데이터시각화이해와실습

Lecture 12. 환경 관련 데이터 분석

동덕여자대학교
데이터사이언스 전공
권 범

목차

01. 분석 대상 데이터 수집

02. 데이터 확인하기
03. 데이터 병합
04. 데이터 분석 및 시각화

01. 분석 대상 데이터 수집

❖ 시작하기 전에 (1/2)

- 최근 몇 년간 중국의 산업화, 도시화의 가속으로 인해 대기오염이 악화되었음
- 오염된 대기는 겨울의 찬 기류와 섞이면서 뿌옇게 보이게 되는데, 이를 미세먼지라고 함
- 미세먼지는 교통, 농업, 공업, 건강 등 여러 측면에서 부정적인 영향을 미치고 있으며, 국민들의 일상생활과 신체 건강에 심각한 영향을 미치고 있음

미세먼지는 시급히 해결해야 할 사회문제임

01. 분석 대상 데이터 수집

❖ 시작하기 전에 (2/2)

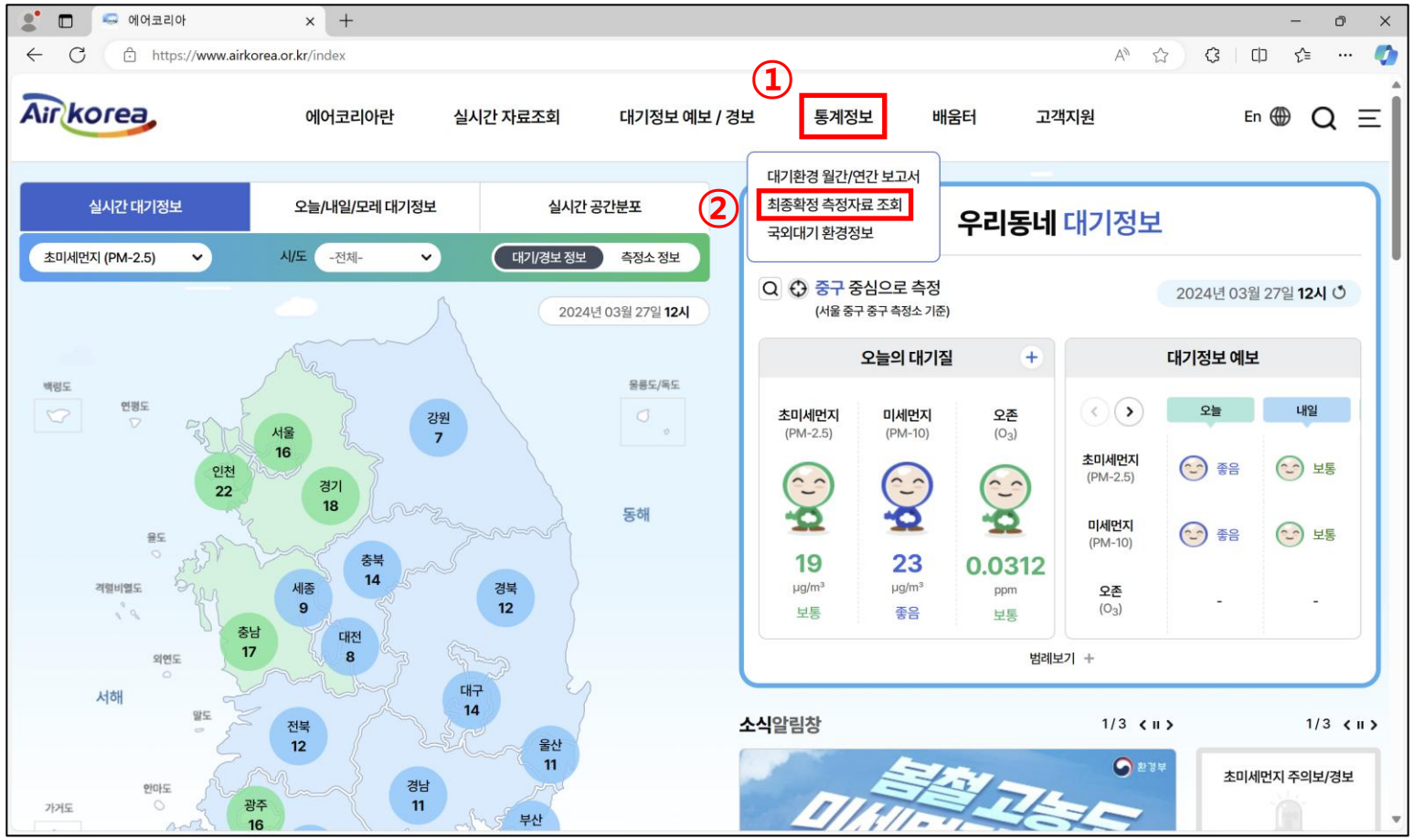
- 이번 수업에서는 미세먼지를 분석 대상으로 정하고,
  관련 데이터를 수집 및 가공하여 미세먼지에 대한 변화 추이를 파악해 보자
- 그리고 미세먼지 농도에 영향을 미치는 주요 변수가 무엇인지 분석해 보자

미세먼지 데이터 분석을 위해,
에어코리아와 기상청에서 미세먼지에 관한 데이터를 수집해 보자

01. 분석 대상 데이터 수집

❖ ① 에어코리아에서 미세먼지 데이터 수집하기 (1/7)

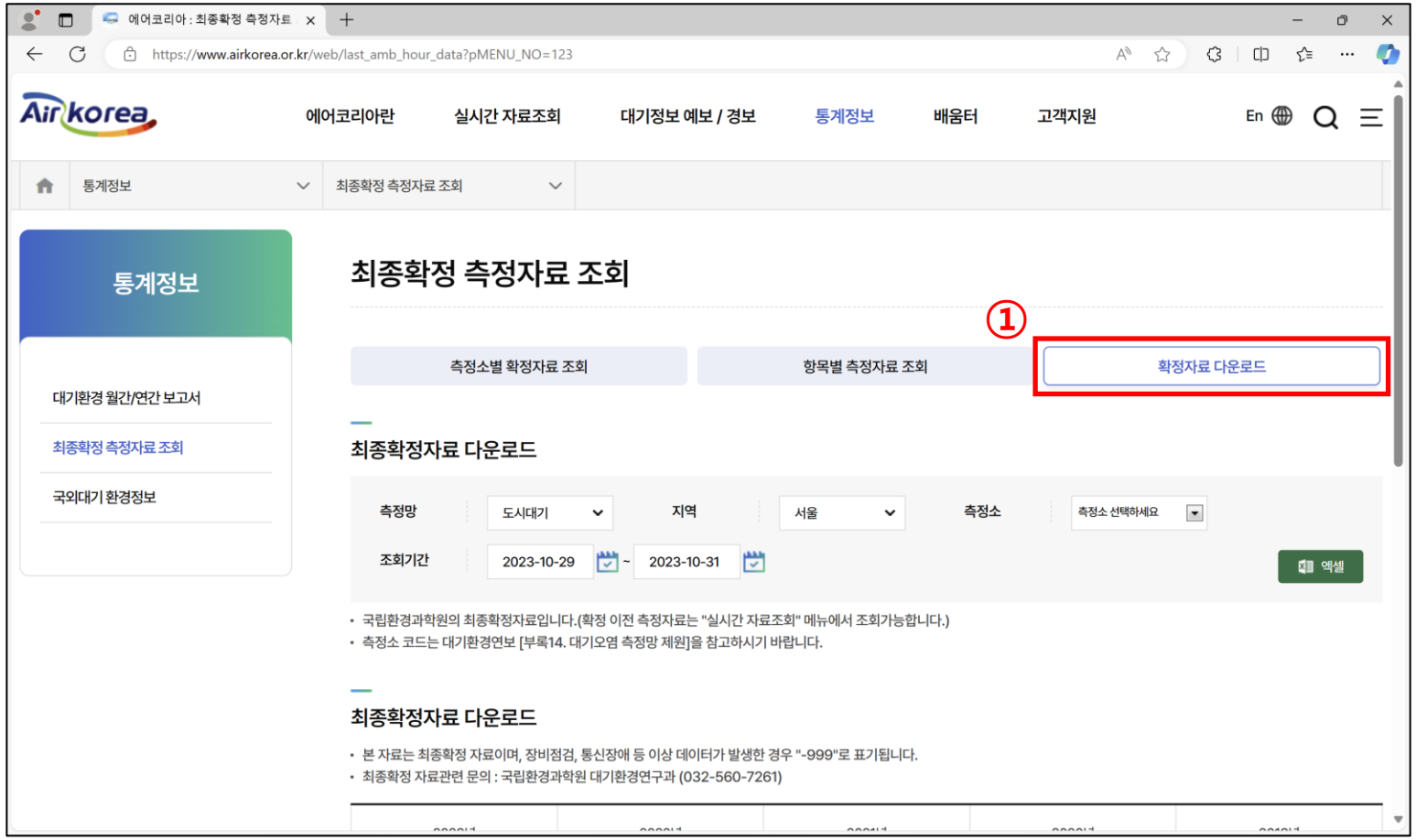
- 에어코리아(https://www.airkorea.or.kr/index)에 접속한 다음,
  메뉴에서 [통계정보] → [최종확정 측정자료 조회]를 클릭



01. 분석 대상 데이터 수집

❖ ① 에어코리아에서 미세먼지 데이터 수집하기 (2/7)

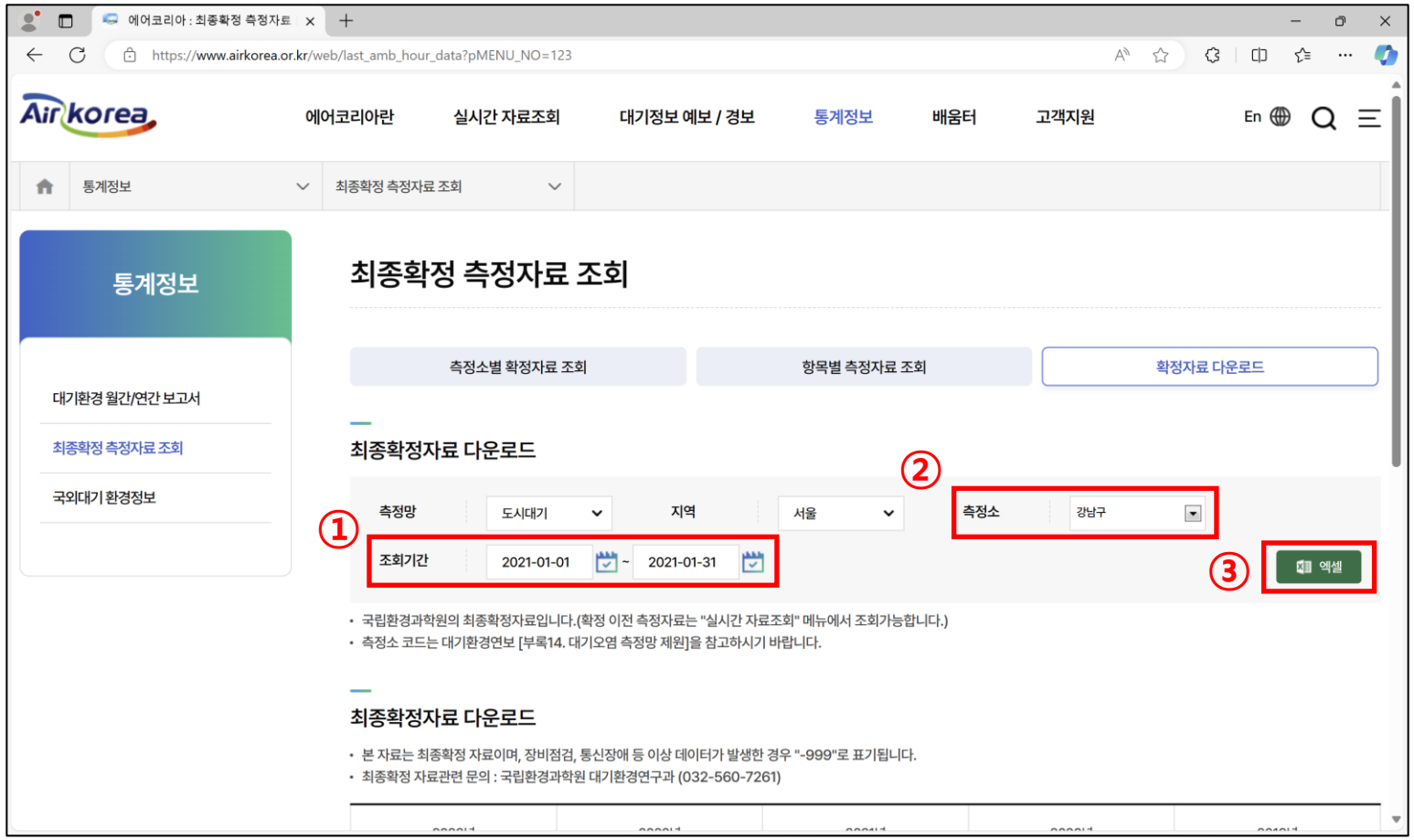
- [확정자료 다운로드] 버튼 클릭



01. 분석 대상 데이터 수집

❖ ① 에어코리아에서 미세먼지 데이터 수집하기 (3/7)

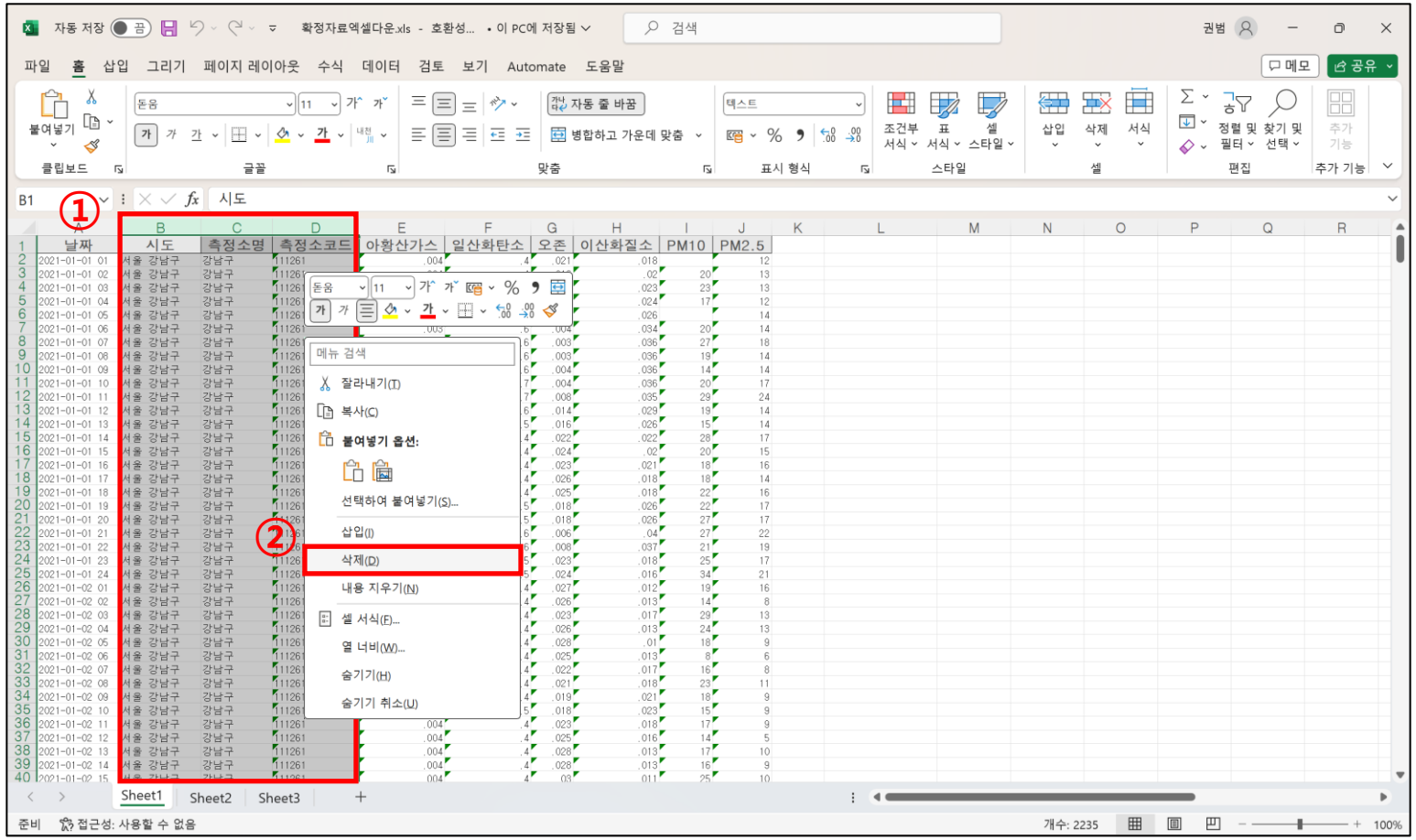
- 아래와 같이 [조회기간]과 [측정소]를 지정한 다음,
  [엑셀] 버튼을 클릭하여 파일을 다운로드하자



01. 분석 대상 데이터 수집

❖ ① 에어코리아에서 미세먼지 데이터 수집하기 (4/7)

- 다운로드한 엑셀을 실행하여 파일을 열고, 데이터 분석에 필요 없는 컬럼(B~D)을 삭제하자



01. 분석 대상 데이터 수집

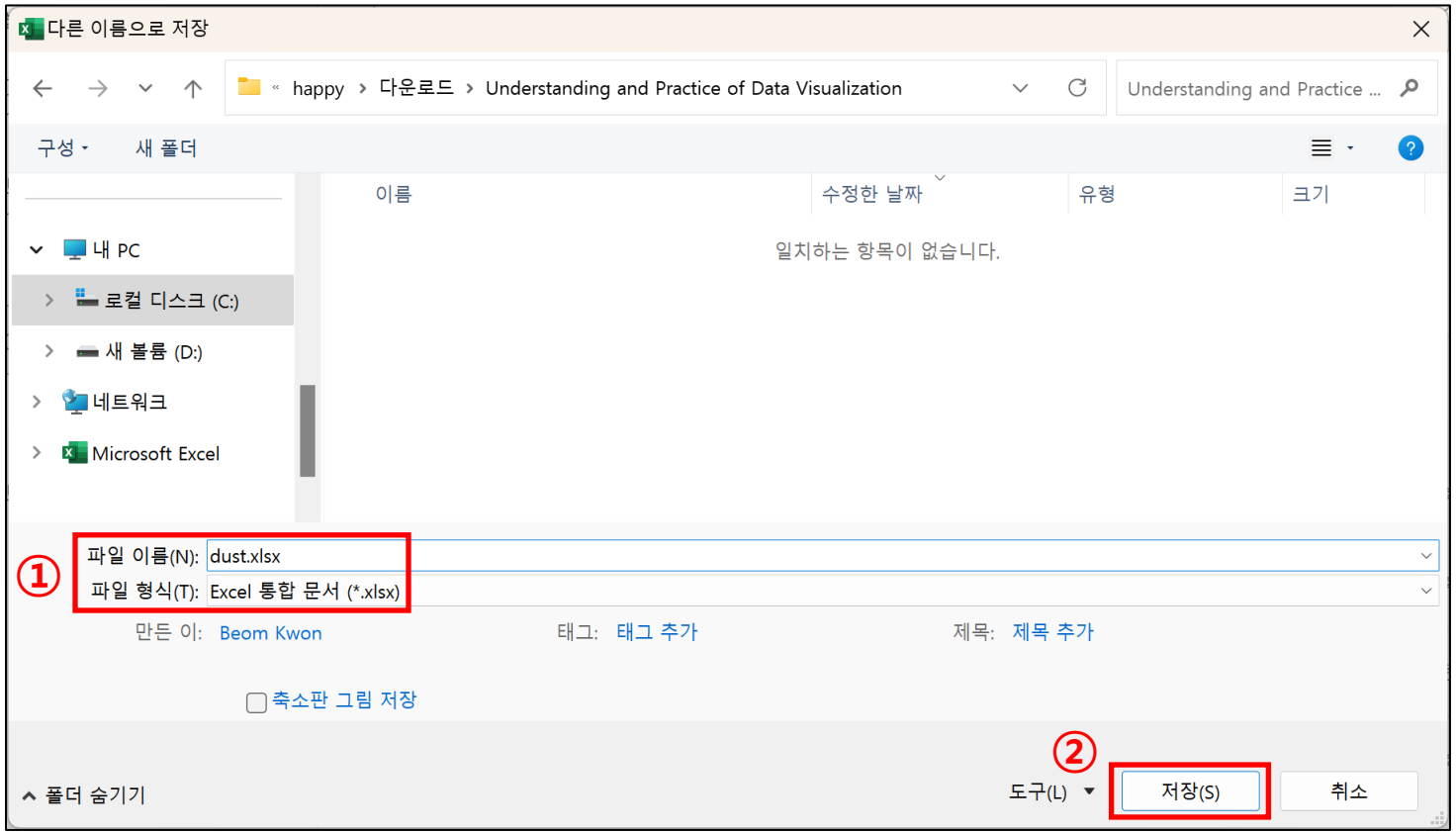
❖ ① 에어코리아에서 미세먼지 데이터 수집하기 (5/7)

- 다운로드한 파일의 형식은 'Microsoft Excel 97-2023 Worksheet.xls(*.xls)'임
- 이 형식의 파일은 코랩에 업로드하여 읽으면 오류가 발생하기 때문에 파일 형식을 'Excel 통합 문서(*.xlsx)'로 변경해야 함

01. 분석 대상 데이터 수집

❖ ① 에어코리아에서 미세먼지 데이터 수집하기 (6/7)

- 엑셀 메뉴에서 [파일] → [다른 이름으로 저장] → [찾아보기]를 클릭
- 대화상자가 표시되면 [파일 형식]을 'Excel 통합 문서(*.xlsx)'로 지정한 다음, [파일 이름]을 'dust'로 변경하고, [저장] 버튼을 클릭



01. 분석 대상 데이터 수집

❖ ① 에어코리아에서 미세먼지 데이터 수집하기 (7/7)

- ‘dust.xlsx’ 파일에 저장된 각 열에 대해서 살펴보자

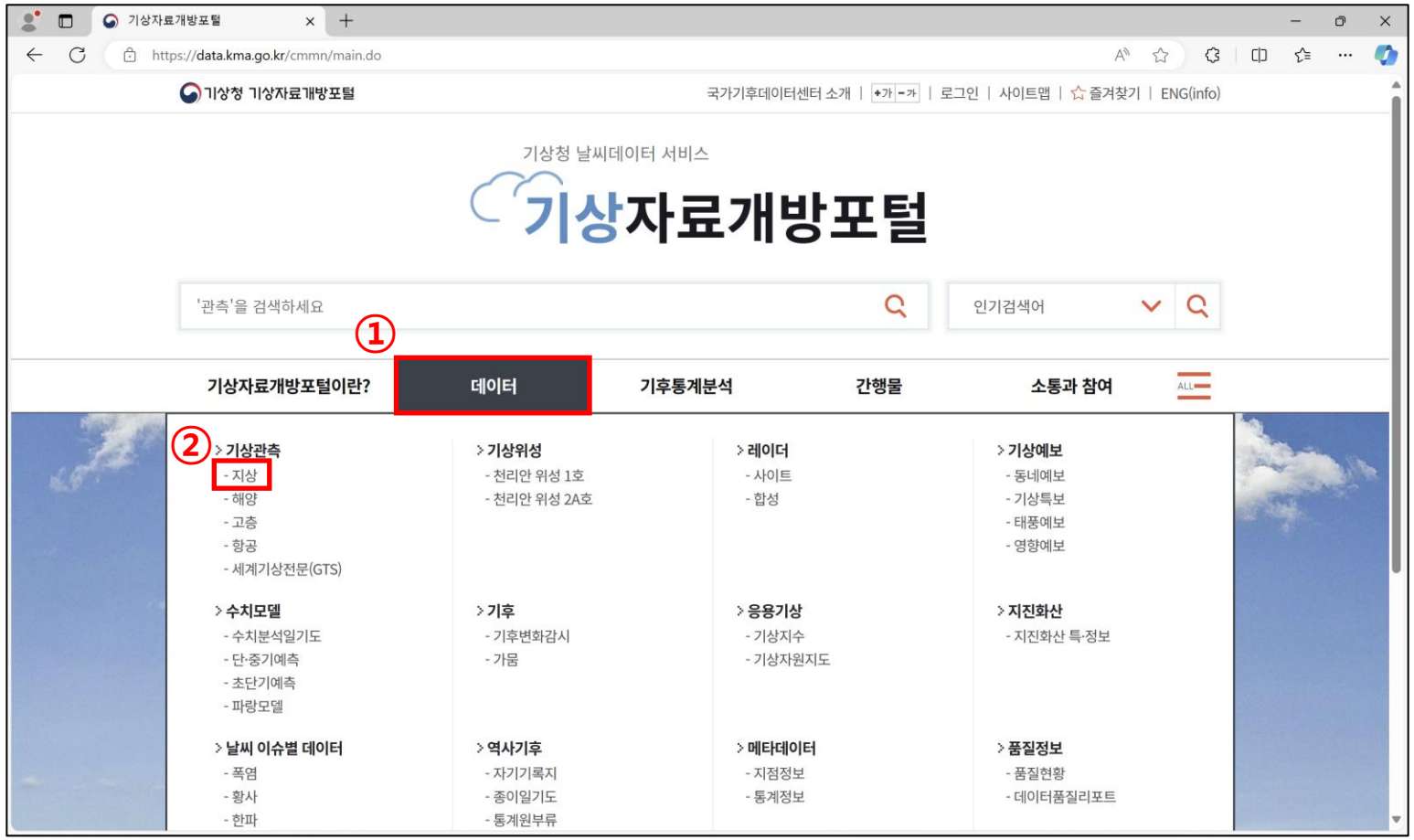
변수명	변수 설명	단위
아황산가스(SO2)	대기오염물질, 아황산가스의 공기 중 농도	ppm
일산화탄소(CO)	대기오염물질, 일산화탄소의 공기 중 농도	ppm
오존(O3)	대기오염물질, 오존의 공기 중 농도	ppm
이산화질소(NO2)	대기오염물질, 이산화질소의 공기 중 농도	ppm
PM10	1000분의 10mm보다 작은 먼지의 공기 중 농도 (미세먼지)	microgram/cubicmeter
PM2.5	1000분의 2.5mm보다 작은 먼지의 공기 중 농도 (초미세먼지)	microgram/cubicmeter

[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

01. 분석 대상 데이터 수집

❖ ② 기상청 사이트에서 날씨 데이터 수집하기 (1/6)

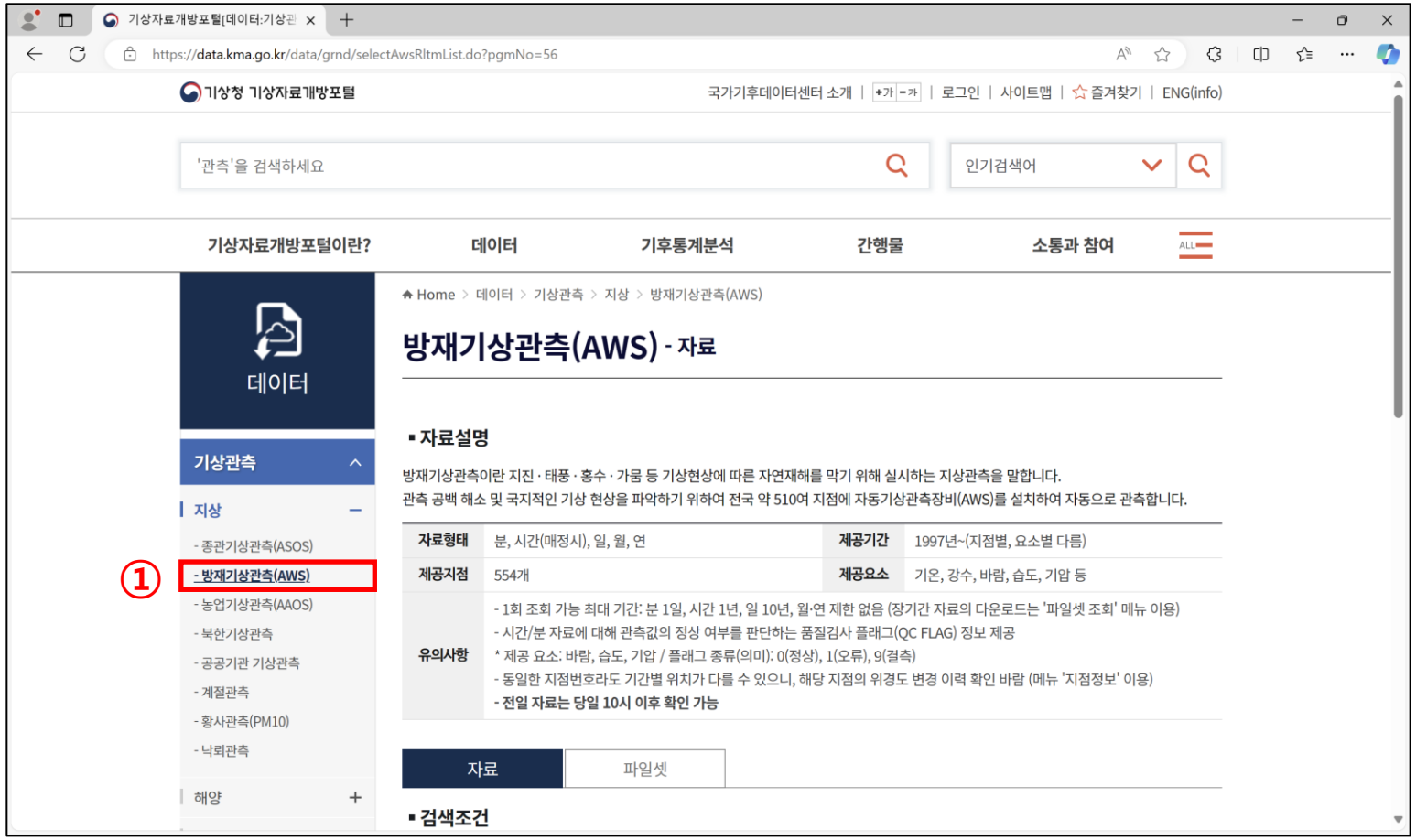
- 기상청(https://data.kma.go.kr/cmmn/main.do)에 접속한 다음, 메뉴에서 [데이터] 버튼을 클릭
- 그다음 왼쪽 메뉴에서 [기상관측] → [지상] 버튼을 클릭



01. 분석 대상 데이터 수집

❖ ② 기상청 사이트에서 날씨 데이터 수집하기 (2/6)

- [방재기상관측(AWS)] 버튼을 클릭



01. 분석 대상 데이터 수집

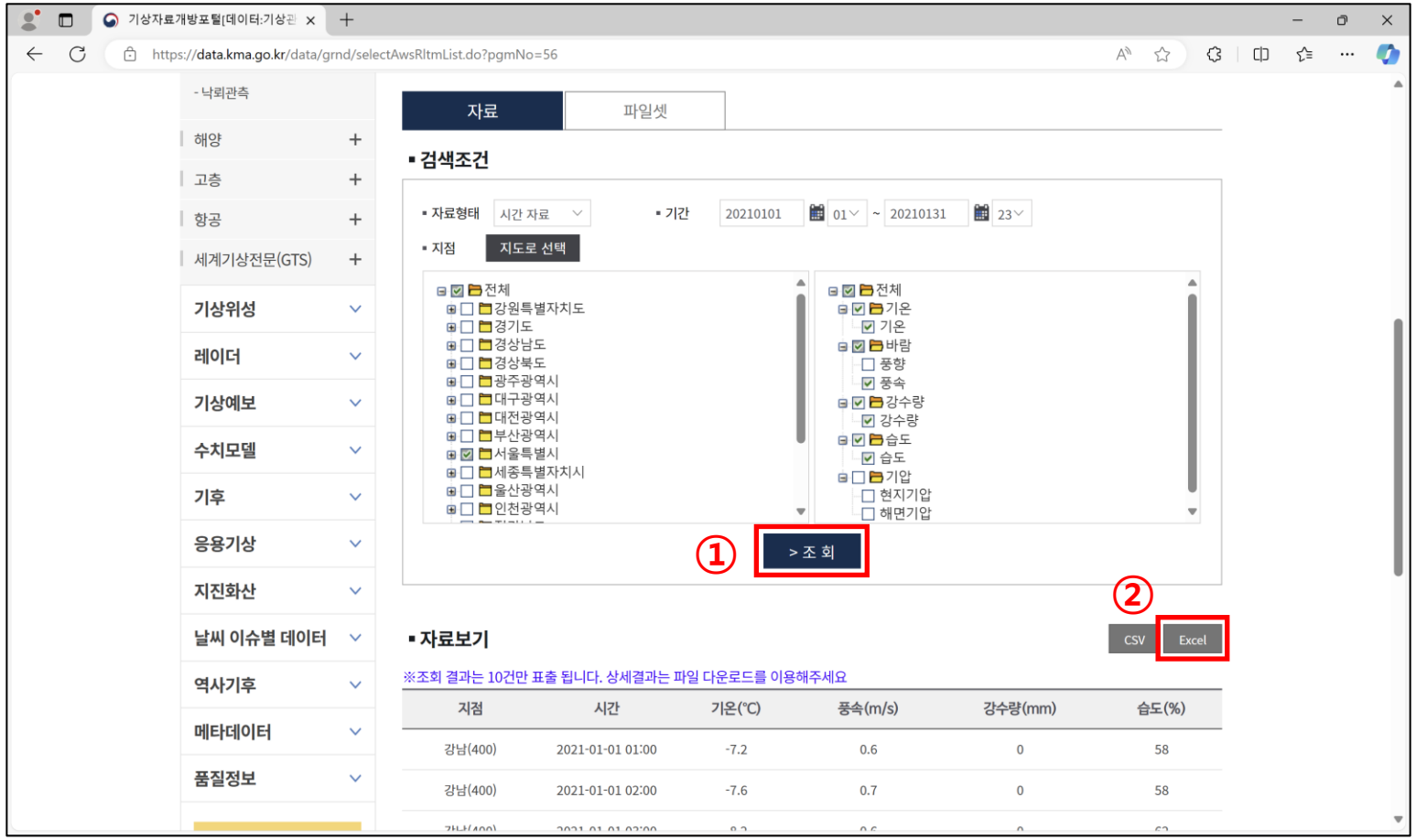
❖ ② 기상청 사이트에서 날씨 데이터 수집하기 (3/6)

- [검색조건]에서 [기간]을 아래와 같이 지정하고, [지점]을 '강남'으로 선택한 다음, 데이터 분석에 필요한 '기온', '풍속', '강수량', '습도'를 클릭함

01. 분석 대상 데이터 수집

❖ ② 기상청 사이트에서 날씨 데이터 수집하기 (4/6)

- [>조회] 버튼을 클릭 후, [Excel] 버튼을 클릭하여 파일을 다운로드하자



01. 분석 대상 데이터 수집

❖ ② 기상청 사이트에서 날씨 데이터 수집하기 (5/6)

- 다운로드한 파일의 형식은 'Microsoft Excel 97-2023 Worksheet.xls(*.xls)'임
- 이 형식의 파일은 코랩에 업로드하여 읽으면 오류가 발생하기 때문에 파일 형식을 'Excel 통합 문서(*.xlsx)'로 변경해야 함

01. 분석 대상 데이터 수집

❖ ② 기상청 사이트에서 날씨 데이터 수집하기 (6/6)

- 엑셀 메뉴에서 [파일] → [다른 이름으로 저장] → [찾아보기]를 클릭
- 대화상자가 표시되면 [파일 형식]을 'Excel 통합 문서(*.xlsx)'로 지정한 다음, [파일 이름]을 'weather'로 변경하고, [저장] 버튼을 클릭
- 'weather.xlsx' 파일에 저장된 각 열에 대해서 살펴보자

변수명	변수 설명	단위
기온	공기의 온도	℃
풍속	바람의 속도	m/s
강수량	강수량 혹은 강우량은 어떤 곳에 일정 기간 동안 내린 물의 총량	mm
습도	공기 중에 포함되어 있는 수증기의 양 또는 비율을 나타내는 단위	%

[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

02. 데이터 확인하기

01. 분석 대상 데이터 수집
03. 데이터 병합
04. 데이터 분석 및 시각화

02. 데이터 확인하기

❖ 미세먼지 데이터: ① 데이터 읽어서 확인하기 (1/3)

- 우선 pandas 라이브러리를 임포트(Import)하자
- 그다음 pandas의 read_excel( ) 함수를 사용하여 'dust.xlsx' 파일을 읽고, 첫 5행을 출력해 보자

```
1 import pandas as pd
2
3 dust = pd.read_excel("dust.xlsx")
4 dust.head()
```

실행결과

	날짜	아황산가스	일산화탄소	오존	이산화질소	PM10	PM2.5
0	2021-01-01 01	0.004	0.4	0.021	0.018	NaN	12.0
1	2021-01-01 02	0.004	0.4	0.019	0.020	20.0	13.0
2	2021-01-01 03	0.004	0.5	0.017	0.023	23.0	13.0
3	2021-01-01 04	0.004	0.5	0.015	0.024	17.0	12.0
4	2021-01-01 05	0.004	0.5	0.010	0.026	NaN	14.0

02. 데이터 확인하기

❖ 미세먼지 데이터: ① 데이터 읽어서 확인하기 (2/3)

- info( ) 함수를 사용해 데이터의 기본 정보를 확인하자
- 데이터프레임의 행과 열의 크기, 컬럼명, 컬럼을 구성하는 값의 자료형 등을 확인할 수 있음

```
1 dust.info()
```

실행결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 744 entries, 0 to 743
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   날짜        744 non-null    object
 1   아황산가스  740 non-null    float64
 2   일산화탄소  740 non-null    float64
 3   오존        740 non-null    float64
 4   이산화질소  740 non-null    float64
 5   PM10        725 non-null    float64
 6   PM2.5       739 non-null    float64
dtypes: float64(6), object(1)
memory usage: 40.8+ KB
```

02. 데이터 확인하기

❖ 미세먼지 데이터: ① 데이터 읽어서 확인하기 (3/3)

- describe( ) 함수를 사용해 데이터의 기초 통계량을 확인하자
- 숫자 데이터의 컬럼별 요약 통계를 확인할 수 있음

```
1 dust.describe()
```

실행결과

	아황산가스	일산화탄소	오존	이산화질소	PM10	PM2.5
count	740.000000	740.000000	740.000000	740.000000	725.000000	739.000000
mean	0.003654	0.563243	0.014154	0.030422	33.325517	21.833559
std	0.000628	0.164593	0.010689	0.014664	19.930029	12.222892
min	0.002000	0.300000	0.001000	0.006000	3.000000	3.000000
25%	0.003000	0.400000	0.003000	0.017000	20.000000	13.000000
50%	0.004000	0.500000	0.014000	0.030000	29.000000	19.000000
75%	0.004000	0.700000	0.024000	0.043000	43.000000	29.000000
max	0.006000	1.200000	0.037000	0.063000	163.000000	72.000000

02. 데이터 확인하기

❖ 미세먼지 데이터: ② 데이터 가공하기 (1/5)

- 한글 컬럼명을 rename( ) 함수를 사용하여 영문명으로 변경하자
- 데이터프레임에 반영하기 위해서는 반드시 inplace 매개변수에 True를 지정해야 함
- 컬럼명이 변경되었는지 확인하기 위해 dust의 첫 5행을 출력해 보자

```
1 dust.rename(columns={"날짜":"date",
2                      "아황산가스":"so2",
3                      "일산화탄소":"co",
4                      "오존":"o3",
5                      "이산화질소":"no2"}, inplace=True)
6 dust.head()
```

실행결과

	date	so2	co	o3	no2	PM10	PM2.5
0	2021-01-01 01	0.004	0.4	0.021	0.018	NaN	12.0
1	2021-01-01 02	0.004	0.4	0.019	0.020	20.0	13.0
2	2021-01-01 03	0.004	0.5	0.017	0.023	23.0	13.0
3	2021-01-01 04	0.004	0.5	0.015	0.024	17.0	12.0
4	2021-01-01 05	0.004	0.5	0.010	0.026	NaN	14.0

02. 데이터 확인하기

❖ 미세먼지 데이터: ② 데이터 가공하기 (2/5)

- 날짜(date) 컬럼에서 데이터 분석에 필요한 '년도-월-일'만 추출하자

```
1 dust["date"] = dust["date"].str[:10]
2 dust.head()
```

실행결과

	date	so2	co	o3	no2	PM10	PM2.5
0	2021-01-01	0.004	0.4	0.021	0.018	NaN	12.0
1	2021-01-01	0.004	0.4	0.019	0.020	20.0	13.0
2	2021-01-01	0.004	0.5	0.017	0.023	23.0	13.0
3	2021-01-01	0.004	0.5	0.015	0.024	17.0	12.0
4	2021-01-01	0.004	0.5	0.010	0.026	NaN	14.0

02. 데이터 확인하기

❖ 미세먼지 데이터: ② 데이터 가공하기 (3/5)

- 날짜(date) 컬럼의 자료형을 날짜형으로 변환해 보자

```
1 dust["date"] = pd.to_datetime(dust["date"])
2 dust.dtypes
```

실행결과

```
date     datetime64[ns]
so2             float64
co              float64
o3              float64
no2             float64
PM10            float64
PM2.5           float64
dtype: object
```

02. 데이터 확인하기

❖ 미세먼지 데이터: ② 데이터 가공하기 (4/5)

- 날짜(date) 컬럼에서 년도, 월, 일을 추출하여 새로운 컬럼을 생성하자

```
1 dust[“year”] = dust[“date”].dt.year
2 dust[“month”] = dust[“date”].dt.month
3 dust[“day”] = dust[“date”].dt.day
4 dust.columns
```

실행결과

```
Index([‘date’, ‘so2’, ‘co’, ‘o3’, ‘no2’, ‘PM10’, ‘PM2.5’, ‘year’, ‘month’,
       ‘day’],
      dtype=‘object’)
```

02. 데이터 확인하기

❖ 미세먼지 데이터: ② 데이터 가공하기 (5/5)

- 컬럼의 순서를 재정렬하자

```
1 dust = dust[[“date”, “year”, “month”, “day”, “so2”, “co”, “o3”, “no2”, “PM10”,
2              “PM2.5”]]
3 dust.columns
```

실행결과

```
Index([‘date’, ‘year’, ‘month’, ‘day’, ‘so2’, ‘co’, ‘o3’, ‘no2’, ‘PM10’,
       ‘PM2.5’],
      dtype=‘object’)
```

02. 데이터 확인하기

❖ 미세먼지 데이터: ③ 데이터 전처리 (1/5)

- 결측치를 확인해 보자

```
1 dust.isna().sum()
```

실행결과

```
date      0
year      0
month     0
day       0
so2       4
co        4
o3        4
no2       4
PM10     19
PM2.5     5
dtype: int64
```

isna( ) 함수와 isnull( ) 함수는 동일한 기능을 수행함

02. 데이터 확인하기

❖ 미세먼지 데이터: ③ 데이터 전처리 (2/5)

- 결측값을 앞 또는 뒤 방향으로 채울 수 있는데, 여기서는 앞 방향으로 채워보자
- 즉, 아래의 빨간색 박스로 표시한 NaN 값을, 이전 시간의 값을 기준으로 채우자

```
1 dust.ffill(inplace=True)          # forward fill → ffill
2 dust.head()
```

실행결과

실행 전

	date	year	month	day	so2	co	o3	no2	PM10	PM2.5
0	2021-01-01	2021	1	1	0.004	0.4	0.021	0.018	NaN	12.0
1	2021-01-01	2021	1	1	0.004	0.4	0.019	0.020	20.0	13.0
2	2021-01-01	2021	1	1	0.004	0.5	0.017	0.023	23.0	13.0
3	2021-01-01	2021	1	1	0.004	0.5	0.015	0.024	17.0	12.0
4	2021-01-01	2021	1	1	0.004	0.5	0.010	0.026	NaN	14.0

실행 후

	date	year	month	day	so2	co	o3	no2	PM10	PM2.5
0	2021-01-01	2021	1	1	0.004	0.4	0.021	0.018	NaN	12.0
1	2021-01-01	2021	1	1	0.004	0.4	0.019	0.020	20.0	13.0
2	2021-01-01	2021	1	1	0.004	0.5	0.017	0.023	23.0	13.0
3	2021-01-01	2021	1	1	0.004	0.5	0.015	0.024	17.0	12.0
4	2021-01-01	2021	1	1	0.004	0.5	0.010	0.026	17.0	14.0

파란색 박스와 같이 결측값의 이전 값이 없으면 어떻게 해야 할까?

02. 데이터 확인하기

❖ 미세먼지 데이터: ③ 데이터 전처리 (3/5)

- 이후 시간의 값으로 결측값을 채우고자 할 경우, bfill( ) 함수를 사용하면 됨
- 이번에는 파란색 박스로 표시한 NaN 값을, 이후 시간의 값을 기준으로 채우자

```
1 dust.bfill(inplace=True)          # backward fill → bfill
2 dust.head()
```

실행결과

실행 전

	date	year	month	day	so2	co	o3	no2	PM10	PM2.5
0	2021-01-01	2021	1	1	0.004	0.4	0.021	0.018	NaN	12.0
1	2021-01-01	2021	1	1	0.004	0.4	0.019	0.020	20.0	13.0
2	2021-01-01	2021	1	1	0.004	0.5	0.017	0.023	23.0	13.0
3	2021-01-01	2021	1	1	0.004	0.5	0.015	0.024	17.0	12.0
4	2021-01-01	2021	1	1	0.004	0.5	0.010	0.026	17.0	14.0

실행 후

	date	year	month	day	so2	co	o3	no2	PM10	PM2.5
0	2021-01-01	2021	1	1	0.004	0.4	0.021	0.018	20.0	12.0
1	2021-01-01	2021	1	1	0.004	0.4	0.019	0.020	20.0	13.0
2	2021-01-01	2021	1	1	0.004	0.5	0.017	0.023	23.0	13.0
3	2021-01-01	2021	1	1	0.004	0.5	0.015	0.024	17.0	12.0
4	2021-01-01	2021	1	1	0.004	0.5	0.010	0.026	17.0	14.0

결측값의 이전 또는 이후 값이 없으면 어떻게 해야 할까?

02. 데이터 확인하기

❖ 미세먼지 데이터: ③ 데이터 전처리 (4/5)

- 만약, 이전/이후 값이 없는 경우, 특정 값으로 채우면 됨

```
1 dust.fillna(20, inplace=True)
2 dust.head()
```

실행결과

	date	year	month	day	so2	co	o3	no2	PM10	PM2.5
0	2021-01-01	2021	1	1	0.004	0.4	0.021	0.018	20.0	12.0
1	2021-01-01	2021	1	1	0.004	0.4	0.019	0.020	20.0	13.0
2	2021-01-01	2021	1	1	0.004	0.5	0.017	0.023	23.0	13.0
3	2021-01-01	2021	1	1	0.004	0.5	0.015	0.024	17.0	12.0
4	2021-01-01	2021	1	1	0.004	0.5	0.010	0.026	17.0	14.0

02. 데이터 확인하기

❖ 미세먼지 데이터: ③ 데이터 전처리 (5/5)

- 결측치를 다시 한번 확인해 보자

```
1 dust.isna().sum()
```

실행결과

```
date     0
year     0
month    0
day      0
so2      0
co       0
o3       0
no2      0
PM10     0
PM2.5    0
dtype: int64
```

02. 데이터 확인하기

❖ 날씨 데이터: ① 데이터 읽어와서 확인하기 (1/2)

- ‘weather.xlsx’ 엑셀 파일을 읽어와 데이터프레임 weather에 저장하고, 첫 5행을 출력해 보자

```
1 weather = pd.read_excel(“weather.xlsx”)
2 weather.head()
```

실행결과

	지점	지점명	일시	기온(°C)	풍속(m/s)	강수량(mm)	습도(%)
0	400	강남	2021-01-01 01:00:00	-7.2	0.6	0.0	57.5
1	400	강남	2021-01-01 02:00:00	-7.6	0.7	0.0	57.5
2	400	강남	2021-01-01 03:00:00	-8.2	0.6	0.0	62.0
3	400	강남	2021-01-01 04:00:00	-8.1	0.5	0.0	60.5
4	400	강남	2021-01-01 05:00:00	-8.7	1.3	0.0	66.4

02. 데이터 확인하기

❖ 날씨 데이터: ① 데이터 읽어와서 확인하기 (2/2)

- info( ) 함수를 사용해 데이터의 기본 정보를 출력해 보자

```
1 weather.info()
```

실행결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 743 entries, 0 to 742
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   지점        743 non-null    int64
 1   지점명      743 non-null    object
 2   일시        743 non-null    datetime64[ns]
 3   기온(°C)    743 non-null    float64
 4   풍속(m/s)   743 non-null    float64
 5   강수량(mm)  743 non-null    float64
 6   습도(%)     743 non-null    float64
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 40.8+ KB
```

02. 데이터 확인하기

❖ 날씨 데이터: ② 데이터 가공하기 (1/5)

- 데이터 분석에 필요 없는 '지점', '지점명' 컬럼을 삭제하자

```
1 weather.drop("지점", axis=1, inplace=True)
2 weather.drop("지점명", axis=1, inplace=True)
3 weather.head()
```

실행결과

	일시	기온(°C)	풍속(m/s)	강수량(mm)	습도(%)
0	2021-01-01 01:00:00	-7.2	0.6	0.0	57.5
1	2021-01-01 02:00:00	-7.6	0.7	0.0	57.5
2	2021-01-01 03:00:00	-8.2	0.6	0.0	62.0
3	2021-01-01 04:00:00	-8.1	0.5	0.0	60.5
4	2021-01-01 05:00:00	-8.7	1.3	0.0	66.4

02. 데이터 확인하기

❖ 날씨 데이터: ② 데이터 가공하기 (2/5)

- 특수기호가 들어간 컬럼들의 이름을 변경하자

```
1 weather.columns = [“date”, “temp”, “wind”, “rain”, “humid”]
2 weather.head()
```

실행결과

	date	temp	wind	rain	humid
0	2021-01-01 01:00:00	-7.2	0.6	0.0	57.5
1	2021-01-01 02:00:00	-7.6	0.7	0.0	57.5
2	2021-01-01 03:00:00	-8.2	0.6	0.0	62.0
3	2021-01-01 04:00:00	-8.1	0.5	0.0	60.5
4	2021-01-01 05:00:00	-8.7	1.3	0.0	66.4

데이터 분석에 필요 없는
시:분:초 정보를 제거해 보자

02. 데이터 확인하기

❖ 날씨 데이터: ② 데이터 가공하기 (3/5)

- 시:분:초 정보를 제거하고, 날짜(date) 컬럼의 자료형을 살펴보자

```
1 weather[“date”] = weather[“date”].dt.date
2 weather.info()
```

실행결과

```
<class ‘pandas.core.frame.DataFrame’>
RangeIndex: 743 entries, 0 to 742
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    743 non-null    object
 1   temp    743 non-null    float64
 2   wind    743 non-null    float64
 3   rain    743 non-null    float64
 4   humid   743 non-null    float64
dtypes: float64(4), object(1)
memory usage: 29.1+ KB
```

**object 형식으로 변경되었으므로,
이를 다시 datetime 형식으로 변경해야 함**

02. 데이터 확인하기

❖ 날씨 데이터: ② 데이터 가공하기 (4/5)

- 2번 줄(Line)의 주석을 해제하고, 셀(Cell)을 다시 실행해 보자

```
1 weather["date"] = pd.to_datetime(weather["date"])
2 weather.info()
```

다음과 같이 astype( ) 함수를 사용해도 됨:
weather["date"].astype("datetime64[ns]")

실행결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 743 entries, 0 to 742
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    743 non-null    datetime64[ns]
 1   temp    743 non-null    float64
 2   wind    743 non-null    float64
 3   rain    743 non-null    float64
 4   humid   743 non-null    float64
dtypes: datetime64[ns](1), float64(4)
memory usage: 29.1 KB
```

datetime 형식으로 변경됨!

02. 데이터 확인하기

❖ 날씨 데이터: ② 데이터 가공하기 (5/5)

- 시:분:초가 제거되었는지 확인하기 위해, head( ) 함수를 사용해 첫 5행을 출력해 보자

```
1 weather.head()
```

실행결과

	date	temp	wind	rain	humid
0	2021-01-01	-7.2	0.6	0.0	57.5
1	2021-01-01	-7.6	0.7	0.0	57.5
2	2021-01-01	-8.2	0.6	0.0	62.0
3	2021-01-01	-8.1	0.5	0.0	60.5
4	2021-01-01	-8.7	1.3	0.0	66.4

03. 데이터 병합

01. 분석 대상 데이터 수집
02. 데이터 확인하기
04. 데이터 분석 및 시각화

03. 데이터 병합

❖ 미세먼지 데이터와 날씨 데이터 병합 (1/3)

- 데이터를 병합하기 전에 미세먼지 데이터(dust)와 날씨 데이터(weather)의 행, 열의 크기를 확인하자

```
1 print("미세먼지 데이터:", dust.shape)
2 print("날씨 데이터:", weather.shape)
```

실행결과

```
미세먼지 데이터: (744, 10)
날씨 데이터: (743, 5)
```

✓ 두 엑셀 파일을 열어서 확인해 보면, 크기 차이가 발생하는 이유를 알 수 있음
✓ 미세먼지 데이터의 경우, 2021-01-31 24:00 데이터가 있음
✓ 날씨 데이터의 경우, 2021-01-31 23:00 데이터까지 기록되어 있음(즉, 24:00 데이터는 없음)

크기가 다른 두 데이터프레임을 어떻게 병합할 수 있을까?

03. 데이터 병합

❖ 미세먼지 데이터와 날씨 데이터 병합 (2/3)

- 미세먼지 데이터에서 2021-01-31 24:00에 해당하는 행을 찾아서 삭제하자

```
1 dust.drop(index=743, inplace=True)
2 print(dust.shape)
```

실행결과

```
(743, 10)
```

**시간 순서대로 저장되어 있으므로,
맨 마지막 행을 삭제하면 됨!**

03. 데이터 병합

❖ 미세먼지 데이터와 날씨 데이터 병합 (3/3)

- dust와 weather 데이터프레임이 동일하게 가진 date 컬럼을 기준으로 병합해 보자

```
1 df = pd.merge(dust, weather, on="date")
2 df.head()
```

실행결과

					dust					weather			
	date	year	month	day	so2	co	o3	no2	PM10	PM2.5	temp	wind	rain	humid
0	2021-01-01	2021	1	1	0.004	0.4	0.021	0.018	20.0	12.0	-7.2	0.6	0.0	57.5
1	2021-01-01	2021	1	1	0.004	0.4	0.021	0.018	20.0	12.0	-7.6	0.7	0.0	57.5
2	2021-01-01	2021	1	1	0.004	0.4	0.021	0.018	20.0	12.0	-8.2	0.6	0.0	62.0
3	2021-01-01	2021	1	1	0.004	0.4	0.021	0.018	20.0	12.0	-8.1	0.5	0.0	60.5
4	2021-01-01	2021	1	1	0.004	0.4	0.021	0.018	20.0	12.0	-8.7	1.3	0.0	66.4

04. 데이터 분석 및 시각화

01. 분석 대상 데이터 수집
02. 데이터 확인하기
03. 데이터 병합

04. 데이터 분석 및 시각화

❖ 데이터 분석 (1/4)

- 상관 계수를 계산하는 corr( ) 함수를 이용해서, 미세먼지 데이터와 날씨 데이터의 모든 요소별 상관관계를 확인해 보자

```
1 df.corr()
```

실행결과

	year	month	day	so2	co	o3	no2	PM10	PM2.5	temp	wind	rain	humid
year	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
month	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
day	NaN	NaN	1.000000	-0.318239	0.226099	-0.118081	0.206333	0.016124	0.051036	0.491312	-0.074944	0.026872	0.176721
so2	NaN	NaN	-0.318239	1.000000	0.141117	-0.068687	0.085989	0.160874	0.147571	-0.375131	0.031460	-0.019894	-0.096445
co	NaN	NaN	0.226099	0.141117	1.000000	-0.756706	0.841594	0.529720	0.692664	0.318052	-0.322431	0.077886	0.338083
o3	NaN	NaN	-0.118081	-0.068687	-0.756706	1.000000	-0.924362	-0.348229	-0.525078	-0.203804	0.355105	-0.097039	-0.288327
no2	NaN	NaN	0.206333	0.085989	0.841594	-0.924362	1.000000	0.420554	0.566387	0.313188	-0.403745	0.110232	0.315524
PM10	NaN	NaN	0.016124	0.160874	0.529720	-0.348229	0.420554	1.000000	0.825433	0.175430	-0.108474	0.026378	0.216753
PM2.5	NaN	NaN	0.051036	0.147571	0.692664	-0.525078	0.566387	0.825433	1.000000	0.190698	-0.202018	0.069463	0.354713
temp	NaN	NaN	0.491312	-0.375131	0.318052	-0.203804	0.313188	0.175430	0.190698	1.000000	-0.211112	0.077955	0.213428
wind	NaN	NaN	-0.074944	0.031460	-0.322431	0.355105	-0.403745	-0.108474	-0.202018	-0.211112	1.000000	-0.078174	-0.461900
rain	NaN	NaN	0.026872	-0.019894	0.077886	-0.097039	0.110232	0.026378	0.069463	0.077955	-0.078174	1.000000	0.284686
humid	NaN	NaN	0.176721	-0.096445	0.338083	-0.288327	0.315524	0.216753	0.354713	0.213428	-0.461900	0.284686	1.000000

04. 데이터 분석 및 시각화

❖ 데이터 분석 (2/4)

- 미세먼지(PM10)를 기준으로 각 변수와의 상관관계를 알아보자

```
1 corr = df.corr()
2 corr[“PM10”].sort_values(ascending=False)
```

실행결과

```
PM10      1.000000
PM2.5     0.825433
co        0.529720
no2       0.420554
humid     0.216753
temp      0.175430
so2       0.160874
rain      0.026378
day       0.016124
wind     -0.108474
o3       -0.348229
year           NaN
month          NaN
Name: PM10, dtype: float64
```

미세먼지(PM10)는 초미세먼지(PM2.5), 일산화탄소(co), 이산화질소(no2)와 양의 상관관계가 있다는 것을 알 수 있음!

04. 데이터 분석 및 시각화

❖ 데이터 분석 (3/4)

- 초미세먼지(PM2.5)를 기준으로 각 변수와의 상관관계를 알아보자

```
1 corr["PM2.5"].sort_values(ascending=False)
```

실행결과

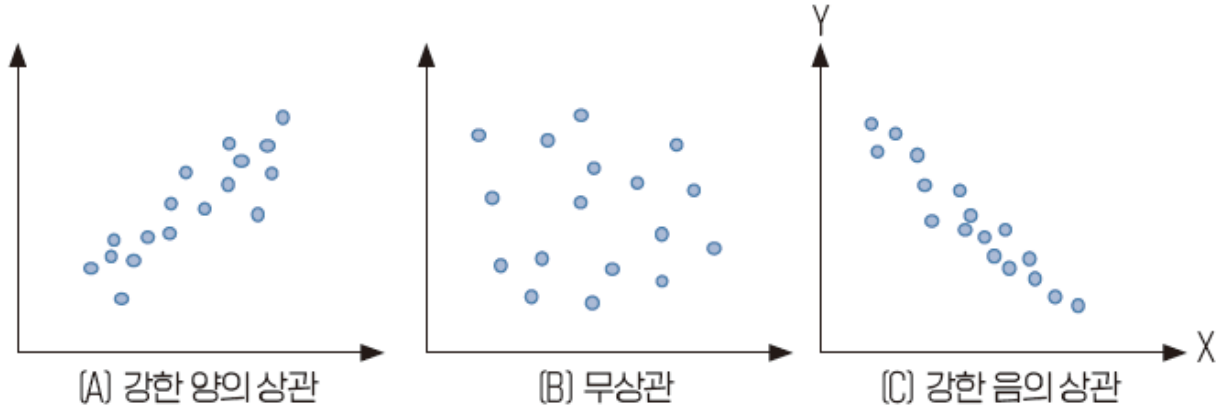
```
PM2.5     1.000000
PM10      0.825433
co        0.692664
no2       0.566387
humid     0.354713
temp      0.190698
so2       0.147571
rain      0.069463
day       0.051036
wind     -0.202018
o3       -0.525078
year           NaN
month          NaN
Name: PM2.5, dtype: float64
```

초미세먼지(PM2.5)도 미세먼지(PM10), 일산화탄소(co), 이산화질소(no2)와 양의 상관관계가 있다는 것을 알 수 있음!

04. 데이터 분석 및 시각화

❖ 데이터 분석 (4/4)

- 상관 계수는 어떻게 해석하는 것일까?



상관	상관 계수
양의 상관	+0.7 ~ +1.00이면, 강한 양의 상관관계 +0.3 ~ +0.70이면, 뚜렷한 양의 상관관계 +0.1 ~ +0.30이면, 약한 양의 상관관계
무상관	−0.1 ~ +0.10이면, 관계가 없음
음의 상관	−1.0 ~ −0.70이면, 강한 음의 상관관계 −0.7 ~ −0.30이면, 뚜렷한 음의 상관관계 −0.3 ~ −0.10이면, 약한 음의 상관관계

[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

04. 데이터 분석 및 시각화

❖ 데이터 시각화: ① 히스토그램으로 시각화 (1/2)

- 각 요소별 히스토그램을 출력해 보자

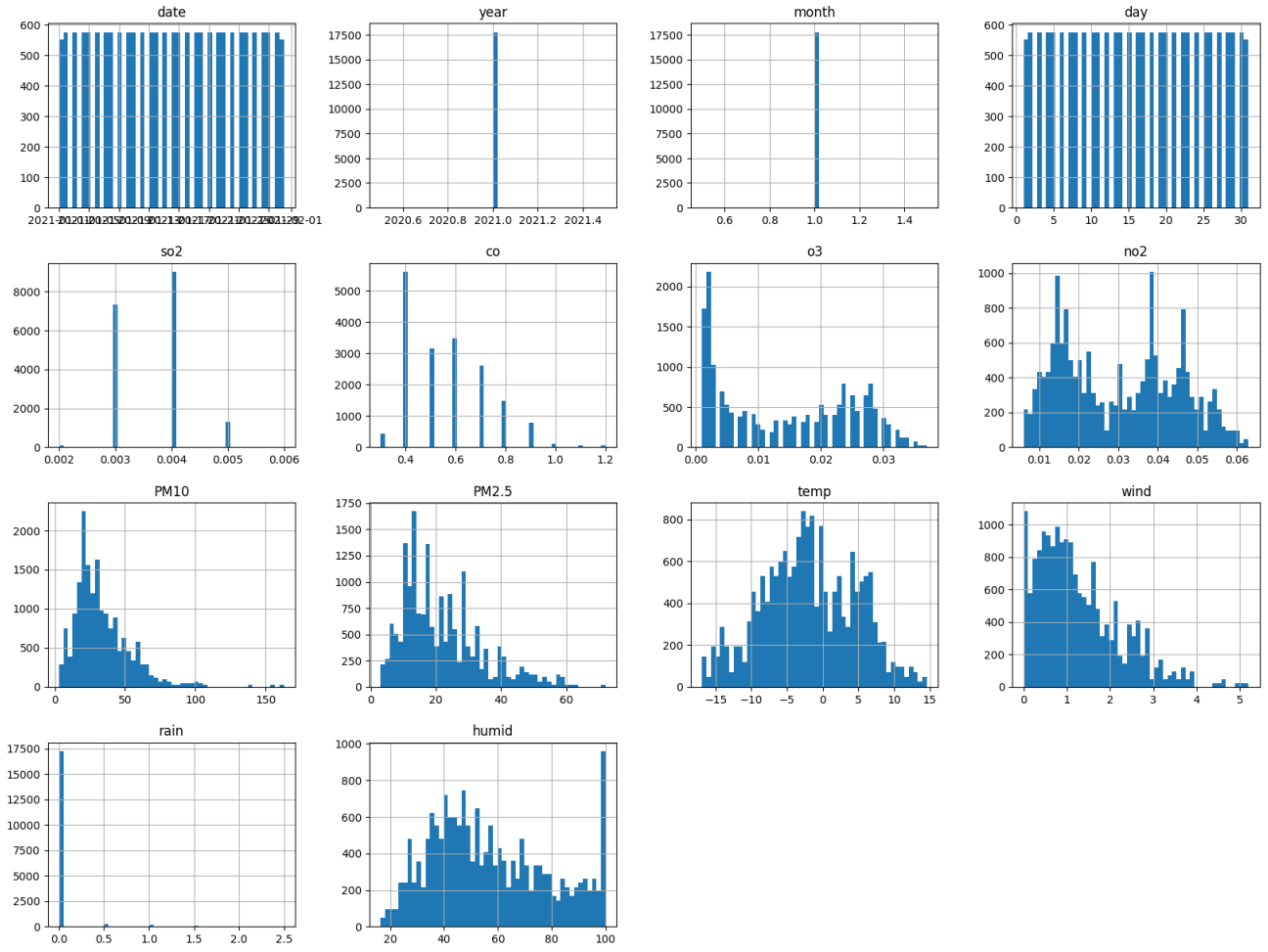
```
1 import matplotlib.pyplot as plt
2
3 df.hist(bins=50, figsize=(20,15))
4 plt.show()
```

✓ 데이터프레임의 hist( ) 함수를 호출하면, matplotlib.pyplot.hist( )가 호출됨
✓ 이 함수는 데이터프레임을 구성하는 시리즈(Series) 즉, 각 컬럼에 대해서 히스토그램을 출력함

04. 데이터 분석 및 시각화

❖ 데이터 시각화: ① 히스토그램으로 시각화 (2/2)

실행결과



04. 데이터 분석 및 시각화

❖ 데이터 시각화: ② 막대 그래프로 시각화 (1/2)

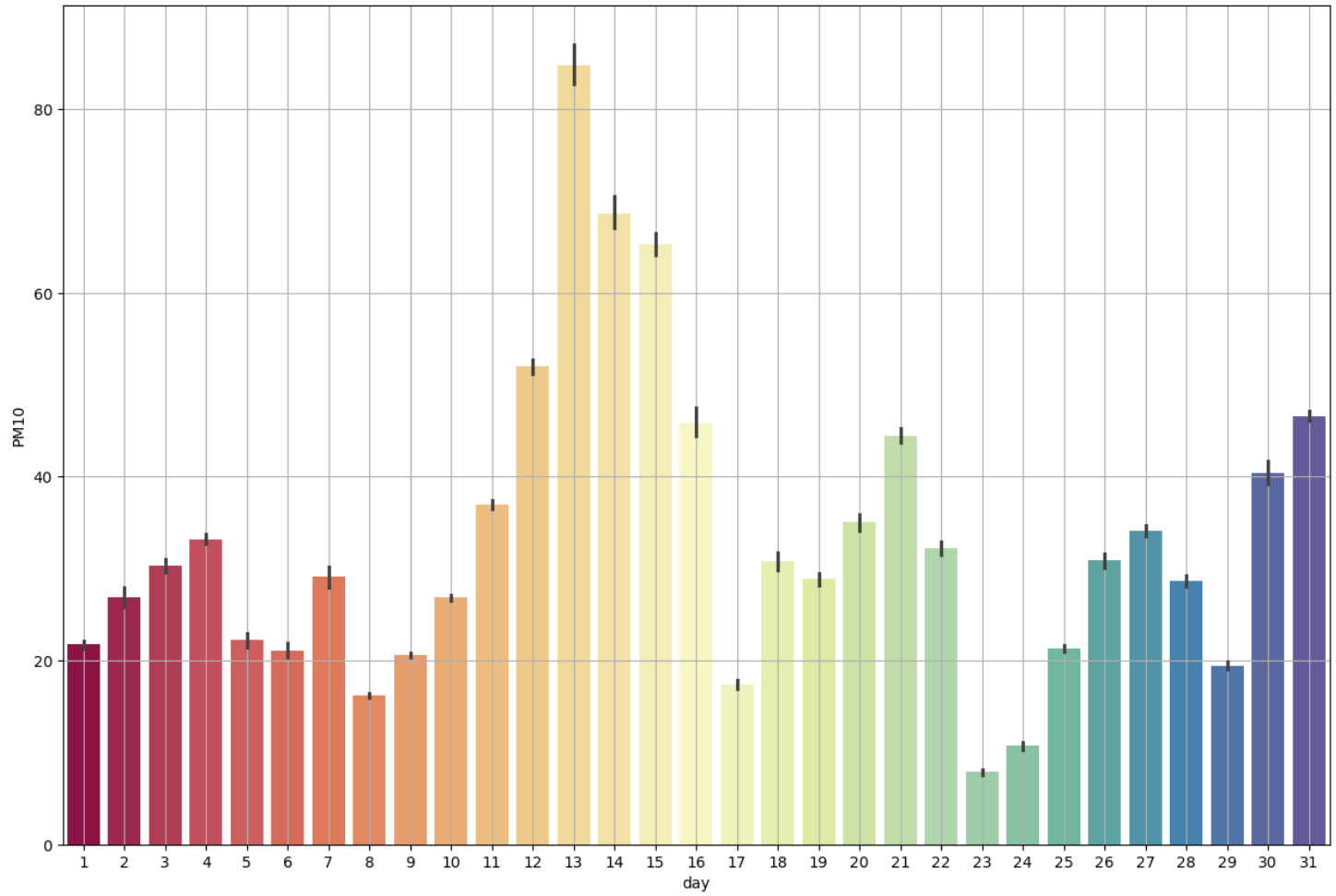
- 일별 미세먼지 현황을 막대 그래프로 출력해 보자
- 시본(Seaborn) 라이브러리를 추가로 임포트하여, 일자별로 다양한 색상으로 출력해 보자

```
1 import seaborn as sns
2
3 plt.figure(figsize=(15, 10))
4 sns.barplot(data=df, x="day", y="PM10", hue="day", palette="Spectral", legend=False)
5 plt.grid()
6 plt.show()
7
```

04. 데이터 분석 및 시각화

❖ 데이터 시각화: ② 막대 그래프로 시각화 (2/2)

실행결과



04. 데이터 분석 및 시각화

❖ 데이터 시각화: ③ 히트맵 그래프로 시각화 (1/2)

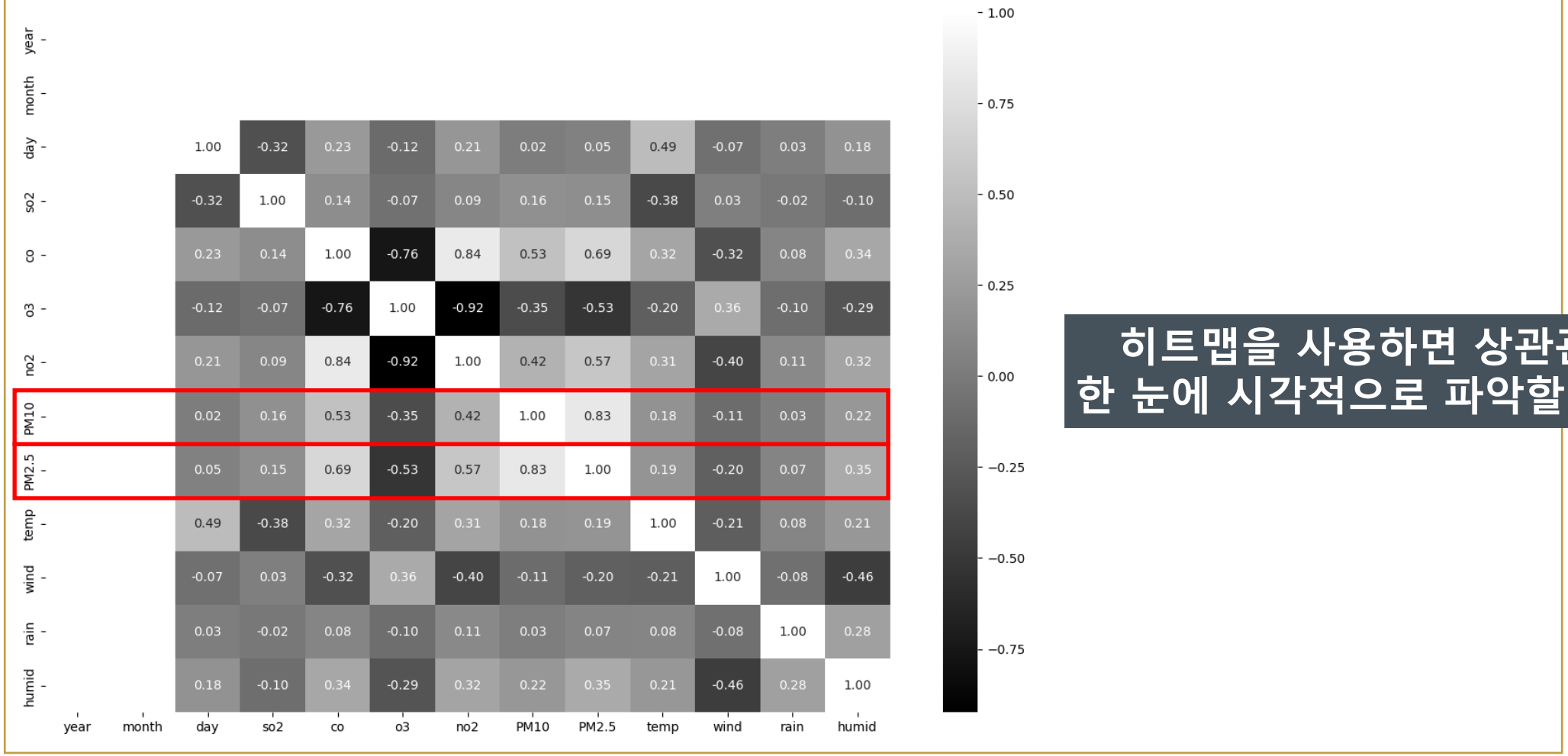
- 각 변수 간의 상관관계를 히트맵으로 출력해 보자
- 앞서 모든 요소별로 어떤 상관 관계가 있는지 corr( )를 이용하여 확인한 데이터를 히트맵 그래프를 이용하여 시각화 해보자

```
1 plt.figure(figsize=(15,10))
2 sns.heatmap(data=corr, annot=True, fmt=“.2f”, cmap=“gray”)
3 plt.show()
```

04. 데이터 분석 및 시각화

❖ 데이터 시각화: ③ 히트맵 그래프로 시각화 (2/2)

실행결과



히트맵을 사용하면 상관관계를 한 눈에 시각적으로 파악할 수 있음!

04. 데이터 분석 및 시각화

❖ 데이터 시각화: ④ 산점도 그래프로 시각화 (1/4)

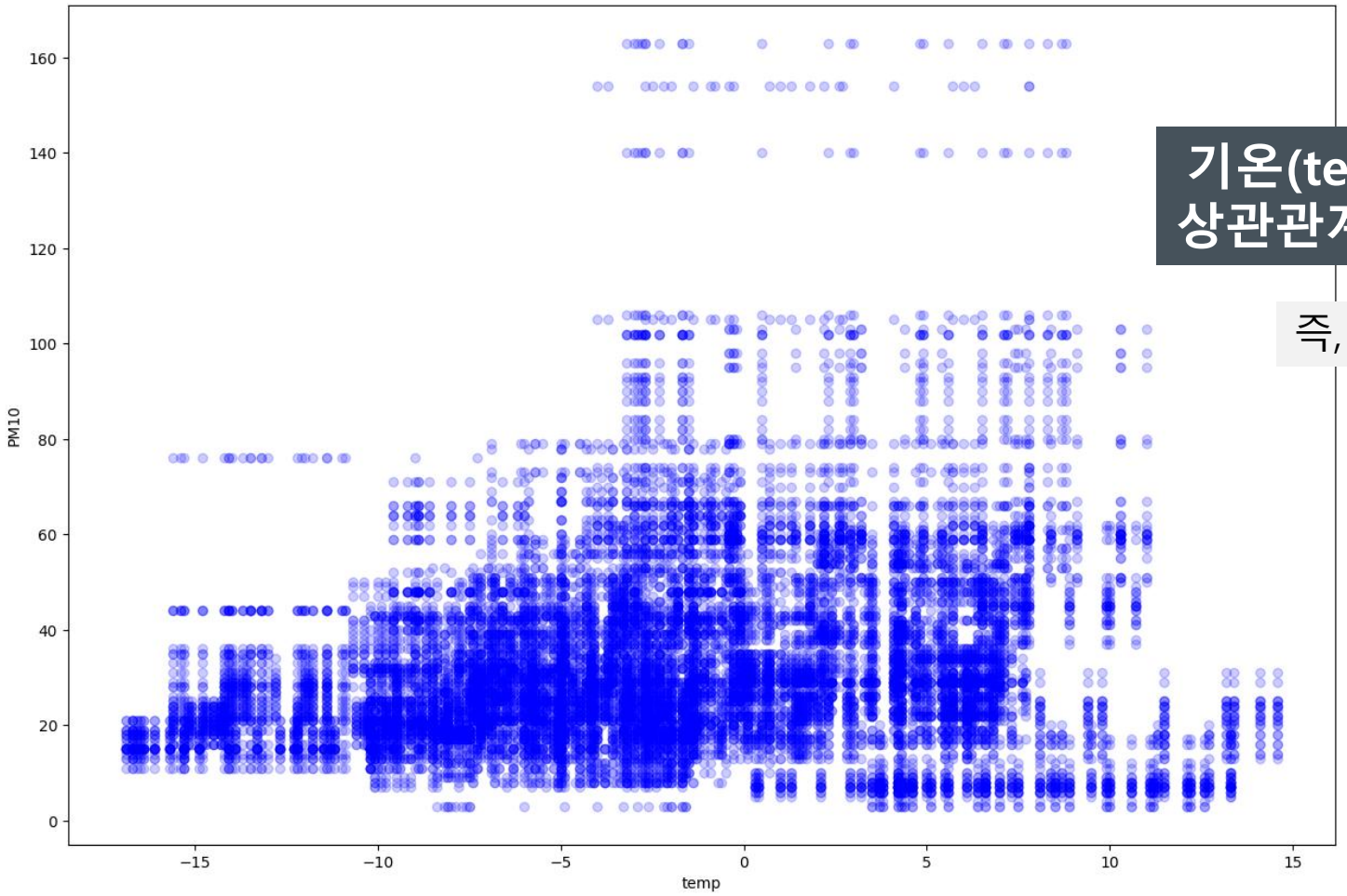
- 온도(temp)와 미세먼지(PM10) 사이의 상관관계를 산점도 그래프로 확인해 보자

```
1 plt.figure(figsize=(15,10))
2 x = df[“temp”]
3 y = df[“PM10”]
4 plt.scatter(x, y, marker=‘o’, color=“blue”, alpha=0.2)
5 plt.xlabel(“temp”)
6 plt.ylabel(“PM10”)
7 plt.show()
```

04. 데이터 분석 및 시각화

❖ 데이터 시각화: ④ 산점도 그래프로 시각화 (2/4)

실행결과



기온(temp)과 미세먼지(PM10)은 상관관계가 없음을 확인할 수 있음

즉, 기온과 미세먼지는 무관함

04. 데이터 분석 및 시각화

❖ 데이터 시각화: ④ 산점도 그래프로 시각화 (3/4)

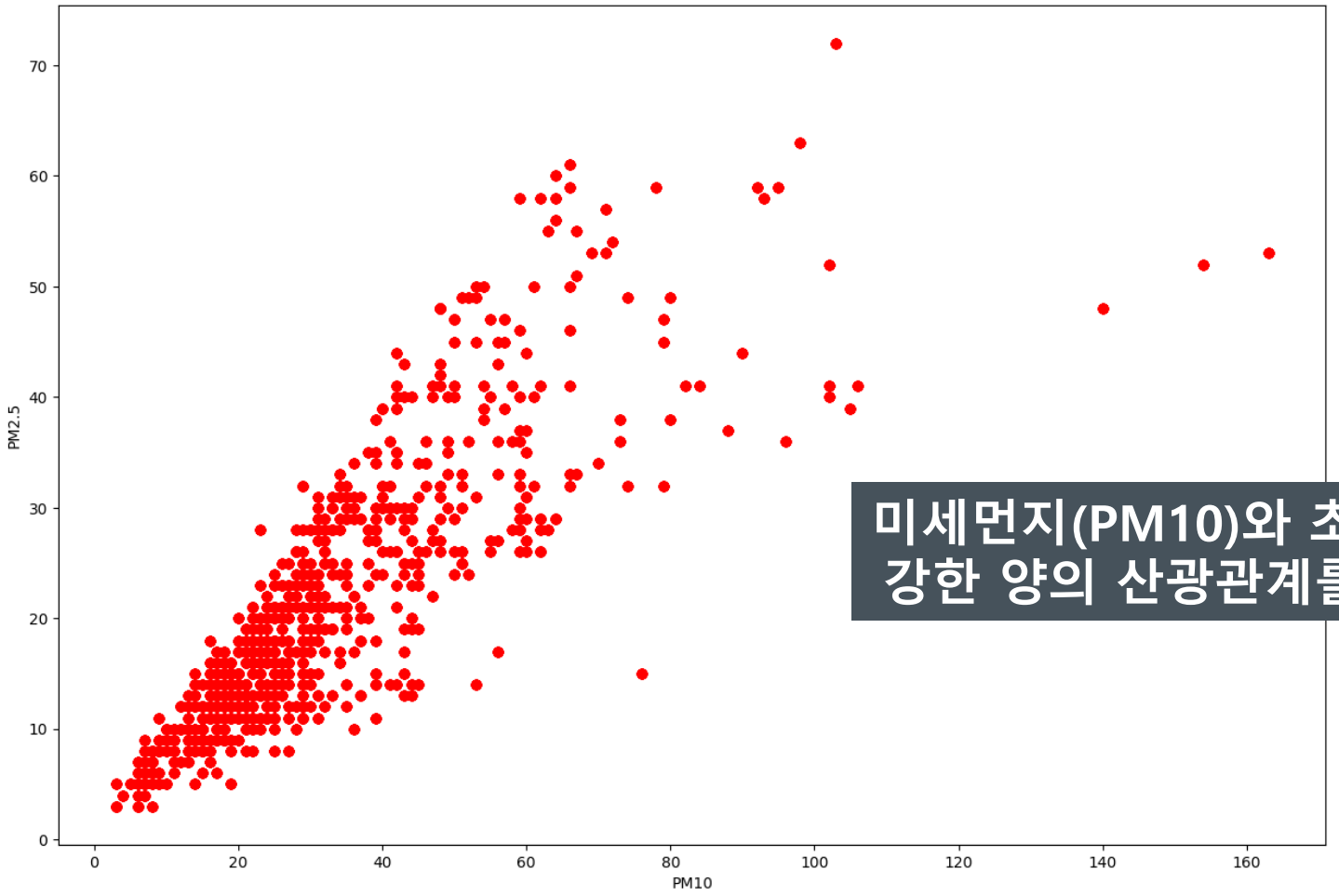
- 미세먼지(PM10)와 초미세먼지(PM2.5) 사이의 상관관계를 산점도 그래프로 확인해 보자

```
1 plt.figure(figsize=(15,10))
2 x = df[“PM10”]
3 y = df[“PM2.5”]
4 plt.scatter(x, y, marker=‘o’, color=“red”, alpha=0.5)
5 plt.xlabel(“PM10”)
6 plt.ylabel(“PM2.5”)
7 plt.show()
```

04. 데이터 분석 및 시각화

❖ 데이터 시각화: ④ 산점도 그래프로 시각화 (4/4)

실행결과



미세먼지(PM10)와 초미세먼지(PM2.5)가 강한 양의 산광관계를 갖는 것이 확인됨!

끝맺음

- ❖ 01. 분석 대상 데이터 수집
- ❖ 02. 데이터 확인하기
- ❖ 03. 데이터 병합
- ❖ 04. 데이터 분석 및 시각화

THANK YOU!
Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: bkwon@dongduk.ac.kr