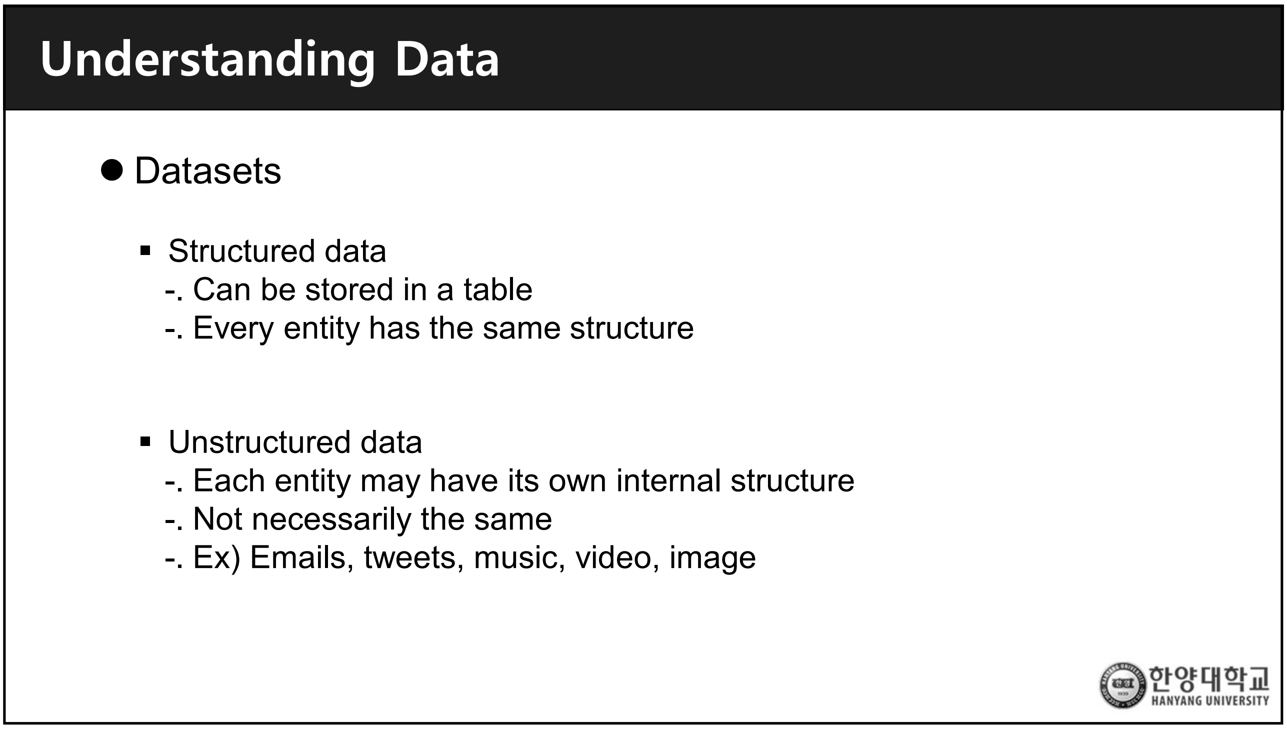
Understanding Data

- Dataset
  - Variable
    individual abstraction, which is also called feature
  - An entity or object
    a set of variables, which is a unit of rows.
  - Dataset
    contains a set of objects and each object is described by a set of variables

Understanding Data

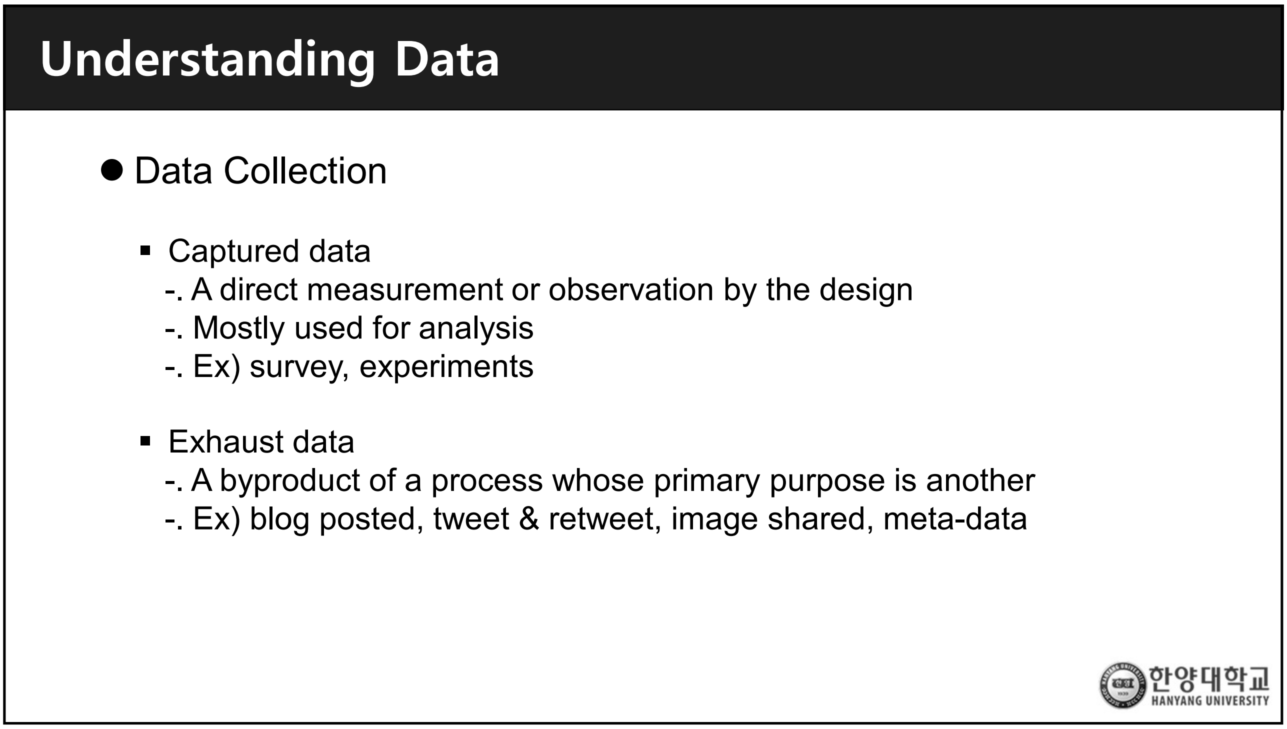
- Standard types of Variables
  - Numeric : measureable quantity

    (1) interval scale : fixed but arbitrary origin ex) time, date
    (2) ratio scale: having true zero-origin ex) height, weight
  - Nominal : categorical
    names for category from a finite set ex) gender
  - ordinal
    similar to nominal but it is possible to apply a rank order



Understanding Data

- Datasets
  - Structured data
    -. Can be stored in a table
    -. Every entity has the same structure
  - Unstructured data
    -. Each entity may have its own internal structure
    -. Not necessarily the same
    -. Ex) Emails, tweets, music, video, image

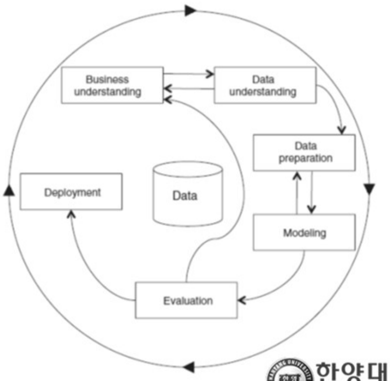


Understanding Data

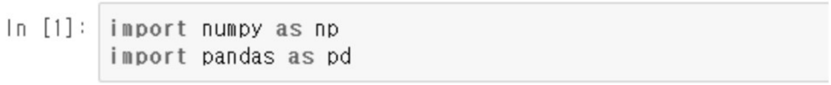
- Data Collection
  - Captured data
    -. A direct measurement or observation by the design
    -. Mostly used for analysis
    -. Ex) survey, experiments
  - Exhaust data
    -. A byproduct of a process whose primary purpose is another
    -. Ex) blog posted, tweet & retweet, image shared, meta-data

Understanding Data

- Cross Industry Standard Process for Data Mining (CRISP-DM)
  - Independent of any software, vendor or analysis techniques
  - Life cycle of 6 stages (Chapman et al, 1999)
    (1) business understanding
    (2) data understanding
    (3) data preparation
    (4) modeling
    (5) evaluation
    (6) deployment



Understanding Data



```
In [1]: import numpy as np
        import pandas as pd
```

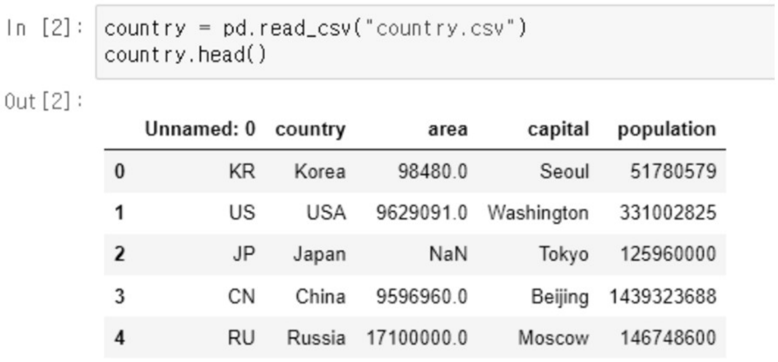
read_csv('file_name.csv', index_col = n)
-. *Read csv files*
-. *Index_col=n : set the index column to n th column*

head(n)
-. n: number of lines to be displayed from the top

tail(n)
-. n: number of lines to be displayed from the bottom

Understanding Data

- Data reading



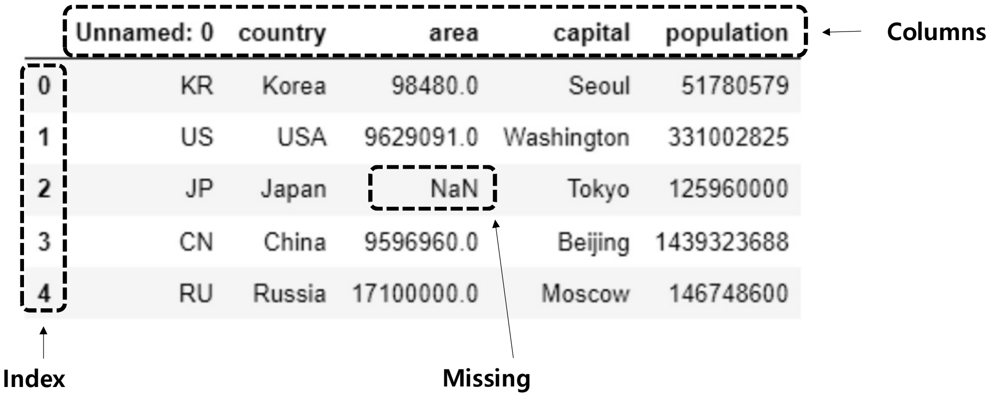
```
In [2]: country = pd.read_csv("country.csv")
        country.head()
```

Out[2]:

	Unnamed: 0	country	area	capital	population
0	KR	Korea	98480.0	Seoul	51780579
1	US	USA	9629091.0	Washington	331002825
2	JP	Japan	NaN	Tokyo	125960000
3	CN	China	9596960.0	Beijing	1439323688
4	RU	Russia	17100000.0	Moscow	146748600

Understanding Data

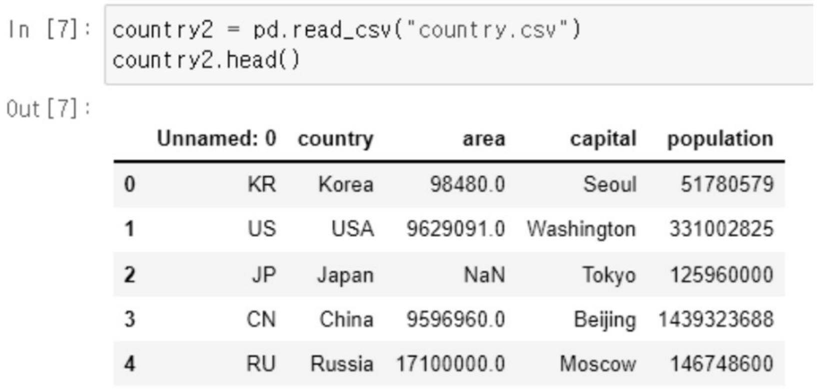
- Dataframe : Index and columns



	Unnamed: 0	country	area	capital	population
0	KR	Korea	98480.0	Seoul	51780579
1	US	USA	9629091.0	Washington	331002825
2	JP	Japan	NaN	Tokyo	125960000
3	CN	China	9596960.0	Beijing	1439323688
4	RU	Russia	17100000.0	Moscow	146748600

Understanding Data

- Index change to 0 column



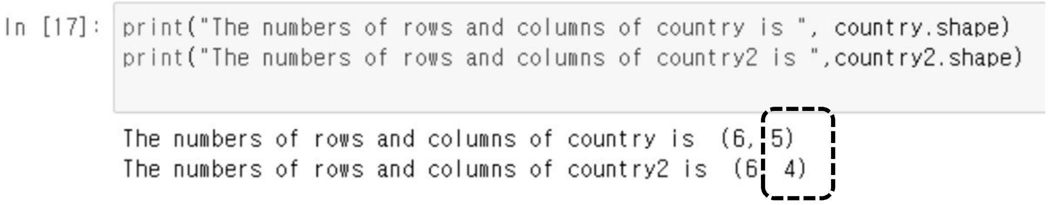
```
In [7]: country2 = pd.read_csv("country.csv")
        country2.head()
```

Out[7]:

	Unnamed: 0	country	area	capital	population
0	KR	Korea	98480.0	Seoul	51780579
1	US	USA	9629091.0	Washington	331002825
2	JP	Japan	NaN	Tokyo	125960000
3	CN	China	9596960.0	Beijing	1439323688
4	RU	Russia	17100000.0	Moscow	146748600

Understanding Data

shape
-. *return the dimension of data frame (#rows, #cols)*
-. *Index_col is not a column*



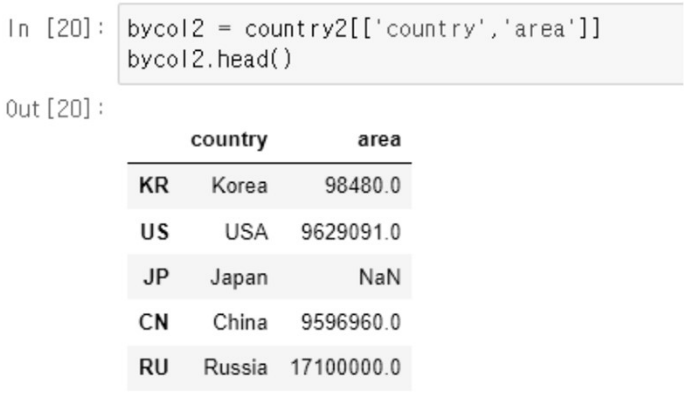
```
In [17]: print("The numbers of rows and columns of country is ", country.shape)
         print("The numbers of rows and columns of country2 is ",country2.shape)

The numbers of rows and columns of country is  (6, 5)
The numbers of rows and columns of country2 is  (6, 4)
```

Understanding Data

- Selection of observations
  - By columns
    : use the column name ex) country2[‘area’]
  - By rows
    : use the row numbers ex) country2[0:2]
    : use the index ex) country2.loc[‘KR’]
  - By columns & rows
    : use the both ex) country2[‘area’][:2]

Understanding Data

- Practice
  (1) Make a new dataframe by select ‘country’ and ‘area’ from country2.

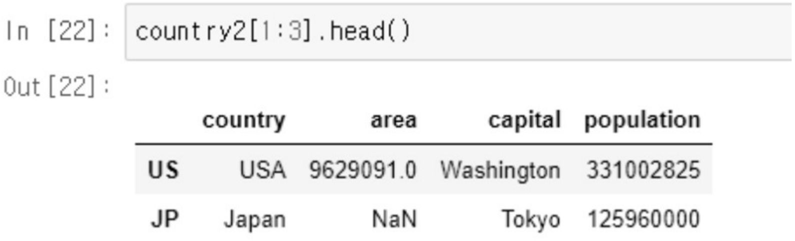


```
In [20]: bycol2 = country2[['country','area']]
         bycol2.head()
```

```
Out[20]:
        country        area
    KR    Korea     98480.0
    US      USA   9629091.0
    JP    Japan         NaN
    CN    China   9596960.0
    RU   Russia  17100000.0
```

Understanding Data

- Practice
  (2) Choose only 2 to 3 rows from country2.

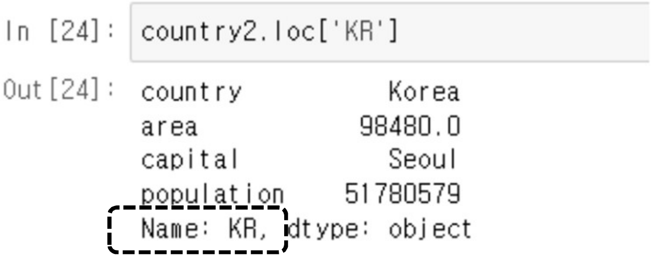


```
In [22]: country2[1:3].head()
Out[22]:
```

	country	area	capital	population
US	USA	9629091.0	Washington	331002825
JP	Japan	NaN	Tokyo	125960000

Understanding Data

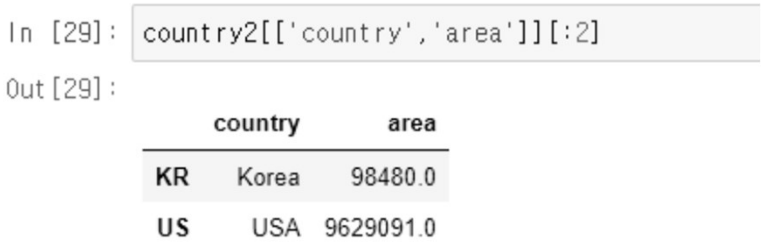
- Practice
  (3) Choose the row whose index is 'KR'.



```
In [24]: country2.loc['KR']
Out[24]: country           Korea
         area            98480.0
         capital           Seoul
         population     51780579
         Name: KR, dtype: object
```

Understanding Data

- Practice
  (4) select ‘country’ and ‘area’ on the first 2 rows.



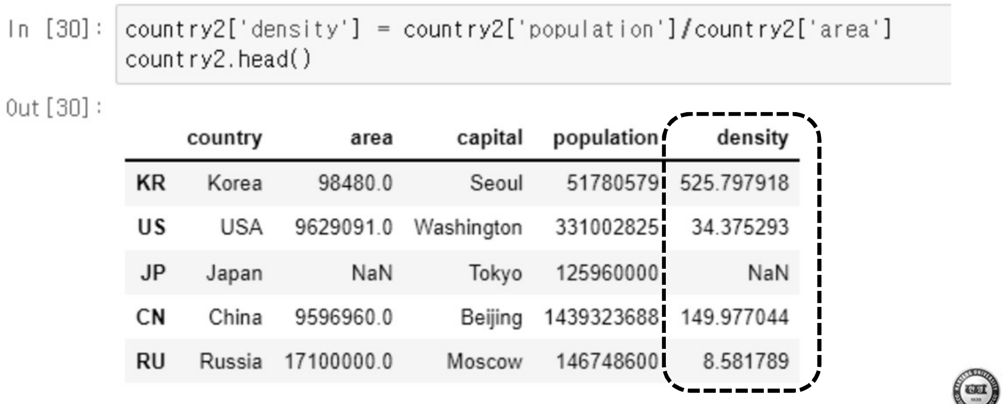
```
In [29]: country2[['country','area']][:2]
```

Out[29]:

	country	area
KR	Korea	98480.0
US	USA	9629091.0

Understanding Data

- Add a new column to the *dataframe*
  - Practice: Add ‘density’ column to’ country2’



```
In [30]: country2['density'] = country2['population']/country2['area']
         country2.head()
```

Out[30]:

	country	area	capital	population	density
KR	Korea	98480.0	Seoul	51780579	525.797918
US	USA	9629091.0	Washington	331002825	34.375293
JP	Japan	NaN	Tokyo	125960000	NaN
CN	China	9596960.0	Beijing	1439323688	149.977044
RU	Russia	17100000.0	Moscow	146748600	8.581789

```
DataFrame.append(other, ignore_index=False, verify_integrity=False, sort=None)
```

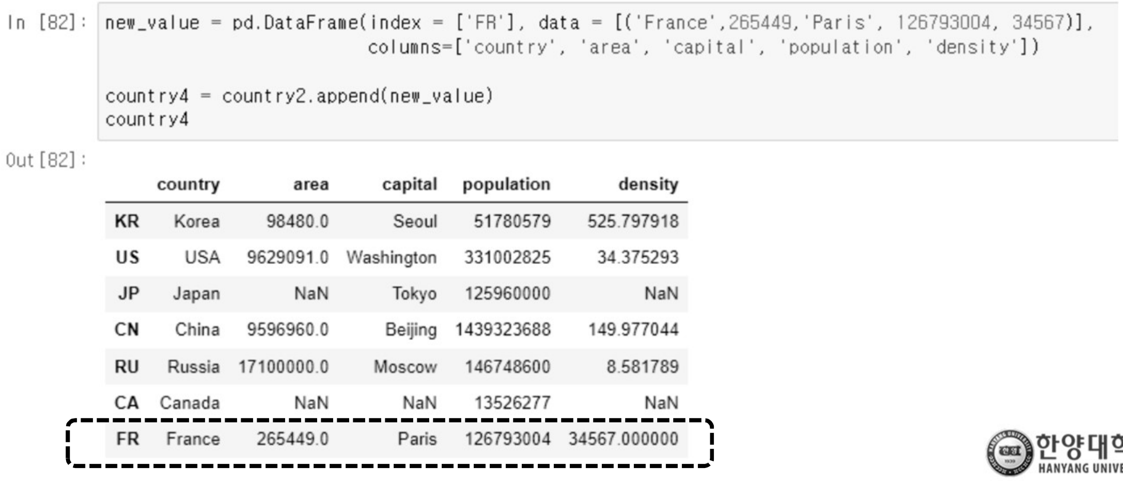
Understanding Data

dataframe.append(others, ignore_index=False, value_integrity=True)

-. Append rows of other dataframe to the end of the given dataframe
-. *others*: dataframe or *series or dic-like observations*
-. *Ignore_index : If True, do not use index labels*
- *value_integrity=True : return 'error' for duplicated indexes*

Understanding Data

- Practice:
  - Add 'FR France 265449.1 Paris 126793004 34567'

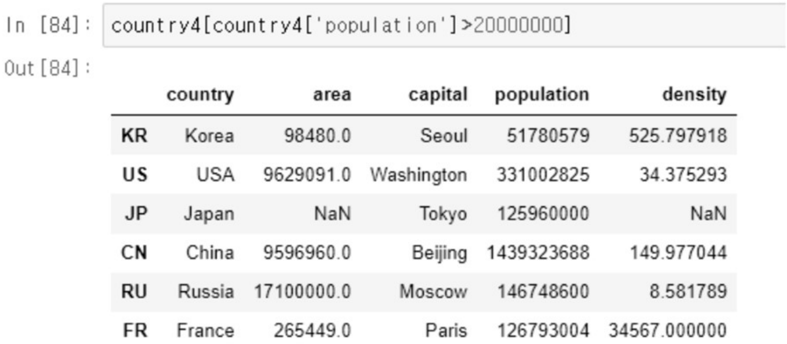


```
In [82]: new_value = pd.DataFrame(index = ['FR'], data = [('France',265449,'Paris', 126793004, 34567)],
                                  columns=['country', 'area', 'capital', 'population', 'density'])

         country4 = country2.append(new_value)
         country4
Out[82]:
```

	country	area	capital	population	density
KR	Korea	98480.0	Seoul	51780579	525.797918
US	USA	9629091.0	Washington	331002825	34.375293
JP	Japan	NaN	Tokyo	125960000	NaN
CN	China	9596960.0	Beijing	1439323688	149.977044
RU	Russia	17100000.0	Moscow	146748600	8.581789
CA	Canada	NaN	NaN	13526277	NaN
FR	France	265449.0	Paris	126793004	34567.000000

Understanding Data

- Filtering
  - Select the observations by the conditional clause
  - Practice: choose observations whose population > 20,000,000



```
In [84]: country4[country4['population']>20000000]
```

Out [84]:

	country	area	capital	population	density
KR	Korea	98480.0	Seoul	51780579	525.797918
US	USA	9629091.0	Washington	331002825	34.375293
JP	Japan	NaN	Tokyo	125960000	NaN
CN	China	9596960.0	Beijing	1439323688	149.977044
RU	Russia	17100000.0	Moscow	146748600	8.581789
FR	France	265449.0	Paris	126793004	34567.000000

Understanding Data

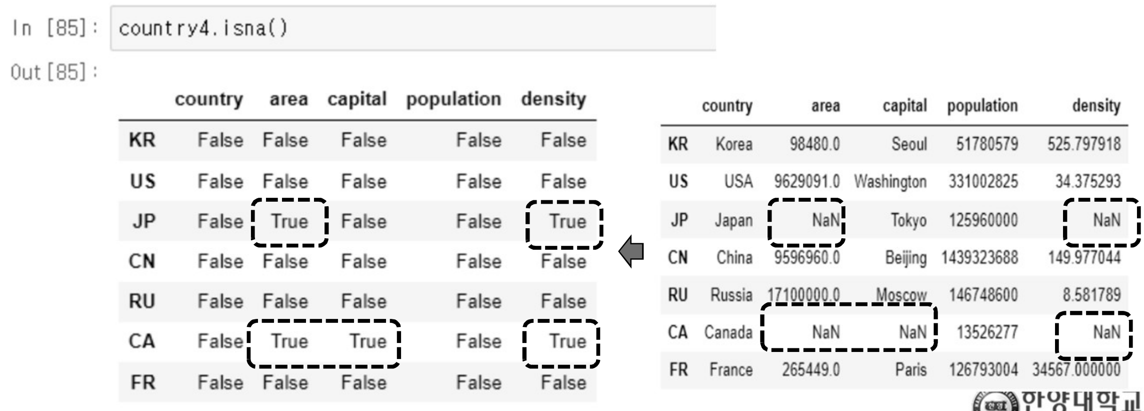
.isna( )
-. Select 'NaN' from the dataframe

.dropna(axis=0, how='any' , inplace=False)
-.remove rows or columns having 'NaN'
-. axis = 0: row, 1: column
-. how = 'any' : any one of the observations is 'NaN' in a row or column
= 'all' : all of the observations are 'NaN' in a row or column

.fillna(value , inplace=True)
-. value: 'NaN' is replaced with 'value'
-. Inplace=True: *fillna()* is operated in the original data

Understanding Data

- Practice
  - Show any ‘NaN’ in country4



```
In [85]: country4.isna()
```

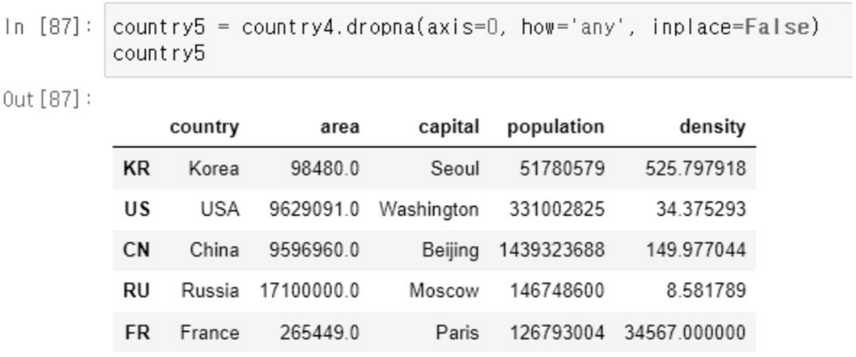
Out [85]:

	country	area	capital	population	density
KR	False	False	False	False	False
US	False	False	False	False	False
JP	False	True	False	False	True
CN	False	False	False	False	False
RU	False	False	False	False	False
CA	False	True	True	False	True
FR	False	False	False	False	False

	country	area	capital	population	density
KR	Korea	98480.0	Seoul	51780579	525.797918
US	USA	9629091.0	Washington	331002825	34.375293
JP	Japan	NaN	Tokyo	125960000	NaN
CN	China	9596960.0	Beijing	1439323688	149.977044
RU	Russia	17100000.0	Moscow	146748600	8.581789
CA	Canada	NaN	NaN	13526277	NaN
FR	France	265449.0	Paris	126793004	34567.000000

Understanding Data

- Practice
  - Remove rows having any ‘NaN’ : JP and CA rows were removed!



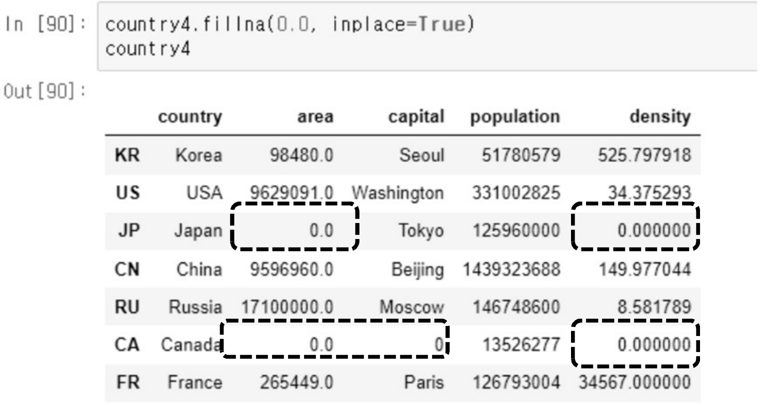
```
In [87]: country5 = country4.dropna(axis=0, how='any', inplace=False)
         country5
```

Out [87]:

	country	area	capital	population	density
KR	Korea	98480.0	Seoul	51780579	525.797918
US	USA	9629091.0	Washington	331002825	34.375293
CN	China	9596960.0	Beijing	1439323688	149.977044
RU	Russia	17100000.0	Moscow	146748600	8.581789
FR	France	265449.0	Paris	126793004	34567.000000

Understanding Data

- Practice
  - Replace any ‘NaN’ with 0 in country4 dataset.



In [90]:

```
country4.fillna(0.0, inplace=True)
country4
```

Out[90]:

	country	area	capital	population	density
KR	Korea	98480.0	Seoul	51780579	525.797918
US	USA	9629091.0	Washington	331002825	34.375293
JP	Japan	0.0	Tokyo	125960000	0.000000
CN	China	9596960.0	Beijing	1439323688	149.977044
RU	Russia	17100000.0	Moscow	146748600	8.581789
CA	Canada	0.0	0	13526277	0.000000
FR	France	265449.0	Paris	126793004	34567.000000