1장: Pandas를 이용한 파이썬 데이터 분석 기초

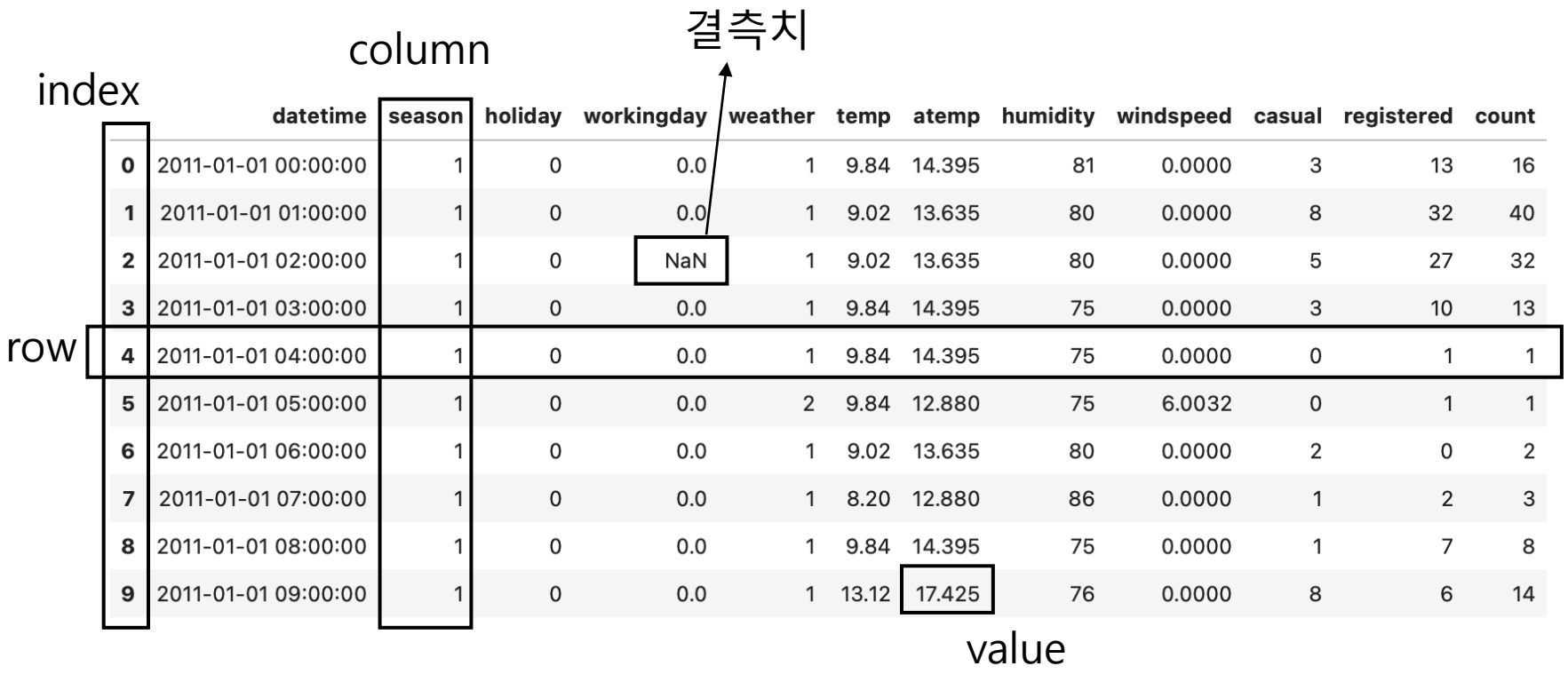
Pandas Dataframe

● **Pandas 라이브러리**

○ 정형화된 데이터를 파이썬에서 다루기 위해 필수적인 라이브러리

```
import pandas as pd
import numpy as np

df = pd.read_csv('./datasets/bike_rentals/bike_rentals.csv')
df.iloc[2, 3] = np.nan # 결측치를 임의로 만들기 위해 추가
df.head(10)
```



	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0.0	1	9.84	14.395	81	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0.0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	NaN	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0.0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0.0	1	9.84	14.395	75	0.0000	0	1	1
5	2011-01-01 05:00:00	1	0	0.0	2	9.84	12.880	75	6.0032	0	1	1
6	2011-01-01 06:00:00	1	0	0.0	1	9.02	13.635	80	0.0000	2	0	2
7	2011-01-01 07:00:00	1	0	0.0	1	8.20	12.880	86	0.0000	1	2	3
8	2011-01-01 08:00:00	1	0	0.0	1	9.84	14.395	75	0.0000	1	7	8
9	2011-01-01 09:00:00	1	0	0.0	1	13.12	17.425	76	0.0000	8	6	14

Dataframe 의 열 및 행 선택: iloc

- **iloc 메서드**
  - ○ 행과 열을 숫자 인덱스로 선택
  - ○ iloc[행 번호, 열 번호]
  - ○ 슬라이싱 가능

```
df.iloc[2:5, 3:6]
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
5	2011-01-01 05:00:00	1	0	0	2	9.84	12.880	75	6.0032	0	1	1
6	2011-01-01 06:00:00	1	0	0	1	9.02	13.635	80	0.0000	2	0	2
7	2011-01-01 07:00:00	1	0	0	1	8.20	12.880	86	0.0000	1	2	3
8	2011-01-01 08:00:00	1	0	0	1	9.84	14.395	75	0.0000	1	7	8
9	2011-01-01 09:00:00	1	0	0	1	13.12	17.425	76	0.0000	8	6	14

	workingday	weather	temp
2	0	1	9.02
3	0	1	9.84
4	0	1	9.84

Dataframe 의 열 및 행 선택: loc

- **loc 메서드**
  - 행과 열을 인덱스 이름으로 선택
  - loc[행 이름, 열 이름]
  - 슬라이싱 가능

```
df.loc[2:4, 'workingday':'temp']
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
5	2011-01-01 05:00:00	1	0	0	2	9.84	12.880	75	6.0032	0	1	1
6	2011-01-01 06:00:00	1	0	0	1	9.02	13.635	80	0.0000	2	0	2
7	2011-01-01 07:00:00	1	0	0	1	8.20	12.880	86	0.0000	1	2	3
8	2011-01-01 08:00:00	1	0	0	1	9.84	14.395	75	0.0000	1	7	8
9	2011-01-01 09:00:00	1	0	0	1	13.12	17.425	76	0.0000	8	6	14

	workingday	weather	temp
2	0	1	9.02
3	0	1	9.84
4	0	1	9.84

Dataframe 의 열 및 행 선택: loc을 이용한 조건 탐색

● **loc 메서드**

○ loc[조건문] 형태로 조건을 만족하는 행 필터링 가능

○ 열까지 선택하려면 loc[조건문, 열 이름] 형식으로 사용

1

```
df.loc[df['season'] == 2]
```

2

```
df.loc[df['season'] == 2, 'casual':]
```

1

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
1323	2011-04-01 00:00:00	2	0	1	3	10.66	12.880	100	11.0014	0	6	6
1324	2011-04-01 01:00:00	2	0	1	3	10.66	12.880	100	11.0014	0	4	4
1325	2011-04-01 02:00:00	2	0	1	3	10.66	12.880	93	12.9980	0	7	7
1326	2011-04-01 03:00:00	2	0	1	2	9.84	11.365	93	16.9979	0	4	4
1327	2011-04-01 04:00:00	2	0	1	2	9.84	11.365	93	16.9979	0	3	3
...	...	...	...	...	...	...	...	...	...	...	...	...
8146	2012-06-19 19:00:00	2	0	1	1	32.80	38.635	59	15.0013	82	432	514
8147	2012-06-19 20:00:00	2	0	1	1	32.80	37.880	55	16.9979	59	399	458
8148	2012-06-19 21:00:00	2	0	1	1	31.16	35.605	62	11.0014	37	239	276
8149	2012-06-19 22:00:00	2	0	1	1	29.52	34.850	79	6.0032	51	240	291
8150	2012-06-19 23:00:00	2	0	1	1	29.52	34.850	79	8.9981	23	102	125

2

	casual	registered	count
1323	0	6	6
1324	0	4	4
1325	0	7	7
1326	0	4	4
1327	0	3	3
...	...	...	...
8146	82	432	514
8147	59	399	458
8148	37	239	276
8149	51	240	291
8150	23	102	125

Dataframe 의 열 및 행 선택: loc을 이용한 조건 탐색

● **loc 메서드 조건문 활용**

○ "season"이 1이 아니면서 weathe가 2가 아닌 모든 행들만 필터링

○ & (and), | (or), != (not equal), ~ (not)

```
df.loc[(df['season'] != 1) & (df['weather'] != 2)]
```

```
df.loc[~(df['season'] == 1) & ~(df['weather'] == 2)]
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
1323	2011-04-01 00:00:00	2	0	1	3	10.66	12.880	100	11.0014	0	6	6
1324	2011-04-01 01:00:00	2	0	1	3	10.66	12.880	100	11.0014	0	4	4
1325	2011-04-01 02:00:00	2	0	1	3	10.66	12.880	93	12.9980	0	7	7
1328	2011-04-01 05:00:00	2	0	1	3	9.84	11.365	93	15.0013	1	11	12
1329	2011-04-01 06:00:00	2	0	1	3	9.84	11.365	93	15.0013	2	26	28
...	...	...	...	...	...	...	...	...	...	...	...	...
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

select_dtypes 을 이용한 열 선택

● **info 메서드**

○ dataframe의 정보 확인 (길이, 열, 데이터타입, 크기 등)

```
df = pd.read_csv('./datasets/bike_rentals/bike_rentals.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  object
 1   season      10886 non-null  int64
 2   holiday     10886 non-null  int64
 3   workingday  10886 non-null  int64
 4   weather     10886 non-null  int64
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

● **select_dtype 메서드**

○ include 인자: 특정 데이터타입을 가지는 열만 필터링

○ exclude 인자: 특정 데이터타입을 가지는 열만 제외

```
df.select_dtypes(include='int')
```

```
df.select_dtypes(exclude='int')
```

	season	holiday	workingday	weather	humidity	casual	registered	count
0	1	0	0	1	81	3	13	16
1	1	0	0	1	80	8	32	40
2	1	0	0	1	80	5	27	32
3	1	0	0	1	75	3	10	13
4	1	0	0	1	75	0	1	1
...	...	...	...	...	...	...	...	...
10881	4	0	1	1	50	7	329	336
10882	4	0	1	1	57	10	231	241
10883	4	0	1	1	61	4	164	168
10884	4	0	1	1	61	12	117	129
10885	4	0	1	1	66	4	84	88

filter 메서드를 이용한 행과 열 선택

● **filter 메서드**

- ○ like 인자: 전달된 값을 포함하는 행/열만 필터링
- ○ items 인자: 전달된 값에 해당하는 행/열만 필터링
- ○ regex 인자: 정규 표현식을 통한 행/열 필터링
- ○ axis 인자: 0은 행, 1은 열을 뜻함 (기본값은 1)

```
df = pd.read_csv('./datasets/bike_rentals/bike_rentals.csv')
df = df.set_index('datetime’)
df.filter(like='00:00:00', axis=0)
```

datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3	13	16
2011-01-02 00:00:00	1	0	0	2	18.86	22.725	88	19.9995	4	13	17
2011-01-03 00:00:00	1	0	1	1	9.02	9.850	44	23.9994	0	5	5
2011-01-04 00:00:00	1	0	1	1	6.56	9.090	55	7.0015	0	5	5
2011-01-05 00:00:00	1	0	1	1	8.20	12.880	64	0.0000	0	6	6
...	...	...	...	...	...	...	...	...	...	...	...
2012-12-15 00:00:00	4	0	0	1	12.30	16.665	70	0.0000	4	90	94
2012-12-16 00:00:00	4	0	0	2	14.76	18.940	62	0.0000	8	102	110
2012-12-17 00:00:00	4	0	1	2	15.58	19.695	87	0.0000	2	26	28
2012-12-18 00:00:00	4	0	1	2	18.04	21.970	94	8.9981	0	18	18
2012-12-19 00:00:00	4	0	1	1	12.30	15.910	61	0.0000	6	35	41

filter 메서드를 이용한 행과 열 선택

● **filter 메서드**

- ○ like 인자: 전달된 값을 포함하는 행/열만 필터링
- ○ items 인자: 전달된 값에 해당하는 행/열만 필터링
- ○ regex 인자: 정규 표현식을 통한 행/열 필터링
- ○ axis 인자: 0은 행, 1은 열을 뜻함 (기본값은 1)

```
df.filter(items=['humidity', 'windspeed'])
```

	humidity	windspeed
0	81	0.0000
1	80	0.0000
2	80	0.0000
3	75	0.0000
4	75	0.0000
...	...	...
10881	50	26.0027
10882	57	15.0013
10883	61	15.0013
10884	61	6.0032
10885	66	8.9981

```
df.filter(regex='in.s')
```

	windspeed
datetime	
2011-01-01 00:00:00	0.0000
2011-01-01 01:00:00	0.0000
2011-01-01 02:00:00	0.0000
2011-01-01 03:00:00	0.0000
2011-01-01 04:00:00	0.0000
...	...
2012-12-19 19:00:00	26.0027
2012-12-19 20:00:00	15.0013
2012-12-19 21:00:00	15.0013
2012-12-19 22:00:00	6.0032
2012-12-19 23:00:00	8.9981

rename 을 사용한 행, 열 이름 변경

- **rename 메서드**
  - ○ 변경 위한 인덱스/열 이름을 {변경전:변경후} 딕셔너리로 전달
  - ○ {"변경 대상 인덱스나 열 이름" : "변경하고자 하는 이름"}
  - ○ axis 인자: 0은 행, 1은 열을 뜻함 (기본값은 0)
  - ○ axis 인자 대신 columns, index 인자 사용 가능

```
df = pd.read_csv('./datasets/bike_rentals/bike_rentals.csv')
df.rename(
    {'registered':'registered_user',
    'casual':'unregistered_user'},
    axis=1
)
```

```
df.rename(
    columns={'registered':'registered_user',
             'casual':'unregistered_user'}
)
```

datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	unregistered_user	registered_user	count
2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3	13	16
2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...
2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

데이터 파악에 기본이 되는 info, describe, value_counts, unique 메서드

● **info 메서드**

○ dataframe의 정보 확인 (길이, 열, 데이터타입, 크기, 결측치 등)

○ 데이터 쿼리 후 가장 먼저 사용하여 데이터 탐색 위한 유익한 정보 확인 가능

```
df = pd.read_csv('./datasets/bookings/bookings.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 525 entries, 0 to 524
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Hotel_Name    525 non-null    object 
 1   Review        325 non-null    object 
 2   Total_Review  325 non-null    object 
 3   Rating        315 non-null    float64
 4   Location      525 non-null    object 
dtypes: float64(1), object(4)
memory usage: 20.6+ KB
```

데이터 파악에 기본이 되는 info, describe, value_counts, unique 메서드

● **describe 메서드**

○ 수치형 변수들에 대한 요약 통계량 확인

○ 데이터의 개수, 평균, 표준편차, 최대/최소, 사분위수

○ include 인자에 "object" 전달 시 범주형 변수에 대한 고유값, 최빈값 확인 가능

```
df = pd.read_csv('./datasets/bookings/bookings.csv')
df.describe()
```

	Rating
count	315.000000
mean	7.883492
std	0.885183
min	1.000000
25%	7.500000
50%	8.000000
75%	8.400000
max	10.000000

```
df.describe(include='object')
```

	Hotel_Name	Review	Total_Review	Location
count	525	325	325	525
unique	498	8	255	25
top	Oakwood Residence Midtown East	Very good	1 review	Manhattan
freq	2	117	5	112

데이터 파악에 기본이 되는 info, describe, value_counts, unique 메서드

● **value_counts 메서드**

○ 범주형 변수에 유용하게 사용

○ 특정 변수의 모든 고유값과 그 고유값의 빈도수 반환

```
df = pd.read_csv('./datasets/bookings/bookings.csv')
df['Review'].value_counts()
```

```
Review
Very good          117
Good               116
Review score        31
Fabulous            30
Superb              16
Superb 9.0           8
Exceptional          5
Exceptional 10       2
Name: count, dtype: int64
```

```
df.loc[df['Review'] == "Superb 9.0", "Review"] = "Superb "
df.loc[df['Review'] == "Exceptional 10", "Review"] = "Exceptional "

df['Review'].value_counts()
```

```
Review
Very good          117
Good               116
Review score        31
Fabulous            30
Superb              24
Exceptional          7
Name: count, dtype: int64
```

데이터 파악에 기본이 되는 info, describe, value_counts, unique 메서드

● **unique 메서드**

○ 특정 변수의 모든 고유값 반환

1

```
df = pd.read_csv('./datasets/bookings/bookings.csv')
df['Total_Review'].unique()
```

1

```
array(['28', '52', '2,870', '975', '13,951', '8,044', '16,148', '343',
       '6,038', '2,028', '9,659', '4,435', '7,298', '11,455', '2,802',
       '1,847', '2,189', '703', '1,382', '4,646', '951', '4,498', '1,475',
       '6,245', '5,866', '959', '1,876', '848', '3,097', '3,477', '1,648',
       '46', '2,289', '148', '664', '1,867', '1,067', '130', '920',
       '2,560', '3,878', '1,698', '719', '7,585', '1,180', '2,035',
       '6,144', '638', '9,240', '3,907', '3,128', '2,383', '833', '558',
       '3,170', '2,857', '981', '192', '1,449', '1,373', '598', '3,859',
       '3,639', '1,048', '2,859', '2,412', '749', '6,639', '1,143',
       '2,353', '3,002', '8,844', '57', '657', '2,535', '946', '1,087',
       '3,627', '844', '483', '1,078', '1,074', '1,797', '728', '415',
       '766', '523', '770', '178', '464', '647', '1,225', '562',
       '1 review', '1,188', '1,260', '4,246', '2,521', '586', '1,643',
                                  <중략>
       '4,288', '787', '355', '1,251', '128', '258', '125', '699', '285',
       '109', '30', '1,754', '437', '60', '200', '2', '453', '769', '465',
       '6', '524', '434', '372', '817', '472', '314', '168', '1,085',
       '1,195', '1,118', '1,471', '416', '173', '352', '385', '1,482',
       '22 external  ', '241', '497', '117', '59', '79', '3',
       '3 external  ', '379', '42', '255', '99', '212', '602', '9', '165',
       '146', '14 external  ', '13 external  ', '7 external  ',
       '4 external  '], dtype=object)
```

2

```
df['Total_Review'] = df['Total_Review'].map(lambda x: str(x).replace('external','').strip())
df['Total_Review'] = df['Total_Review'].map(lambda x: str(x).replace('review','').strip())
df['Total_Review'] = df['Total_Review'].map(lambda x: str(x).replace(',',''))
df['Total_Review'] = df['Total_Review'].astype('float')

df['Total_Review'].describe()
```

2

```
count      325.000000
mean      1771.630769
std       2530.013514
min          1.000000
25%        343.000000
50%        920.000000
75%       2189.000000
max      16148.000000
Name: Total_Review, dtype: float64
```

결측치를 처리하는 fillna, dropna 메서드

- **isna 메서드**
  - ○ 값이 결측치인 경우 True를, 아닌 경우 False를 반환
  - ○ sum 메서드와 함께 사용하여 결측치의 개수 확인 가능
- **fillna 메서드**
  - ○ 결측치를 인자에 전달하는 값으로 대체

```
df = pd.read_csv('./datasets/bookings/bookings.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 525 entries, 0 to 524
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Hotel_Name    525 non-null    object
 1   Review        325 non-null    object
 2   Total_Review  325 non-null    object
 3   Rating        315 non-null    float64
 4   Location      525 non-null    object
dtypes: float64(1), object(4)
memory usage: 20.6+ KB
```

```
#Rating 열에서 결측치인 상위 5개 인덱스 확인
index = df[df['Rating'].isna()].head(5).index
index
```

```
Index([167, 168, 169, 170, 171], dtype='int64')
```

```
# 평균으로 결측치 대체
df['Rating'] = df['Rating'].fillna(df['Rating'].mean())

# 좀 전의 인덱스 확인
df.loc[index, 'Rating']
```

```
167    7.883492
168    7.883492
169    7.883492
170    7.883492
171    7.883492
Name: Rating, dtype: float64
```

결측치를 처리하는 fillna, dropna 메서드

● **fillna 메서드**

○ 결측치를 인자에 전달하는 값으로 대체

○ method 인자에 "ffill" (front fill) 전달 시 앞의 값으로 뒤의 결측치 대체

○ method 인자에 "bfill" (back fill) 전달 시 뒤의 값으로 앞의 결측치 대체

```
import numpy as np
s = pd.Series([1, np.nan, np.nan,2, np.nan, 3])
s
```

```
0    1.0
1    NaN
2    NaN
3    2.0
4    NaN
5    3.0
dtype: float64
```

```
s.fillna(method='ffill')
```

```
0    1.0
1    1.0
2    1.0
3    2.0
4    2.0
5    3.0
dtype: float64
```

```
s.fillna(method='bfill')
```

```
0    1.0
1    2.0
2    2.0
3    2.0
4    3.0
5    3.0
dtype: float64
```

결측치를 처리하는 fillna, dropna 메서드

● **dropna 메서드**

- ○ 결측치가 포함된 특정 행/열을 삭제
- ○ axis 인자: 0은 행, 1은 열을 뜻함
- ○ subset 인자: 결측치를 제거할 레이블 특정
- ○ how 인자: "all" 전달 시모든 값이 결측치인 경우에만 삭제

1

```
df = pd.read_csv('./datasets/bookings/bookings.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 525 entries, 0 to 524
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Hotel_Name    525 non-null    object
 1   Review        325 non-null    object
 2   Total_Review  325 non-null    object
 3   Rating        315 non-null    float64
 4   Location      525 non-null    object
dtypes: float64(1), object(4)
memory usage: 20.6+ KB
```

2

```
df = df.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 307 entries, 0 to 333
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Hotel_Name    307 non-null    object
 1   Review        307 non-null    object
 2   Total_Review  307 non-null    object
 3   Rating        307 non-null    float64
 4   Location      307 non-null    object
dtypes: float64(1), object(4)
memory usage: 14.4+ KB
```

3

```
df = df.dropna(subset='Review', axis=0)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 325 entries, 0 to 343
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Hotel_Name    325 non-null    object
 1   Review        325 non-null    object
 2   Total_Review  317 non-null    object
 3   Rating        307 non-null    float64
 4   Location      325 non-null    object
dtypes: float64(1), object(4)
memory usage: 15.2+ KB
```

4

```
df['Review'].isna().sum()
```

```
0
```

데이터의 분위수를 구하는 quantile 메서드

● **quantile 메서드**

○ 수치형 데이터의 분위수 반환

○ 0부터 1 사이의 분위수 값을 인자로 전달

○ interpolation 인자에 linear, lower, higher, nearest, midpoint 전달 가능 (기본값은 linear)

```
# 데이터 쿼리 및 Total review 열의 이상값 변환
df = pd.read_csv('./datasets/bookings/bookings.csv')

df['Total_Review'] = df['Total_Review'].map(lambda x: str(x).replace('external','').strip())
df['Total_Review'] = df['Total_Review'].map(lambda x: str(x).replace('review','').strip())
df['Total_Review'] = df['Total_Review'].map(lambda x: str(x).replace(',',''))
df['Total_Review'] = df['Total_Review'].astype('float')
```

```
quantile = [0, 0.2, 0.4, 0.6, 0.8, 1]

for idx in quantile:
    q = df['Total_Review'].quantile(idx, interpolation='lower')
    print(f'quantile({idx}) is {q}')
```

```
quantile(0) is 1.0
quantile(0.2) is 227.0
quantile(0.4) is 657.0
quantile(0.6) is 1169.0
quantile(0.8) is 2620.0
quantile(1) is 16148.0
```

```
for idx in quantile:
    q = df['Total_Review'].quantile(idx, interpolation='nearest')
    print(f'quantile({idx}) is {q}')
```

```
quantile(0) is 1.0
quantile(0.2) is 230.0
quantile(0.4) is 663.0
quantile(0.6) is 1169.0
quantile(0.8) is 2620.0
quantile(1) is 16148.0
```

원하는 데이터만 필터링 하는 query 메서드

- **query 메서드**
  - 인자로 조건식을 나타내는 문자열을 전달하여 행 필터링
  - ==, !=, and, or, ~ 사용 가능
  - 다양한 문자열 메서드 사용 가능 (contains, startswith, endswith)
  - @를 사용하여 외부 변수 참조 가능

```
import seaborn as sns

df = sns.load_dataset('penguins')
df.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

```
df.query('bill_length_mm > 55')
```

```
# 아래와 동일 결과
df[df['bill_length_mm'] > 55]
```

```
# 아래와 동일 결과
df.loc[df['bill_length_mm'] > 55]
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
169	Chinstrap	Dream	58.0	17.8	181.0	3700.0	Female
215	Chinstrap	Dream	55.8	19.8	207.0	4000.0	Male
253	Gentoo	Biscoe	59.6	17.0	230.0	6050.0	Male
321	Gentoo	Biscoe	55.9	17.0	228.0	5600.0	Male
335	Gentoo	Biscoe	55.1	16.0	230.0	5850.0	Male

원하는 데이터만 필터링 하는 query 메서드

- **query 메서드**
  - 인자로 조건식을 나타내는 문자열을 전달하여 행 필터링
  - ==, !=, and, or, ~ 사용 가능
  - 다양한 문자열 메서드 사용 가능 (contains, startswith, endswith)
  - @를 사용하여 외부 변수 참조 가능

```
df.query('bill_length_mm > 55 and species == "Gentoo"')
```

```
length = 55
species = 'Gentoo'

df.query('bill_length_mm >= @length and species == @species')
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
253	Gentoo	Biscoe	59.6	17.0	230.0	6050.0	Male
321	Gentoo	Biscoe	55.9	17.0	228.0	5600.0	Male
335	Gentoo	Biscoe	55.1	16.0	230.0	5850.0	Male

원하는 데이터만 필터링 하는 query 메서드

- **query 메서드**
  - 인자로 조건식을 나타내는 문자열을 전달하여 행 필터링
  - ==, !=, and, or, ~ 사용 가능
  - 다양한 문자열 메서드 사용 가능 (contains, startswith, endswith)
  - @를 사용하여 외부 변수 참조 가능

```
df.query('island.str.contains("oe")', engine='python')
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
20	Adelie	Biscoe	37.8	18.3	174.0	3400.0	Female
21	Adelie	Biscoe	37.7	18.7	180.0	3600.0	Male
22	Adelie	Biscoe	35.9	19.2	189.0	3800.0	Female
23	Adelie	Biscoe	38.2	18.1	185.0	3950.0	Male
24	Adelie	Biscoe	38.8	17.2	180.0	3800.0	Male
...	...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	Female
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Female
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Male

```
df.query('species.str.endswith("e")', engine='python')
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
...	...	...	...	...	...	...	...
147	Adelie	Dream	36.6	18.4	184.0	3475.0	Female
148	Adelie	Dream	36.0	17.8	195.0	3450.0	Female
149	Adelie	Dream	37.8	18.1	193.0	3750.0	Male
150	Adelie	Dream	36.0	17.1	187.0	3700.0	Female
151	Adelie	Dream	41.5	18.5	201.0	4000.0	Male

원하는 데이터만 필터링 하는 query 메서드

● **query 메서드**

○ 인자로 조건식을 나타내는 문자열을 전달하여 행 필터링

○ ==, !=, and, or, ~ 사용 가능

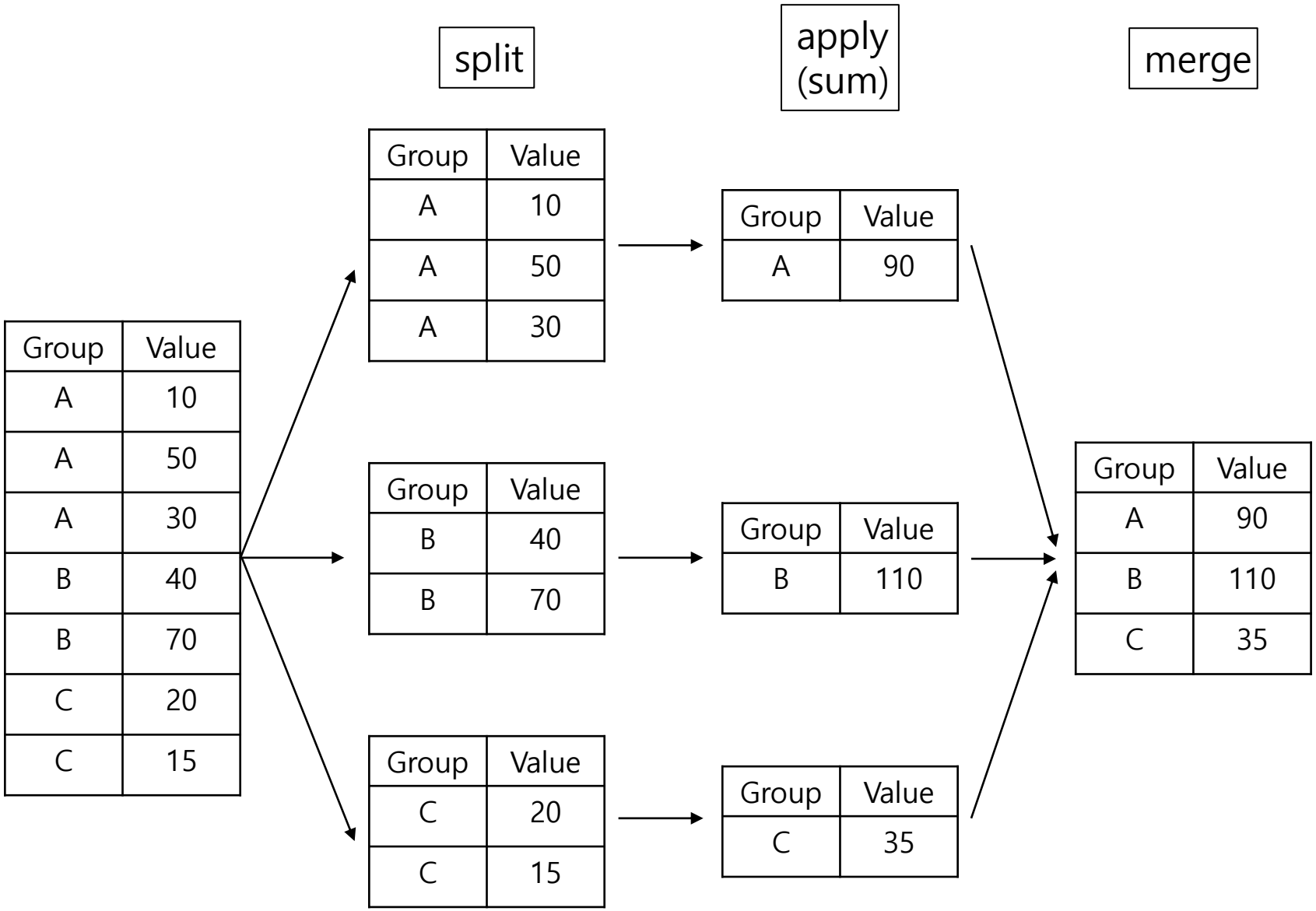
○ 다양한 문자열 메서드 사용 가능 (contains, startswith, endswith)

○ @를 사용하여 외부 변수 참조 가능

```
filtering = ["Adelie", "Chinstrap"]
df.query('species.isin(@filtering)', engine='python')
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
...	...	...	...	...	...	...	...
215	Chinstrap	Dream	55.8	19.8	207.0	4000.0	Male
216	Chinstrap	Dream	43.5	18.1	202.0	3400.0	Female
217	Chinstrap	Dream	49.6	18.2	193.0	3775.0	Male
218	Chinstrap	Dream	50.8	19.0	210.0	4100.0	Male
219	Chinstrap	Dream	50.2	18.7	198.0	3775.0	Female

Pandas 의 꽃, 그룹별 연산을 위한 groupby 메서드



Pandas 의 꽃, 그룹별 연산을 위한 groupby 메서드

● **groupby 메서드**

○ 특정 변수의 값 별로 묶어서 각종 연산 가능

○ 수치형 변수의 통계 계산, 문자열 연산 등 가능

○ 사용자 정의 함수 사용 가능

```
import seaborn as sns

df = sns.load_dataset('titanic')
df.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

Pandas 의 꽃, 그룹별 연산을 위한 groupby 메서드

● **groupby 메서드**

○ 특정 변수의 값 별로 묶어서 각종 연산 가능

○ 수치형 변수의 통계 계산, 문자열 연산 등 가능

○ 사용자 정의 함수 사용 가능

```
df.groupby('sex')['survived'].mean()
```

```
sex
female    0.742038
male      0.188908
Name: survived, dtype: float64
```

```
df.groupby(['sex','class'])['survived'].mean()
```

```
sex     class
female  First     0.968085
        Second    0.921053
        Third     0.500000
male    First     0.368852
        Second    0.157407
        Third     0.135447
Name: survived, dtype: float64
```

```
df.groupby(['sex','class'])['survived'].agg(['mean','count'])
```

		mean	count
sex	**class**		
female	**First**	0.968085	94
	Second	0.921053	76
	Third	0.500000	144
male	**First**	0.368852	122
	Second	0.157407	108
	Third	0.135447	347

Pandas 의 꽃, 그룹별 연산을 위한 groupby 메서드

● **groupby 메서드**

○ 특정 변수의 값 별로 묶어서 각종 연산 가능

○ 수치형 변수의 통계 계산, 문자열 연산 등 가능

○ 사용자 정의 함수 사용 가능

```
# 변수 별 별도 통계 연산 가능
df.groupby(['sex','class'])[['survived','age']].agg({'survived':'mean', 'age':'max'})
```

		survived	age
sex	**class**		
female	**First**	0.968085	63.0
	Second	0.921053	57.0
	Third	0.500000	63.0
male	**First**	0.368852	80.0
	Second	0.157407	70.0
	Third	0.135447	74.0

Pandas 의 꽃, 그룹별 연산을 위한 groupby 메서드

● **groupby 메서드**

○ 특정 변수의 값 별로 묶어서 각종 연산 가능

○ 수치형 변수의 통계 계산, 문자열 연산 등 가능

○ 사용자 정의 함수 사용 가능

```
def get_IQR(data):
    _3rd = data.quantile(.75)
    _1st = data.quantile(.25)
    return (np.abs(_3rd - _1st) * 1.5)

df.groupby(['sex','class'])['age'].apply(get_IQR)
```

```
sex     class
female  First     31.5000
        Second    20.6250
        Third     23.4375
male    First     31.5000
        Second    20.6250
        Third     19.5000
Name: age, dtype: float64
```

Pandas 의 꽃, 그룹별 연산을 위한 groupby 메서드

● **groupby 메서드**

○ 특정 변수의 값 별로 묶어서 각종 연산 가능

○ 수치형 변수의 통계 계산, 문자열 연산 등 가능

○ 사용자 정의 함수 사용 가능

1

```
df = sns.load_dataset('penguins')
# 결측치 개수 확인
df.isna().sum()
```

```
species              0
island               0
bill_length_mm       2
bill_depth_mm        2
flipper_length_mm    2
body_mass_g          2
sex                 11
dtype: int64
```

2

```
# 그룹별 평균값 확인
df.groupby('species')[[
    'bill_length_mm','bill_depth_mm','flipper_length_mm','body_mass_g'
    ]].mean()
```

species	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
Adelie	38.791391	18.346358	189.953642	3700.662252
Chinstrap	48.833824	18.420588	195.823529	3733.088235
Gentoo	47.504878	14.982114	217.186992	5076.016260

3

```
# 결측치를 그룹별 평균값으로 대체
df.groupby('species')[[
    'bill_length_mm','bill_depth_mm','flipper_length_mm','body_mass_g'
    ]].apply(lambda x: x.fillna(x.mean()))
```

		bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
species					
	0	39.100000	18.700000	181.000000	3750.000000
	1	39.500000	17.400000	186.000000	3800.000000
Adelie	2	40.300000	18.000000	195.000000	3250.000000
	3	38.791391	18.346358	189.953642	3700.662252
	4	36.700000	19.300000	193.000000	3450.000000
...	...	...	...	...	...
	339	47.504878	14.982114	217.186992	5076.016260
	340	46.800000	14.300000	215.000000	4850.000000
Gentoo	341	50.400000	15.700000	222.000000	5750.000000
	342	45.200000	14.800000	212.000000	5200.000000
	343	49.900000	16.100000	213.000000	5400.000000

Pandas 의 꽃, 그룹별 연산을 위한 groupby 메서드

● **groupby 메서드**

○ 특정 변수의 값 별로 묶어서 각종 연산 가능

○ 수치형 변수의 통계 계산, 문자열 연산 등 가능

○ 사용자 정의 함수 사용 가능

```
df = pd.DataFrame(
    {'group':['A','A','A','B','B'],
    'value':[1, 1, 1, 10, 10]}
)
df
```

	group	value
0	A	1
1	A	1
2	A	1
3	B	10
4	B	10

```
df.groupby('group')['value'].sum()
```

```
group
A     3
B    20
Name: value, dtype: int64
```

```
# groupby 인자로 열 이름 이외 다른 형태의 데이터 전달
df.groupby([0,0,1,1,1])['value'].sum()
```

```
0     2
1    21
Name: value, dtype: int64
```

```
# groupby 인자로 열 이름 이외 다른 형태의 데이터 전달
s = pd.Series([False, False, True, True, True])
df.groupby(s)['value'].sum()
```

```
False     2
True     21
Name: value, dtype: int64
```

시계열 데이터의 특징

- **시계열 데이터**
  - 시간에 따라 변화하는 특성을 가지거나, 분석의 기준이 시간이 되는 경우
  - 시계열 행 슬라이싱 사용 가능
  - 특정 시간 단위로 묶어 연산을 수행하는 resample 메서드 사용 가능

```
df = pd.read_csv('datasets/APPL_price/APPL_price.csv')
df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1980-12-12	0.128348	0.128906	0.128348	0.128348	0.100178	469033600
1	1980-12-15	0.122210	0.122210	0.121652	0.121652	0.094952	175884800
2	1980-12-16	0.113281	0.113281	0.112723	0.112723	0.087983	105728000
3	1980-12-17	0.115513	0.116071	0.115513	0.115513	0.090160	86441600
4	1980-12-18	0.118862	0.119420	0.118862	0.118862	0.092774	73449600

```
df['Date'] = pd.to_datetime(df['Date'])
df = df.set_index('Date')
df.head()
```

Date	Open	High	Low	Close	Adj Close	Volume
1980-12-12	0.128348	0.128906	0.128348	0.128348	0.100178	469033600
1980-12-15	0.122210	0.122210	0.121652	0.121652	0.094952	175884800
1980-12-16	0.113281	0.113281	0.112723	0.112723	0.087983	105728000
1980-12-17	0.115513	0.116071	0.115513	0.115513	0.090160	86441600
1980-12-18	0.118862	0.119420	0.118862	0.118862	0.092774	73449600

시계열 데이터의 특징

- **시계열 데이터**
  - ○ 시간에 따라 변화하는 특성을 가지거나, 분석의 기준이 시간이 되는 경우
  - ○ 시계열 행 슬라이싱 사용 가능
  - ○ 특정 시간 단위로 묶어 연산을 수행하는 resample 메서드 사용 가능

```
df['1980-12-13':'1980-12-18']
```

Date	Open	High	Low	Close	Adj Close	Volume
1980-12-15	0.122210	0.122210	0.121652	0.121652	0.094952	175884800
1980-12-16	0.113281	0.113281	0.112723	0.112723	0.087983	105728000
1980-12-17	0.115513	0.116071	0.115513	0.115513	0.090160	86441600
1980-12-18	0.118862	0.119420	0.118862	0.118862	0.092774	73449600

```
df['2015-02':'2015-02']
```

Date	Open	High	Low	Close	Adj Close	Volume
2015-02-02	29.512501	29.792500	29.020000	29.657499	26.777473	250956400
2015-02-03	29.625000	29.772499	29.402500	29.662500	26.781982	207662800
2015-02-04	29.625000	30.127501	29.577499	29.889999	26.987389	280598800
2015-02-05	30.004999	30.057501	29.812500	29.985001	27.180010	168984800
2015-02-06	30.004999	30.062500	29.612499	29.732500	26.951132	174826400
2015-02-09	29.637501	29.959999	29.607500	29.930000	27.130157	155559200
2015-02-10	30.042500	30.537500	30.040001	30.504999	27.651367	248034000
2015-02-11	30.692499	31.230000	30.625000	31.219999	28.299484	294247200
2015-02-12	31.514999	31.870001	31.392500	31.615000	28.657528	297898000
2015-02-13	31.820000	31.820000	31.412500	31.770000	28.798038	217088800
2015-02-17	31.872499	32.220001	31.730000	31.957500	28.967993	252609600
2015-02-18	31.907499	32.195000	31.862499	32.180000	29.169676	179566800
2015-02-19	32.119999	32.257500	32.082500	32.112499	29.108494	149449600
2015-02-20	32.154999	32.375000	32.012501	32.375000	29.346434	195793600
2015-02-23	32.505001	33.250000	32.415001	33.250000	30.139582	283896400
2015-02-24	33.235001	33.400002	32.792500	33.042500	29.951496	276912400
2015-02-25	32.889999	32.900002	32.037498	32.197498	29.185539	298846800
2015-02-26	32.197498	32.717499	31.652500	32.605000	29.554924	365150000
2015-02-27	32.500000	32.642502	32.060001	32.115002	29.110762	248059200

시계열 데이터 그룹화를 위한 resample 메서드

● **resample 메서드**

○ datetime 데이터형이 인덱스인 경우 특정 날짜/시간 단위로 데이터 분절 후 연산 가능

○ offset string을 인자로 전달하여 그룹화 할 날짜/시간 단위 설정

○ closed, label 인자 등 사용 가능

```
df = pd.read_csv('datasets/APPL_price/APPL_price.csv')
df['Date'] = pd.to_datetime(df['Date'])
df = df.set_index('Date')

df.resample('7d').mean()
```

Date	Open	High	Low	Close	Adj Close	Volume
1980-12-12	0.119643	0.119978	0.119420	0.119420	0.093209	182107520.0
1980-12-19	0.135324	0.135882	0.135324	0.135324	0.105623	45236800.0
1980-12-26	0.157366	0.157645	0.157087	0.157087	0.122610	63341600.0
1981-01-02	0.144755	0.144978	0.144308	0.144308	0.112636	39612160.0
1981-01-09	0.139509	0.139955	0.139174	0.139174	0.108628	19322240.0
...	...	...	...	...	...	...
2022-05-20	138.701996	142.411996	136.613998	141.072000	141.072000	108473860.0
2022-05-27	148.047501	150.837502	146.659996	149.600002	149.600002	85332900.0
2022-06-03	146.788001	148.672000	144.690003	146.166003	146.166003	70260240.0
2022-06-10	134.529999	135.915997	132.235999	133.451999	133.451999	99617240.0
2022-06-17	130.070007	133.080002	129.809998	131.559998	131.559998	134118500.0

Offset string	의미
B	Business day
W	주
M	Month end
MS	Month begin
BMS	Business month begin
BM	Business month end
Q	Quarter end
QS	Quarter begin
A	Year end
D	One day
H	One hour
T	One minute
S	One second