

제6장. Pandas



- 다양한 데이터 읽고 쓰기
- 기본 및 상세 정보 확인
- group by (aggregation)
- sort
- columns와 index
- 데이터 형태 변환
- 조건과 slicing, selecting, query
- 컬럼 변환 및 추가
- 데이터 합치기(merge)
- 결측치 대체, 이상치 및 중복값 처리

Study Point

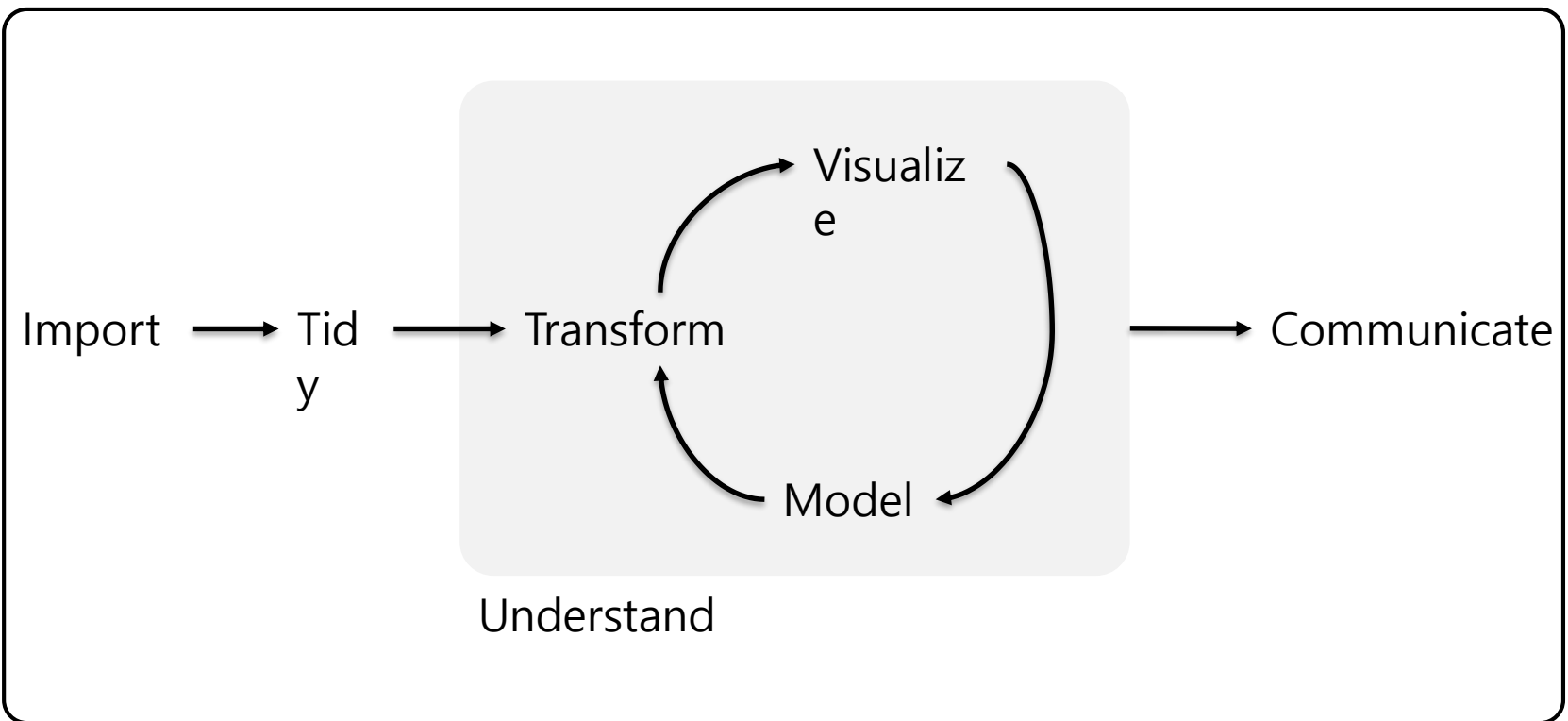
- **Pandas : Data Manipulation**

- DataFrame이 무엇인지에 대해 이해한다.
- csv파일, excel파일, DB의 데이터를 가져오고 저장해 본다.
- 탐색적 데이터 분석(Exploratory Data Analysis)을 수행해 본다.
- 필요한 데이터를 선택해 가져와 본다.
- 변수를 만들어 DataFrame 에 추가하고 삭제한다.
- DataFrame을 서로 연결한다.

Pandas

- 현실 세계 데이터 분석을 위한 빠르고 유연한 데이터 구조를 제공
- 손쉽게 다룰 수 있는 데이터
 - DataFrame: DB 테이블이나 엑셀 시트처럼 존재된 데이터 타입의 컬럼을 갖는 자료
 - Time series data: 시간에 따라 변화하는 시계열 자료(Stochastic data)
 - Tabular data: 행과 열이 라벨된 행렬 구조의 데이터
- 특징
 - 대용량 데이터 처리(다른 DataFrame이나 고차원 객체로 부터 컬럼을 추가 삭제)
 - Aggregation 기능, group by 함수
 - 라벨 등을 이용한 부분 데이터 집합 추출
 - 데이터를 가로 세로로 합치기
 - 외부 데이터 연동의 견고함(csv, excel, database, HDF5, SAS, STATA 등 지원)
 - 결측치 데이터 처리(missing data(NaN) 처리)
- import 관례
 - Import pandas as pd

Pandas_여러분이 배울 것

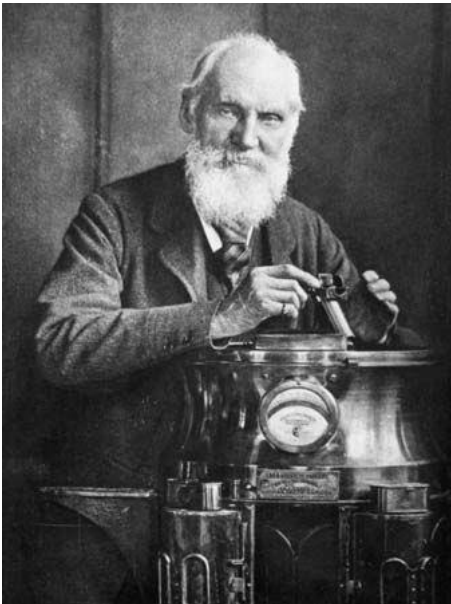


Program

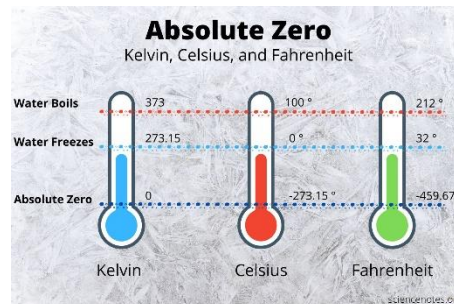
Pandas_빅데이터 표현

여러분이 말하고 있는 것을 측정하여 숫자로 표현할 수 있을 때, 비로소 여러분은 그것에 대해 어느 정도 알고 있는 것입니다. 만약 그렇지 못하다면, 여러분의 지식은 빈약하고 만족스럽지 못한 것입니다. 그 것은 지식의 시작일 수 있지만, (빅데이터) 과학의 단계까지는 아직 한참 멀었다고 할 수 있습니다.

Lord Kelvin




"When you can measure what you are speaking about, and express it in numbers, you know something about it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely, in your thoughts advanced to the stage of science."



Pandas_External Library

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



About us ▾ Getting started Documentation Community ▾ Contribute

pandas


pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

[Install pandas now!](#)

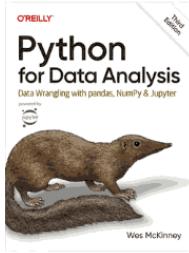
Latest version: 1.5.3

- What's new in 1.5.3
- Release date: Jan 19, 2023
- Documentation (web)
- Download source code

Follow us



Get the book




Previous versions

- 1.5.2 (Nov 22, 2022) [changelog](#) | [docs](#) | [code](#)
- 1.5.1 (Oct 19, 2022) [changelog](#) | [docs](#) | [code](#)
- 1.5.0 (Sep 19, 2022) [changelog](#) | [docs](#) | [code](#)
- 1.4.4 (Aug 31, 2022) [changelog](#) | [docs](#) | [code](#)

[Show more](#)


Coming from...

Are you familiar with other software for manipulating tabular data? Learn the pandas-equivalent operations compared to software you already know:




The R programming language provides the `data.frame` data structure and multiple packages, such as `tidyverse` use and extend `data.frame` for convenient data handling functionalities similar to pandas.

[Learn more](#)




Already familiar to `SELECT`, `GROUP BY`, `JOIN`, etc? Most of these SQL manipulations do have equivalents in pandas.

[Learn more](#)




The `data set` included in the STATA statistical software suite corresponds to the pandas `DataFrame`. Many of the operations known from STATA have an equivalent in pandas.

[Learn more](#)



Users of Excel or other spreadsheet programs will find that many of the concepts are transferrable to pandas.

[Learn more](#)



The SAS statistical software suite also provides the `data set` corresponding to the pandas `DataFrame`. Also SAS vectorized operations, filtering, string processing operations, and more have similar functions in pandas.

[Learn more](#)

Getting started

- Install pandas
- Getting started











Documentation

- User guide
- API reference
- Contributing to pandas
- Release notes

Community

- About pandas
- Ask a question
- Ecosystem

With the support of:

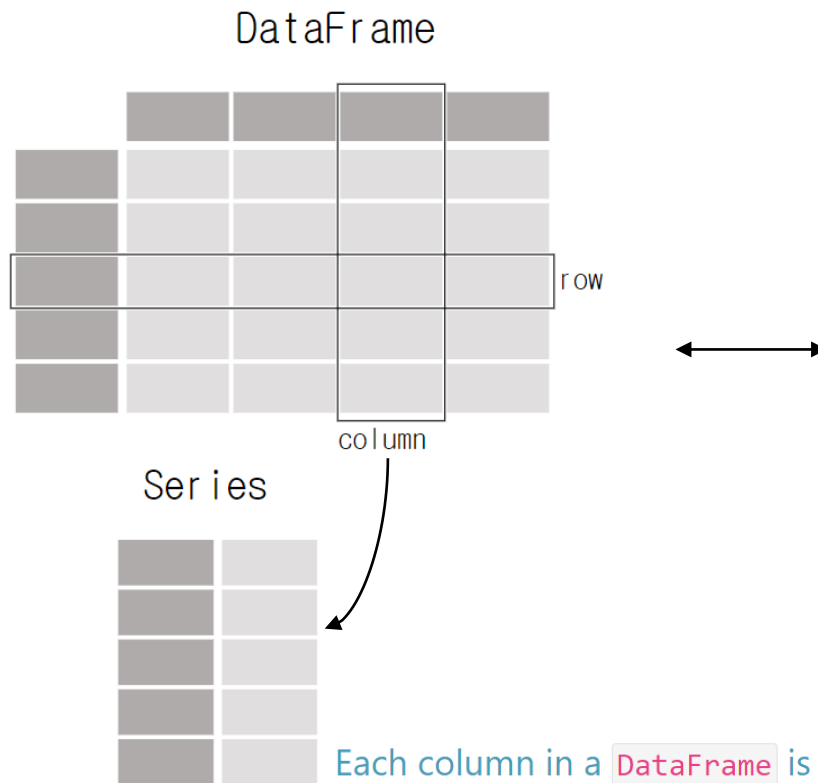


https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

Pandas

! pip install pandas

pandas data table representation



Untitled 1 - LibreOffice Calc

	A	B	C	D	E	F
1		Name	Age	Sex		
2	0	Braund, Mr. Owen Harris	22	male		
3	1	Allen, Mr. William Henry	35	male		
4	2	Bonnell, Miss. Elizabeth	58	female		
5						
6						
7						
8						

Sheet1

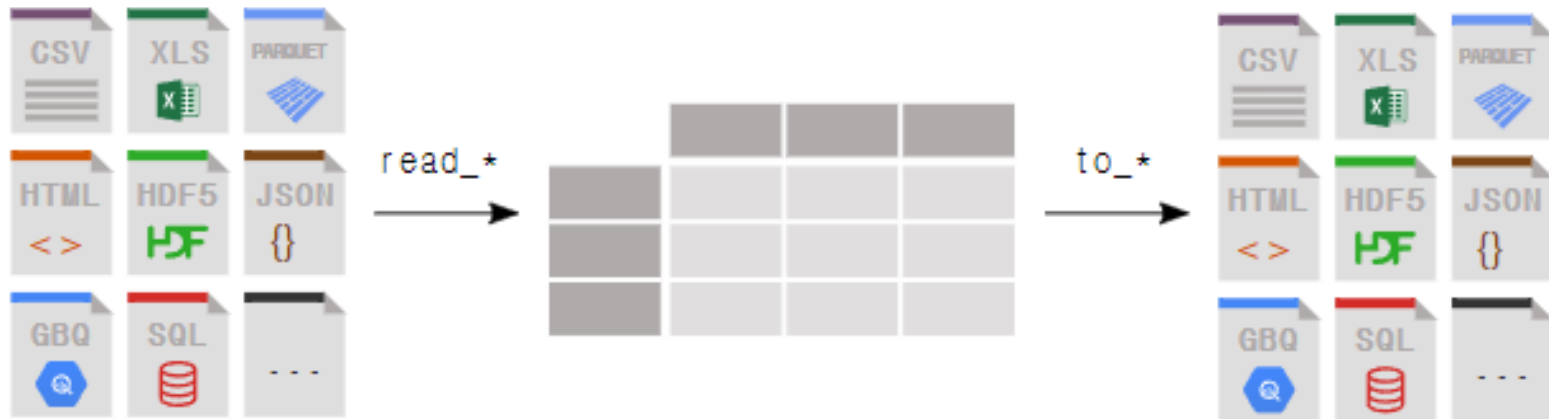
Sheet 1 of 1 | Default | English (USA) | Average: ; Sum: 0

Pandas – 외부데이터 읽기 쓰기

```
import pandas as pd
```

```
df = pd.read_*(‘읽어올 디렉토리/파일명.파일확장자’)  
df.to_*(‘내보낼 디렉토리/파일명.파일확장자’)
```

How do I read and write tabular data?



Pandas – 외부데이터 읽기 쓰기

```
import pandas as pd
```

```
# 인터넷 사이트에서 읽기
```

```
url="https://raw.githubusercontent.com/cs109/2014_data/master/countries.csv"
```

```
df = pd.read_csv(url) # url에서 읽기
```

```
df
```

```
url = 'https://raw.githubusercontent.com/e9t/nsmc/master/raw/10001.json'
```

```
df = pd.read_json(url)
```

```
df
```

```
# 구글에서 ‘영화매출순위’를 검색하여 해당 페이지의 html파일 불러오기
```

```
url = 'https://ko.wikipedia.org/wiki/%EC%98%81%ED%99%94\_%EB%A7%A4%EC%B6%9C\_%EC%88%9C%EC%9C%84\_%EB%AA%A9%EB%A1%9D'
```

```
df = pd.read_html(url)
```

```
dfs = pd.read_html(url)
```

```
dfs[2]
```

Pandas – 외부데이터 읽기 쓰기

```
# 엑셀로 만들어 나의 로컬 PC에 'test.xlsx'로 다운로드하기 (Co lab 환경)
df.to_excel('test.xlsx')                                     # 엑셀형식으로 내보내기
files.download('test.xlsx')

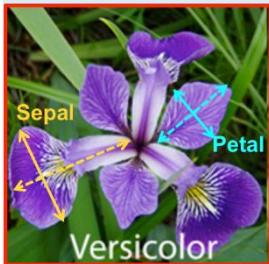
# 로컬 PC에서 읽기 (구글로 upload)
from google.colab import files
uploaded = files.upload()
df = pd.read_excel('06_무작위난수와 균등분포.xlsx')          # 엑셀파일 읽기
df
```

Pandas – scikit-learn 데이터 읽기

```
# iris dataset 읽기 # https://scikit-learn.org/stable/datasets.html
from sklearn.datasets import load_iris
load_iris()                                # 사전형태의 데이터
load_iris().keys()
# dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
print(load_iris()['DESCR'])

load_iris()['data'][:5]                    # numpy.ndarray
load_iris()['data'].shape                   # (150,4)
type(load_iris()['data'])                  # numpy.ndarray
load_iris()['feature_names']               # ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
type(load_iris()['feature_names'])         # list

df= pd.DataFrame(load_iris()['data'], columns = load_iris()['feature_names'])
df
```



Pandas – 엑셀을 읽고 DataFrame 생성

필요한 데이터 읽기

로컬 PC에서 읽기 (구글로 upload)

```
from google.colab import files  
uploaded = files.upload()
```

로컬에 파이썬 작업 중 이라면

```
path = r'C:\Users\조상구\Downloads'
```

```
pd.read_csv(path + '/도로교통공단_사망 교통사고 정보_20211231.csv')
```

내 PC에서 그대로('r') copy하여 사용

csv 파일 읽어 오기

```
pd.read_csv('도로교통공단_사망 교통사고 정보_20211231.csv') #Error, 구글선생에게 문의
```

```
-----  
UnicodeDecodeError                                Traceback (most recent call last)  
<ipython-input-63-0c4886e9c540> in <module>  
----> 1 pd.read_csv('도로교통공단_사망 교통사고 정보_20211231.csv')  
  
-----  
      9 frames  
/usr/local/lib/python3.9/dist-packages/pandas/_libs/parsers.pyx in pandas._libs.parsers.raise_parser_error()  
  
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xb9 in position 0: invalid start byte
```



UnicodeDecodeError: 'utf-8' codec can't decode byte 0xb9 in pc X

전체 이미지 뉴스 쇼핑 지도 더보기 도구

검색결과 약 1,500개 (0.35초)

tistory.com
https://zephyrus1111.tistory.com > ...

'utf-8' codec can't decode byte 0xb9 in position 0: invalid start ...
2020. 8. 13. — UnicodeDecodeError: 'utf-8' codec can't decode byte 0xb9 in position 0:
invalid start byte 저 같은 경우 한글이 포함된 csv파일이나 텍스트 ...

Pandas – 엑셀을 읽고 DataFrame 생성

- 로컬 PC에서 직접 drag하여 Co lab에 갖다 놓으면 구글에 upload 됨

The image shows a CoLab notebook interface on the left and a Windows File Explorer on the right. In the CoLab notebook, the file explorer on the left shows a folder named 'sample_data' containing several CSV files. One file, '도로교통공단_사망 교통사고 정보_20211231.csv', is highlighted with a dashed box. In the code cell on the right, the following code is shown:

```
[2]: !ls
      '도로교통공단_사망 교통사고 정보_20211231.csv' sample_data
```

The Windows File Explorer shows the local file system. The path is '내 PC > 로컬 디스크 (C:) > 빅데이터과 > 파이썬프로그래밍'. A file named '도로교통공단_사망 교통사고 정보_2021...' is highlighted with a dashed box. An arrow points from this file to the code cell in the CoLab notebook, indicating the upload process.

Pandas – 엑셀을 읽고 DataFrame 생성

csv 파일 읽어 오기

```
df = pd.read_csv('도로교통공단_사망 교통사고 정보_20211231.csv', encoding='cp949')
```

```
df = pd.read_csv('도로교통공단_사망 교통사고 정보_20211231.csv', encoding='euc-kr')
```

```
df = pd.read_csv('도로교통공단_사망 교통사고 정보_20211231.csv', sep=',', encoding='euc-kr')
```

csv 파일 쓰기

```
df.to_csv('ggdata.csv')
```

```
df.to_csv('ggdata.csv', encoding= ' euc-kr ' )
```

깨짐

OK

	발생 년	발생년월일 시	주 야	요일	사망 자수	부상 자수	중상 자수	경상 자수	부상신고 자수	발생지 시도	...	사고유 형	가해자법규 위반	도로형태 대분류	도로형태	가해자_당사 자종별	피해자_당사 자종별	발생위치 X(UTM)	발생위치 Y(UTM)	경도	위도
0	2021	2021-01-01 03:00	야	금	1	3	0	3	0	경북	...	추돌	안전운전 의무 불이행	교차로	교차로부근	승용차	승용차	1097010.0	1793385.0	128.578152	36.132653
1	2021	2021-01-01 09:00	주	금	1	0	0	0	0	충남	...	공작물 충돌	안전운전 의무 불이행	단일로	기타단일로	승용차	없음	902369.0	1847109.0	126.408201	36.616845
2	2021	2021-01-01 15:00	주	금	1	0	0	0	0	강원	...	측면충 돌	안전운전 의무 불이행	교차로	교차로내	원동기장치자 전거	승용차	1123975.0	1974509.0	128.907484	37.761842
3	2021	2021-01-01 19:00	야	금	1	0	0	0	0	전남	...	횡단충 돌	안전운전 의무 불이행	단일로	기타단일로	화물차	보행자	886507.0	1613961.0	126.263573	34.513391
4	2021	2021-01-01 21:00	야	금	1	0	0	0	0	경기	...	기타	기타	단일로	기타단일로	승용차	보행자	953522.0	1915403.0	126.976011	37.236327
...
2811	2021	2021-12-31 16:00	주	금	1	0	0	0	0	경북	...	정면충 돌	안전운전 의무 불이행	교차로	교차로내	승용차	이륜차	1119020.0	1766895.0	128.818730	35.891434
2812	2021	2021-12-31 17:00	주	금	1	0	0	0	0	제주	...	추돌	안전운전 의무 불이행	단일로	기타단일로	화물차	화물차	940588.0	1503049.6	126.860248	33.517699
2813	2021	2021-12-31 18:00	야	금	1	0	0	0	0	강원	...	횡단충 돌	보행자 보호의 무 위반	단일로	기타단일로	승용차	보행자	1023127.0	1982332.0	127.762845	37.840465
2814	2021	2021-12-31 19:00	야	금	1	0	0	0	0	경북	...	횡단충 돌	보행자 보호의 무 위반	교차로	교차로횡단 보도내	승용차	보행자	1058805.0	1824755.0	128.155943	36.418521
2815	2021	2021-12-31 21:00	야	금	1	0	0	0	0	강원	...	전복	중앙선 침범	단일로	기타단일로	승용차	없음	1042559.0	2010975.0	127.985386	38.097913

2816 rows × 23 columns

Pandas – 엑셀파일내 여러 시트 읽기

```
# 여러 개 시트가 있는 엑셀 파일의 경우
df = pd.DataFrame([[1,2,3], [4,5,6], [7,8,9]])
df
wb = pd.ExcelFile('04_다양한 예제 실습.xlsx')
wb
wb.sheet_names # list
df = pd.read_excel('04_다양한 예제 실습.xlsx', sheet_name='1부터 100 더하기')
df.head(3)
for i in wb.sheet_names:
    pd.read_excel('04_다양한 예제 실습.xlsx', sheet_name= i)
```

5	41	42	43	44	45	46	47	48	49	50		455
6	51	52	53	54	55	56	57	58	59	60		555
7	61	62	63	64	65	66	67	68	69	70		655
8	71	72	73	74	75	76	77	78	79	80		755
9	81	82	83	84	85	86	87	88	89	90		855
10	91	92	93	94	95	96	97	98	99	100		955
11												
12	460	470	480	490	500	510	520	530	540	550		5050
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												

1부터 100 더하기 | 이진법과 십진법 | 구구단 | 직선을 찾는 알고리즘 | +

Pandas – DBMS

```
# DBMS(sqlite)
! pip install sqlalchemy
```

```
from sqlalchemy import create_engine
engine = create_engine('sqlite://', echo=False)
engine = create_engine('sqlite:///memory:')
df.to_sql('test.df', my_engine, if_exists = 'append')
```

```
tf = pd.read_sql_query('SELECT * FROM test_df', my_engine)
tf
```

Comparison with other tools

Comparison with R / R libraries

Quick reference

Base R

plyr

reshape / reshape2

Comparison with SQL

Copies vs. in place operations

SELECT

WHERE

GROUP BY

JOIN

UNION

LIMIT

pandas equivalents for some SQL analytic and aggregate functions

UPDATE

DELETE

Comparison with spreadsheets

Data structures

Data input / output

Data operations

String processing

Merging

Other considerations

Comparison with SAS

Data structures

Data input / output

Data operations

String processing

Merging

Missing data

GroupBy

Other considerations

Comparison with Stata

SQLAlchemy

home features news documentation community download

The Python SQL Toolkit and Object Relational Mapper

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.

It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

Documentation

• Current Documentation (version 2.0) - learn SQLAlchemy here

- Documentation Overview
- Installation Guide
- ORM Quickstart
- Comprehensive Tutorial
- Reference Guides
 - Object Relational Mapping (ORM)
 - Core (Connections, Schema Management, SQL)
 - Dialects (specific backends)

• Documentation by Version

- Version 2.0
- Version 1.4
- Version 1.3

Pandas – 파일 압축하기

pickle로 파일 압축하기

```
import numpy as np
```

```
%time df = pd.DataFrame(np.random.random((int(1048576/20), int(16384/20))))
```

```
df
```

```
%time df.to_pickle('test.pkl')
```

%time은 명령문 수행 시간을 나타냄

```
%time df = pd.read_pickle('test.pkl')
```

```
df.info(memory_usage='deep')
```

```
%time df.to_pickle('test1.pkl', compression='gzip')
```

```
%time df = pd.read_pickle('test1.pkl', compression='gzip')
```

```
df.info(memory_usage='deep')
```

small 데이터로 자료 분석시 유용, 엑셀로 육안으로 확인하면서 분석식 가능

```
df.sample(frac=0.2)
```

전체 데이터의 20%만 무작위층화 sampling

```
df.sample(frac=0.2).shape
```

```
df.sample(10000)
```

전체 데이터에서 10,00개만 random stratified sampling

```
df.sample(10000).shape
```

Pandas – DataFrame 생성 – 직접 만들기

```
# list로 부터 만들기
df = pd.DataFrame([[1,2,3], [4,5,6], [7,8,9]])
df
# 행과 열에 이름 짓기
df = pd.DataFrame([[1,2,3], [4,5,6], [7,8,9]],
                  columns = ['a', 'b', 'c'],
                  index = [10,11,12])
df
# dictionary로 부터 만들기
df = pd.DataFrame({'a': [1,2,3], 'b': [4,5,6], 'c': [7,8,9]}, index = [10,11,12])
df
# numpy array로 부터 만들기
import numpy as np
arr = np.arange(9).reshape(3,3).T
df = pd.DataFrame(arr, columns = list('abc'), index = [10,11,12])
df
type(df)                                # pandas.core.frame.DataFrame

# 이렇게 하면 어떤 결과가 나올까?
df = pd.DataFrame({'a': [1,2,3], 'b': [4,5,6], 'c': [7,8,9]}, columns = ['c','b','a'])
```

Pandas – DataFrame 생성 – 직접 만들기

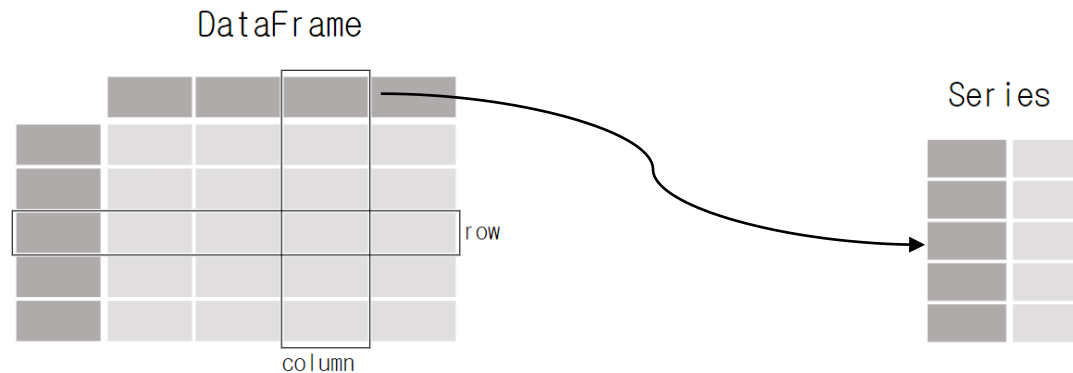
```
# 데이터프레임의 모든 컬럼은 시리즈이다.  
# 모든 시리즈를 컬럼으로 합치면 데이터프레임이다.
```

```
df['c']  
type(df['c'])
```

```
# pandas.core.series.Series
```

```
ages = pd.Series([22, 35, 58], name="Age")  
ages  
type(ages)
```

pandas data table representation



제 6 장. Pandas 배우기



- 다양한 데이터 읽고 쓰기
- 기본 및 상세 정보 확인
- group by (aggregation)
- sort
- columns와 index
- 데이터 형태 변환
- 조건과 slicing, selecting, query
- 컬럼 변환 및 추가
- 데이터 합치기(merge)
- 결측치 대체, 이상치 및 중복값 처리

Pandas – 공공데이터 – data.go.kr



도로교통공단_전국_사망



전체

이미지

뉴스

동영상

지도

더보기

도구

검색결과 약 3,960,000개 (0.45초)



data.go.kr

https://www.data.go.kr > data > openapi

도로교통공단_사망교통사고정보서비스 - 공공데이터포털

사망 교통사고에 대한 개별 정보를 제공(위치 데이터 포함)

https://www.data.go.kr > data > fileData

도로교통공단_사망 교통사고 정보 - 공공데이터포털

도로교통공단_사망 교통사고 정보 - 사망 교통사고에 대한 개별정보 제공(발생일시, 사고유형, 위치좌표 등)

치좌표 등) - 부상자수 = 중상자수 + 경상

파일데이터명: 도로교통공단_사망 교통

<https://www.data.go.kr/data/15070340/fileData.do>

DATA .GO.KR

데이터찾기

국가데이터맵

데이터요청

데이터활용

정보공유

이용안내

데이터 상세



도로교통공단_사망 교통사고 정보

- 사망 교통사고에 대한 개별정보 제공(발생일시, 사고유형, 위치좌표 등)

- 부상자수 = 중상자수 + 경상자수 + 부상신고자수

0 0 관심

파일데이터

오픈API

주천데이터

공공데이터활용지원센터는 공공데이터포털에 개방되는 3단계 이상의 오픈 포맷 파일데이터를 오픈 API(RestAPI 기반의 JSON/XML)로 자동변환하여 제공합니다.

오픈 API를 활용하기 위해서는 공공데이터포털 회원 가입 및 활용신청이 필요하며, 활용 관련 문의는 공공데이터활용지원센터로 연락주시기 바라며,

데이터 자체에 대한 문의는 아래 제공기관의 관리부서 전화번호로 연락주시기 바랍니다.

파일데이터는 로그인 없이 다운로드를 통해 이용하실 수 있습니다.

CSV 도로교통공단_사망 교통사고 정보

다운로드

Pandas – 공공데이터

공공데이터 읽기

```
df = pd.read_csv('도로교통공단_사망 교통사고 정보_20211231.csv', encoding='euc-kr')
```

```
df.columns
```

실습에 필요없는 컬럼 제거

```
df = df.drop(['발생년', '부상신고자수', '사망자수', '사고유형', '가해자법규위반',  
            '가해자_당사자종별', '피해자_당사자종별',  
            '발생위치X(UTMK)', '발생위치Y(UTMK)', '경도', '위도'], axis=1)
```

```
df
```

도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고 정보_20211231 - Excel																						
도로교통공단_사망 교통사고																						

Pandas – 기본 정보 확인

```
# 내용 확인
df.head()
df.tail()
# 크기 확인
df.shape
df.columns
df.index
df.values
len(df)
len(df.index)
len(df.columns)
# 기본 속성 확인
#df.describe()           # Series or DataFrame을 return
df.describe?            # df.describe(include='all')
df.dtypes
df.info()
df.count()
df.nunique()
df['가해자_당사자종별'].unique()
```

Pandas – 기본 정보 확인

```
# describe()에서 찾아 보기
```

```
df.describe().index
```

```
df.describe().columns
```

```
# null 개수
```

```
df.isnull()
```

```
df.notnull()
```

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

```
df.notnull().sum()
```

```
# aggregation 함수로 기본 정보 확인
```

```
df.sum()
```

```
df.count()
```

```
df.median() # 0.5가 median
```

```
df.quantile([0.25, 0.5, 0.75])
```

```
df.min()
```

```
df.max()
```

```
df.mean()
```

```
df.var()
```

```
df.std()
```


Pandas - 기본 정보 확인

```
# 변수별로 확인하기, pd.Series, 평균과 분산(표준편차)
# pandas visualization
# https://pandas.pydata.org/docs/user_guide/visualization.html
df['경상자수'].plot()
df['경상자수'].plot(figsize=(15,4)) # index(2,816개) 기준으로 수자를 표시
df['경상자수'].shape
#df['경상자수'].sum()/len(df)
average = df['경상자수'].sum()/len(df)
average

# ()으로 아주 긴 명령문을 가독성있게
import numpy as np
np.sqrt(
    (
        (df['경상자수'] - average)**2
    ).sum()
)/len(df)
)
np.sqrt((((df['경상자수'] - average)**2).sum())/len(df))
```

Pandas – 상세 정보 확인

수치형 특정 컬럼 내용 살펴보기 (int, float type)

df['경상자수'].head()

df.경상자수.head()

df['경상자수'].value_counts()

df['경상자수'].value_counts().plot(kind='hist')

df['경상자수'].value_counts().hist()

df['경상자수'].value_counts().plot(kind='bar', rot=60, figsize=(12,4))

df['경상자수'].hist()

df['경상자수'].plot(kind='pie')

df['경상자수'].max()

df['경상자수'].min()

df['경상자수'].mean()

df['경상자수'].std()

df['경상자수'].quantile([0.25, 0.5, 0.75])

df['경상자수'].isnull().head()

df['경상자수'].isnull.sum()

권장

중간에 공백이 있는 변수명 인식 에러

Pandas – 상세 정보 확인

```
# 범주형 특정 컬럼 내용 살펴보기 (object type)
df['발생지 시도'].head()
df.발생지 시도.head()
df['발생지 시도'].value_counts()
df['발생지 시도'].value_counts().plot(kind='hist')
df['발생지 시도'].value_counts().hist()
df['발생지 시도'].value_counts().plot(kind='bar', rot=60, figsize=(12,4))
df['발생지 시도'].hist()
df['발생지 시도'].plot(kind='pie')
df['발생지 시도'].max()
df['발생지 시도'].min()
df['발생지 시도'].mean()
df['발생지 시도'].std()
df['발생지 시도'].quantile([0.25, 0.5, 0.75])
df['발생지 시도'].isnull().head()
df['발생지 시도'].isnull.sum()
```

권장

Pandas – 상세 정보 확인

```
# value_counts()의 필요 정보 추출하기
df['발생지시도'].value_counts().index
df['발생지시도'].value_counts().values
df['발생지시도'].value_counts()['서울']
df['발생지시도'].value_counts().sort_values()
df['발생지시도'].value_counts().sort_values(ascending=False).plot(kind='bar')
df['발생지시도'].value_counts().sort_index()

# Nan을 고려한 value_counts
import numpy as np
tf = pd.DataFrame([[1,2,3], [4,'NaN',6], [7,8,np.nan]],
                  columns = ['a', 'b', 'c'],
                  index = [10,11,12])

tf
tf['a'].value_counts()           # 해당 컬럼에 NaN이 없는 경우
tf['b'].value_counts()           # 해당 컬럼에 NaN이 있는 경우
tf['b'].fillna('missing').value_counts()
tf['b'].value_counts(dropna=False)
tf['b'].value_counts(normalize=True)
tf['b'].value_counts(normalize=True).sum()
```

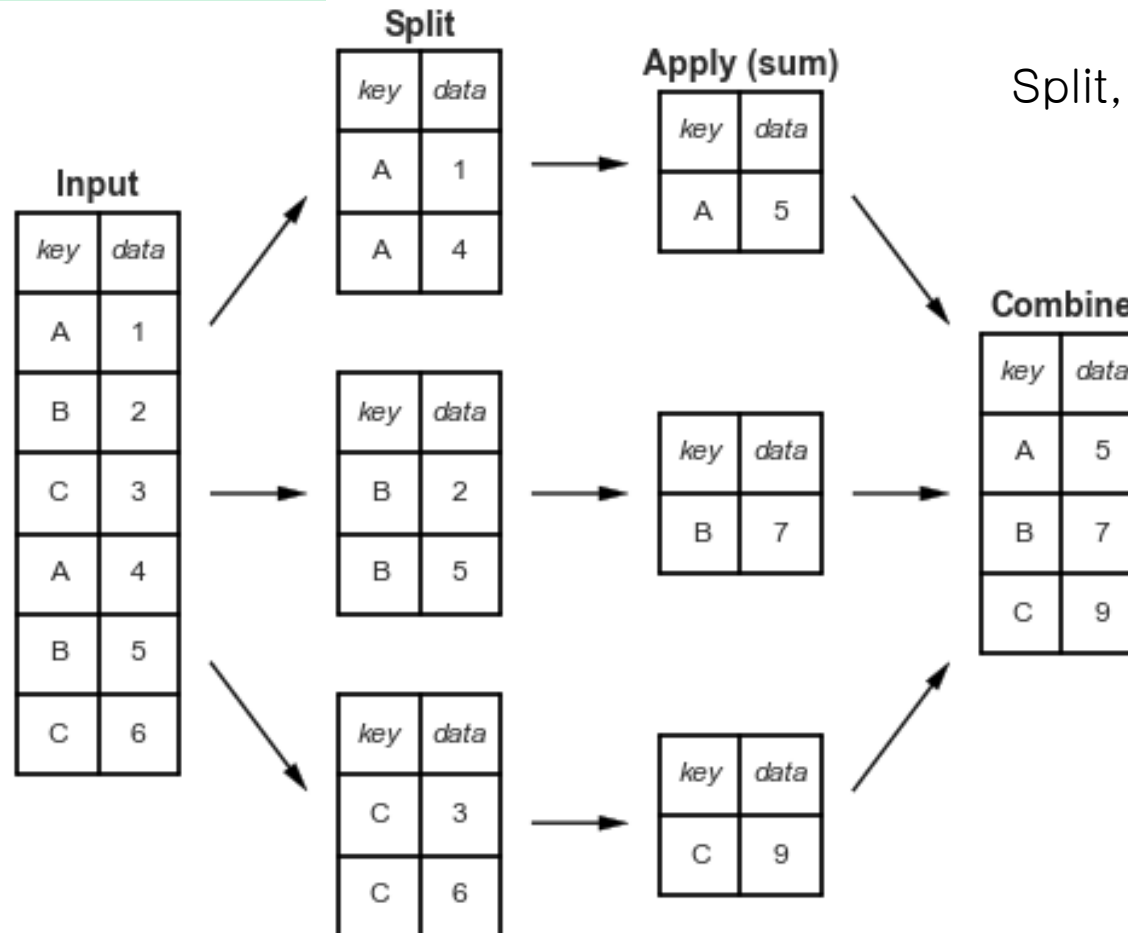
제 6 장. Pandas 배우기

- 다양한 데이터 읽고 쓰기
- 기본 및 상세 정보 확인
- group by (aggregation)
- sort
- columns와 index
- 데이터 형태 변환
- 조건과 slicing, selecting, query
- 컬럼 변환 및 추가
- 데이터 합치기(merge)
- 결측치 대체, 이상치 및 중복값 처리



Pandas – groupby

```
>>> input.groupby('key')['data'].sum()
```



from the [Python Data Science Handbook](#) by Jake VanderPlas; Jupyter notebooks are available [on GitHub](#).

Pandas – groupby

```
# 변수별로 확인하기
# groupby aggregation과 value_counts
# '발생지시도별'로 '부상자수'를 확인하는 방법은 무엇일까?
df.groupby('발생지시도')['부상자수']

for g, sf in df.groupby('발생지시도'):      # 발생지시도별로 aggregation하여 데이터 세트를 구성
    print('group by', g)
    sf.head(2)                             # 발생지시도별로 dataframe 생성

for g, sf in df.groupby('발생지시도'):      # 발생지시도별로 aggregation하여 데이터 세트를 구성
    print('group by', g)
    sf['부상자수'].value_counts()           # 발생지시도별로 dataframe 생성
    print('*'*100)

# 발생지시도별로 부상자수를 확인하는 방법 groupby
df.groupby('발생지시도')['부상자수']
df.groupby('발생지시도')['부상자수'].value_counts()
df.groupby('발생지시도')['부상자수'].value_counts().unstack()
df.groupby('발생지시도')['부상자수'].value_counts().unstack().fillna(0)
df.groupby('발생지시도')['부상자수'].value_counts().unstack().fillna(0).astype(int)
```

Pandas – groupby

```
# 테이블 시각화
# https://pandas.pydata.org/docs/reference/api/pandas.io.formats.style.Styler.background_gradient.html
# ()를 사용하여 아주 긴 명령문을 줄바꾸기기
(
df.groupby('발생지시도')['부상자수'].value_counts().unstack().          #
fillna(0).                      # NaN(결측치)을 숫자 0으로 대체
(imputation)
astype(int).                    # 실수형 정수로 변환(astype)
style.background_gradient(axis=0) # 열을 기준으로 크기에 따라 색깔 표
시, (axis=1)
)

# crosstab 형태로 전환
pd.crosstab(df['발생지시도'], df['부상자수'])

# pivot_table 형태로 전환
df.pivot_table(index='발생지시도', columns='도로형태', values='부상자수', aggfunc='sum')
df.pivot_table(index='발생지시도', columns='부상자수', values='요일',
               aggfunc='count') #.fillna(0)
```


Pandas – groupby

```
# plot 자료 시각화
df.groupby('발생지 시도')['부상자수'].value_counts().unstack().plot(kind='bar', figsize=(25,6))

df.pivot_table(index=['발생지 시도'], columns=['부상자수'], values=['요일'],
aggfunc='count').fillna(0).plot(kind='bar', figsize=(25,6))

pd.crosstab(df['발생지 시도'], df['부상자수']).plot(kind='bar', figsize=(25,6))

cross = df.pivot_table(index=['발생지 시도'], columns=['부상자수'], values=['요일'],
aggfunc='count') #.fillna(0)

# NaN을 고려한 value_counts
cross.T['강원'].value_counts()
cross.T['강원'].fillna('missing').value_counts()
cross.T['강원'].value_counts(dropna=False)
cross.T['강원'].value_counts(normalize=True)
cross.T['강원'].value_counts(normalize=True).sum()
```

Pandas – groupby

```
# groupby + sum vs. value_counts
df.groupby('발생지시도')['부상자수'].sum()
df.pivot_table(values=['부상자수'], index=['발생지시도'], aggfunc=sum)
df['발생지시도'].value_counts().sort_index()
df.groupby(['발생지시도', '요일'])['부상자수'].sum().unstack()
df.groupby(['발생지시도', '요일', '주야'])['부상자수'].sum().unstack()
# plot
df.groupby('발생지시도')['부상자수'].sum().plot(kind='bar')
# pivot_table로 구현
df.pivot_table(values=['부상자수'], index=['발생지시도'], aggfunc=sum)
df.pivot_table(values=['부상자수'], index=['발생지시도', '요일'], aggfunc=sum)
df.pivot_table(values=['부상자수'], index=['발생지시도', '요일'], aggfunc=sum).unstack()
df.pivot_table(values=['부상자수'], index=['발생지시도', '요일'], aggfunc=sum)
df.pivot_table(values=['부상자수'], index=['발생지시도'], columns=['요일'], aggfunc=sum)
# size 함수 : element 의 개수
df.groupby('발생지시도')['부상자수'].sum()
df.groupby('발생지시도')['부상자수'].size()
df.groupby('발생지시도')['부상자수'].size().sort_values(ascending=False)
df['발생지시도'].value_counts()
```

Pandas – groupby

aggregation 함수, groupby 결과도 시리즈나 데이터프레임이다.

```
df.groupby('발생지시도')['부상자수'].sum()
df.groupby('발생지시도')['부상자수'].count()
df.groupby('발생지시도')['부상자수'].median()
df.groupby('발생지시도')['부상자수'].quantile([0.25, 0.5, 0.75])
df.groupby('발생지시도')['부상자수'].min()
df.groupby('발생지시도')['부상자수'].max()
df.groupby('발생지시도')['부상자수'].mean()
df.groupby('발생지시도')['부상자수'].var()
df.groupby('발생지시도')['부상자수'].std()
```

agg함수로 한번에 표현

```
df.groupby('발생지시도')['부상자수'].agg(['sum', 'count', 'median', 'max', 'min', 'mean', 'var', 'std'])
```

제 6 장. Pandas 배우기



- 다양한 데이터 읽고 쓰기
- 기본 및 상세 정보 확인
- group by (aggregation)
- sort
- columns와 index
- 데이터 형태 변환
- 조건과 slicing, selecting, query
- 컬럼 변환 및 추가
- 데이터 합치기(merge)
- 결측치 대체, 이상치 및 중복값 처리

Pandas – 기본 정보 확인 – sort

```
df['발생지시도'].value_counts()
type(df['발생지시도'].value_counts())
# index에 있던 시도를 컬럼으로 reset하여 Series를 DataFrame으로 전환
tf = df['발생지시도'].value_counts().reset_index()
tf

# columns명을 부여
tf.columns = ['시도', '회수']
# 컬럼을 추출하고 sort
tf['시도'].sort_values()
tf['회수'].sort_values()
tf['회수'].sort_values(ascending=False)
tf['회수'] # 원래에는 영향을 미치지 않음

# 전체 테이블에 대해 sort
tf.sort_values('시도')
tf.sort_values('시도', ascending=False)
tf.sort_values(['회수', '시도'])
tf.sort_values(['회수', '시도'], ascending=[False, True]) # 각 컬럼별로 sorting 기준을 적용
tf.sort_values('시도')['시도']
```

제 6 장. Pandas 배우기



- 다양한 데이터 읽고 쓰기
- 기본 및 상세 정보 확인
- group by (aggregation)
- sort
- columns와 index
- 데이터 형태 변환
- 조건과 slicing, selecting, query
- 컬럼 변환 및 추가
- 데이터 합치기(merge)
- 결측치 대체, 이상치 및 중복값 처리

Pandas – column 명

```
# Column 명
ta = df[['발생지시도', '부상자수']]          # 2개 컬럼으로만 데이터프레임 구성
ta.head()
# 컬럼명 변경 - rename 메소드 이용
ta = ta.rename(columns={'발생지시도': '시도', '부상자수': '부상'})
ta.columns

# 컬럼명 변경 - columns값 치환
ta.columns= ['발생지시도', '부상자수']
ta.columns

# 컬럼명 조건에 의해 동시 변경
ta.columns = ta.columns.str.replace('발생지', ''); ta.columns
ta.columns= ['발생지시도', '부상자수']
ta.columns.str.replace('발생지|자수', repr)    # representation 해당 건을 보여줌
ta.columns = ta.columns.str.replace('발생지|자수', ''); ta.columns
ta.columns= ['발생지시도', '부상자수']; ta.columns

# 컬럼 순서 바꾸기
ta[['부상자수', '발생지시도']]
```

Pandas – 구조 변환 – index

```
# DataFrame 생성시 index 자동 생성
import numpy as np
tf = pd.DataFrame({'학생': np.random.choice(range(1, 11), 10, replace=False),
                  '국어': np.random.randint(70, 101, 10),
                  '영어': np.random.randint(60, 101, 10)})

tf

# 학생 컬럼을 index로 사용
tf.set_index('학생')
tf.set_index('학생', inplace=True) # tf = tf.set_index('학생')과 같은 명령어

# index로 sorting
tf = tf.sort_index(ascending=True)

# 국어 컬럼으로 내림차순 sort
tf = tf.sort_values('국어', ascending=False)

# index인 학생 컬럼을 데이터 컬럼으로 환원
tf.reset_index()
tf.reset_index(inplace=True)
```


제 6 장. Pandas 배우기



- 다양한 데이터 읽고 쓰기
- 기본 및 상세 정보 확인
- group by (aggregation)
- sort
- columns와 index
- 데이터 형태 변환
- 조건과 slicing, selecting, query
- 컬럼 변환 및 추가
- 데이터 합치기(merge)
- 결측치 대체, 이상치 및 중복값 처리

Pandas – 구조 변환 – dtype

```
# 각 컬럼의 date type을 확인하고 각각 변경
```

```
# 수치형의 경우
```

```
df['부상자수'].dtypes
```

```
df['부상자수'] = df['부상자수'].astype(float)
```

```
df['부상자수'].dtypes
```

```
df['부상자수'].dtypes == float
```

```
for col in df.columns:
```

```
    if df[col].dtype == int: # float로 boolean
```

```
        print(col)
```

```
        df[col] = df[col].astype(float)
```

```
df['부상자수'].dtype
```

Pandas – 구조 변환 – category type

```
# category형의 경우
df.info()          # df.dtypes

# 특정 컬럼의 data type을 category type으로 변경
df['발생지시도'].dtype          # 파이썬은 문자열을 string, 판다스는 object라고 한다.
for col in df.columns:
    if df[col].dtype == object: # float로 boolean
        print(col)
        df[col] = df[col].astype('category')

df.info()          # 메모리 사용량 확인

df.발생지시도
sorted(df.발생지시도.unique())
df.발생지시도.cat.codes
df.발생지시도.cat.codes.unique()
```

Pandas – 구조 변환 – category type

```
# dictionary 자료형을 사용하여 특정 data type만 추출
zip(df.발생지시도, df.발생지시도.cat.codes)
dict(zip(df.발생지시도, df.발생지시도.cat.codes))
dict(df.dtypes)
dict(df.dtypes).items()
dict(df.dtypes).keys()
dict(df.dtypes).values()
```

```
[key for key in dict(df.dtypes).keys() if dict(df.dtypes)[key] in ['float64', 'int64']]
[key for key in dict(df.dtypes).keys() if dict(df.dtypes)[key] in ['category']]
[key for key in dict(df.dtypes).keys() if dict(df.dtypes)[key] in ['object']]
```

```
# category type을 모두 object type으로 일괄 변경
for k in dict(df.dtypes).keys(): # for k in dict(df.dtypes):
    if dict(df.dtypes)[k] == 'category':
        df[k] = df[k].astype('object')
df.info()
for k in dict(df.dtypes).keys(): # for k in dict(df.dtypes):
    if dict(df.dtypes)[k] == 'object':
        df[k] = df[k].astype('category')
```

Pandas – 구조 변환 – category type

```
# dictionary의 key, value를 각각 추출해서 비교
[k for k, v in dict(df.dtypes).items() if v == 'int64']
[k for k, v in dict(df.dtypes).items() if v == 'float64']

# tf 데이터프레임에 현장학습 컬럼 추가
tf = pd.DataFrame({'학생': np.random.choice(range(1, 11), 10, replace=False),
                  '국어': np.random.randint(70, 101, 10), '영어': np.random.randint(60, 101, 10)})
tf['현장학습'] = pd.Series(np.random.choice(['상', '중', '하', 'X'], 10)) # pd.DataFrame(~~)
tf.sort_values('현장학습')      # X를 맨 뒤에 놓고 싶으면,

# 순서있는 category type 만들기
from pandas.api.types import CategoricalDtype
cat_type = CategoricalDtype(categories= ['상', '중', '하', 'X'], ordered = True)

# 순서있는 category type 적용용
tf['현장학습'] = tf['현장학습'].astype(cat_type)
tf.sort_values('현장학습')

# Category에 순서(order)가 있으니 '중'이하 성적 학생을 뽑으면
tf.loc[tf['현장학습'] >= '중', :]
```

제 6 장. Pandas 배우기

- 다양한 데이터 읽고 쓰기
- 기본 및 상세 정보 확인
- group by (aggregation)
- sort
- columns와 index
- 데이터 형태 변환
- 조건과 slicing, selecting, query
- 컬럼 변환 및 추가
- 데이터 합치기(merge)
- 결측치 대체, 이상치 및 중복값 처리



Pandas – loc, iloc – column 선택

```
# loc
df.loc[0, :]
df.loc[[0,1,2], :]
df.loc[0:3, :]          # 0,1,2 까지 index가 아니라 0,1,2,3 까지
df.loc[:, ['주야', '요일']]

df[df['발생지시도'] == '경기']
df.loc[df['발생지시도'] == '경기', :]

df[df['발생지시도'] == '경기']['부상자수']
df.loc[df['발생지시도'] == '경기', :]['부상자수']

df[df['발생지시도'] == '경기']['부상자수', '사고유형']
df.loc[df['발생지시도'] == '경기', :]['부상자수', '사고유형']

# iloc --> numpy array slicing과 유사
df.iloc[:, 0:4]
df.iloc[0:3, :]
df[0:5]
```

Pandas – loc, iloc – column 선택

```
# column 선택
df['부상자수'].head()
df.부상자수.head()
```

```
# 권장
# 2 개 이상 컬럼 지정 못함
```

```
# 두개 이상의 컬럼 선택, loc와 iloc
df[['요일', '부상자수']].head()
df.iloc[:, [5,7,9]].head()
```

```
# 두개 이상의 연속된 컬럼 선택
df.loc[:, '부상자수':'발생지시군구'].head()
df.iloc[:, 4:11].head()
df.filter(regex='자수').head()
```

```
# '자수' 글자가 포함된 컬럼인 경우(True)
```

```
# 조건과 함께 선택
df['부상신고자수']>2
df[df['부상신고자수']>2]
df[df['부상신고자수']>2]['발생지시군구']
df[df['부상신고자수']>2][['발생지시군구', '도로형태']]
df.loc[df['부상신고자수']>2, '발생지시군구']
```

```
# Boolean type
```

```
# df[df['부상신고자수']>2]['발생지시군구']
```


Pandas – index 선택

```
# index 선택
df.index
df['발생지시도'] == '경기'
df.index[df['발생지시도'] == '경기']
df.index[df['발생지시도'] == '경기'].tolist()
```

Pandas – select – 조건(slicing) – query

```
# 부상신고자수가 2 이상으로 사고 유무 여부(boolean)를 list로 만들기
injurs = []
for injur in df['부상신고자수']:      # df['부상신고자수']는 iterators
    if injur >=3:
        injurs.append(True)
    else:
        injurs.append(False)

len(injurs)  # df.shape[0]
injurs[0:5]

# boolean list의 값이 참(True)인 row들만 가져오기
pd.Series(injurs)      # pd.DataFrame(injurs)
tf_injurs = pd.Series(injurs)
df[tf_injurs]

# list comprehension을 사용해서
tf_injurs = [injur >= 3 for injur in df['부상신고자수']]
tf_injurs[:5]
```

Pandas – select – 조건(slicing) – query

단순 조건문

```
df[df['부상신고자수'] >= 2]
df[df['부상신고자수'] >= 2]['발생지시도']
df.loc[df['부상신고자수'] >= 2, '발생지시도']
```

복합 조건문

```
(df['부상신고자수'] >= 2) & (df['발생지시도'] == '경기')
df[(df['부상신고자수'] >= 2) & (df['발생지시도'] == '경기')]
df[(df['부상신고자수'] >= 2) | (df['발생지시도'] == '경기')]
```

복합조건문은 isin 사용, 주로 category 자료형에

```
df[(df['발생지시도'] == '서울') | (df['발생지시도'] == '경기')]
df['발생지시도'].isin(['서울', '경기'])
df[df['발생지시도'].isin(['서울', '경기'])]
```

값에 특정문자가 포함되어 있는지 조건으로 활용시

```
df['발생지시도'].str
[i for i in df['발생지시도'].str][:5]
[i for i in df['발생지시도'].str[0]][:5]
df[df['발생지시도'].str.contains('경')] # [i for i in df['발생지시도'].str[1]][:5]
```

Pandas – select – 조건(slicing) – query

query에 의한 선택

```
df.query('부상자수 > 3')[['발생년', '발생년월일시', '주야', '요일']]
```

비율에 의한 sample

```
df.sample(frac=0.2)
```

개수에 의한 sample

```
df.sample(300)            # df.sample(n = 300)
```

Pandas – 모양 바꾸기 – 값 바꾸기

```
# 국영수 성적 데이터에 값 하나로 모든 row의 값 추가
tf = pd.DataFrame({'학생': np.random.choice(range(1, 11), 10, replace=False),
                  '국어': np.random.randint(70, 101, 10),
                  '영어': np.random.randint(60, 101, 10)})
tf['현장학습'] = pd.Series(np.random.choice(['상', '중', '하', 'X'], 10))

tf['추가컬럼'] = 6
tf
tf.set_index('학생', inplace=True)
tf

# index가 달라 값이 치환되지 않음
tf['현장학습'] = pd.DataFrame(np.random.choice(['상', '중', '하', 'X'], 10))
tf
tf.reset_index(inplace=True)
tf['현장학습'] = pd.DataFrame(np.random.choice(['상', '중', '하', 'X'], 10))
tf
```

제 6 장. Pandas 배우기



- 다양한 데이터 읽고 쓰기
- 기본 및 상세 정보 확인
- group by (aggregation)
- sort
- columns와 index
- 데이터 형태 변환
- 조건과 slicing, selecting, query
- 컬럼 변환 및 추가
- 데이터 합치기(merge)
- 결측치 대체, 이상치 및 중복값 처리

Pandas – 모양 바꾸기 – 컬럼 변환 / 추가, 특성공학

```
# 숫자와 문자가 같이 있는 데이터는 문자를 제거 후 숫자만 활용
import numpy as np
sf = pd.DataFrame({'월': range(1,13),
                  '용돈': [i+ '원' for i in (np.random.randint(100, 300,12).astype(str))]},
                  columns= ['월', '용돈'])

sf
sf.용돈.mean()          # Error

sf.용돈.str.replace('원', "").astype(int).mean()
```

Pandas – 모양 바꾸기 – 컬럼 변환 / 추가, 특성공학

```
# 새로운 변수 생성 '발생년월일시'에서 '시'만 추출
df['발생시간'] = df['발생년월일시'].astype(str).str.slice(8,11)

# 년도, 월, 일, 시간 추출시 pandas 기능 이용
pd.to_datetime(df['발생년월일시']).dt.year
pd.to_datetime(df['발생년월일시']).dt.month # dt.month_name()
pd.to_datetime(df['발생년월일시']).dt.day   # dt.day_name()
pd.to_datetime(df['발생년월일시']).dt.hour
df['발생시간'].value_counts()
df['발생시간'].value_counts().sort_index()
df['발생시간'].value_counts().sort_index().plot.bar()

# 시각을 '새벽', '아침', '점심', '저녁', '밤'으로 구분
df['시간대'] = pd.cut(df['발생시간'].astype('int'), [-1, 6, 10, 15, 20, 24], # 구분 개수보다 1개 더
                    labels=['새벽', '아침', '점심', '저녁', '밤'])
df['시간대']
df['시간대'].fillna('밤', inplace=True)
df['시간대'].value_counts().plot.bar()
```


Pandas – 모양 바꾸기 – 컬럼 변환 / 추가, 특성공학

```
df['부상자수'].value_counts()  
df['부상자수'].shift(1)  
df['부상자수'].shift(-1)  
df['부상자수'].diff()  
df['부상자수'].cumsum()
```

컬럼을 삭제하는 방법

```
df.drop(['시간대'], axis=1).head()  
df.loc[:, df.columns.difference(['시간대'])].head()  
'시간대' in df.loc[:, df.columns.difference(['시간대'])].columns  
del df['시간대']
```

행(record)을 제거하는 방법

```
df.drop(2, axis=0).head()  
df.drop([2,4], axis=0).head()  
df.drop(range(0, len(df), 2), axis=0).head() # 짝수행 제거  
df.drop(range(1, len(df), 2), axis=0).head() # 홀수행 제거
```

Pandas – 모양 바꾸기 – melt, pivot

```
# 데이터프레임 모양 바꾸기 - melt, pivot
tf = pd.DataFrame({'A': list('abc'),
                  'B': [1,2,3],
                  'C': [4,5,6]})

tf
# long type으로 분해하기
pd.melt(tf)
pd.melt(tf, id_vars= ['A'], value_vars=['B'])
tf = pd.melt(tf, id_vars= ['A'], value_vars=['B', 'C'])
tf
# wide type으로 조립하기
tf = tf.pivot(index='A', columns='variable', values = 'value')
tf
# 원래대로 변환
tf.reset_index()
```

Pandas – 모양 바꾸기 – 삭제

```
df['발생시간'] = df['발생년월일시'].astype(str).str.slice(8,11)
df.drop(['발생시간'], axis=1).head()
'발생시간' in df.columns
df.drop(['발생시간'], axis=1, inplace=True)
'발생시간' in df.columns

# index 적용
df.set_index('발생년월일시', inplace=True)
df = df.set_index('발생년월일시')
df.index
```

제 6 장. Pandas 배우기

- 다양한 데이터 읽고 쓰기
- 기본 및 상세 정보 확인
- group by (aggregation)
- sort
- columns와 index
- 데이터 형태 변환
- 조건과 slicing, selecting, query
- 컬럼 변환 및 추가
- 데이터 합치기(merge)
- 결측치 대체, 이상치 및 중복값 처리



Pandas – 데이터프레임 생성

```
# 실행결과 모두 보기
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
import pandas as pd

# 가로에 모든 결과 화면 보이기
from IPython.display import display_html
def display_side_by_side(*args):
    """여러 데이터프레임 비교가 쉽게 옆쪽으로 표시한다"""
    html_str=""
    for df in args:
        html_str += df.to_html()
    display_html(html_str.replace('table','table style="display:inline"'), raw=True)

# 예제 데이터프레임 만들기
def make_df(var, obs):
    data = {c: [str(c) + str(i) for i in obs]
            for c in var}
    return pd.DataFrame(data, obs)
```

Pandas – 데이터프레임 생성

```
display_side_by_side(
    make_df('ABCD', [0,1,2,3]), make_df('012', ['A','B']), make_df('ACEGIJ', [1,3,5,7,9])
)
```

	A	B	C	D		0	1	2		A	C	E	G	I	J
0	A0	B0	C0	D0	A	0A	1A	2A	1	A1	C1	E1	G1	I1	J1
1	A1	B1	C1	D1	B	0B	1B	2B	3	A3	C3	E3	G3	I3	J3
2	A2	B2	C2	D2					5	A5	C5	E5	G5	I5	J5
3	A3	B3	C3	D3					7	A7	C7	E7	G7	I7	J7
									9	A9	C9	E9	G9	I9	J9

```
df1 = make_df('ABCD', [0,1,2,3])
df2 = make_df('DEFG', [1,2,3,4])
display_side_by_side(df1, df2, pd.concat([df1, df1]))
```

	A	B	C	D		D	E	F	G		A	B	C	D
0	A0	B0	C0	D0	1	D1	E1	F1	G1	0	A0	B0	C0	D0
1	A1	B1	C1	D1	2	D2	E2	F2	G2	1	A1	B1	C1	D1
2	A2	B2	C2	D2	3	D3	E3	F3	G3	2	A2	B2	C2	D2
3	A3	B3	C3	D3	4	D4	E4	F4	G4	3	A3	B3	C3	D3
										0	A0	B0	C0	D0
										1	A1	B1	C1	D1
										2	A2	B2	C2	D2
										3	A3	B3	C3	D3

Pandas – 데이터프레임 합치기 – concat

실행결과 모두 보기

display_side_by_side(df1, df1, pd.concat([df1, df1])) # 기본은 상하로 합치기(vertical)

	A	B	C	D		A	B	C	D		A	B	C	D
0	A0	B0	C0	D0	0	A0	B0	C0	D0	0	A0	B0	C0	D0
1	A1	B1	C1	D1	1	A1	B1	C1	D1	1	A1	B1	C1	D1
2	A2	B2	C2	D2	2	A2	B2	C2	D2	2	A2	B2	C2	D2
3	A3	B3	C3	D3	3	A3	B3	C3	D3	3	A3	B3	C3	D3
										0	A0	B0	C0	D0
										1	A1	B1	C1	D1
										2	A2	B2	C2	D2
										3	A3	B3	C3	D3

display_side_by_side(df1, df1, pd.concat([df1, df1], axis=1))

axis=1, 좌우로 합치기(horizontal)

	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
0	A0	B0	C0	D0	0	A0	B0	C0	D0	0	A0	B0	C0	D0	0	A0	B0	C0	D0
1	A1	B1	C1	D1	1	A1	B1	C1	D1	1	A1	B1	C1	D1	1	A1	B1	C1	D1
2	A2	B2	C2	D2	2	A2	B2	C2	D2	2	A2	B2	C2	D2	2	A2	B2	C2	D2
3	A3	B3	C3	D3	3	A3	B3	C3	D3	3	A3	B3	C3	D3	3	A3	B3	C3	D3

Pandas – 데이터프레임 합치기 – concat

실행결과 모두 보기

```
display_side_by_side(df1, df2, pd.concat([df1, df2]))
```

기본은 상하로 합치기, 합칠 때 동일한 이름의 컬럼이 없으면 nan

	A	B	C	D		D	E	F	G		A	B	C	D	E	F	G
0	A0	B0	C0	D0	1	D1	E1	F1	G1	0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	2	D2	E2	F2	G2	1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	3	D3	E3	F3	G3	2	A2	B2	C2	D2	NaN	NaN	NaN
3	A3	B3	C3	D3	4	D4	E4	F4	G4	3	A3	B3	C3	D3	NaN	NaN	NaN
										1	NaN	NaN	NaN	D1	E1	F1	G1
										2	NaN	NaN	NaN	D2	E2	F2	G2
										3	NaN	NaN	NaN	D3	E3	F3	G3
										4	NaN	NaN	NaN	D4	E4	F4	G4

```
display_side_by_side(df1, df2,
    pd.concat([df1, df2], ignore_index=True)
) # index를 새롭게 만들기
```

	A	B	C	D	E	F	G
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	NaN	NaN	NaN
3	A3	B3	C3	D3	NaN	NaN	NaN
4	NaN	NaN	NaN	D1	E1	F1	G1
5	NaN	NaN	NaN	D2	E2	F2	G2
6	NaN	NaN	NaN	D3	E3	F3	G3
7	NaN	NaN	NaN	D4	E4	F4	G4

Pandas – 데이터프레임 합치기 – concat

```
display_side_by_side(df1, df2, pd.concat([df1, df2], axis=1)) # 수평으로 합치기
```

```
display_side_by_side(df1, df2, pd.concat([df1, df2], join='inner'))
```

inner join(기본은 outer join), 서로 컬럼이 있는 경우만 즉 inner는 교집합, outer는 합집합

```
display_side_by_side(df1, df2, pd.concat([df1, df2], axis=1, join='inner'))
```

수평으로 서로 인덱스가 경우만

```
display_side_by_side(df1.set_index('D'), df2.set_index('D'),  
                    pd.concat([df1.set_index('D'), df2.set_index('D')], axis=1)
```

) # 수평으로 특정 인덱스를 기준으로

	A	B	C		E	F	G		A	B	C	E	F	G
D				D				D						
D0	A0	B0	C0	D1	E1	F1	G1	D0	A0	B0	C0	NaN	NaN	NaN
D1	A1	B1	C1	D2	E2	F2	G2	D1	A1	B1	C1	E1	F1	G1
D2	A2	B2	C2	D3	E3	F3	G3	D2	A2	B2	C2	E2	F2	G2
D3	A3	B3	C3	D4	E4	F4	G4	D3	A3	B3	C3	E3	F3	G3
								D4	NaN	NaN	NaN	E4	F4	G4

Pandas – 데이터프레임 합치기 – append

```
display_side_by_side(df1, df1, df1.append(df1))
```

	A	B	C	D		A	B	C	D		A	B	C	D
0	A0	B0	C0	D0	0	A0	B0	C0	D0	0	A0	B0	C0	D0
1	A1	B1	C1	D1	1	A1	B1	C1	D1	1	A1	B1	C1	D1
2	A2	B2	C2	D2	2	A2	B2	C2	D2	2	A2	B2	C2	D2
3	A3	B3	C3	D3	3	A3	B3	C3	D3	3	A3	B3	C3	D3
										0	A0	B0	C0	D0
										1	A1	B1	C1	D1
										2	A2	B2	C2	D2
										3	A3	B3	C3	D3

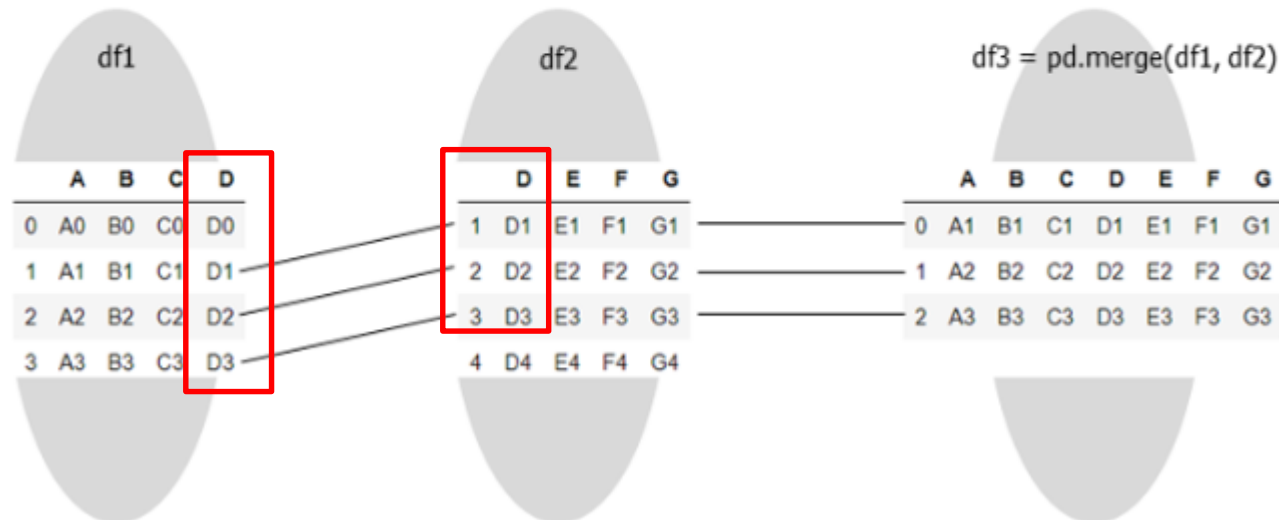
Pandas - 데이터프레임 합치기 - merge

```
df3 = pd.merge(df1, df2)  
df1; df2; df3
```

Merge (1:1)

Merge(df1, df2)는 df1을 기준으로 df2를 merge

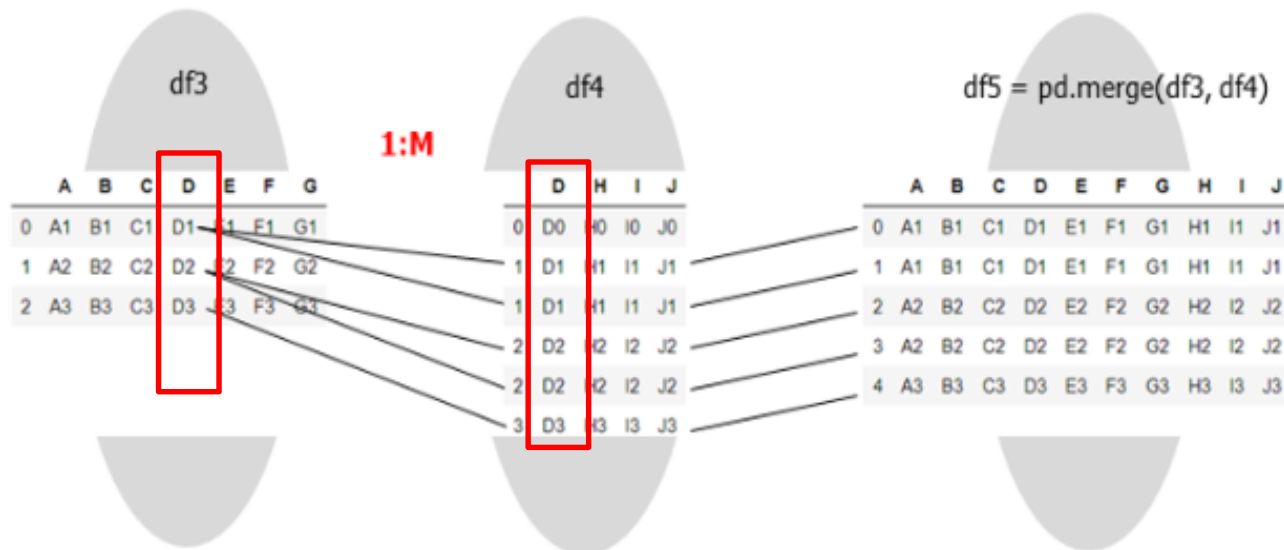
- `pd.merge(df1, df2, how='left', on=['D'])`



Pandas – 데이터프레임 합치기 – merge

```
df4 = make_df('DHIJ', [0,1,1,2,2,3])  
df5 = pd.merge(df3, df4)  
df3; df4; df5
```

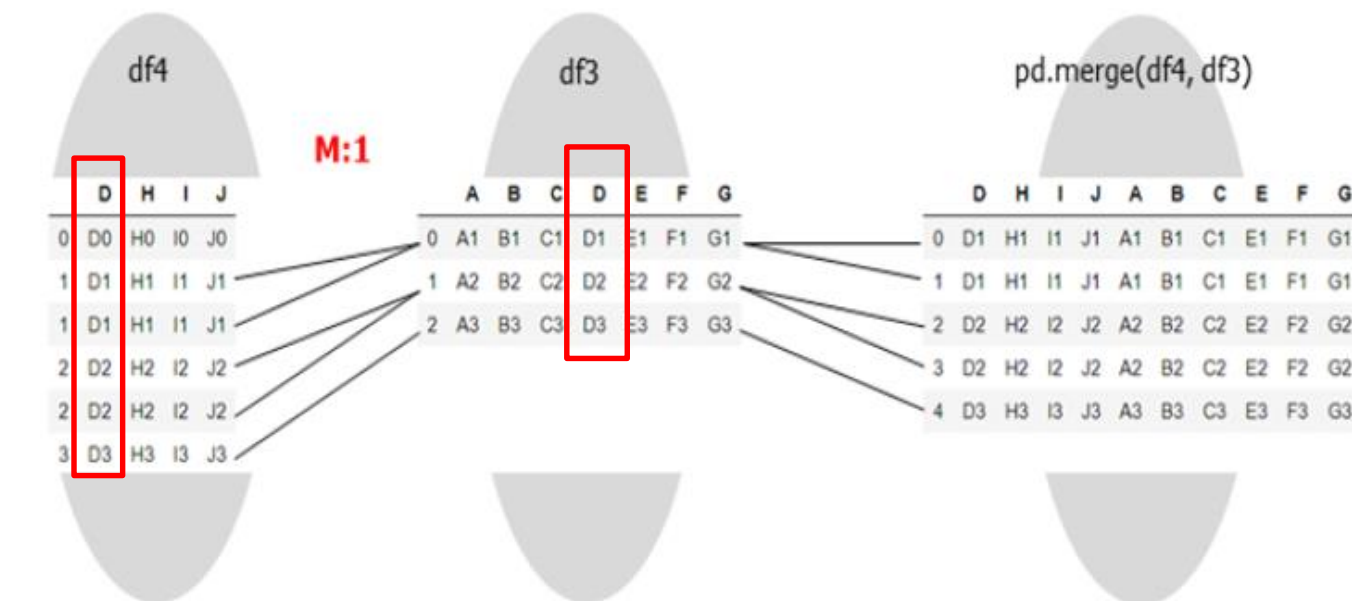
Merge (1:M)



Pandas – 데이터프레임 합치기 – merge

```
df4; df3; pd.merge(df4, df3)  
df4.merge(df3)
```

Merge (M:1)



Pandas – 데이터프레임 합치기 – merge

```
df4; df5; pd.merge(df4, df5)
```

Merge (M:M)



Pandas – 데이터프레임 합치기 – merge

```
df3 = pd.merge(df1, df2)
display_side_by_side(df1, df2, df3)
```

1:1

동일컬럼 'D'를 기준으로

	A	B	C	D		D	E	F	G		A	B	C	D	E	F	G
0	A0	B0	C0	D0	1	D1	E1	F1	G1	0	A1	B1	C1	D1	E1	F1	G1
1	A1	B1	C1	D1	2	D2	E2	F2	G2	1	A2	B2	C2	D2	E2	F2	G2
2	A2	B2	C2	D2	3	D3	E3	F3	G3	2	A3	B3	C3	D3	E3	F3	G3
3	A3	B3	C3	D3	4	D4	E4	F4	G4								

```
df4 = make_df('DHIJ', [0,1,1,2,2,3])
df5 = pd.merge(df3, df4)
display_side_by_side(df3, df4, df5)
```

1:M

동일컬럼 'D'를 기준으로

	A	B	C	D	E	F	G		D	H	I	J		A	B	C	D	E	F	G	H	I	J
0	A1	B1	C1	D1	E1	F1	G1	0	D0	H0	I0	J0	0	A1	B1	C1	D1	E1	F1	G1	H1	I1	J1
1	A2	B2	C2	D2	E2	F2	G2	1	D1	H1	I1	J1	1	A1	B1	C1	D1	E1	F1	G1	H1	I1	J1
2	A3	B3	C3	D3	E3	F3	G3	1	D1	H1	I1	J1	2	A2	B2	C2	D2	E2	F2	G2	H2	I2	J2
								2	D2	H2	I2	J2	3	A2	B2	C2	D2	E2	F2	G2	H2	I2	J2
								2	D2	H2	I2	J2	4	A3	B3	C3	D3	E3	F3	G3	H3	I3	J3
								3	D3	H3	I3	J3											

Pandas – 데이터프레임 합치기 – merge

```
display_side_by_side(df4, df3, pd.merge(df4, df3))
#df4.merge(df3)
```

	D	H	I	J		A	B	C	D	E	F	G		D	H	I	J	A	B	C	E	F	G
0	D0	H0	I0	J0	0	A1	B1	C1	D1	E1	F1	G1	0	D1	H1	I1	J1	A1	B1	C1	E1	F1	G1
1	D1	H1	I1	J1	1	A2	B2	C2	D2	E2	F2	G2	1	D1	H1	I1	J1	A1	B1	C1	E1	F1	G1
1	D1	H1	I1	J1	2	A3	B3	C3	D3	E3	F3	G3	2	D2	H2	I2	J2	A2	B2	C2	E2	F2	G2
2	D2	H2	I2	J2									3	D2	H2	I2	J2	A2	B2	C2	E2	F2	G2
2	D2	H2	I2	J2									4	D3	H3	I3	J3	A3	B3	C3	E3	F3	G3
3	D3	H3	I3	J3																			

M:1

동일컬럼 'D'를 기준으로

```
display_side_by_side(df4, df5, pd.merge(df4, df5))
```

	D	H	I	J		A	B	C	D	E	F	G	H	I	J		D	H	I	J	A	B	C	E	F	G
0	D0	H0	I0	J0	0	A1	B1	C1	D1	E1	F1	G1	H1	I1	J1	0	D1	H1	I1	J1	A1	B1	C1	E1	F1	G1
1	D1	H1	I1	J1	1	A1	B1	C1	D1	E1	F1	G1	H1	I1	J1	1	D1	H1	I1	J1	A1	B1	C1	E1	F1	G1
1	D1	H1	I1	J1	2	A2	B2	C2	D2	E2	F2	G2	H2	I2	J2	2	D1	H1	I1	J1	A1	B1	C1	E1	F1	G1
2	D2	H2	I2	J2	3	A2	B2	C2	D2	E2	F2	G2	H2	I2	J2	3	D1	H1	I1	J1	A1	B1	C1	E1	F1	G1
2	D2	H2	I2	J2	4	A3	B3	C3	D3	E3	F3	G3	H3	I3	J3	4	D2	H2	I2	J2	A2	B2	C2	E2	F2	G2
3	D3	H3	I3	J3												5	D2	H2	I2	J2	A2	B2	C2	E2	F2	G2
																6	D2	H2	I2	J2	A2	B2	C2	E2	F2	G2
																7	D2	H2	I2	J2	A2	B2	C2	E2	F2	G2
																8	D3	H3	I3	J3	A3	B3	C3	E3	F3	G3

M:M

동일컬럼 'D', 'H'를 기준으로

Pandas – 데이터프레임 합치기 – merge

```
display_side_by_side(df4, df5, pd.merge(df4, df5, how = 'left'))
```

왼쪽 파일을 기준으로(how='left') merge하기 때문에 왼쪽 파일의 D컬럼의 'D0'원소값을 포함한다

	D	H	I	J		A	B	C	D	E	F	G	H	I	J		D	H	I	J	A	B	C	E	F	G
0	D0	H0	I0	J0	0	A1	B1	C1	D1	E1	F1	G1	H1	I1	J1	0	D0	H0	I0	J0	NaN	NaN	NaN	NaN	NaN	NaN
1	D1	H1	I1	J1	1	A1	B1	C1	D1	E1	F1	G1	H1	I1	J1	1	D1	H1	I1	J1	A1	B1	C1	E1	F1	G1
1	D1	H1	I1	J1	2	A2	B2	C2	D2	E2	F2	G2	H2	I2	J2	2	D1	H1	I1	J1	A1	B1	C1	E1	F1	G1
2	D2	H2	I2	J2	3	A2	B2	C2	D2	E2	F2	G2	H2	I2	J2	3	D1	H1	I1	J1	A1	B1	C1	E1	F1	G1
2	D2	H2	I2	J2	4	A3	B3	C3	D3	E3	F3	G3	H3	I3	J3	4	D1	H1	I1	J1	A1	B1	C1	E1	F1	G1
3	D3	H3	I3	J3												5	D2	H2	I2	J2	A2	B2	C2	E2	F2	G2
																6	D2	H2	I2	J2	A2	B2	C2	E2	F2	G2
																7	D2	H2	I2	J2	A2	B2	C2	E2	F2	G2
																8	D2	H2	I2	J2	A2	B2	C2	E2	F2	G2
																9	D3	H3	I3	J3	A3	B3	C3	E3	F3	G3

Pandas – 데이터프레임 합치기 – merge

```
df6= df5[['A', 'B', 'D', 'H', 'I']]
```

```
display_side_by_side(df4, df6, pd.merge(df4, df6, on =['D', 'H']))
```

merge할 경우 동일한 컬럼명인 경우 앞에 있는 컬럼에_x, 뒤의 컬럼에_y

	D	H	I	J		A	B	D	H	I		D	H	I_x	J	A	B	I_y
0	D0	H0	I0	J0	0	A1	B1	D1	H1	I1	0	D1	H1	I1	J1	A1	B1	I1
1	D1	H1	I1	J1	1	A1	B1	D1	H1	I1	1	D1	H1	I1	J1	A1	B1	I1
1	D1	H1	I1	J1	2	A2	B2	D2	H2	I2	2	D1	H1	I1	J1	A1	B1	I1
2	D2	H2	I2	J2	3	A2	B2	D2	H2	I2	3	D1	H1	I1	J1	A1	B1	I1
2	D2	H2	I2	J2	4	A3	B3	D3	H3	I3	4	D2	H2	I2	J2	A2	B2	I2
3	D3	H3	I3	J3							5	D2	H2	I2	J2	A2	B2	I2
											6	D2	H2	I2	J2	A2	B2	I2
											7	D2	H2	I2	J2	A2	B2	I2
											8	D3	H3	I3	J3	A3	B3	I3

Pandas – 데이터프레임 합치기 – merge

```
df6= df5[['A', 'B', 'D', 'H', 'I']]
display_side_by_side(df4, df6, pd.merge(df4, df6, on =['D', 'H'], suffixes=('_left', '_right')))
```

	D	H	I	J		A	B	D	H	I		D	H	I_left	J	A	B	I_right
0	D0	H0	I0	J0	0	A1	B1	D1	H1	I1	0	D1	H1	I1	J1	A1	B1	I1
1	D1	H1	I1	J1	1	A1	B1	D1	H1	I1	1	D1	H1	I1	J1	A1	B1	I1
1	D1	H1	I1	J1	2	A2	B2	D2	H2	I2	2	D1	H1	I1	J1	A1	B1	I1
2	D2	H2	I2	J2	3	A2	B2	D2	H2	I2	3	D1	H1	I1	J1	A1	B1	I1
2	D2	H2	I2	J2	4	A3	B3	D3	H3	I3	4	D2	H2	I2	J2	A2	B2	I2
3	D3	H3	I3	J3							5	D2	H2	I2	J2	A2	B2	I2
											6	D2	H2	I2	J2	A2	B2	I2
											7	D2	H2	I2	J2	A2	B2	I2
											8	D3	H3	I3	J3	A3	B3	I3

Pandas – 데이터프레임 합치기 – merge

실행결과 모두 보기

display_side_by_side(df1, df6, pd.merge(df1, df6, on=['D']))# 'D'를 기준으로 서로 합치기

	A	B	C	D		A	B	D	H	I		A_x	B_x	C	D	A_y	B_y	H	I
0	A0	B0	C0	D0	0	A1	B1	D1	H1	I1	0	A1	B1	C1	D1	A1	B1	H1	I1
1	A1	B1	C1	D1	1	A1	B1	D1	H1	I1	1	A1	B1	C1	D1	A1	B1	H1	I1
2	A2	B2	C2	D2	2	A2	B2	D2	H2	I2	2	A2	B2	C2	D2	A2	B2	H2	I2
3	A3	B3	C3	D3	3	A2	B2	D2	H2	I2	3	A2	B2	C2	D2	A2	B2	H2	I2
					4	A3	B3	D3	H3	I3	4	A3	B3	C3	D3	A3	B3	H3	I3

display_side_by_side(df1, df6, pd.merge(df1, df6, on=['A'])) # 'A'를 기준으로 서로 합치기

	A	B	C	D		A	B	D	H	I		A	B_x	C	D_x	B_y	D_y	H	I
0	A0	B0	C0	D0	0	A1	B1	D1	H1	I1	0	A1	B1	C1	D1	B1	D1	H1	I1
1	A1	B1	C1	D1	1	A1	B1	D1	H1	I1	1	A1	B1	C1	D1	B1	D1	H1	I1
2	A2	B2	C2	D2	2	A2	B2	D2	H2	I2	2	A2	B2	C2	D2	B2	D2	H2	I2
3	A3	B3	C3	D3	3	A2	B2	D2	H2	I2	3	A2	B2	C2	D2	B2	D2	H2	I2
					4	A3	B3	D3	H3	I3	4	A3	B3	C3	D3	B3	D3	H3	I3

Pandas – 데이터프레임 합치기 – merge

실행결과 모두 보기

```
display_side_by_side(df1, df6, pd.merge(df1, df6, on='A', suffixes=('_left', '_right')))
```

	A	B	C	D		A	B	D	H	I		A	B_left	C	D_left	B_right	D_right	H	I
0	A0	B0	C0	D0	0	A1	B1	D1	H1	I1	0	A1	B1	C1	D1	B1	D1	H1	I1
1	A1	B1	C1	D1	1	A1	B1	D1	H1	I1	1	A1	B1	C1	D1	B1	D1	H1	I1
2	A2	B2	C2	D2	2	A2	B2	D2	H2	I2	2	A2	B2	C2	D2	B2	D2	H2	I2
3	A3	B3	C3	D3	3	A2	B2	D2	H2	I2	3	A2	B2	C2	D2	B2	D2	H2	I2
					4	A3	B3	D3	H3	I3	4	A3	B3	C3	D3	B3	D3	H3	I3

```
df6['J'] = df1['A']; df7 = df6[['H', 'J']]
```

```
display_side_by_side(df1, df7, pd.merge(df1, df7, left_on='A', right_on='J'))
```

컬럼명이 틀린 경우에도 서로 이름만 다를 경우

	A	B	C	D		H	J		A	B	C	D	H	J
0	A0	B0	C0	D0	0	H1	A0	0	A0	B0	C0	D0	H1	A0
1	A1	B1	C1	D1	1	H1	A1	1	A1	B1	C1	D1	H1	A1
2	A2	B2	C2	D2	2	H2	A2	2	A2	B2	C2	D2	H2	A2
3	A3	B3	C3	D3	3	H2	A3	3	A3	B3	C3	D3	H2	A3
					4	H3	NaN							

제 6 장. Pandas 배우기



- 다양한 데이터 읽고 쓰기
- 기본 및 상세 정보 확인
- group by (aggregation)
- sort
- columns와 index
- 데이터 형태 변환
- 조건과 slicing, selecting, query
- 컬럼 변환 및 추가
- 데이터 합치기(merge)
- 결측치 대체, 이상치 및 중복값 처리

Pandas – 결측치 대체

```
df = pd.DataFrame(np.random.randn(5, 3),
                  index=list('acefh'),
                  columns=["one", "two", "three"])

df['four'] = 'bar'
df['five'] = df['one'] > 0
df2 = df.reindex(list('abcdefgh'))
df2

# 무작위로 5개의 nan 발생 기록
for i in range(5):
    (i, j) = np.random.randint(8), np.random.randint(5)
    df2.iloc[i, j] = np.nan
df2

# isnull, notnull --> sum하면 개수를 알 수 있음
df2.isnull().sum()
df2['one'].value_counts(normalize=True)
df2['one'].value_counts(dropna=False)

# df2.isna().sum()
```

Pandas - 결측치 대체

dropna, any는 어느 하나라도 있으면, all은 모두 NaN인 경우

```
df2.shape
```

```
df2.dropna(how='any').shape
```

```
df2.dropna(how='all').shape
```

```
df2.dropna(subset=['one', 'five'], how='all').shape
```

```
df2.dropna(subset=['one', 'five'], how='any').shape
```

결측치 대체 (missing value imputation), NaN을 다른 값으로 치환

```
df2.fillna('missing')
```

```
df2.fillna(0)
```

```
df2['one'].fillna(df2['one'].mean())
```


Pandas - 중복 값 처리

```
# 중복 확인 duplicates
np.random.seed(321)
df = pd.DataFrame({'동전': np.random.choice(['앞', '뒤'], 20),
                  '주사위': np.random.choice(range(1,7), 20)})

df
df.shape

# 중복항목들을 확인
df.duplicated()
df[df.duplicated()]
df['동전'].duplicated()

# 중복개수 확인
df.duplicated().sum()
df['동전'].duplicated().sum()
df.duplicated(subset=['동전', '주사위']).sum()

# 중복 값 처리
df.duplicated(keep='first')
df.duplicated(keep='last')
```

Pandas - 중복 값 처리

```
# Boolean 조건식으로 찾기  
df[df.duplicated(keep='first')]  
df[df.duplicated(keep='last')]
```

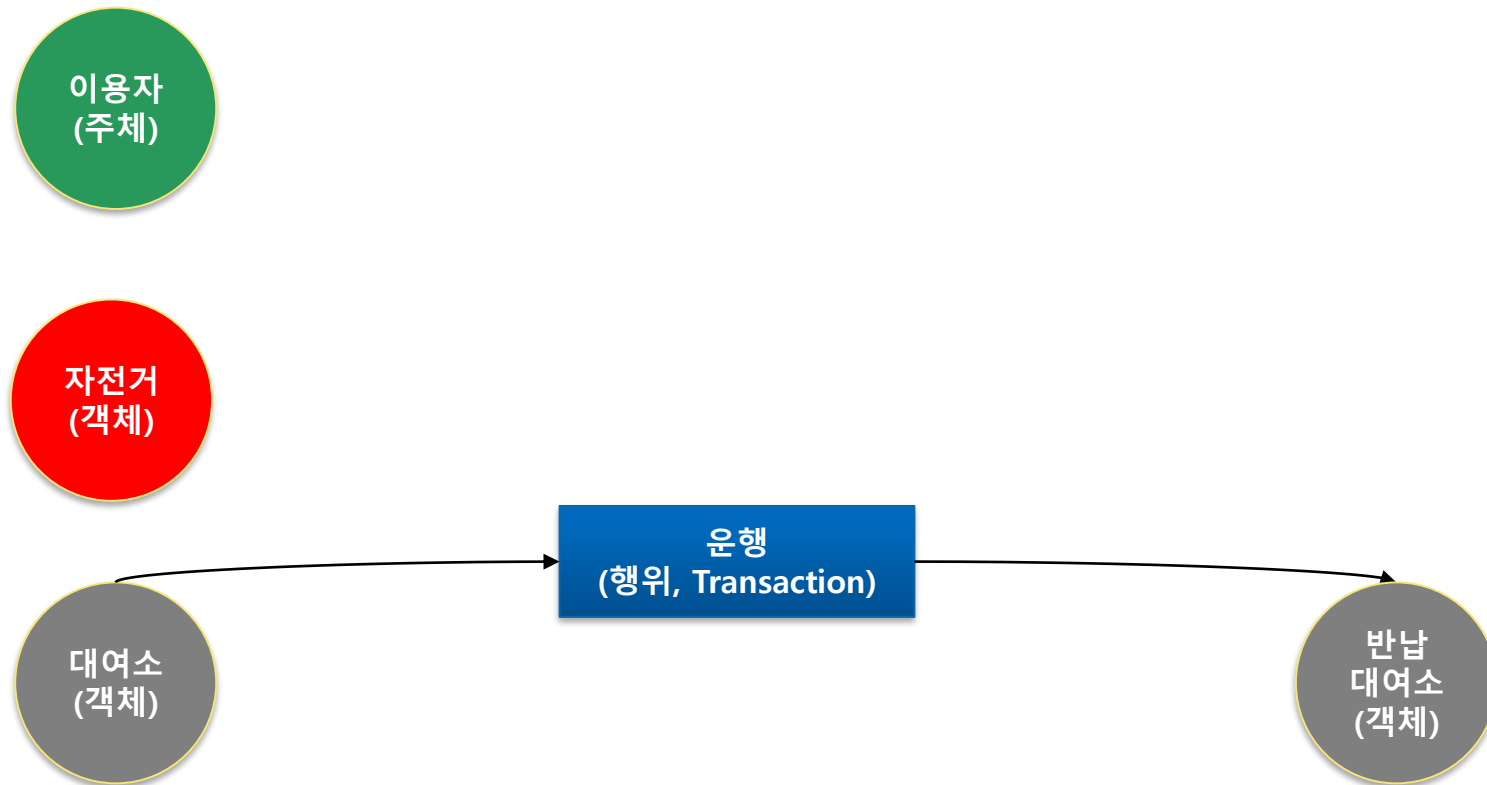
```
# loc로 찾기  
df.loc[df.duplicated(keep='first'), :]  
df.loc[df.duplicated(keep='last'), :]
```

```
# 중복 중복 데이터 제거  
df[~df.duplicated()].shape  
df.drop_duplicates(keep='first').shape  
df.drop_duplicates(subset=['동전', '주사위'], keep='first').shape
```



Appendix. 데이터와 프로세스 이해

- Entity의 작동(process)으로 인해 데이터(data)가 발생
- 사람은 생존하는 동안 권리의 행사와 의무의 이행은 신의에 좇아 성실히 하여야 한다.



Appendix. 데이터 생성 이해

은행
(행위, Transaction)

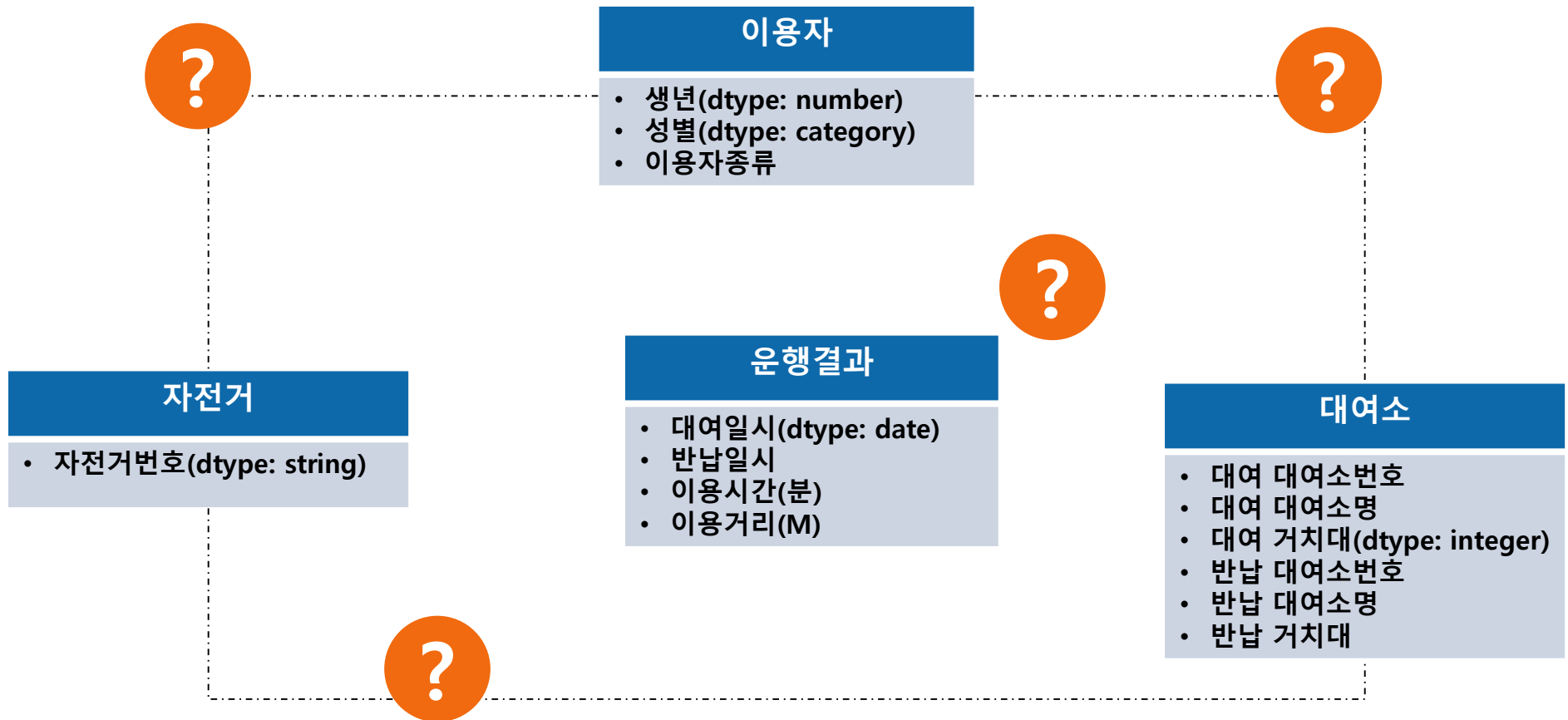
자전거
(객체)

대여소
(객체)

이용자
(주체)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	자전거번호	대여일시	대여 대여소번호	대여 대여소명	대여거치대	반납일시	반납대여소번호	반납대여소명	반납거치대	이용시간(분)	이용거리(M)	생년	성별	이용자종류	대여대여소ID	반납대여소ID
2	SPB-44695	2022-12-01 0000	1933	개봉푸르지오아파트 상가	0	2022-12-01 0000	1933	개봉푸르지오아파트 상가	0	0	0	1994	M	내국인	ST-678	ST-678
3	SPB-31562	2022-12-01 0000	3007	MBC 앞	0	2022-12-01 0000	3007	MBC 앞	0	0	111.2	1995	F	내국인	ST-2165	ST-2165
4	SPB-56324	2022-12-01 0001	4468	가락1동주민센터	0	2022-12-01 0001	4468	가락1동주민센터	0	0	0	1994	M	내국인	ST-2583	ST-2583
5	SPB-30175	2022-12-01 0003	652	답십리 래미안엘파인아파트 입구	0	2022-12-01 0003	652	답십리 래미안엘파인아파트 입	0	0	0	1981	M	내국인	ST-1447	ST-1447
6	SPB-37639	2022-12-01 0000	1047	강동 한신휴플러스	0	2022-12-01 0003	1075	천동초교 삼거리	0	3	492.34	1989	M	내국인	ST-1369	ST-1836
7	SPB-39909	2022-12-01 0003	4508	유원목동아파트 103동 앞	0	2022-12-01 0003	4508	유원목동아파트 103동 앞	0	0	0	1985	M	내국인	ST-2760	ST-2760
8	SPB-31850	2022-12-01 0000	646	장한평역 1번출구	0	2022-12-01 0004	668	서울죽산농협(장안지점)	0	3	911.71	1992	WN	내국인	ST-1298	ST-2268
9	SPB-36226	2022-12-01 0000	1175	대한항공 인력개발센터	0	2022-12-01 0004	1118	종미역 3번출구뒤(동춘두산위	0	4	660.88	1970	F	내국인	ST-1512	ST-516
10	SPB-53863	2022-12-01 0001	1210	롯데월드타워(잠실역2번출구 쪽)	0	2022-12-01 0004	2608	송파구청	0	3	613.24	2000	F	내국인	ST-891	ST-1681
11	SPB-30127	2022-12-01 0001	203	국회의사당역 3번출구 옆	0	2022-12-01 0004	201	진미파라곤 앞	0	2	469.46	1967	WN	내국인	ST-51	ST-46
12	SPB-55046	2022-12-01 0000	1344	아리랑시네센터 앞	0	2022-12-01 0004	1386	동선고가차도 초입	0	4	913.6	1959	M	내국인	ST-1208	ST-2227
13	SPB-50449	2022-12-01 0000	498	연남동주민센터 앞	0	2022-12-01 0004	113	홍대입구역 2번출구 앞	0	4	905.64	1984	M	내국인	ST-2157	ST-18
14	SPB-52881	2022-12-01 0000	5306	동신아파트 후문 옆	0	2022-12-01 0004	2915	서울과학기술대학교(여학교육)	0	4	791.01	1999	M	내국인	ST-3060	ST-2081
15	SPB-50627	2022-12-01 0000	1351	안암2교 옆	0	2022-12-01 0004	3416	동묘앞역 6번출구	0	4	1249.23	1997	F	내국인	ST-1215	ST-1813
16	SPB-61436	2022-12-01 0000	191	서우빌딩	0	2022-12-01 0005	4255	래미안 루센티아 아파트 108동	0	4	1390	1993	M	내국인	ST-347	ST-2897
17	SPB-61011	2022-12-01 0003	2524	반포소풍타운 8동 앞	0	2022-12-01 0005	2523	뉴코아 킴스클럽 앞	0	1	0	2000	M	내국인	ST-1909	ST-1908
18	SPB-59271	2022-12-01 0000	5064	양천향교역8번출구	0	2022-12-01 0005	3783	마곡근로여울림아파트	0	4	670	WN	M	내국인	ST-2943	ST-2490
19	SPB-59269	2022-12-01 0005	1297	석촌호수교차로(동호 팔각정 앞)	0	2022-12-01 0005	1297	석촌호수교차로(동호 팔각정 앞)	0	0	0	1981	M	내국인	ST-1586	ST-1586
20	SPB-43585	2022-12-01 0001	1735	도봉역 1,2번 출구사이 건너편	0	2022-12-01 0005	1727	서울도봉초등학교인근	0	3	495.4	1978	M	내국인	ST-1467	ST-1297
21	SPB-65898	2022-12-01 0002	1531	미아사거리 1번 출구	0	2022-12-01 0005	1560	도봉세무서 앞	0	3	660	1997	WN	내국인	ST-1105	ST-2132
22	SPB-32179	2022-12-01 0000	3007	MBC 앞	0	2022-12-01 0005	4238	상암한샘빌딩앞	0	4	695.74	1995	F	내국인	ST-2165	ST-2981

Appendix. 데이터 모델링



Appendix. Power BI로-데이터 모델 이해



샘플 데이터를 사용하는 두 가지 방법



온라인 자습서 사용

정확하고 세부적인 보고서를 단계별로 만드는 방법을 알아봅니다.

[자습서 시작](#)



직접 실험

시각적 개체를 직접 만들려면 예제 데이터를 로드합니다.

[예제 데이터 로드](#)

자습서

Power BI Desktop: Excel에서 보고서로

Power BI Desktop: 보고할 차원 모델

Power BI 서비스: Excel에서 보고서로

Power BI 샘플 살펴보기

분해 트리 만들기

온라인 과정: 페이지를 매긴 보고서

페이지를 매긴 보고서 만들기 및 업로드

