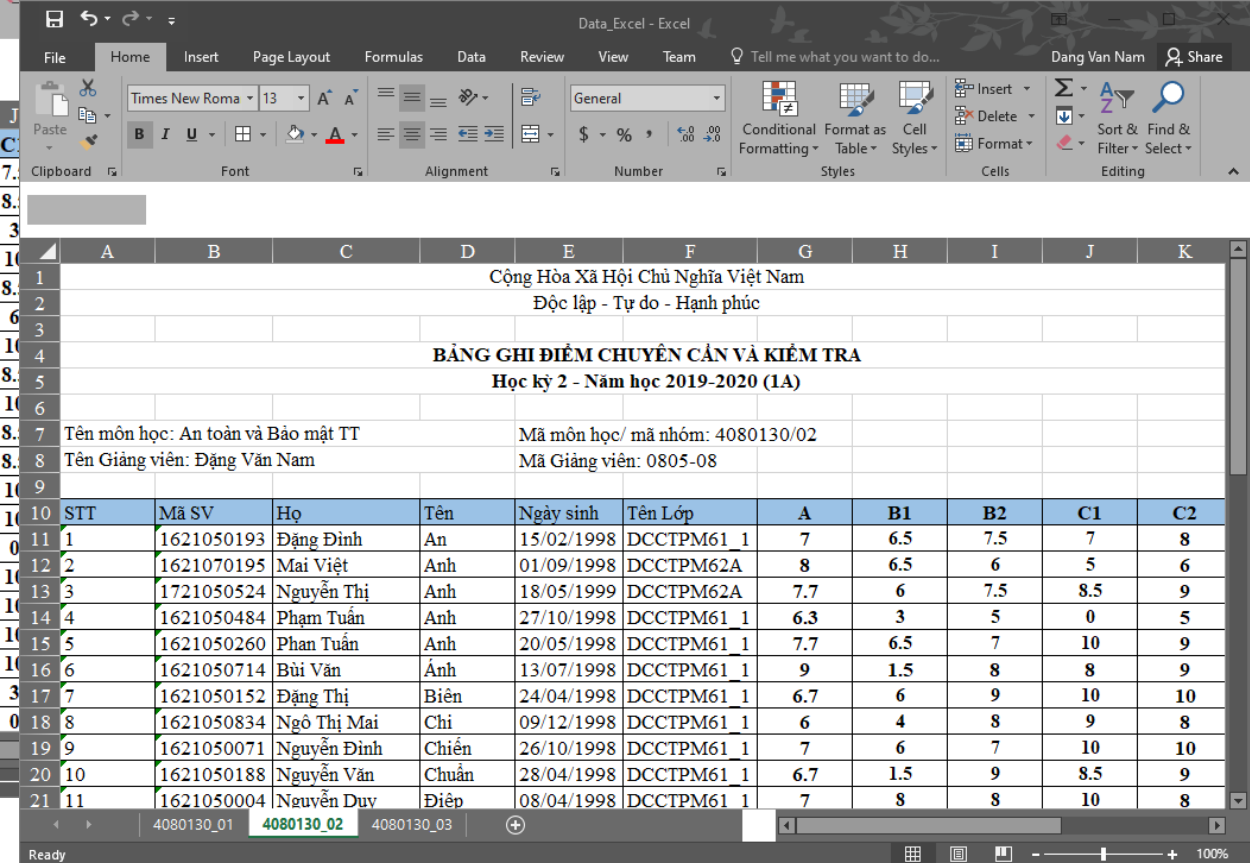
3.2 Đọc dữ liệu từ file Excel

3.2 Đọc file Excel

- File dữ liệu Excel demo gồm 3 sheet:



STT	Mã SV	Họ	Tên	Ngày sinh	Tên Lớp	A	B1	B2	C1	C2
1	1621050322	Phạm Trường	An	04/10/1998	DCCTPM61_1	8	0	5	7.5	8.5
2	1621050512	Nguyễn Quang Duy	Anh	08/10/1998	DCCTPM61_1	6	3	7.5	8.5	8.5
3	1621050211	Nguyễn Thế	Anh	26/08/1998	DCCTPM61_1	6.7	4	6.5	3	3
4	1621050827	Đỗ Xuân	Bách	13/07/1998	DCCTPM61_1	8	6.5	8	10	10
5	1621050298	Dương Trí	Bách	25/09/1998	DCCTPM61_1	7	5	8	8.5	8.5
6	1621050351	Nguyễn Văn	Bắc	04/02/1998	DCCTPM61_1	4.3	5	5	6	6
7	1621050422	Phạm Tiến	Cảnh	20/03/1998	DCCTPM61_1	7	6.5	9	10	10
8	1621050281	Trần Minh	Chiến	02/03/1998	DCCTPM61_1	5.3	3.5	6	8.5	8.5
9	1621050753	Vũ Trung	Chiến	22/01/1998	DCCTPM61_1	6	5	6.5	10	10
10	1621050283	Tạ Xuân	Công	21/08/1997	DCCTPM61_1	6	5.5	7	8.5	8.5
11	1621050122	Nguyễn Văn	Cường	11/07/1998	DCCTPM61_1	5.3	2	6	8.5	8.5
12	1621050203	Nguyễn Ngọc	Diện	05/08/1998	DCCTPM61_1	6	8	8	10	10
13	1621050090	Nguyễn Mạnh	Dũng	11/04/1998	DCCTPM61_1	6	5	6.5	10	10
14	1621050802	Đặng Ngọc	Dương	26/09/1998	DCCTPM61_1	7	0	8	0	0
15	1621050434	Lê Nhật	Dương	23/02/1998	DCCTPM61_1	6	9	7	10	10
16	1621050240	Phùng Thế	Đại	15/10/1998	DCCTPM61_1	6	9	7	10	10
17	1621050729	Đoàn Thành	Đạt	24/12/1998	DCCTPM61_1	7.3	5.5	6	10	10
18	1621050798	Hoàng Văn	Đạt	09/10/1998	DCCTPM61_1	6	7	6	10	10
19	1621050023	Lê Quý	Đạt	20/02/1998	DCCTPM61_1	5.7	4	6	3	3
20	1621050059	Ngô Xuân	Đạt	28/03/1998	DCCTPM61_1	6	2	8	0	0



STT	Mã SV	Họ	Tên	Ngày sinh	Tên Lớp	A	B1	B2	C1	C2
1	1621050193	Đặng Đình	An	15/02/1998	DCCTPM61_1	7	6.5	7.5	7	8
2	1621070195	Mai Việt	Anh	01/09/1998	DCCTPM62A	8	6.5	6	5	6
3	1721050524	Nguyễn Thị	Anh	18/05/1999	DCCTPM62A	7.7	6	7.5	8.5	9
4	1621050484	Phạm Tuấn	Anh	27/10/1998	DCCTPM61_1	6.3	3	5	0	5
5	1621050260	Phan Tuấn	Anh	20/05/1998	DCCTPM61_1	7.7	6.5	7	10	9
6	1621050714	Bùi Văn	Anh	13/07/1998	DCCTPM61_1	9	1.5	8	8	9
7	1621050152	Đặng Thị	Biên	24/04/1998	DCCTPM61_1	6.7	6	9	10	10
8	1621050834	Ngô Thị Mai	Chi	09/12/1998	DCCTPM61_1	6	4	8	9	8
9	1621050071	Nguyễn Đình	Chiến	26/10/1998	DCCTPM61_1	7	6	7	10	10
10	1621050188	Nguyễn Văn	Chuẩn	28/04/1998	DCCTPM61_1	6.7	1.5	9	8.5	9
11	1621050004	Nguyễn Duy	Diệp	08/04/1998	DCCTPM61_1	7	8	8	10	8

3.2 Đọc file Excel

- Sử dụng phương thức **pd.read_excel()** để đọc dữ liệu từ file excel.
 - Lưu ý 2 tham số **sheetname=""** xác định sheet muốn đọc dữ liệu (Mặc định là sheet đầu tiên)

```
1 import pandas as pd
2 path_excel = 'Data_Excercise\Data_Excel.xlsx'
3 #Đọc dữ liệu từ file excel
4 data_ex = pd.read_excel(path_excel)
5 data_ex.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66 entries, 0 to 65
Data columns (total 11 columns):
#   Column      Non-Null Count  Dtype
---  -
0   STT          66 non-null     int64
1   Mã SV        66 non-null     int64
2   Họ           66 non-null     object
3   Tên          66 non-null     object
4   Ngày sinh    66 non-null     object
5   Tên Lớp      66 non-null     object
6   A            66 non-null     float64
7   B1           66 non-null     float64
8   B2           66 non-null     float64
9   C1           66 non-null     float64
10  C2           66 non-null     float64
dtypes: float64(5), int64(2), object(4)
memory usage: 5.8+ KB
```

```
1 #Hiển thị 5 dòng dữ liệu đầu tiên
2 data_ex.head()
```

	STT	Mã SV	Họ	Tên	Ngày sinh	Tên Lớp	A	B1	B2	C1	C2
0	1	1621050322	Phạm Trường	An	04/10/1998	DCCTPM61_1	8.0	0.0	5.0	7.5	8.0
1	2	1621050512	Nguyễn Quang Duy	Anh	08/10/1998	DCCTPM61_1	6.0	3.0	7.5	8.5	9.0
2	3	1621050211	Nguyễn Thế	Anh	26/08/1998	DCCTPM61_1	6.7	4.0	6.5	3.0	5.0
3	4	1621050827	Đỗ Xuân	Bách	13/07/1998	DCCTPM61_1	8.0	6.5	8.0	10.0	9.0
4	5	1621050298	Dương Trí	Bách	25/09/1998	DCCTPM61_1	7.0	5.0	8.0	8.5	9.0

3.2 Đọc file Excel

- Một vài tham số quan trọng trong phương thức `pd.read_excel()` để đọc dữ liệu từ file excel.

Argument	Description
io	A string containing the pathname of the given Excel file.
sheet_name	The Excel sheet name, or sheet number, of the data you want to import. The sheet number can be an integer where 0 is the first sheet, 1 is the second, etc. If a list of sheet names/numbers are given, then the output will be a dictionary of DataFrames. The default is to read all the sheets and output a dictionary of DataFrames.
header	Row number to use for the list of column labels. The default is 0, indicating that the first row is assumed to contain the column labels. If the data does not have a row of column labels, None should be used.
names	A separate Python list input of column names. This option is None by default. This option is the equivalent of assigning a list of column names to the columns attribute of the output DataFrame.
index_col	Specifies which column should be used for row indices. The default option is None, meaning that all columns are included in the data, and a range of numbers is used as the row indices.
usecols	An integer, list of integers, or string that specifies the columns to be imported into the DataFrame. The default is to import all columns. If a string is given, then Pandas uses the standard Excel format to select columns (e.g. "A:C,F,G" will import columns A, B, C, F, and G).
skiprows	The number of rows to skip at the top of the Excel sheet. Default is 0. This option is useful for skipping rows in Excel that contain explanatory information about the data below it.

3.2 Đọc file Excel

- Sử dụng phương thức `pd.read_excel()` với một số tham số cơ bản.

```
1 #Ví dụ:  
2 #Đọc dữ liệu tại sheet đầu tiên,  
3 #Chỉ lấy dữ liệu cột Mã SV và các cột điểm  
4 #Thiết lập cột đầu tiên làm index  
5 data_ex1 = pd.read_excel(path_excel,  
6                           sheet_name=0,  
7                           usecols=[1,6,7,8,9,10],  
8                           index_col=0)  
9 data_ex1.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 66 entries, 1621050322 to 1621050013  
Data columns (total 5 columns):  
#   Column   Non-Null Count  Dtype  
---  ---  
0    A        66 non-null    float64  
1    B1        66 non-null    float64  
2    B2        66 non-null    float64  
3    C1        66 non-null    float64  
4    C2        66 non-null    float64  
dtypes: float64(5)  
memory usage: 3.1 KB
```

```
1 #Hiển thị 5 dòng dữ liệu đầu tiên  
2 data_ex1.head()
```

	A	B1	B2	C1	C2
Mã SV					
1621050322	8.0	0.0	5.0	7.5	8.0
1621050512	6.0	3.0	7.5	8.5	9.0
1621050211	6.7	4.0	6.5	3.0	5.0
1621050827	8.0	6.5	8.0	10.0	9.0
1621050298	7.0	5.0	8.0	8.5	9.0

3.2 Đọc file Excel

- Sử dụng phương thức **pd.read_excel()** với một số tham số cơ bản.
 - Đọc dữ liệu sheet 2 ['4080130_02'], từ dòng 9.

```
1 #Ví dụ 3:  
2 #Đọc dữ liệu tại sheet 2, từ dòng 9  
3 data_ex3 = pd.read_excel(path_excel,  
4                           sheet_name='4080130_02',  
5                           skiprows=9)  
6 data_ex3.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 39 entries, 0 to 38  
Data columns (total 11 columns):  
#   Column      Non-Null Count  Dtype  
---  -  
0   STT          39 non-null      int64  
1   Mã SV        39 non-null      int64  
2   Họ           39 non-null      object  
3   Tên          39 non-null      object  
4   Ngày sinh    39 non-null      object  
5   Tên Lớp      39 non-null      object  
6   A            39 non-null      float64  
7   B1           39 non-null      float64  
8   B2           39 non-null      float64  
9   C1           39 non-null      float64  
10  C2           39 non-null      float64  
dtypes: float64(5), int64(2), object(4)  
memory usage: 3.5+ KB
```

```
1 #Hiển thị 5 dòng dữ liệu đầu tiên  
2 data_ex3.head()
```

	STT	Mã SV	Họ	Tên	Ngày sinh	Tên Lớp	A	B1	B2	C1	C2
0	1	1621050193	Đặng Đình	An	15/02/1998	DCCTPM61_1	7.0	6.5	7.5	7.0	8.0
1	2	1621070195	Mai Việt	Anh	01/09/1998	DCCTPM62A	8.0	6.5	6.0	5.0	6.0
2	3	1721050524	Nguyễn Thị	Anh	18/05/1999	DCCTPM62A	7.7	6.0	7.5	8.5	9.0
3	4	1621050484	Phạm Tuấn	Anh	27/10/1998	DCCTPM61_1	6.3	3.0	5.0	0.0	5.0
4	5	1621050260	Phan Tuấn	Anh	20/05/1998	DCCTPM61_1	7.7	6.5	7.0	10.0	9.0

3.2 Đọc file Excel

- Sử dụng phương thức `pd.read_excel()` với một số tham số cơ bản.
 - Đọc dữ liệu sheet 3 ['4080130_03'], không có dòng header

```
1 #Ví dụ 4
2 #Đọc dữ liệu từ sheet: '4080130_03'
3 #Dữ liệu không chứa dòng header
4 data_ex4 = pd.read_excel(path_excel,
5                           sheet_name='4080130_03',
6                           header=None)
7 data_ex4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39 entries, 0 to 38
Data columns (total 11 columns):
#   Column  Non-Null Count  Dtype  
---  -
0    0         39 non-null    int64  
1    1         39 non-null    int64  
2    2         39 non-null    object  
3    3         39 non-null    object  
4    4         39 non-null    object  
5    5         39 non-null    object  
6    6         39 non-null    float64 
7    7         39 non-null    float64 
8    8         39 non-null    float64 
9    9         39 non-null    float64 
10   10        39 non-null    float64 
dtypes: float64(5), int64(2), object(4)
memory usage: 3.5+ KB
```

```
1 #Hiển thị 5 dòng dữ liệu đầu tiên
2 data_ex4.head()
```

	0	1	2	3	4	5	6	7	8	9	10
0	1	1621050041	Đào Tuấn	Anh	22/10/1998	DCCTPM61_1	6.7	9.0	5.5	8.5	8.0
1	2	1621050262	Vũ Thị Lan	Anh	26/09/1998	DCCTPM61_1	6.7	7.0	9.0	8.5	6.0
2	3	1621050083	Trịnh Như	Bình	06/04/1998	DCCTPM61_1	7.3	8.5	9.5	10.0	9.0
3	4	1621050113	Trần Văn	Cương	19/06/1998	DCCTPM61_1	5.7	5.0	6.0	10.0	5.0
4	5	1621050384	Nguyễn Sỹ	Dũng	02/10/1998	DCCTPM61_1	7.0	0.0	7.5	8.5	9.0

3.2 Đọc file Excel

- Sử dụng phương thức **pd.read_excel()** với một số tham số cơ bản.
 - Đọc dữ liệu sheet 3 ['4080130_03'],
 - Không có dòng header
 - Chỉ lấy dữ liệu cột 1,6,7,8,9,10
 - Đặt tên cho các cột lần lượt là ['Mã SV', 'A', 'B1','B2','C1','C2']
 - Thiết lập cột đầu tiên làm Index

```
5 data_ex41 = pd.read_excel(path_excel,
6                             sheet_name='4080130_03',
7                             header=None,
8                             usecols=[1,6,7,8,9,10],
9                             names=['Mã SV', 'A', 'B1', 'B2', 'C1', 'C2'],
10                            index_col=0)
11 data_ex41.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 39 entries, 1621050041 to 1621050034
Data columns (total 5 columns):
#   Column  Non-Null Count  Dtype
---  -
0    A      39 non-null         float64
1   B1      39 non-null         float64
2   B2      39 non-null         float64
3   C1      39 non-null         float64
4   C2      39 non-null         float64
dtypes: float64(5)
memory usage: 1.8 KB
```

```
1 #Hiển thị 5 dòng dữ liệu đầu tiên
2 data_ex41.head()
```

	A	B1	B2	C1	C2
Mã SV					
1621050041	6.7	9.0	5.5	8.5	8.0
1621050262	6.7	7.0	9.0	8.5	6.0
1621050083	7.3	8.5	9.5	10.0	9.0
1621050113	5.7	5.0	6.0	10.0	5.0
1621050384	7.0	0.0	7.5	8.5	9.0



Thực hành 2

4. Quan sát và truy cập dữ liệu trong DataFrame

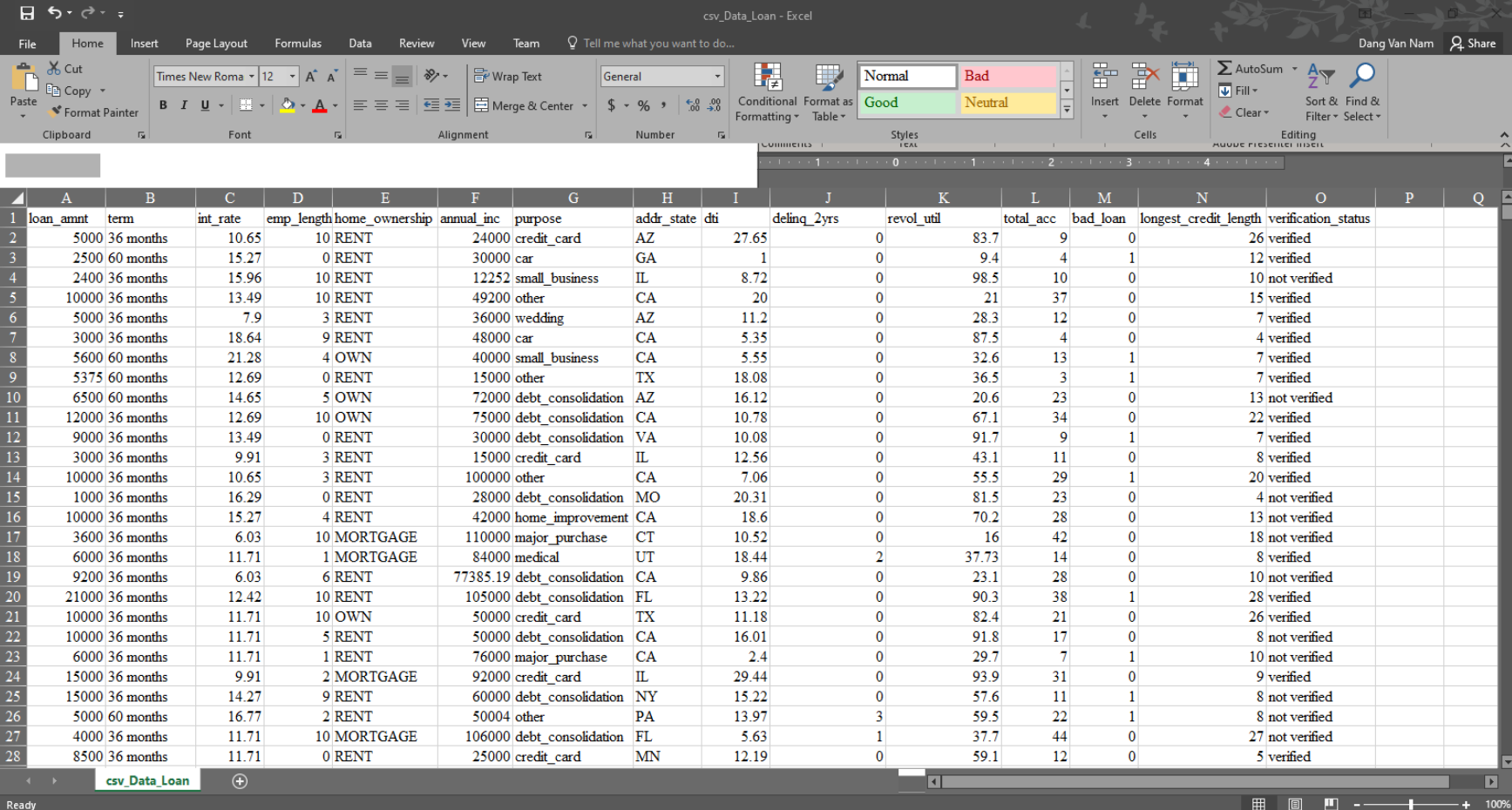
4.1 Quan sát dữ liệu



VINBIGDATA



- Đọc file dữ liệu mẫu: **csv_Data_loan**
- Đây là file dữ liệu cho biết thông tin về các khoản vay cho các mục đích khác nhau của người dùng Mỹ.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	loan_amnt	term	int_rate	emp_length	home_ownership	annual_inc	purpose	addr_state	dti	delinq_2yrs	revol_util	total_acc	bad_loan	longest_credit_length	verification_status		
2	5000	36 months	10.65	10	RENT	24000	credit_card	AZ	27.65	0	83.7	9	0	26	verified		
3	2500	60 months	15.27	0	RENT	30000	car	GA	1	0	9.4	4	1	12	verified		
4	2400	36 months	15.96	10	RENT	12252	small_business	IL	8.72	0	98.5	10	0	10	not verified		
5	10000	36 months	13.49	10	RENT	49200	other	CA	20	0	21	37	0	15	verified		
6	5000	36 months	7.9	3	RENT	36000	wedding	AZ	11.2	0	28.3	12	0	7	verified		
7	3000	36 months	18.64	9	RENT	48000	car	CA	5.35	0	87.5	4	0	4	verified		
8	5600	60 months	21.28	4	OWN	40000	small_business	CA	5.55	0	32.6	13	1	7	verified		
9	5375	60 months	12.69	0	RENT	15000	other	TX	18.08	0	36.5	3	1	7	verified		
10	6500	60 months	14.65	5	OWN	72000	debt_consolidation	AZ	16.12	0	20.6	23	0	13	not verified		
11	12000	36 months	12.69	10	OWN	75000	debt_consolidation	CA	10.78	0	67.1	34	0	22	verified		
12	9000	36 months	13.49	0	RENT	30000	debt_consolidation	VA	10.08	0	91.7	9	1	7	verified		
13	3000	36 months	9.91	3	RENT	15000	credit_card	IL	12.56	0	43.1	11	0	8	verified		
14	10000	36 months	10.65	3	RENT	100000	other	CA	7.06	0	55.5	29	1	20	verified		
15	1000	36 months	16.29	0	RENT	28000	debt_consolidation	MO	20.31	0	81.5	23	0	4	not verified		
16	10000	36 months	15.27	4	RENT	42000	home_improvement	CA	18.6	0	70.2	28	0	13	not verified		
17	3600	36 months	6.03	10	MORTGAGE	110000	major_purchase	CT	10.52	0	16	42	0	18	not verified		
18	6000	36 months	11.71	1	MORTGAGE	84000	medical	UT	18.44	2	37.73	14	0	8	verified		
19	9200	36 months	6.03	6	RENT	77385.19	debt_consolidation	CA	9.86	0	23.1	28	0	10	not verified		
20	21000	36 months	12.42	10	RENT	105000	debt_consolidation	FL	13.22	0	90.3	38	1	28	verified		
21	10000	36 months	11.71	10	OWN	50000	credit_card	TX	11.18	0	82.4	21	0	26	verified		
22	10000	36 months	11.71	5	RENT	50000	debt_consolidation	CA	16.01	0	91.8	17	0	8	not verified		
23	6000	36 months	11.71	1	RENT	76000	major_purchase	CA	2.4	0	29.7	7	1	10	not verified		
24	15000	36 months	9.91	2	MORTGAGE	92000	credit_card	IL	29.44	0	93.9	31	0	9	verified		
25	15000	36 months	14.27	9	RENT	60000	debt_consolidation	NY	15.22	0	57.6	11	1	8	not verified		
26	5000	60 months	16.77	2	RENT	50004	other	PA	13.97	3	59.5	22	1	8	not verified		
27	4000	36 months	11.71	10	MORTGAGE	106000	debt_consolidation	FL	5.63	1	37.7	44	0	27	not verified		
28	8500	36 months	11.71	0	RENT	25000	credit_card	MN	12.19	0	59.1	12	0	5	verified		

4.1 Quan sát dữ liệu



VINBIGDATA



- **df.info()** : Hiển thị thông tin chi tiết biến DataFrame
- **df.head(n)**: Hiển thị n dòng đầu tiên của biến df (default = 5)
- **df.tail(n)** : Hiển thị n dòng cuối cùng biến df (default = 5)
- **df.shape** : Hiển thị kích thước (rows x columns) của biến df
- **df.columns**: Tên các cột trong biến df
- **df.isnull()** : Kiểm tra dữ liệu rỗng trong biến df
- **df.isnull().sum()** : Tính tổng các dòng dữ liệu null trong df
- **df.count()** : Tổng số dòng dữ liệu không null trong df
- **df.size** : Số phần tử của biến df (=rows x columns)
- **df.dtypes** : Kiểu dữ liệu của từng columns trong df



4.1 Quan sát dữ liệu

- **df.describe()** : Một số đặc trưng thống kê của biến df
 - Tham số include = 'O': thống kê các cột có kiểu dữ liệu Object
 - Tham số include = 'all': Thống kê tất cả các cột trong df

```
1 #Quan sát một số đặc trưng thống kê của df
2 #Thống kê các cột dữ liệu Object
3 df_loan.describe(include='O')
```

	term	home_ownership	purpose	addr_state	verification_status
count	163987	163987	163987	163987	163987
unique	2	6	14	50	2
top	36 months	MORTGAGE	debt_consolidation	CA	verified
freq	129950	79714	93261	28702	104832

4.2 Truy cập dữ liệu

- `df[['Col1', 'Col2', 'Col3']]`: Chỉ truy cập dữ liệu của các cột có tên **Col1, Col2, Col3** trong dataframe `df`

```
1 #Truy xuất dữ liệu theo cột
2 #Lấy dữ liệu của một cột
3 df_state = df_loan[['addr_state']]
4 df_state.head()
```

	addr_state
0	AZ
1	GA
2	IL
3	CA
4	AZ

```
1 #Truy xuất dữ liệu theo cột
2 #Chỉ lấy dữ liệu của 3 cột: loan_amnt, int_rate, purpose
3 df_loan1 = df_loan[['loan_amnt', 'int_rate', 'purpose']]
4 df_loan1.head()
```

	loan_amnt	int_rate	purpose
0	5000	10.65	credit_card
1	2500	15.27	car
2	2400	15.96	small_business
3	10000	13.49	other
4	5000	7.90	wedding

4.2 Truy cập dữ liệu

- `df.iloc[[index_row],[index_col]]`: Truy cập tới dữ liệu của hàng và cột qua **chỉ số `index_row`, `index_col` (tương tự như với Numpy)**

```
1 #Sử dụng .iloc truy xuất dữ liệu như với Numpy
2 #Truy xuất 10 dòng dữ liệu từ [10 --> 20) tất cả các cột
3 df_loan.iloc[10:20,:]
```

	loan_amnt	term	int_rate	emp_length	home_ownership	annual_inc	purpose	addr_state	dti	delinq_2yrs	revol_util	total_acc	bad_loan	l
10	9000	36 months	13.49	0.0	RENT	30000.00	debt_consolidation	VA	10.08	0.0	91.70	9.0	1	
11	3000	36 months	9.91	3.0	RENT	15000.00	credit_card	IL	12.56	0.0	43.10	11.0	0	
12	10000	36 months	10.65	3.0	RENT	100000.00	other	CA	7.06	0.0	55.50	29.0	1	
13	1000	36 months	16.29	0.0	RENT	28000.00	debt_consolidation	MO	20.31	0.0	81.50	23.0	0	
14	10000	36 months	15.27	4.0	RENT	42000.00	home_improvement	CA	18.60	0.0	70.20	28.0	0	
15	3600	36 months	6.03	10.0	MORTGAGE	110000.00	major_purchase	CT	10.52	0.0	16.00	42.0	0	

4.2 Truy cập dữ liệu

- `df.loc[[name_index],[name_col]]`: Truy cập tới dữ liệu của hàng và cột qua **name_index, name_column**

```
1 #Truy cập từ dòng 20 đến dòng 25 của df
2 #chỉ lấy dữ liệu 4 cột: loan_amnt, home_ownership, purpose, addr_state
3 df_loan.loc[20:25,['loan_amnt','home_ownership','purpose','addr_state']]
```

	loan_amnt	home_ownership	purpose	addr_state
20	10000	RENT	debt_consolidation	CA
21	6000	RENT	major_purchase	CA
22	15000	MORTGAGE	credit_card	IL
23	15000	RENT	debt_consolidation	NY
24	5000	RENT	other	PA
25	4000	MORTGAGE	debt_consolidation	FL

4.2 Truy cập dữ liệu



VINBIGDATA



Type	Notes
<code>df[val]</code>	Select single column or sequence of columns from the DataFrame; special case conveniences: boolean array (filter rows), slice (slice rows), or boolean DataFrame (set values based on some criterion)
<code>df.loc[val]</code>	Selects single row or subset of rows from the DataFrame by label
<code>df.loc[:, val]</code>	Selects single column or subset of columns by label
<code>df.loc[val1, val2]</code>	Select both rows and columns by label
<code>df.iloc[where]</code>	Selects single row or subset of rows from the DataFrame by integer position
<code>df.iloc[:, where]</code>	Selects single column or subset of columns by integer position
<code>df.iloc[where_i, where_j]</code>	Select both rows and columns by integer position
<code>df.at[label_i, label_j]</code>	Select a single scalar value by row and column label
<code>df.iat[i, j]</code>	Select a single scalar value by row and column position (integers)
reindex method	Select either rows or columns by labels
get_value, set_value methods	Select single value by row and column label

5. Replacing Values, Rename columns

5.1 Replacing Values

- Thay thế 1 giá trị trong Dataframe, thực hiện tương tự như với Numpy. Sử dụng **.loc**; **.iloc** để xác định phần tử cần cập nhật, thay đổi giá trị

	loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	credit_card	AZ
1	2500	RENT	car	GA
2	2400	RENT	small_business	IL
3	10000	RENT	other	CA
4	5000	RENT	wedding	AZ
5	3000	RENT	car	CA
6	5600	OWN	small_business	CA
7	5375	RENT	other	TX
8	6500	OWN	debt_consolidation	AZ
9	12000	OWN	debt_consolidation	CA
10	9000	RENT	debt_consolidation	VA

```
1 #Thay thế giá trị purpose: credit_card--> wedding
2 #của index đầu tiên
3 df_new.loc[0,'purpose'] = 'wedding'
4 df_new
```

	loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	wedding	AZ
1	2500	RENT	car	GA
2	2400	RENT	small_business	IL
3	10000	RENT	other	CA

```
1 #Thay thế giá trị thuộc tính loan_amnt: 2400 --> 8800
2 #của index = 2
3 df_new.iloc[2,0] = 8800
4 df_new
```

	loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	wedding	AZ
1	2500	RENT	car	GA
2	8800	RENT	small_business	IL
3	10000	RENT	other	CA

5.1 Replacing Values

- **df.replace():** Thay thế các giá trị trong toàn bộ DataFrame. (tham số inplace=True|False áp dụng thay đổi cho dataframe hiện tại hay không?).

	loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	credit_card	AZ
1	2500	RENT	car	GA
2	2400	RENT	small_business	IL
3	10000	RENT	other	CA
4	5000	RENT	wedding	AZ
5	3000	RENT	car	CA
6	5600	OWN	small_business	CA
7	5375	RENT	other	TX
8	6500	OWN	debt_consolidation	AZ
9	12000	OWN	debt_consolidation	CA
10	9000	RENT	debt_consolidation	VA

```
1 #Khi muốn thay đổi áp dụng lên DataFrame hiện tại
2 #Thiết lập tham số inplace=True
3 df_new.replace({'RENT':'MORTGAGE',
4                 'car':'small_business'}, inplace=True)
5 df_new
```

	loan_amnt	home_ownership	purpose	addr_state
0	5000	MORTGAGE	credit_card	AZ
1	2500	MORTGAGE	small_business	GA
2	2400	MORTGAGE	small_business	IL

02

```
1 #Thay thế nhiều giá trị trong DataFrame
2 #RENT --> MORTGAGE
3 #car --> small_business
4 df_new.replace({'RENT':'MORTGAGE',
5                 'car':'small_business'})
```

01

	loan_amnt	home_ownership	purpose	addr_state
0	5000	MORTGAGE	wedding	AZ
1	2500	MORTGAGE	small_business	GA
2	8800	MORTGAGE	small_business	IL
3	10000	MORTGAGE	other	CA
4	5000	MORTGAGE	wedding	AZ
5	3000	MORTGAGE	small_business	CA
6	5600	OWN	small_business	CA
7	5375	MORTGAGE	other	TX
8	6500	OWN	debt_consolidation	AZ
9	12000	OWN	debt_consolidation	CA
10	9000	MORTGAGE	debt_consolidation	VA

5.1 Replacing Values

- **df.replace():** Thay thế các giá trị theo từng cột (tham số inplace=True|False áp dụng thay đổi cho dataframe hiện tại hay không?).

	loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	credit_card	AZ
1	2500	RENT	car	GA
2	2400	RENT	small_business	IL
3	10000	RENT	other	CA
4	5000	RENT	wedding	AZ
5	3000	RENT	car	CA
6	5600	OWN	small_business	CA
7	5375	RENT	other	TX
8	6500	OWN	debt_consolidation	AZ
9	12000	OWN	debt_consolidation	CA
10	9000	RENT	debt_consolidation	VA

```
1 #Thay thế tên viết tắt bằng tên đầy đủ.
2 state_name={'AZ':'Arizona',
3             'GA':'Georgia',
4             'IL':'Illinois',
5             'CA':'California',
6             'TX':'Texas',
7             'VA':'Virginia'}
8 #Trong cột addr_state
9 df_new['addr_state'].replace(state_name,inplace=True)
10 df_new
```

	loan_amnt	home_ownership	purpose	addr_state
0	5000	RENT	credit_card	Arizona
1	2500	RENT	car	Georgia
2	2400	RENT	small_business	Illinois
3	10000	RENT	other	California
4	5000	RENT	wedding	Arizona
5	3000	RENT	car	California

5.2 Rename Columns

- `df.rename()`: thay đổi tên cột trong DataFrame

```
1 #Muốn áp dụng thay đổi vào trực tiếp biến df, sử dụng inplace=True
2 df_new.rename(columns={'loan_amnt':'Số tiền vay',
3                        'home_ownership':'Tình trạng nhà ở',
4                        'purpose': 'Mục đích vay tiền',
5                        'addr_state':'Địa chỉ'}, inplace=True)
6 df_new.head()
```

01

	Số tiền vay	Tình trạng nhà ở	Mục đích vay tiền	Địa chỉ
0	5000	RENT	credit_card	AZ
1	2500	RENT	car	GA
2	2400	RENT	small_business	IL
3	10000	RENT	other	CA
4	5000	RENT	wedding	AZ

```
1 #Đổi tên cột sang viết hoa
2 df_new.rename(str.upper, axis='columns')
```

02

	SỐ TIỀN VAY	TÌNH TRẠNG NHÀ Ở	MỤC ĐÍCH VAY TIỀN	ĐỊA CHỈ
0	5000	RENT	credit_card	AZ
1	2500	RENT	car	GA
2	2400	RENT	small_business	IL
3	10000	RENT	other	CA
4	5000	RENT	wedding	AZ

Thực hành 3

6. Filter Data

6. Filter Data



VINBIGDATA



- Để lọc dữ liệu trong DataFrame có thể sử dụng nhiều cách khác nhau

```
1 #Lọc danh sách người giới tính nam
2 #Cách 1:
3 df_male1 = df_bmi[df_bmi.Gender=='Male']
4 df_male1.head(2)
```

	Personal	Gender	Height_cm	Weight_kg
0	P1	Male	174	96
1	P2	Male	189	87

01

```
1 #Cách 2: sử dụng phương thức query
2 df_male2 = df_bmi.query('Gender=="Male"')
3 df_male2.head(2)
```

	Personal	Gender	Height_cm	Weight_kg
0	P1	Male	174	96
1	P2	Male	189	87

02

```
1 #Cách 3: sử dụng iloc
2 df_male3 = df_bmi.loc[(df_bmi.Gender=="Male")]
3 df_male3.head(2)
```

	Personal	Gender	Height_cm	Weight_kg
0	P1	Male	174	96
1	P2	Male	189	87

03

6. Filter Data

- Sử dụng toán tử **& (and)** - **| (or)** - **~ (not)** để kết hợp nhiều điều kiện trong khi lọc dữ liệu

```
1 #Kết hợp nhiều tiêu chí lọc dữ liệu
2 #Lọc người có giới tính Female và cân nặng dưới 70kg
3 df_p1 = df_bmi[(df_bmi.Gender == 'Female') & (df_bmi.Weight_kg < 70)]
4 df_p1
```

	Personal	Gender	Height_cm	Weight_kg
24	P25	Female	172	67
25	P26	Female	151	64
32	P33	Female	195	65
51	P52	Female	176	54

01

```
1 #Kết hợp nhiều tiêu chí tìm kiếm
2 #Lọc người có chiều cao > 195 cm hoặc cân nặng > 150kg
3 df_p2 = df_bmi[(df_bmi.Height_cm > 195) | (df_bmi.Weight_kg > 150)]
4 df_p2
```

	Personal	Gender	Height_cm	Weight_kg
28	P29	Female	163	159
29	P30	Male	179	152
34	P35	Female	157	153
36	P37	Female	197	114

02

```
1 # toán tử ~ - Not
2 df_p3 = df_bmi[~(df_bmi.Weight_kg < 155)]
3 df_p3
```

	Personal	Gender	Height_cm	Weight_kg
28	P29	Female	163	159
65	P66	Female	179	158

03

6. Filter Data



VINBIGDATA



- Sử dụng phương thức `.isin()` để kết lọc dữ liệu theo một tập hợp

```
1 #Lọc ra những người có cân nặng bằng 150, 155 và 160kg
2 # phương thức isin (tương tự như in)
3 df_p4 = df_bmi[df_bmi.Weight_kg.isin([150,155,160])]
4 df_p4
```

	Personal	Gender	Height_cm	Weight_kg
102	P103	Male	161	155
106	P107	Male	166	160
123	P124	Female	184	160
134	P135	Female	171	155
135	P136	Female	183	150

7. Tính toán đặc trưng thống kê trong DataFrame

7. Đặc trưng thống kê

- Sử dụng phương thức `.max()`, `.min()`, `.sum()`, `.mean()`, `.median()`, `.cumsum()`, `.std()` để tính các đặc trưng thống kê cho DataFrame hoặc theo từng cột.

```
1 #tìm Max, Min của thuộc tính cân nặng
2 w_max = df_bmi['Weight_kg'].max()
3 w_min = df_bmi['Weight_kg'].min()
4 print('Cân nặng lớn nhất:', w_max, '(kg)')
5 print('Cân nặng nhỏ nhất:', w_min, '(kg)')
```

Cân nặng lớn nhất: 160 (kg)

Cân nặng nhỏ nhất: 50 (kg)

```
1 #tìm Mean, Median của chiều cao
2 h_mean = df_bmi['Height_cm'].mean()
3 h_median = df_bmi['Height_cm'].median()
4 print('Chiều cao trung bình:', h_mean, '(cm)')
5 print('Trung vị:', h_median, '(cm)')
```

Chiều cao trung bình: 169.944 (cm)

Trung vị: 170.5 (cm)

```
1 #tìm độ lệch chuẩn của chiều cao, cân nặng
2 h_std = df_bmi['Height_cm'].std()
3 w_std = df_bmi['Weight_kg'].std()
4 print('sdt của chiều cao:', h_std)
5 print('sdt của cân nặng:', w_std)
```

sdt của chiều cao: 16.37526067959376

sdt của cân nặng: 32.38260746964435

8. Xác định giá trị duy nhất (Unique)

8. Unique

- **df.unique():** liệt kê danh sách các giá trị khác nhau trong một cột dữ liệu của DataFrame.
- **df.value_counts():** Tính tổng số theo từng giá trị khác nhau trong một cột dữ liệu của DataFrame. Kết quả là một đối tượng series.

```
1 #Xác định giá trị duy nhất trong một cột
2 df_bmi['Gender'].unique()
```

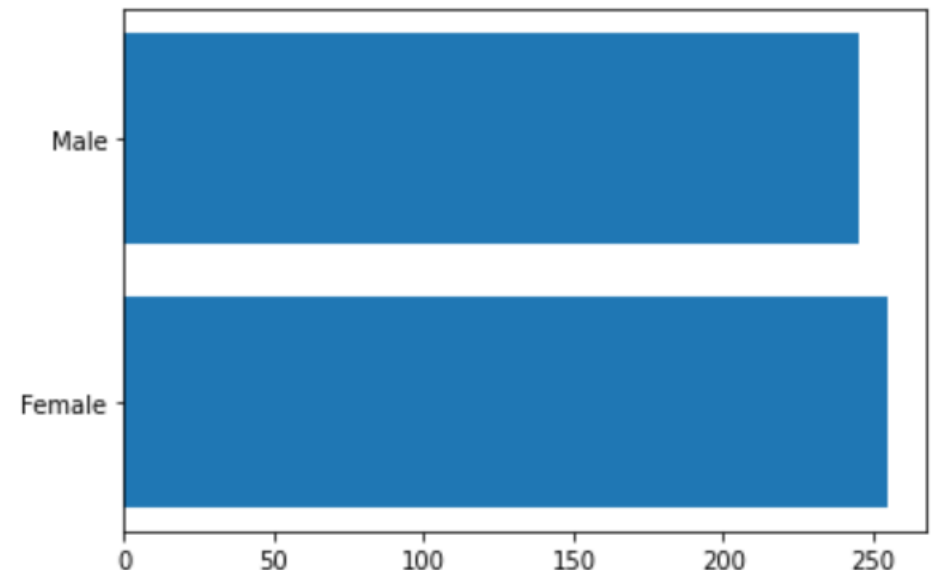
```
array(['Male', 'Female'], dtype=object)
```

```
1 #Thống kê số lượng theo giá trị duy nhất
2 unique_gender = df_bmi['Gender'].value_counts()
3 unique_gender
```

```
Female    255
Male      245
Name: Gender, dtype: int64
```

```
1 #Vẽ đồ thị thể hiện kết quả
2 plt.barh(unique_gender.index, unique_gender.values)
```

<BarContainer object of 2 artists>



Thực hành 4

9. Phân tích dữ liệu Time series

(Tiếp cận từ bài toán thực tế)

Mô tả bài toán

Fremont Bridge

Trang web

Chỉ đường

Lưu

4,4 ★★★★★ 127 đánh giá trên Google

Cầu ở Portland, Oregon

Được dịch từ tiếng Anh - Cầu Fremont là cây cầu vòm bằng thép bắc qua sông Willamette nằm ở Portland, Oregon, Hoa Kỳ. Nó mang giao thông Interstate 405 và US 30 giữa trung tâm thành phố và Bắc Portland, nơi giao nhau với Xa lộ liên tiểu bang 5.

[Wikipedia \(tiếng Anh\)](#)

Xem mô tả gốc ▾

Địa chỉ: Stadium Fwy, Portland, OR 97232, Hoa Kỳ

Chiều cao: 116 m

Bắt đầu xây dựng: 1968

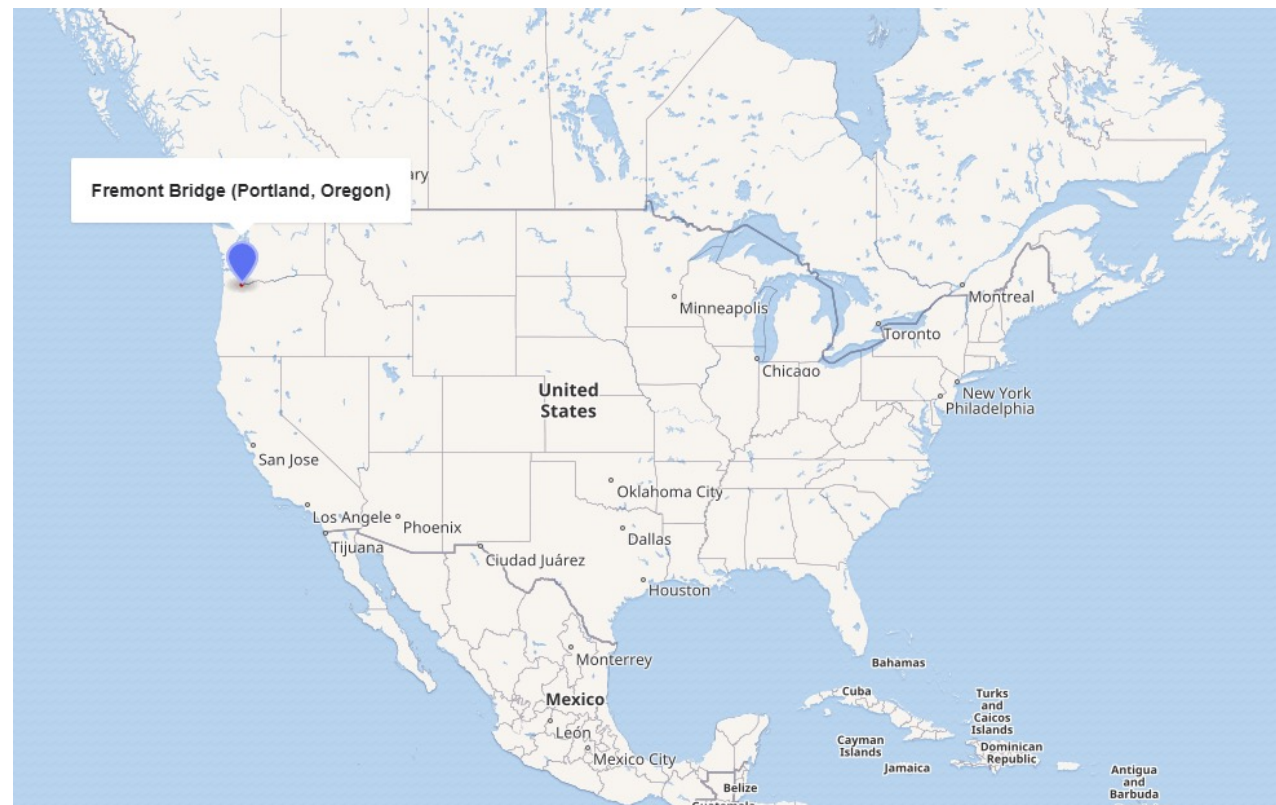
Tổng chiều dài: 656 m

Khoảng hở bên dưới: 53 m



+ Người ta lắp đặt thiết bị để đếm số lượng xe đạp đi qua chiều phía đông và phía tây của cây cầu Fremont Bridge theo từng giờ.

+ Chi tiết: <https://data.seattle.gov/Transportation/Fremont-Bridge-Bicycle-Counter/65db-xm6k>



- Tập dữ liệu là số lượng xe đạp đi qua cây cầu Fremont Bridge. Dữ liệu này được thu thập tự động thông qua các cảm biến ở 2 lối đi bộ ở phía đông và phía tây của cây cầu. Số lượng xe đạp được tổng hợp theo từng giờ.
- Tập dữ liệu bao gồm 4 cột:
 - Date: Thời gian (ngày - giờ): 10/03/2012 12:00:00 AM (Kiểu thời gian)
 - Fremont Bridge Total: Tổng số xe đi theo cả 2 lối đông và tây (Kiểu số nguyên)
 - Fremont Bridge East Sidewalk: Số xe đạp đi qua lối phía đông của cầu tương ứng với thời gian (Kiểu số nguyên)
 - Fremont Bridge West Sidewalk: Số xe đạp đi qua lối phía tây của cầu tương ứng với thời gian (Kiểu số nguyên)



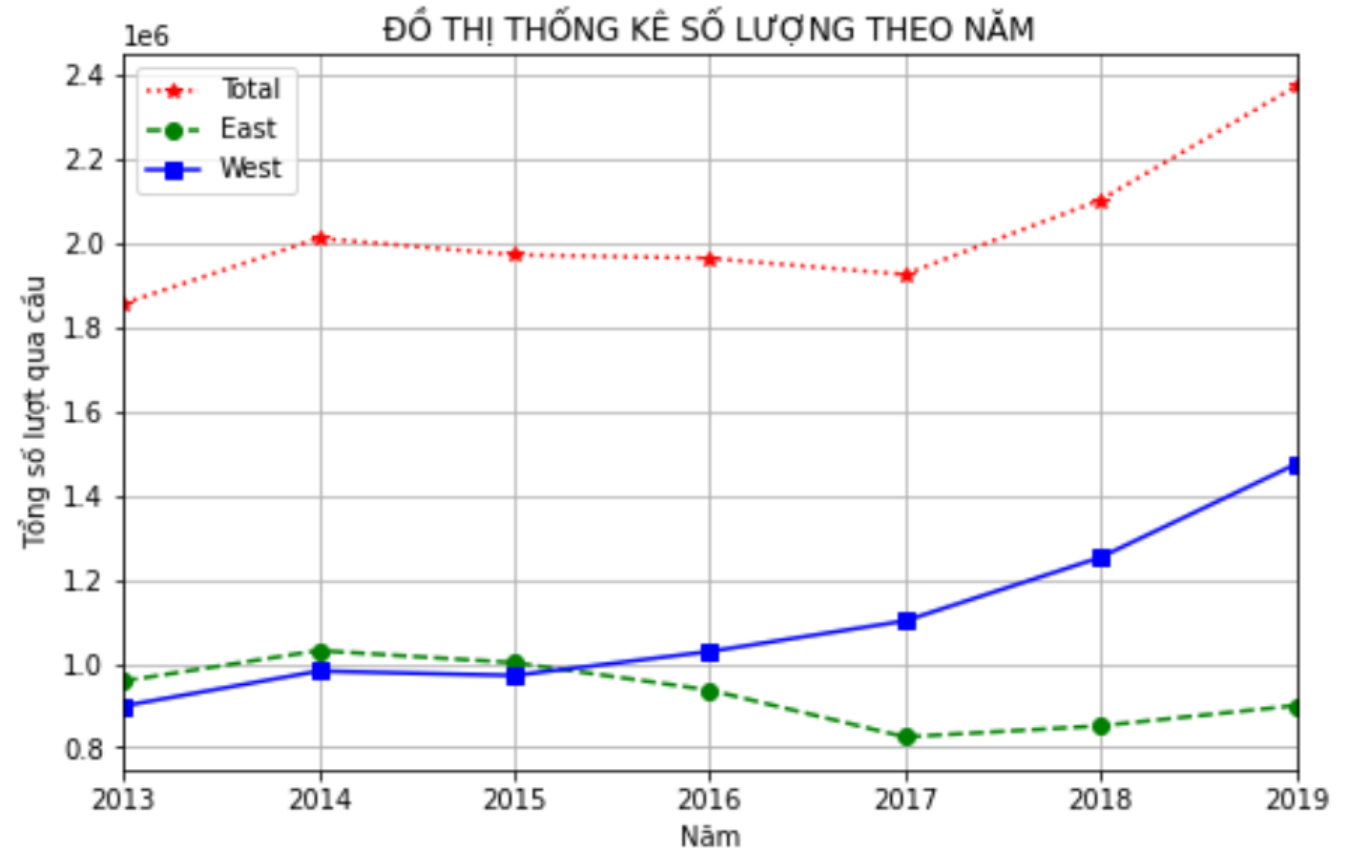
Date	Fremont Bridge Total	Fremont Bridge East Sidewalk	Fremont Bridge West Sidewalk
10/03/2012 12:00:00 AM	13	4	9
10/03/2012 01:00:00 AM	10	4	6
10/03/2012 02:00:00 AM	2	1	1
10/03/2012 03:00:00 AM	5	2	3
10/03/2012 04:00:00 AM	7	6	1
10/03/2012 05:00:00 AM	31	21	10
10/03/2012 06:00:00 AM	155	105	50
10/03/2012 07:00:00 AM	352	257	95
10/03/2012 08:00:00 AM	437	291	146
10/03/2012 09:00:00 AM	276	172	104
10/03/2012 10:00:00 AM	118	72	46
10/03/2012 11:00:00 AM	42	10	32
10/03/2012 12:00:00 PM	76	35	41
10/03/2012 01:00:00 PM	90	42	48
10/03/2012 02:00:00 PM	128	77	51
10/03/2012 03:00:00 PM	164	72	92
10/03/2012 04:00:00 PM	315	133	182

- Phân tích dữ liệu chuỗi thời gian (Time Series Data) sử dụng Pandas.
- Kết hợp với các biểu đồ để tìm ra được những Insight ẩn chứa trong tập dữ liệu.

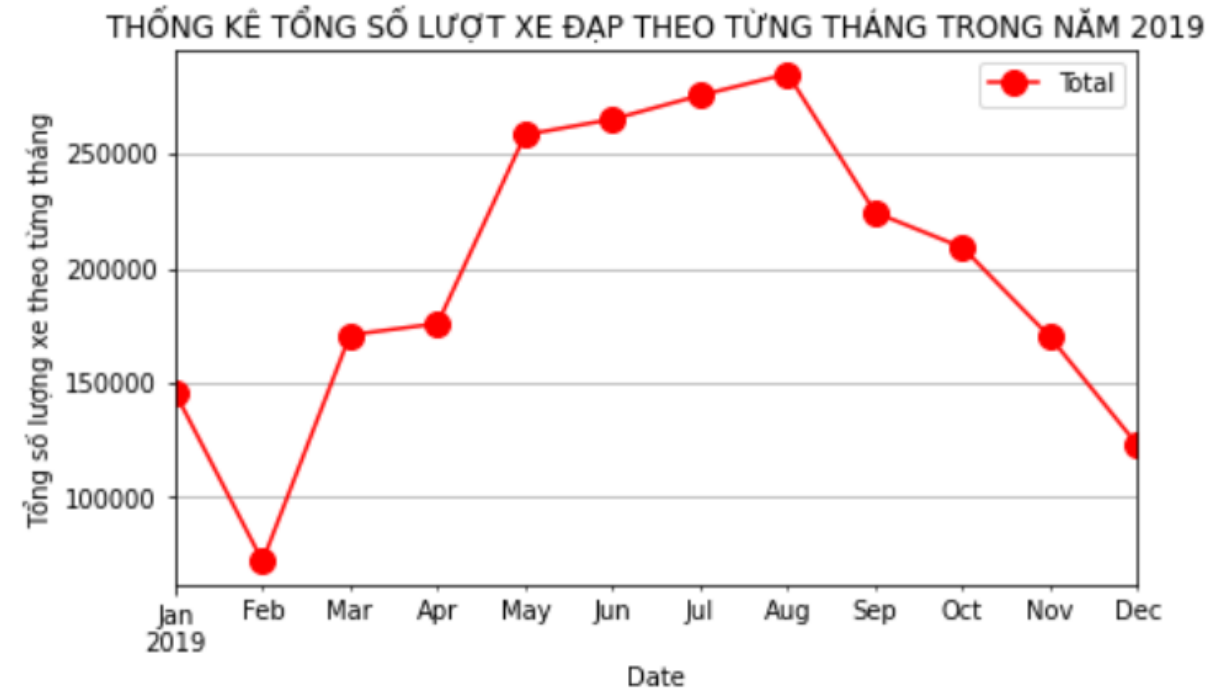
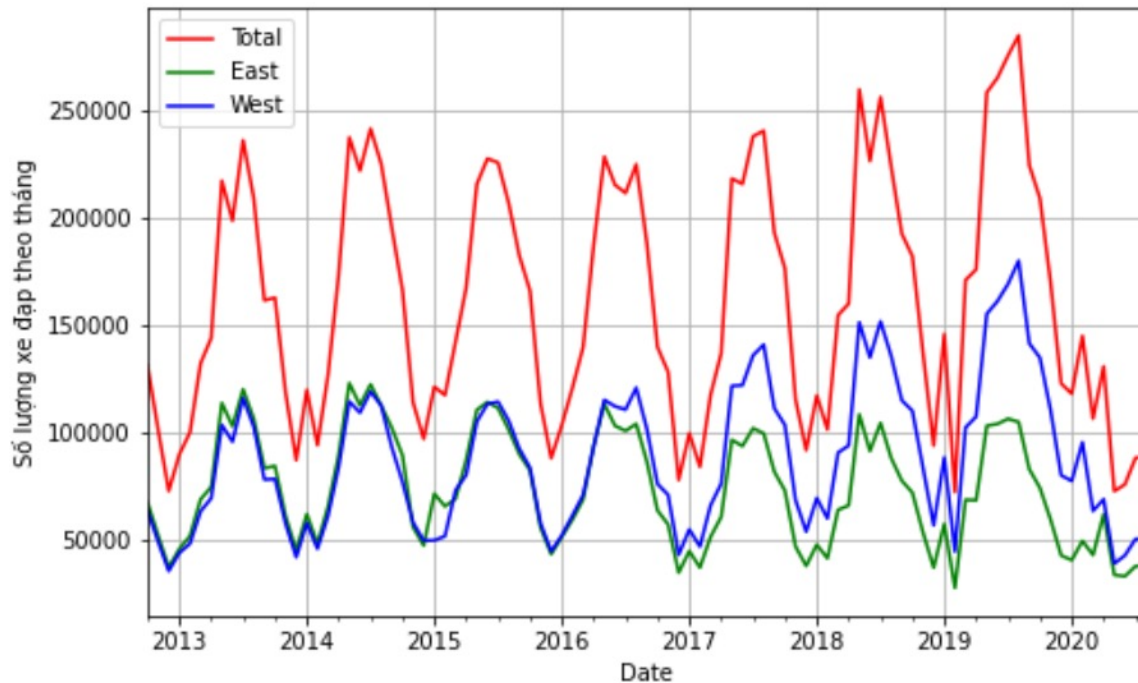
INSIGHT



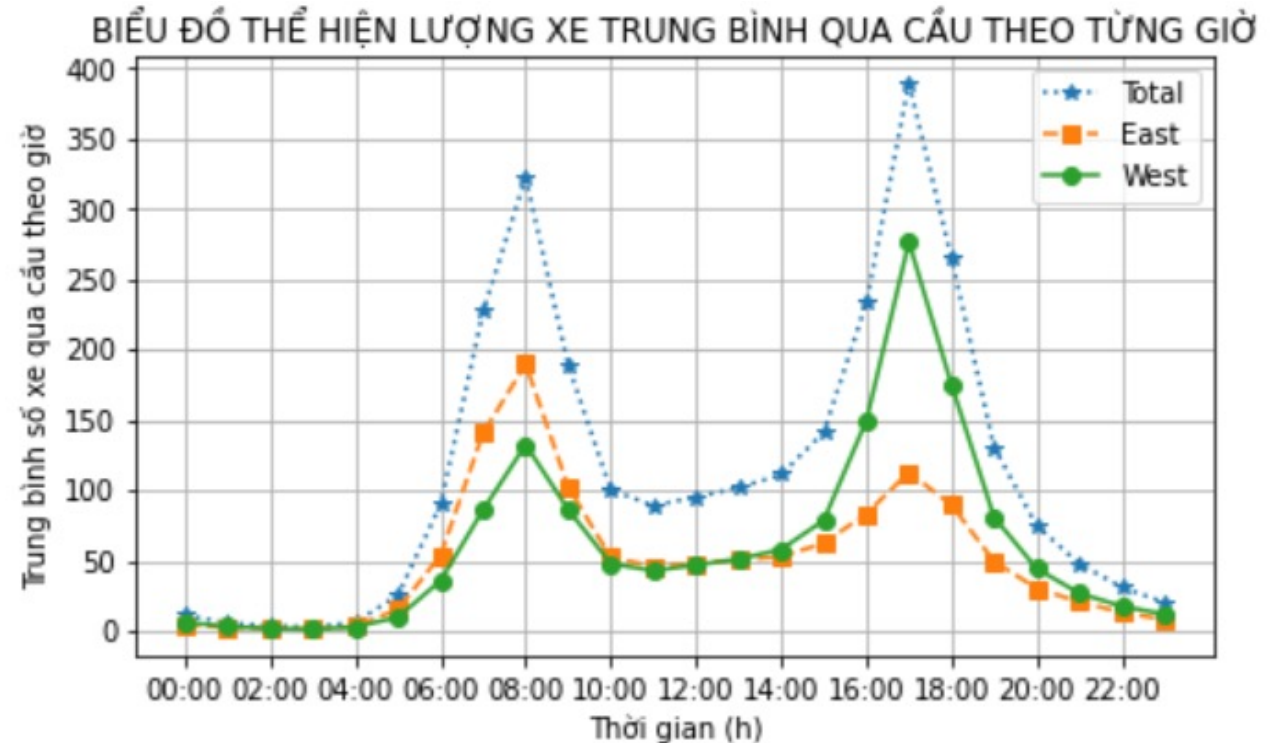
- 1) Từ biểu đồ thống kê tổng số xe đạp qua cầu theo năm ta thấy:
 - Số lượng người đi xe đạp qua cầu Fremont có xu hướng tăng lên theo từng năm, những năm gần đây tăng nhanh.
 - Lượt xe đạp qua lối đi phía tây nhiều hơn lối đi phía đông, và cũng có xu hướng tăng nhanh trong những năm gần đây.



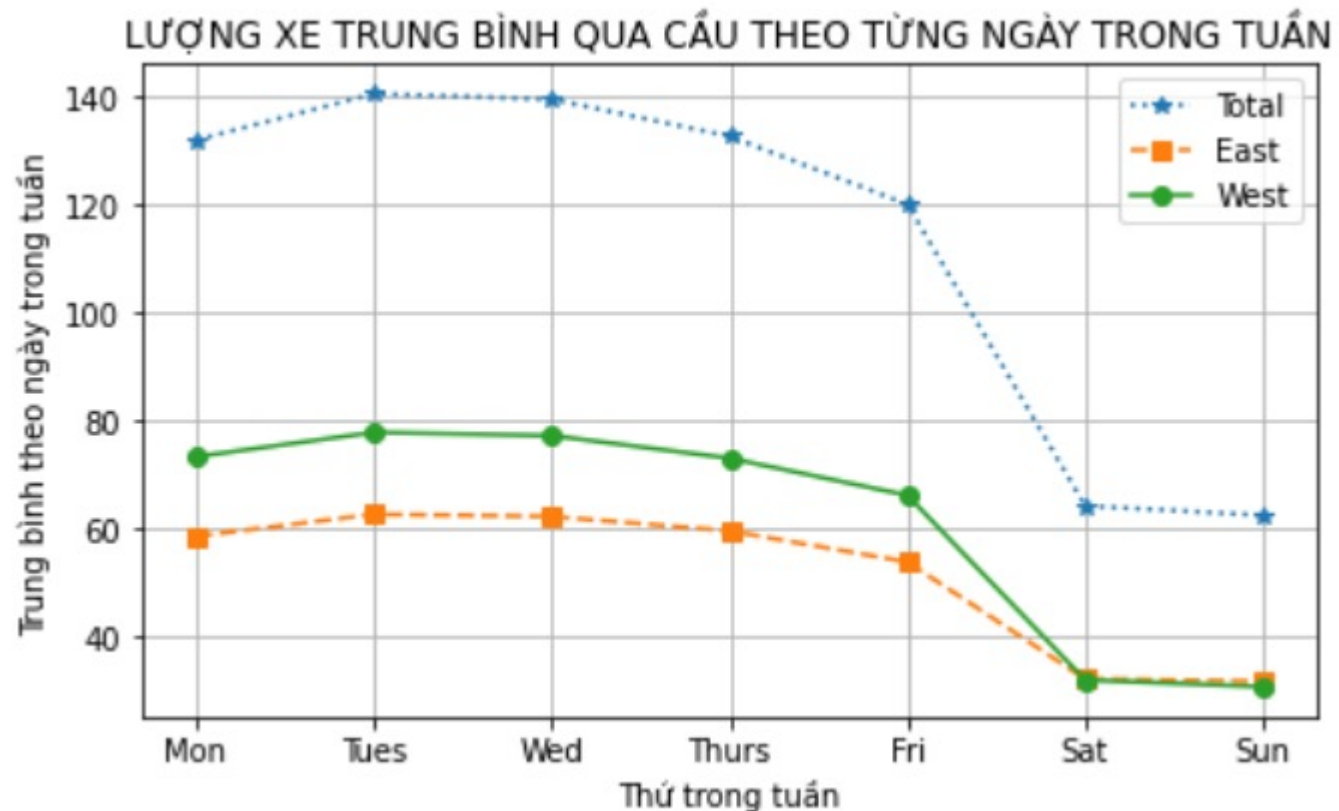
- 2) Từ biểu đồ thể hiện lượng xe đạp qua cầu theo tháng ta thấy:
 - Mọi người đạp xe nhiều hơn vào các tháng mùa hè và ít hơn vào các tháng mùa đông (4 tháng có số lượng người đạp xe nhiều nhất: 5, 6, 7 và 8)
 - Dữ liệu chuỗi thời gian về lượng xe đạp qua cầu có tính xu hướng (tăng dần) và tính thời vụ (số lượng nhiều hơn vào các tháng mùa hè và ít hơn vào các tháng mùa đông)



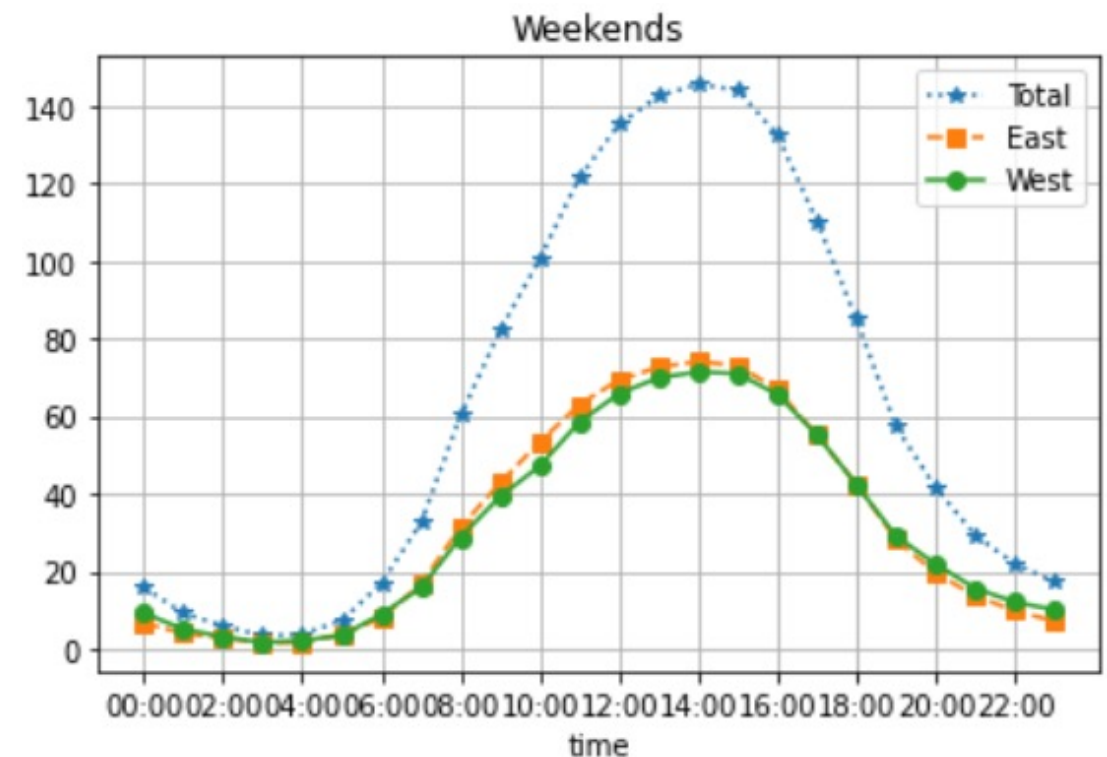
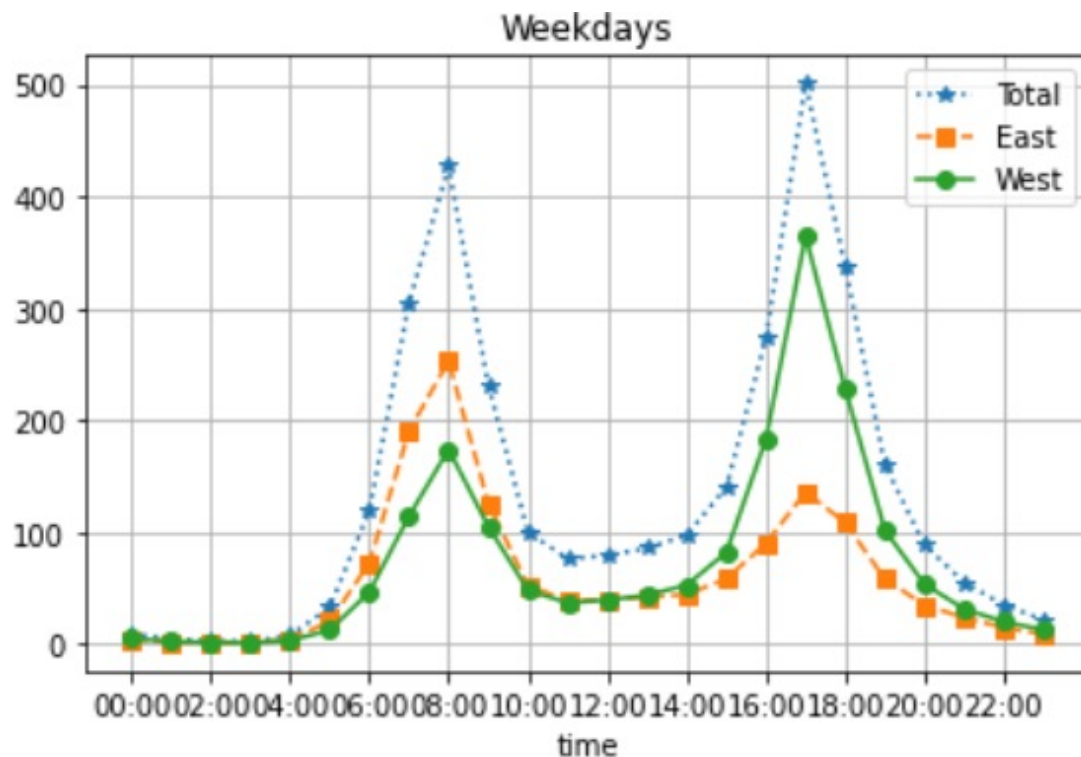
- 3) Từ biểu đồ thể hiện lượng xe đạp qua cầu trung bình theo giờ ta thấy:
 - Lượng người đi xe đạp qua cầu chủ yếu tập trung vào thời điểm 7,8,9h buổi sáng | 16, 17, 18h buổi chiều
 - Lượng người đi nhiều nhất vào thời điểm 8(h) sáng - 17h chiều.
 - Thời điểm buổi sáng lượng người đi qua cầu làn phía Đông (East) Lớn hơn làn phía Tây (Đi từ bên ngoài vào trung tâm thành phố Seattle) | Buổi chiều lượng người đi qua cầu làn phía Tây (West) lớn hơn (đi ra khỏi trung tâm thành phố).



- 4) Từ biểu đồ thể hiện lượng xe đạp qua cầu trung bình theo ngày trong tuần:
 - Lượng người đi xe đạp qua cầu chủ yếu vào các ngày làm việc trong tuần [thứ 2 --> thứ 6]; Cuối tuần [Thứ 7, CN] lượng người đi qua cầu giảm đi đáng kể. Lượng người đi qua cầu ngày làm việc gấp đôi ngày cuối tuần.



- 5) Từ biểu đồ thể hiện lượng xe đạp qua cầu trung bình theo các ngày trong tuần và các ngày cuối tuần theo từng giờ ta thấy:
 - Vào các ngày làm việc trong tuần lượng người đi xe đạp qua cầu chủ yếu tập trung vào thời điểm 7,8,9h buổi sáng | 16, 17, 18h buổi chiều. Lượng người đi nhiều nhất vào thời điểm 8(h) sáng - 17h chiều.
 - Vào các ngày cuối tuần, người đi xe đạp chủ yếu qua cầu trong thời gian từ 12-16h



Thực hành 5

Q & A
Thank you!