In [3]:

```
import pandas as pd
import numpy as np

# csv 읽지 않고, 특정 df로 설정한다.
df = pd.DataFrame([
    [np.nan, 2, np.nan, 0],
    [3, 4, np.nan, 1],
    [np.nan, np.nan, np.nan, 5],
    [np.nan, 3, np.nan, 4]],
    columns = list('ABCD')
    )
df
```

Out[3]:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5
3	NaN	3.0	NaN	4

1. 결측치란 :

- 비어있는 값을 의미
- **csv** 파일에서 **NaN** 값으로 나오는 값을 의미한다.
- 공백이 아니라, 아예 없는 값을 의미

1.1 결측치 확인하기

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

- 로 확인한다.
- **df.isnull()** 함수는 각각 값이 **NaN** 혹은 **null** 값일 때 **true**를 반환한다.
- 따라서 **column**별 **null**의 개수를 확인할 수 있다.

In [5]:

```
# isnull
df.isnull()
```

Out[5]:

	A	B	C	D
0	True	False	True	False
1	False	False	True	False
2	True	True	True	False
3	True	False	True	False

In [7]:

```
df.isnull().sum() # 컬럼 단위
```

Out[7]:

```
A    3
```

```
B    1
C    4
D    0
dtype: int64
```

In [6]:

```
# info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   A       1 non-null      float64
 1   B       3 non-null      float64
 2   C       0 non-null      float64
 3   D       4 non-null      int64  
dtypes: float64(3), int64(1)
memory usage: 256.0 bytes
```

2. 결측치 처리하기

``` **df.dropna() :** 결측치가 존재하는 모든 행 삭제 **df.dropna(axis = 1):** 결측치가 존재하는 모든 열 삭제

In [8]:

```
# 결측치가 존재하는 모든 행 삭제
df.dropna()
# 모든 행 삭제 : 모든 행에 결측치가 존재한다.
```

Out[8]:

| | A | B | C | D |
|---|---|---|---|---|

In [10]:

```
# 결측치가 존재하는 모든 열 삭제
df.dropna(axis = 1)
```

Out[10]:

	D
0	0
1	1
2	5
3	4

3. 결측치 대치하기

1. 특정 값으로 채우기
   `df.fillna(특정 값)`

1. 주변 값으로 채우기

- 이전 값으로 채우기:
  `df.fillna(method="ffill")`
- 다음 값으로 채우기:
  `df.fillna(method="bfill")`

1. 컬럼 별로 값을 지정하여 치우기 : (딕셔너리를 매개변수로 제공)
```

`df.fillna({'컬럼명1' : 값1, '컬럼명2' : 값2, ,,,})`

3.1. 특정 값으로 채우기

`df.fillna(특정 값)`

In [11]:

```
# 0으로 채움
df.fillna(0)
```

Out[11]:

	A	B	C	D
0	0.0	2.0	0.0	0
1	3.0	4.0	0.0	1
2	0.0	0.0	0.0	5
3	0.0	3.0	0.0	4

In [15]:

```
# 평균 값으로 채우기
df.fillna(df.mean())
# 그러나 C열은 모두 NaN이라서 갱신되지 않는다.
```

Out[15]:

	A	B	C	D
0	3.0	2.0	NaN	0
1	3.0	4.0	NaN	1
2	3.0	3.0	NaN	5
3	3.0	3.0	NaN	4

3.2. 주변 값으로 채우기

- 이전 값으로 채우기:
  `df.fillna(method="ffill")`
- 다음 값으로 채우기:
  `df.fillna(method="bfill")`

In [16]:

```
# 이전 값으로 채우기
df.fillna(method = 'ffill')
# 0번째 행은 이전 값이 없기 때문에 null이 그대로 반영된다.
# C열은 각 행의 값이 모두 NaN이기 때문에 그대로 반영되어, 결측치가 처리되지 않는다.
```

Out[16]:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	3.0	4.0	NaN	5
3	3.0	3.0	NaN	4

In [17]:

```
df.fillna(method="bfill")
```

Out[17]:

	A	B	C	D
0	3.0	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	3.0	NaN	5
3	NaN	3.0	NaN	4

3.3. 컬럼 별로 대치할 값을 지정하여 채우기

- 딕셔너리를 매개변수로 제공 `df.fillna({'컬럼명1' : 값1, '컬럼명2' : 값2, ,,,})`

In [19]:

```
# {'A' : 0, 'B' : 1, 'C' : 2, 'D' : 3}
df.fillna({'A' : 0, 'B' : 1, 'C' : 2, 'D' : 3})
```

Out[19]:

	A	B	C	D
0	0.0	2.0	2.0	0
1	3.0	4.0	2.0	1
2	0.0	1.0	2.0	5
3	0.0	3.0	2.0	4

4. 결측치와 통계값

- 결측치는 없는 데이터로 간주한다.
- 통계값을 구할 때 데이터의 개수에 영향을 끼친다.

In [22]:

```
df
```

Out[22]:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5
3	NaN	3.0	NaN	4

In [20]:

```
df['A'].mean()
```

Out[20]:

```
3.0
```

In [26]:

```
df['A'].value_counts()
# 총 행의 개수는 4개인데, 값의 개수는 1개이다.
```

Out[26]:

```
3.0    1
Name: A, dtype: int64
```

In [23]:

```
df['B'].mean()
```

Out[23]:

```
3.0
```

In [25]:

```
df['B'].value_counts()
# 총 행의 개수는 4개인데, 값으로 세어지는 개수는 3개이다.
```

Out[25]:

```
2.0    1
4.0    1
3.0    1
Name: B, dtype: int64
```

In [27]:

```
df
```

Out[27]:

	A	B	C	D
0	**NaN**	**2.0**	**NaN**	**0**
1	**3.0**	**4.0**	**NaN**	**1**
2	**NaN**	**NaN**	**NaN**	**5**
3	**NaN**	**3.0**	**NaN**	**4**

In [29]:

```
# 결측치 확인
df.isnull().sum()
```

Out[29]:

```
A    3
B    1
C    4
D    0
dtype: int64
```

In [30]:

```
# scores.csv를 isnull 함수로 결측치 확인하기
df = pd.read_csv('./data/scores.csv')
df.isnull().sum()
```

Out[30]:

```
name    0
kor     3
eng     2
math    1
dtype: int64
```

In [32]:

```
# scores.csv를 info 함수로 결측치 확인하기
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 4 columns):
```

```
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   name    30 non-null     object
 1   kor     27 non-null     float64
 2   eng     28 non-null     float64
 3   math    29 non-null     float64
dtypes: float64(3), object(1)
memory usage: 1.1+ KB
```

In [33]:

```
# 결측치를 0으로 채우기
df = df.fillna(0)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   name    30 non-null     object
 1   kor     30 non-null     float64
 2   eng     30 non-null     float64
 3   math    30 non-null     float64
dtypes: float64(3), object(1)
memory usage: 1.1+ KB
```