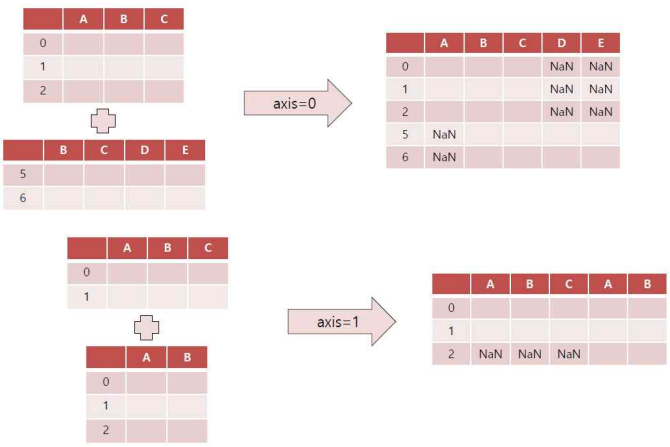
05. 데이터프레임 합치기

1. 여러 데이터프레임 연결

○ 데이터프레임 연결

- 서로 다른 데이터프레임들의 구성 형태와 속성이 균일하다면, 행 또는 열 중에 어느 한 방향으로 이어 붙여도 데이터의 일관성을 유지할 수 있음.
- 즉, 기존 데이터프레임의 형태를 유지하면서 이어 붙이는 개념으로 concat() 함수를 활용할 수 있음.
- concat() 함수를 사용하는 방법은 다음과 같음.

```
from pandas import concat
concat(데이터프레임의 리스트, ignore_index=False/True, axis=0/1, join='outer'/'inner')
```



- 연결하고자 하는 데이터프레임을 리스트형태로 입력하여 여러개의 데이터프레임을 연결함. 연결하고자 하는 방향은 axis 옵션을 통해 결정함. 0을 입력하면 행을 추가하는 형태로 연결하게 되고, 1을 입력하면 열을 추가하는 형태로 연결하게 됨.
- 열 이름은 join 옵션에 의해 결정되며, 기본인 outer를 입력하면 합집합의 형태로 데이터프레임을 연결하게 됨. 즉, 적어도 하나의 데이터프레임에 존재하는 열 이름이 있다면, 열 이름이 생성되고 데이터가 존재하지 않는 경우에는 NaN이 입력됨.
- 그리고 join 옵션에 inner를 입력하게 되면 교집합의 형태로 데이터프레임을 연결하게 됨. 즉, 모든 데이터프레임에 공통으로 속하는 열 이름의

경우에 열 이름이 생성되고 나머지는 삭제 됨.

- 데이터프레임이 결합될 때, 기본적으로 각 데이터프레임의 행 인덱스는 본래 형태를 유지함. 만약 결합한 후 새롭게 0부터 행 인덱스를 설정하고자 한다면 ignore_index 옵션에 True를 입력하여 새롭게 행 인덱스를 설정할 수 있음.
- 다음의 2개의 데이터프레임을 concat() 함수를 이용하여 하나의 데이터프레임으로 연결해보도록 하겠음.

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

In [2]: df1 = pd.DataFrame({'a': ['a0', 'a1', 'a2', 'a3'], 'b': ['b0', 'b1', 'b2', 'b3'],
   ...:                     'c': ['c0', 'c1', 'c2', 'c3']},
   ...:                    index=[0, 1, 2, 3])

In [3]: print(df1)
    a   b   c
0  a0  b0  c0
1  a1  b1  c1
2  a2  b2  c2
3  a3  b3  c3
```

```
In [4]: df2 = pd.DataFrame({'a': ['a2', 'a3', 'a4', 'a5'], 'b': ['b2', 'b3', 'b4', 'b5'],
   ...:                     'c': ['c2', 'c3', 'c4', 'c5'], 'd': ['d2', 'd3', 'd4', 'd5']},
   ...:                    index=[2, 3, 4, 5])

In [5]: print(df2)
    a   b   c   d
2  a2  b2  c2  d2
3  a3  b3  c3  d3
4  a4  b4  c4  d4
5  a5  b5  c5  d5
```

- 먼저 기본 옵션인 axis=0, ignore_index=False, join='outer'을 활용하여 두 데이터프레임을 결합하였음. 즉, 행을 추가하는 데이터프레임 연결을 실시함에 있어 합집합의 형태와 기존의 행 인덱스를 유지하는 방법으로 연결을 실시하였음.

```
In [6]: result1 = pd.concat([df1, df2])

In [7]: print(result1)
    a   b   c    d
0  a0  b0  c0  NaN
1  a1  b1  c1  NaN
2  a2  b2  c2  NaN
3  a3  b3  c3  NaN
2  a2  b2  c2   d2
3  a3  b3  c3   d3
4  a4  b4  c4   d4
5  a5  b5  c5   d5
```

- 그 결과 첫 번째 데이터프레임(df1)에 존재하지 않지만, 두 번째 데이터프레임(df2)에 존재하는 변수 d가 추가된 것을 확인할 수 있음. 이 때, 첫 번재 데이터프레임에는 변수 d가 존재하지 않으므로 변수 d의 데이터에는 NaN이 생성된 것을 확인할 수 있음. 그리고 기존의 행 인덱스를 유지하는 방법으로 연결을 실시하였으므로 기존의 행 인덱스가 유지되고 있는 것을 확인할 수 있음.
- 다음으로 join 옵션에 'inner'를 입력하여 공통적인 열 이름 요소만 연결하여 데이터프레임을 연결하도록 하겠음.

```
In [8]: result2 = pd.concat([df1, df2], join='inner')

In [9]: print(result2)
    a   b   c
0  a0  b0  c0
1  a1  b1  c1
2  a2  b2  c2
3  a3  b3  c3
2  a2  b2  c2
3  a3  b3  c3
4  a4  b4  c4
5  a5  b5  c5
```

- 그 결과 두 번째 데이터프레임(df2)에 존재하지만, 첫 번째 데이터프레임(df1)에 존재하지 않는 변수 d는 삭제된 것을 확인할 수 있음. 그리고 기존의 행 인덱스를 유지하는 방법으로 연결을 실시하였으므로 기존의 행 인덱스가 유지되고 있는 것을 확인할 수 있음.
- 다음으로 ignore_index 옵션에 True를 입력하여 기존의 행 인덱스를 무시하고 새롭게 행 인덱스를 설정하여 데이터프레임을 연결하도록 하겠음.

```
In [10]: result3 = pd.concat([df1, df2], ignore_index=True)

In [11]: print(result3)
    a   b   c    d
0  a0  b0  c0  NaN
1  a1  b1  c1  NaN
2  a2  b2  c2  NaN
3  a3  b3  c3  NaN
4  a2  b2  c2   d2
5  a3  b3  c3   d3
6  a4  b4  c4   d4
7  a5  b5  c5   d5
```

- 그 결과 기존의 행 인덱스가 삭제되고 새롭게 0부터 행 인덱스가 설정된 것을 확인할 수 있음.
- 다음으로 axis옵션에 1을 입력하여 열을 추가하는 형태로 두 데이터프레임을 연결하고자 함.

```
In [6]: result1 = pd.concat([df1, df2], axis=1)

In [7]: print(result1)
     a    b    c    a    b    c    d
0   a0   b0   c0  NaN  NaN  NaN  NaN
1   a1   b1   c1  NaN  NaN  NaN  NaN
2   a2   b2   c2   a2   b2   c2   d2
3   a3   b3   c3   a3   b3   c3   d3
4  NaN  NaN  NaN   a4   b4   c4   d4
5  NaN  NaN  NaN   a5   b5   c5   d5
```

- 그 결과 첫 번째 데이터프레임(df1)의 오른쪽에 두 번째 데이터프레임(df2)이 위치하는 것을 확인할 수 있음. 이 때, 두 데이터프레임이 공통으로 존재하는 행 인덱스인 2행과 3행에 데이터가 입력되고 첫 번째 데이터프레임에 존재하지 않는 행 인덱스인 4행과 5행에 NaN이 입력되고, 두 번째 데이터프레임에 존재하지 않는 행 인덱스인 0행과 1행에 NaN이 입력되어 있는 것을 확인할 수 있음.

- 다음으로 join 옵션에 'inner'를 입력하여 공통적인 행 인덱스 요소만 연결하여 데이터프레임을 연결하도록 하겠음.

```
In [8]: result2 = pd.concat([df1, df2], axis=1, join='inner')

In [9]: print(result2)
    a   b   c   a   b   c   d
2  a2  b2  c2  a2  b2  c2  d2
3  a3  b3  c3  a3  b3  c3  d3
```

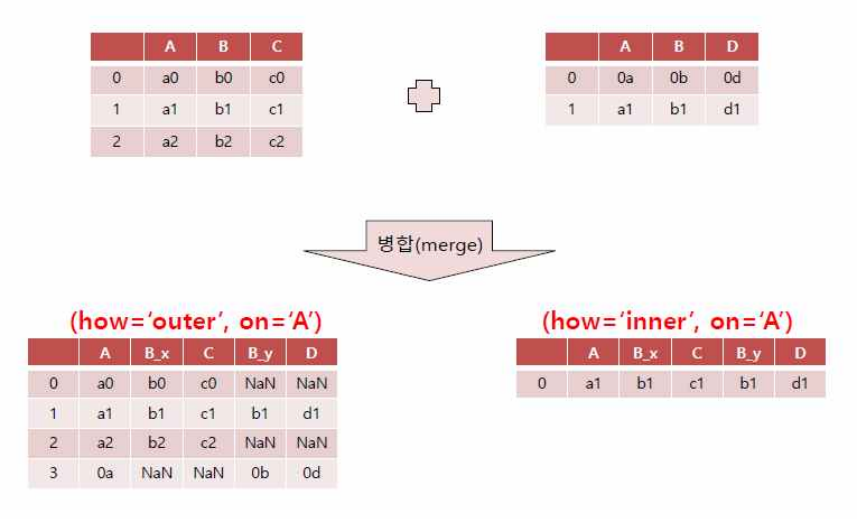
- 그 결과 첫 번째 데이터프레임(df1)과 두 번째 데이터프레임(df2)에 공통적으로 존재하는 행 인덱스인 2, 3인 경우만 데이터가 연결되어 있고 나머지 행 인덱스는 삭제된 것을 확인할 수 있음.

2. 여러 데이터프레임 병합

○ 데이터프레임 병합

- 앞에서 살펴본 concat() 함수가 여러 데이터프레임을 이어 붙이는 듯이 연결하는 함수라고 한다면, 데이터프레임 병합은 어떤 기준에 의해서 두 데이터프레임을 연결하는 것을 말함.
- 이 때, 사용되는 기준이 되는 열이나 인덱스를 키(key)라고 함. 주의해야 할 점은 키가 되는 열이나 인덱스는 반드시 병합하고자 하는 두 데이터프레임에 모두 존재해야 함.
- 두 데이터프레임을 병합하는 merge() 함수를 사용하는 방법은 다음과 같음.

```
from pandas import merge
merge(첫번째 데이터프레임, 두 번째 데이터프레임, on=기준변수,
how='inner'/'outer'/'left'/'right', left_on= , right_on= )
```



	A	B	C
0	a0	b0	c0
1	a1	b1	c1
2	a2	b2	c2

	A	B	D
0	0a	0b	0d
1	a1	b1	d1

	A	B_x	C	B_y	D
0	a0	b0	c0	NaN	NaN
1	a1	b1	c1	b1	d1
2	a2	b2	c2	NaN	NaN
3	0a	NaN	NaN	0b	0d

	A	B_x	C	B_y	D
0	a1	b1	c1	b1	d1

- 병합하고자 하는 두 데이터프레임을 차례로 입력하여 두 데이터프레임을 병합함. 병합에 기준이 되는 키 변수명은 on 옵션을 통해 결정함. 만약 두 데이터프레임의 키 변수명이 동일하지 않다면, 첫 번째 데이터프레임에 존재하는 키 변수명은 left_on 옵션에 입력하고, 두 번째 데이터프레임에 존재하는 키 변수명은 right_on 옵션에 입력함. 이 때, 키 변수명 외에 동일한 변수명이 두 데이터프레임에 존재하는 경우 변수명에 각각 _x, _y가 추가됨.
- 병합 방식은 how 옵션에 의해 결정되며, 기본인 inner를 입력하게 되면 교집합의 형태로 데이터프레임을 병합하게 됨. 즉, 두 데이터프레임에 키 변수의 데이터를 기준으로 두 데이터프레임에 동일한 데이터만 병합되고 나머지 데이터는 삭제 됨.
- 그리고 how 옵션에 outer를 입력하면 합집합의 형태로 두 데이터프레임을 병합하게 됨. 즉, 두 데이터프레임에 키 변수를 기준으로 적어도 하나의 키 변수의 데이터는 병합되고, 키 변수의 데이터에 다른 열 이름 데이터가 존재하지 않는 경우에는 NaN이 입력됨.
- 또한, how 옵션에 left를 입력하면, 첫 번째 데이터프레임의 키 변수에 속하는 데이터값을 기준으로 병합하게 되고, right를 입력하면 두 번째 데이터프레임의 키 변수에 속하는 데이터값을 기준으로 병합하게 됨.
- 다음의 2개의 데이터프레임을 merge() 함수를 이용하여 하나의 데이터프레임으로 병합해보도록 하겠음.

	A	B	C	D
1	id	name	eps	per
2	145990	삼양사	5741	14.28322592
3	161390	한국타이어	5648.5	7.453306187
4	185750	종근당	3990.333333	25.18586584
5	204210	모두투어리츠	85.16666667	40.80234834
6	207940	삼성바이오로직스	4644.166667	89.79005921
7				

	A	B	C	D	E
1	id	stock_name	price		
2	145990	삼양사	82000		
3	185750	종근당	100500		
4	192400	쿠쿠홀딩스	177500		
5	199800	툴젠	115400		
6	204210	모두투어리츠	3475		
7					

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

In [2]: df1 = pd.read_excel('C:/Users/LeeKJ/Desktop/price.xlsx')

In [3]: print(df1)
       id stock_name   price
0  145990       삼양사   82000
1  185750       종근당  100500
2  192400     쿠쿠홀딩스  177500
3  199800       툴젠  115400
4  204210     모두투어리츠    3475
```

```
In [4]: df2 = pd.read_excel('C:/Users/LeeKJ/Desktop/valuation.xlsx')

In [5]: print(df2)
       id       name         eps        per
0  145990       삼양사  5741.000000  14.283226
1  161390     한국타이어  5648.500000   7.453306
2  185750       종근당  3990.333333  25.185866
3  204210    모두투어리츠    85.166667  40.802348
4  207940  삼성바이오로직스  4644.166667  89.790059
```

- 먼저 기본 옵션인 how='inner'를 활용하여 두 데이터프레임을 병합하였음. 두 데이터프레임에 공통적으로 나타나있는 변수인 id 변수를 키 변수로 활용하기 위해 on 옵션에 id를 입력하였고, 두 데이터프레임 병합을 실시함에 있어 교집합의 방법으로 병합을 실시하였음.

```
In [6]: merge_inner = pd.merge(df1, df2, on='id')

In [7]: print(merge_inner)
       id stock_name   price    name         eps        per
0  145990       삼양사   82000    삼양사  5741.000000  14.283226
1  185750       종근당  100500    종근당  3990.333333  25.185866
2  204210    모두투어리츠    3475  모두투어리츠    85.166667  40.802348
```

- 그 결과 첫 번째 데이터프레임(df1)과 두 번째 데이터프레임에 모두 존재하는 id 145990, 185750, 204210의 데이터가 병합된 것을 확인할 수 있음. 첫 번재 데이터프레임의 열 이름인 id, stock_name, price와 두 번째 데이터프레임의 열 이름인 name, eps, per가 병합되어 하나의 데이터프레임으로 나타난 것을 확인할 수 있음. 그리고 첫 번째 데이터프레임과 두 번째 데이터프레임에 공통적으로 나타나지 않는 id 192400, 199800, 161390, 207940은 삭제된 것을 확인할 수 있음.
- 다음으로 how 옵션에 'outer'를 입력하여 두 데이터프레임을 병합하였음. 두 데이터프레임에 공통적으로 나타나있는 변수인 id 변수를 키 변수로 활용하기 위해 on 옵션에 id를 입력하였고, 두 데이터프레임 병합을 실시함에 있어 합집합의 방법으로 병합을 실시하였음.

```
In [8]: merge_outer = pd.merge(df1, df2, how='outer', on='id')

In [9]: print(merge_outer)
       id stock_name     price      name          eps        per
0  145990       삼양사   82000.0      삼양사  5741.000000  14.283226
1  185750       종근당  100500.0      종근당  3990.333333  25.185866
2  192400      쿠쿠홀딩스  177500.0       NaN          NaN        NaN
3  199800        툴젠  115400.0       NaN          NaN        NaN
4  204210     모두투어리츠    3475.0    모두투어리츠    85.166667  40.802348
5  161390       NaN       NaN     한국타이어  5648.500000   7.453306
6  207940       NaN       NaN  삼성바이오로직스  4644.166667  89.790059
```

- 그 결과 두 번째 데이터프레임(df2)에 존재하지만, 첫 번째 데이터프레임(df1)에 존재하지 않는 변수 d는 삭제된 것을 확인할 수 있음. 그리고 기존의 행 인덱스를 유지하는 방법으로 연결을 실시하였으므로 기존의 행 인덱스가 유지되고 있는 것을 확인할 수 있음.
- 그 결과 첫 번째 데이터프레임(df1)과 두 번째 데이터프레임에 모두 존재하는 id 145990, 185750, 204210의 데이터뿐만 아니라 첫 번째 데이터프레임에만 존재하는 id 192400, 199800과 두 번째 데이터프레임에만 존재하는 id 161390, 207940 역시 병합된 것을 확인할 수 있음. inner 옵션을 사용했을 때와 마찬가지로 첫 번째 데이터프레임의 열 이름인 id, stock_name, price와 두 번째 데이터프레임의 열 이름인 name, eps, per가 병합되어 하나의 데이터프레임으로 나타난 것을 확인할 수 있음. 그리고 첫 번째 데이터프레임에만 존재하는 id의 값이 병합되었을 때 두 번째 데이터프레임의 값은 모두 NaN이 입력된 것을 확인할 수 있고, 두 번째 데이터프레임에만 존재하는 id의 값이 병합되었을 때 첫 번째 데이터프레임의 값은 모두 NaN이 입력된 것을 확인할 수 있음.
- 다음으로 left_on과 right_on 옵션을 활용하여 두 데이터프레임에 공통적으로 존재하지 않는 열 이름을 키 변수로 활용하여 두 데이터프레임을 병합하도록 하겠음. 즉, 첫 번째 데이터프레임에 존재하는 stock_name과 두 번째 데이터프레임에 존재하는 name이 모두 회사이름을 나타내는 변수명이므로 두 변수를 활용하여 병합하도록 하겠음.

```
In [10]: merge_left = pd.merge(df1, df2, how='inner', left_on='stock_name', right_on='name')

In [11]: print(merge_left)
     id_x stock_name   price    id_y    name          eps        per
0  145990       삼양사   82000  145990    삼양사  5741.000000  14.283226
1  185750       종근당  100500  185750    종근당  3990.333333  25.185866
2  204210     모두투어리츠    3475  204210  모두투어리츠    85.166667  40.802348
```

- 그 결과 첫 번째 데이터프레임과 두 번째 데이터프레임에 모두 존재하는

id 145990, 185750, 204210의 데이터가 병합된 것을 확인할 수 있음. 그리고 첫 번째 데이터프레임과 두 번째 데이터프레임에 모두 id라는 열 이름이 존재하므로 첫 번째 데이터프레임의 id는 id_x로 변경된 것을 확인할 수 있고, 두 번째 데이터프레임의 id는 id_y로 변경된 것을 확인할 수 있음.

- 다음으로 how 옵션에 'left'을 입력하여 첫 번째 데이터프레임의 키 열에 속하는 데이터값을 기준으로 두 데이터프레임을 병합하고자 함.

```
In [12]: merge_left = pd.merge(df1, df2, how='left', left_on='stock_name', right_on='name')

In [13]: print(merge_left)
     id_x stock_name   price      id_y    name          eps        per
0  145990       삼양사   82000  145990.0     삼양사  5741.000000  14.283226
1  185750       종근당  100500  185750.0     종근당  3990.333333  25.185866
2  192400      쿠쿠홀딩스  177500       NaN     NaN          NaN        NaN
3  199800        툴젠  115400       NaN     NaN          NaN        NaN
4  204210     모두투어리츠    3475  204210.0  모두투어리츠    85.166667  40.802348
```

- 그 결과 첫 번째 데이터프레임(df1)의 키 변수 (stock_name)에 존재하는 값을 기준으로 두 번째 데이터프레임의 데이터가 병합된 것을 확인할 수 있음. 이 때, 합집합의 형태로 두 데이터프레임을 결합하는 방식과는 달리 첫 번째 데이터프레임에 존재하는 데이터를 기준으로 공통되는 키 변수의 값을 가지는 경우는 병합하고 그렇지 않은 경우는 NaN으로 입력되어 병합되는 것을 확인할 수 있음.
- 다음으로 how 옵션에 'right'을 입력하여 두 번째 데이터프레임의 키 열에 속하는 데이터값을 기준으로 두 데이터프레임을 병합하고자 함.

```
In [14]: merge_right = pd.merge(df1, df2, how='right', left_on='stock_name', right_on='name')

In [15]: print(merge_right)
       id_x stock_name     price    id_y      name          eps        per
0  145990.0       삼양사   82000.0  145990      삼양사  5741.000000  14.283226
1       NaN        NaN       NaN  161390    한국타이어  5648.500000   7.453306
2  185750.0       종근당  100500.0  185750      종근당  3990.333333  25.185866
3  204210.0     모두투어리츠    3475.0  204210    모두투어리츠    85.166667  40.802348
4       NaN        NaN       NaN  207940  삼성바이오로직스  4644.166667  89.790059
```

- 그 결과 두 번째 데이터프레임(df2)의 키 변수 (name)에 존재하는 값을 기준으로 첫 번째 데이터프레임의 데이터가 병합된 것을 확인할 수 있음. 이 때, 합집합의 형태로 두 데이터프레임을 결합하는 방식과는 달리 두 번째 데이터프레임에 존재하는 데이터를 기준으로 공통되는 키 변수의 값을 가지는 경우는 병합하고 그렇지 않은 경우는 NaN으로 입력되

어 병합되는 것을 확인할 수 있음.

3. 여러 데이터프레임 결합

○ 데이터프레임 결합

- 데이터프레임 결합은 데이터프레임 병합과 비슷한 방법임. 데이터프레임 병합은 기준 변수인 키 변수를 기준으로 병합하는 방법인 반면, 데이터프레임 결합은 행 인덱스를 기준으로 결합하는 방법으로 차이가 있음.
- 하지만 데이터프레임 결합에서도 옵션 설정에 따라 데이터프레임 병합과 동일하게 적용할 수 있음.
- 두 데이터프레임을 결합하는 join() 함수를 사용하는 방법은 다음과 같음.

첫 번째 데이터프레임.join(두 번째 데이터프레임, how='left'/'right'/'inner'/'outer', on=기준변수)

- join() 함수는 첫 번째 데이터프레임에 두 번째 데이터프레임을 결합시키는 것이 기본적인 형태임. 즉, how 옵션에 left가 기본적으로 지정되어져 있는 형태임. 이를 두 번째 데이터프레임에 첫 번째 데이터프레임을 결합하고자 한다면 how 옵션에 right를 지정할 수 있음. 또한 두 데이터프레임에 공통적으로 존재하는 행 인덱스를 기준으로 결합하고자 한다면 how 옵션에 inner를 입력하고, 합집합의 형태로 결합하고자 한다면 how 옵션에 outer를 입력할 수 있음.
- 그리고 행 인덱스를 기준으로 하지 않고 다른 변수명을 키 변수로 활용하고자 한다면 on 옵션에 키 변수명을 입력하여 결합할 수 있음.
- 다음의 2개의 데이터프레임을 join() 함수를 이용하여 하나의 데이터프레임으로 결합해보도록 하겠음.

	A	B	C	D	E
1	id	name	eps	per	
2	145990	삼양사	5741	14.28322592	
3	161390	한국타이어	5648.5	7.453306187	
4	185750	종근당	3990.333333	25.18586584	
5	204210	모두투어리츠	85.16666667	40.80234834	
6	207940	삼성바이오로직스	4644.166667	89.79005921	
7					

	A	B	C	D	E
1	id	stock_name	price		
2	145990	삼양사	82000		
3	185750	종근당	100500		
4	192400	쿠쿠홀딩스	177500		
5	199800	툴젠	115400		
6	204210	모두투어리츠	3475		
7					

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

In [2]: df1 = pd.read_excel('C:/Users/LeeKJ/Desktop/price.xlsx')
   ...: df1.set_index('id', inplace=True)

In [3]: print(df1)
       stock_name   price
id
145990        삼양사   82000
185750        종근당  100500
192400      쿠쿠홀딩스  177500
199800         톨젠  115400
204210     모두투어리츠    3475
```

```
In [4]: df2 = pd.read_excel('C:/Users/LeeKJ/Desktop/valuation.xlsx')
   ...: df2.set_index('id', inplace=True)

In [5]: print(df2)
              name         eps        per
id
145990        삼양사  5741.000000  14.283226
161390     한국타이어  5648.500000   7.453306
185750        종근당  3990.333333  25.185866
204210     모두투어리츠    85.166667  40.802348
207940  삼성바이오로직스  4644.166667  89.790059
```

- 먼저 두 데이터프레임의 행 인덱스를 생성하기 위하여 set_index() 함수를 활용하여 id 변수를 행 인덱스로 지정하였음.
- 다음으로 기본 옵션인 how='left'를 활용하여 첫 번째 데이터프레임을 기준으로 하여 두 번째 데이터프레임이 결합하였음.

```
In [6]: join_left = df1.join(df2)

In [7]: print(join_left)
       stock_name   price    name          eps        per
id
145990        삼양사   82000    삼양사  5741.000000  14.283226
185750        종근당  100500    종근당  3990.333333  25.185866
192400      쿠쿠홀딩스  177500     NaN          NaN        NaN
199800         톨젠  115400     NaN          NaN        NaN
204210     모두투어리츠    3475  모두투어리츠    85.166667  40.802348
```

- 그 결과 첫 번째 데이터프레임(df1)의 행 인덱스에 존재하는 값을 기준으로 두 번째 데이터프레임의 데이터가 결합된 것을 확인할 수 있음.
- 다음으로 how 옵션에 'right'을 입력하여 두 번째 데이터프레임을 기준으로 하여 첫 번째 데이터프레임이 결합하였음.

```
In [8]: merge_right = df1.join(df2, how='right')

In [9]: print(merge_right)
       stock_name     price      name          eps        per
id
145990        삼양사   82000.0      삼양사  5741.000000  14.283226
161390        NaN       NaN   한국타이어  5648.500000   7.453306
185750        종근당  100500.0      종근당  3990.333333  25.185866
204210     모두투어리츠    3475.0   모두투어리츠    85.166667  40.802348
207940        NaN       NaN  삼성바이오로직스  4644.166667  89.790059
```

- 그 결과 두 번째 데이터프레임(df2)의 행 인덱스에 존재하는 값을 기준으로 첫 번째 데이터프레임의 데이터가 결합된 것을 확인할 수 있음.
- 다음으로 how 옵션에 'inner'를 활용하여 첫 번째 데이터프레임을 기준으로 하여 두 번째 데이터프레임이 결합하였음. 즉, 두 데이터프레임에 공통적으로 나타나있는 행 인덱스를 이용하여 결합 함에 있어 교집합의

방법으로 실시하였음.

```
In [10]: join_inner = df1.join(df2, how='inner')

In [11]: print(join_inner)
       stock_name   price    name          eps         per
id
145990        삼양사   82000     삼양사  5741.000000  14.283226
185750        종근당  100500     종근당  3990.333333  25.185866
204210     모두투어리츠    3475  모두투어리츠    85.166667  40.802348
```

- 그 결과 첫 번째 데이터프레임(df1)과 두 번째 데이터프레임에 모두 존재하는 행 인덱스 값인 145990, 185750, 204210의 데이터가 결합된 것을 확인할 수 있음.
- 다음으로 how 옵션에 'outer'를 활용하여 첫 번째 데이터프레임을 기준으로 하여 두 번째 데이터프레임이 결합하였음. 즉, 두 데이터프레임에 모두 나타나 있는 행 인덱스를 이용하여 결합 함에 있어 합집합의 방법으로 실시하였음.

```
In [12]: join_outer = df1.join(df2, how='outer')

In [13]: print(join_outer)
       stock_name     price      name          eps         per
id
145990        삼양사   82000.0      삼양사  5741.000000  14.283226
161390        NaN       NaN    한국타이어  5648.500000   7.453306
185750        종근당  100500.0      종근당  3990.333333  25.185866
192400      쿠쿠홀딩스  177500.0       NaN          NaN         NaN
199800         툴젠  115400.0       NaN          NaN         NaN
204210     모두투어리츠    3475.0  모두투어리츠    85.166667  40.802348
207940        NaN       NaN  삼성바이오로직스  4644.166667  89.790059
```

- 그 결과 첫 번째 데이터프레임(df1)과 두 번째 데이터프레임에 존재하는 모든 행 인덱스 값인 145990, 185750, 192400, 199800, 204210, 161390, 207940의 데이터가 결합된 것을 확인할 수 있음.

4. 여러 데이터프레임의 연결/병합/결합 실습

※ 여러 데이터프레임의 연결

- 다음의 자료는 total_bill, tip, sex, smoker, day, time, size의 자료가 기록된 ex_1.xlsx와 total_bill, tip, sex, smoker, size의 자료가 기록된 ex_2.xlsx임.
- ex_1.xlsx는 143명의 자료를 기록하였고, ex_2.xlsx는 그 이후 101명의 자료를 기록한 데이터임.
- 다음의 2개의 자료를 하나의 데이터프레임으로 만들어 보세요.

	A	B	C	D	E	F	G
1	total_bill	tip	sex	smoker	day	time	size
2	16.99	1.01	Female	No	Sun	Dinner	2
3	10.34	1.66	Male	No	Sun	Dinner	3
4	21.01	3.5	Male	No	Sun	Dinner	3
5	23.68	3.31	Male	No	Sun	Dinner	2
6	24.59	3.61	Female	No	Sun	Dinner	4
7	25.29	4.71	Male	No	Sun	Dinner	4
8	8.77	2	Male	No	Sun	Dinner	2
9	26.88	3.12	Male	No	Sun	Dinner	4
10	15.04	1.96	Male	No	Sun	Dinner	2
11	14.78	3.23	Male	No	Sun	Dinner	2
12	10.27	1.71	Male	No	Sun	Dinner	2
13	35.26	5	Female	No	Sun	Dinner	4
14	15.42	1.57	Male	No	Sun	Dinner	2
15	18.43	3	Male	No	Sun	Dinner	4

	A	B	C	D	E	F	G
1	total_bill	tip	sex	smoker	size		
2	27.05	5	Female	No	6		
3	16.43	2.3	Female	No	2		
4	8.35	1.5	Female	No	2		
5	18.64	1.36	Female	No	3		
6	11.87	1.63	Female	No	2		
7	9.78	1.73	Male	No	2		
8	7.51	2	Male	No	2		
9	14.07	2.5	Male	No	2		
10	13.13	2	Male	No	2		
11	17.26	2.74	Male	No	3		
12	24.55	2	Male	No	4		
13	19.77	2	Male	No	4		
14	29.85	5.14	Female	No	5		
15	48.17	5	Male	No	6		

- 먼저 ex_1.xlsx와 ex_2.xlsx 자료를 먼저 데이터프레임으로 각각 df1과 df2라는 객체이름으로 불러와 보도록 하겠음.

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

In [2]: df1 = pd.read_excel('C:/Users/LeeKJ/Desktop/ex_1.xlsx')

In [3]: print(df1)
     total_bill   tip     sex smoker   day    time  size
0         16.99  1.01  Female     No   Sun  Dinner     2
1         10.34  1.66    Male     No   Sun  Dinner     3
2         21.01  3.50    Male     No   Sun  Dinner     3
3         23.68  3.31    Male     No   Sun  Dinner     2
4         24.59  3.61  Female     No   Sun  Dinner     4
..          ...   ...     ...    ...   ...     ...   ...
138       16.00  2.00    Male    Yes  Thur   Lunch     2
139       13.16  2.75  Female     No  Thur   Lunch     2
140       17.47  3.50  Female     No  Thur   Lunch     2
141       34.30  6.70    Male     No  Thur   Lunch     6
142       41.19  5.00    Male     No  Thur   Lunch     5

[143 rows x 7 columns]
```

```
In [4]: df2 = pd.read_excel('C:/Users/LeeKJ/Desktop/ex_2.xlsx')

In [5]: print(df2)
     total_bill   tip     sex smoker  size
0         27.05  5.00  Female     No     6
1         16.43  2.30  Female     No     2
2          8.35  1.50  Female     No     2
3         18.64  1.36  Female     No     3
4         11.87  1.63  Female     No     2
..          ...   ...     ...    ...   ...
96        29.03  5.92    Male     No     3
97        27.18  2.00  Female    Yes     2
98        22.67  2.00    Male    Yes     2
99        17.82  1.75    Male     No     2
100       18.78  3.00  Female     No     2

[101 rows x 5 columns]
```

- 먼저 기본 옵션인 axis=0, ignore_index=False, join='outer'을 활용하여 두 데이터프레임을 결합하였음. 즉, 행을 추가하는 데이터프레임 연결을 실시함에 있어 합집합의 형태와 기존의 행 인덱스를 유지하는 방법으로 연결을 실시하였음.

```
In [6]: result1 = pd.concat([df1, df2])

In [7]: print(result1)
     total_bill   tip     sex smoker  day    time  size
0         16.99  1.01  Female     No  Sun  Dinner     2
1         10.34  1.66    Male     No  Sun  Dinner     3
2         21.01  3.50    Male     No  Sun  Dinner     3
3         23.68  3.31    Male     No  Sun  Dinner     2
4         24.59  3.61  Female     No  Sun  Dinner     4
..          ...   ...     ...    ...  ...     ...   ...
96        29.03  5.92    Male     No  NaN     NaN     3
97        27.18  2.00  Female    Yes  NaN     NaN     2
98        22.67  2.00    Male    Yes  NaN     NaN     2
99        17.82  1.75    Male     No  NaN     NaN     2
100       18.78  3.00  Female     No  NaN     NaN     2

[244 rows x 7 columns]
```

- 그 결과 첫 번째 데이터프레임(df1)에 존재하지만, 두 번째 데이터프레임(df2)에 존재하지 않는 변수 day와 time가 두 번째 데이터 프레임에서 추가되어 연결된 것을 확인할 수 있음. 이 때, 두 번재 데이터프레임에는 변수 day, time가 존재하지 않으므로 변수 day와 time의 데이터에는 NaN이 생성된 것을 확인할 수 있음.
- 그리고 기존의 행 인덱스를 유지하는 방법으로 연결을 실시하였으므로 기존의 행 인덱스가 유지되어 244개의 rows가 있다고 되어 있지만 인덱스가 100으로 끝나는 것을 확인할 수 있음.
- 따라서 ignore_index 옵션에 True를 입력하여 기존의 인덱스를 무시하고 새롭게 설정하도록 하겠음.

```
In [8]: result2 = pd.concat([df1, df2], ignore_index=True)

In [9]: print(result2)
     total_bill   tip     sex smoker  day    time  size
0         16.99  1.01  Female     No  Sun  Dinner     2
1         10.34  1.66    Male     No  Sun  Dinner     3
2         21.01  3.50    Male     No  Sun  Dinner     3
3         23.68  3.31    Male     No  Sun  Dinner     2
4         24.59  3.61  Female     No  Sun  Dinner     4
..          ...   ...     ...    ...  ...     ...   ...
239       29.03  5.92    Male     No  NaN     NaN     3
240       27.18  2.00  Female    Yes  NaN     NaN     2
241       22.67  2.00    Male    Yes  NaN     NaN     2
242       17.82  1.75    Male     No  NaN     NaN     2
243       18.78  3.00  Female     No  NaN     NaN     2

[244 rows x 7 columns]
```

- 그 결과 기존의 행 인덱스가 삭제되고 새롭게 0부터 행 인덱스가 설정된 것을 확인할 수 있음.

- 다음으로 두 데이터 프레임에 공통적으로만 존재하는 변수인 total_bill, tip, sex, smoker, size 데이터만 연결해 보도록 하겠음. 즉, join 옵션에 'inner'를 입력하여 행을 추가하는 데이터프레임 연결을 실시함에 있어 교집합의 형태와 기존의 행 인덱스를 유지하는 방법으로 연결을 실시하였음.

```
In [10]: result3 = pd.concat([df1, df2], ignore_index=True, join='inner')

In [11]: print(result3)
     total_bill   tip     sex smoker  size
0         16.99  1.01  Female     No     2
1         10.34  1.66    Male     No     3
2         21.01  3.50    Male     No     3
3         23.68  3.31    Male     No     2
4         24.59  3.61  Female     No     4
..          ...   ...     ...    ...   ...
239       29.03  5.92    Male     No     3
240       27.18  2.00  Female    Yes     2
241       22.67  2.00    Male    Yes     2
242       17.82  1.75    Male     No     2
243       18.78  3.00  Female     No     2

[244 rows x 5 columns]
```

- 그 결과 day와 time 변수는 사라지고 공통적으로만 존재하는 변수인 total_bill, tip, sex, smoker, size만 존재하는 것을 확인할 수 있음.

※ 여러 데이터프레임의 병합

- 다음의 자료는 공통의 ID를 통해서 ex1_1.xlsx에서는 method, number, orbital_period를 측정하여 기록한 자료이고, ex1_2.xlsx는 mass, distance, year를 측정하여 기록한 자료임.
- 다음의 2개의 자료를 불러와 하나의 데이터프레임으로 만든 후, mass와 distance의 관측값에 결측이 존재하는 경우 삭제하세요.

	A	B	C	D	E
1	ID	method	number	orbital_period	
2	0	Radial Velocity	1	269.3	
3	1	Radial Velocity	1	874.774	
4	2	Radial Velocity	1	763	
5	3	Radial Velocity	1	326.03	
6	4	Radial Velocity	1	516.22	
7	5	Radial Velocity	1	185.84	
8	6	Radial Velocity	1	1773.4	
9	7	Radial Velocity	1	798.5	
10	8	Radial Velocity	1	993.3	
11	9	Radial Velocity	2	452.8	

	A	B	C	D	E	F	G
1	ID	mass	distance	year			
2	0	7.1	77.4	2006			
3	1	2.21	56.95	2008			
4	2	2.6	19.84	2011			
5	3	19.4	110.62	2007			
6	4	10.5	119.47	2009			
7	5	4.8	76.39	2008			
8	6	4.64	18.15	2002			
9	7		21.41	1996			

- 먼저 ex1_1.xlsx와 ex1_2.xlsx 자료를 먼저 데이터프레임으로 각각 df1과 df2라는 객체이름으로 불러와 보도록 하겠음.

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

In [2]: df1 = pd.read_excel('C:/Users/LeeKJ/Desktop/ex1_1.xlsx')

In [3]: print(df1)
        ID          method  number  orbital_period
0        0  Radial Velocity       1      269.300000
1        1  Radial Velocity       1      874.774000
2        2  Radial Velocity       1      763.000000
3        3  Radial Velocity       1      326.030000
4        4  Radial Velocity       1      516.220000
...    ...             ...     ...             ...
1030  1030         Transit       1        3.941507
1031  1031         Transit       1        2.615864
1032  1032         Transit       1        3.191524
1033  1033         Transit       1        4.125083
1034  1034         Transit       1        4.187757

[1035 rows x 4 columns]
```

```
In [4]: df2 = pd.read_excel('C:/Users/LeeKJ/Desktop/ex1_2.xlsx')

In [5]: print(df2)
        ID   mass  distance  year
0        0   7.10     77.40  2006
1        1   2.21     56.95  2008
2        4  10.50    119.47  2009
3        5   4.80     76.39  2008
4        6   4.64     18.15  2002
...    ...    ...       ...   ...
1026  1028    NaN   3200.00  2012
1027  1029    NaN     10.10  2012
1028  1030    NaN    172.00  2006
1029  1031    NaN    148.00  2007
1030  1034    NaN    260.00  2008

[1031 rows x 4 columns]
```

- 두 자료를 살펴보면 df1의 자료에서는 1035개의 rows를 가지고 있는 반

면 df2의 자료에서는 1031개의 rows를 가지고 있는 것을 알 수 있음. 따라서 데이터를 살펴보면 df1의 자료에서는 ID가 2, 3, 1032, 1033인 자료가 존재하는 반면 df2에서는 ID가 2, 3, 1032, 1033인 자료가 존재하지 않는 것을 확인할 수 있음.

- 따라서 두 데이터셋을 병합하는 방법 중 두 데이터셋에 동시에 존재하는 ID에 대해서만 병합을 먼저 실시하도록 하겠음.
- 두 데이터프레임에 공통적으로 나타나있는 변수인 ID 변수를 키 변수로 활용하기 위해 on 옵션에 ID를 입력하였고, how='inner'를 활용하여 교집합의 형태로 두 데이터프레임을 병합하였음.

```
In [6]: merge_inner = pd.merge(df1, df2, how='inner', on='ID')

In [7]: print(merge_inner)
        ID           method  number  orbital_period   mass  distance  year
0        0  Radial Velocity       1      269.300000   7.10     77.40  2006
1        1  Radial Velocity       1      874.774000   2.21     56.95  2008
2        4  Radial Velocity       1      516.220000  10.50    119.47  2009
3        5  Radial Velocity       1      185.840000   4.80     76.39  2008
4        6  Radial Velocity       1     1773.400000   4.64     18.15  2002
...    ...              ...     ...             ...    ...       ...   ...
1026  1028          Transit       1        3.352057    NaN   3200.00  2012
1027  1029          Imaging       1             NaN    NaN     10.10  2012
1028  1030          Transit       1        3.941507    NaN    172.00  2006
1029  1031          Transit       1        2.615864    NaN    148.00  2007
1030  1034          Transit       1        4.187757    NaN    260.00  2008

[1031 rows x 7 columns]
```

- 그 결과 ID가 2, 3, 1032, 1033을 제외한 1031개의 rows가 병합된 것을 확인할 수 있음.
- 그리고 mass와 distance의 결측자료값이 존재하는지를 확인하기 위해 info() 함수를 활용하여 결합된 데이터셋을 확인해 보았습니다.

```
In [8]: merge_inner.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1031 entries, 0 to 1030
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              1031 non-null   int64
 1   method          1031 non-null   object
 2   number          1031 non-null   int64
 3   orbital_period  988 non-null    float64
 4   mass            511 non-null    float64
 5   distance        804 non-null    float64
 6   year            1031 non-null   int64
dtypes: float64(3), int64(3), object(1)
memory usage: 64.4+ KB
```

- 그 결과 mass에는 511개의 유효데이터가, distance에는 804개의 유효데이터가 존재하는 것을 알 수 있음.
- 따라서 mass와 distance에 존재하는 결측 데이터를 삭제하기 위해 dropna() 함수를 활용하여 결측 데이터를 삭제하도록 하겠음.

- dropna() 함수의 subset 옵션에 기준이 되는 두 변수명 mass와 distance를 입력하고 두 변수 중 하나라도 결측 데이터가 존재하면 삭제하기 위해 how 옵션에 any를 입력하였음.

```
In [9]: df = merge_inner.dropna(subset=['mass', 'distance'], axis=0, how='any')

In [10]: df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 496 entries, 0 to 782
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              496 non-null    int64
 1   method          496 non-null    object
 2   number          496 non-null    int64
 3   orbital_period  496 non-null    float64
 4   mass            496 non-null    float64
 5   distance        496 non-null    float64
 6   year            496 non-null    int64
dtypes: float64(3), int64(3), object(1)
memory usage: 31.0+ KB
```

- 그 결과 총 496개의 유효 데이터만 존재하는 것을 확인할 수 있음.

- 다음으로 두 데이터셋을 병합하는 방법 중 두 데이터셋에 하나라도 존재하는 ID에 대해서 모두 병합을 실시하도록 하겠음.
- 두 데이터프레임에 공통적으로 나타나있는 변수인 ID 변수를 키 변수로 활용하기 위해 on 옵션에 ID를 입력하였고, how='outer'를 활용하여 합집합의 형태로 두 데이터프레임을 병합하였음.

```
In [11]: merge_outer = pd.merge(df1, df2, how='outer', on='ID')

In [12]: print(merge_outer)
        ID           method  number  orbital_period   mass  distance    year
0        0  Radial Velocity       1      269.300000   7.10     77.40  2006.0
1        1  Radial Velocity       1      874.774000   2.21     56.95  2008.0
2        2  Radial Velocity       1      763.000000    NaN       NaN     NaN
3        3  Radial Velocity       1      326.030000    NaN       NaN     NaN
4        4  Radial Velocity       1      516.220000  10.50    119.47  2009.0
...    ...              ...     ...             ...    ...       ...     ...
1030  1030          Transit       1        3.941507    NaN    172.00  2006.0
1031  1031          Transit       1        2.615864    NaN    148.00  2007.0
1032  1032          Transit       1        3.191524    NaN       NaN     NaN
1033  1033          Transit       1        4.125083    NaN       NaN     NaN
1034  1034          Transit       1        4.187757    NaN    260.00  2008.0

[1035 rows x 7 columns]
```

- 그 결과 ID가 2, 3, 1032, 1033의 경우 mass, distance, year에서 모두 결측의 형태로 자료가 병합되어 총 1035개의 rows가 병합된 것을 확인할 수 있음.
- 그리고 mass와 distance의 결측자료값이 존재하는지를 확인하기 위해 info() 함수를 활용하여 결합된 데이터셋을 확인해 보았습니다.

```
In [13]: merge_outer.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1035 entries, 0 to 1034
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              1035 non-null   int64
 1   method          1035 non-null   object
 2   number          1035 non-null   int64
 3   orbital_period  992 non-null    float64
 4   mass            511 non-null    float64
 5   distance        804 non-null    float64
 6   year            1031 non-null   float64
dtypes: float64(4), int64(2), object(1)
memory usage: 64.7+ KB
```

- 그 결과 mass에는 511개의 유효데이터가, distance에는 804개의 유효 데이터가 존재하는 것을 알 수 있음.
- 따라서 mass와 distance에 존재하는 결측 데이터를 삭제하기 위해 dropna() 함수를 활용하여 결측 데이터를 삭제하도록 하겠음.
- dropna() 함수의 subset 옵션에 기준이 되는 두 변수명 mass와 distance를 입력하고 두 변수 중 하나라도 결측 데이터가 존재하면 삭제하기 위해 how 옵션에 any를 입력하였음.

```
In [14]: df = merge_outer.dropna(subset=['mass', 'distance'], axis=0, how='any')

In [15]: df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 496 entries, 0 to 784
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              496 non-null    int64
 1   method          496 non-null    object
 2   number          496 non-null    int64
 3   orbital_period  496 non-null    float64
 4   mass            496 non-null    float64
 5   distance        496 non-null    float64
 6   year            496 non-null    float64
dtypes: float64(4), int64(2), object(1)
memory usage: 31.0+ KB
```

- 그 결과 총 496개의 유효 데이터만 존재하는 것을 확인할 수 있음.

※ 여러 데이터프레임의 병합

- 다음의 자료는 공통의 ID를 통해서 ex1_1.xlsx에서는 method, number, orbital_period를 측정하여 기록한 자료이고, ex1_2.xlsx는 mass, distance, year를 측정하여 기록한 자료임.
- 다음의 2개의 자료를 ID를 인덱스로 설정하여 불러와 하나의 데이터프레임으로 만들어 보세요.

	A	B	C	D	E
1	ID	method	number	orbital_period	
2	0	Radial Velocity	1	269.3	
3	1	Radial Velocity	1	874.774	
4	2	Radial Velocity	1	763	
5	3	Radial Velocity	1	326.03	
6	4	Radial Velocity	1	516.22	
7	5	Radial Velocity	1	185.84	
8	6	Radial Velocity	1	1773.4	
9	7	Radial Velocity	1	798.5	
10	8	Radial Velocity	1	993.3	
11	9	Radial Velocity	2	452.8	

	A	B	C	D	E	F	G
1	ID	mass	distance	year			
2	0	7.1	77.4	2006			
3	1	2.21	56.95	2008			
4	2	2.6	19.84	2011			
5	3	19.4	110.62	2007			
6	4	10.5	119.47	2009			
7	5	4.8	76.39	2008			
8	6	4.64	18.15	2002			
9	7		21.41	1996			

- 먼저 두 데이터프레임의 ID를 행 인덱스로 설정하여 불러오기 위하여 read_excel() 함수의 index_col 옵션에 ID를 입력함.

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

In [2]: df1 = pd.read_excel('C:/Users/LeeKJ/Desktop/ex1_1.xlsx', index_col='ID')

In [3]: print(df1)
              method  number  orbital_period
ID
0     Radial Velocity       1      269.300000
1     Radial Velocity       1      874.774000
2     Radial Velocity       1      763.000000
3     Radial Velocity       1      326.030000
4     Radial Velocity       1      516.220000
...               ...     ...             ...
1030          Transit       1        3.941507
1031          Transit       1        2.615864
1032          Transit       1        3.191524
1033          Transit       1        4.125083
1034          Transit       1        4.187757

[1035 rows x 3 columns]
```

```
In [4]: df2 = pd.read_excel('C:/Users/LeeKJ/Desktop/ex1_2.xlsx', index_col='ID')

In [5]: print(df2)
      mass  distance  year
ID
0     7.10     77.40  2006
1     2.21     56.95  2008
4    10.50    119.47  2009
5     4.80     76.39  2008
6     4.64     18.15  2002
...    ...       ...   ...
1028   NaN   3200.00  2012
1029   NaN     10.10  2012
1030   NaN    172.00  2006
1031   NaN    148.00  2007
1034   NaN    260.00  2008

[1031 rows x 3 columns]
```

- 다음으로 첫 번째 첫 번째 데이터프레임을 기준으로 하여 join() 함수를 적용하여 두 번째 데이터프레임이 병합하였음.

```
In [6]: df = df1.join(df2)

In [7]: print(df)
              method  number  orbital_period   mass  distance    year
ID
0     Radial Velocity       1      269.300000   7.10     77.40  2006.0
1     Radial Velocity       1      874.774000   2.21     56.95  2008.0
2     Radial Velocity       1      763.000000    NaN       NaN     NaN
3     Radial Velocity       1      326.030000    NaN       NaN     NaN
4     Radial Velocity       1      516.220000  10.50    119.47  2009.0
...               ...     ...             ...    ...       ...     ...
1030          Transit       1        3.941507    NaN    172.00  2006.0
1031          Transit       1        2.615864    NaN    148.00  2007.0
1032          Transit       1        3.191524    NaN       NaN     NaN
1033          Transit       1        4.125083    NaN       NaN     NaN
1034          Transit       1        4.187757    NaN    260.00  2008.0

[1035 rows x 6 columns]
```

- 그 결과 첫 번째 데이터프레임(df1)의 행 인덱스에 존재하는 값을 기준으로 두 번째 데이터프레임의 데이터가 결합된 것을 확인할 수 있음. 합집합 형태로 결합되었으므로 행 인덱스 번호인 2, 3, 1032, 1033의 경우 mass, distance, year에서 모두 결측의 형태로 자료가 병합된 것을 확인할 수 있음.

- 다음으로 how 옵션에 'inner'을 입력하여 두 번째 데이터프레임을 기준

으로 하여 첫 번째 데이터프레임이 병합하였음.

```
In [8]: df = df1.join(df2, how='inner')

In [9]: print(df)
              method  number  orbital_period   mass  distance  year
ID
0     Radial Velocity       1      269.300000   7.10     77.40  2006
1     Radial Velocity       1      874.774000   2.21     56.95  2008
4     Radial Velocity       1      516.220000  10.50    119.47  2009
5     Radial Velocity       1      185.840000   4.80     76.39  2008
6     Radial Velocity       1     1773.400000   4.64     18.15  2002
...               ...     ...             ...    ...       ...   ...
1028          Transit       1        3.352057    NaN   3200.00  2012
1029          Imaging       1             NaN    NaN     10.10  2012
1030          Transit       1        3.941507    NaN    172.00  2006
1031          Transit       1        2.615864    NaN    148.00  2007
1034          Transit       1        4.187757    NaN    260.00  2008

[1031 rows x 6 columns]
```

- 그 결과 첫 번째 데이터프레임(df1)의 행 인덱스에 존재하는 값을 기준으로 두 번째 데이터프레임의 데이터가 결합된 것을 확인할 수 있음. 교집합 형태로 결합되었으므로 행 인덱스 번호인 2, 3, 1032, 1033의 경우 제외된 형태로 자료가 병합된 것을 확인할 수 있음.