In [1]:
```
# module 불러오기
import pandas as pd
```

데이터 가공 1 - 데이터 합치기

pandas.merge()

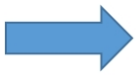
두 개의 데이터프레임을 A**컬럼**의 값을 기준으로 합치고 싶은 경우, **merge()**함수를 사용

df1

	A	B
0	1	3
1	1	2
2	2	5

df2

	A	C
0	1	Big
1	2	Data
2	3	analytics



결과

	A	B	C
0	1	3	Big
1	1	2	Big
2	2	5	Data

In [2]:
```
# sample data 생성
df1 = pd.DataFrame(data = [[1, 3], [1, 2], [2, 5]],
                   columns = list('AB'))   # columns = ['A', 'B']

df2 = pd.DataFrame(data = [[1, 'Big'], [2, 'Data'], [3, 'analytics']],
                   columns = list('AC'))   # columns = ['A', 'C']
```

In [3]:
```
print(df1)
print(df2)
```

```
   A  B
0  1  3
1  1  2
2  2  5
   A          C
0  1        Big
1  2       Data
2  3  analytics
```

In [4]:
```
# 'A'를 기준으로 합치기
# pd.merge(df1, df2, on = 'A')
pd.merge(df1, df2, on = 'A')
```

Out[4]:

	A	B	C
0	1	3	Big
1	1	2	Big
2	2	5	Data

주요 파라미터

- 키를 기준으로 DataFrame의 로우를 합친다. SQL이나 다른 관계형 데이터베이스의 join 연산과 동일함.
- 주요 파라미터

  . left, right : merge할 DataFrame 객체이름

  . how = 'inner', #left, right, outer

  . on = None, #merge의 기준이 되는 컬럼

  . left_on = None, #left DataFrame의 기준 컬럼

. right_on = None, #right DataFrame의 기준 컬럼

In [5]:
```
df_left = pd.DataFrame({'KEY': ['k0', 'k1', 'k2', 'k3'],
                        'A': ['a0', 'a1', 'a2', 'a3'],
                        'B': ['b0', 'b1', 'b2', 'b3']})
df_right = pd.DataFrame({'KEY': ['k2', 'k3', 'k4', 'k5'],
                         'C': ['c2', 'c3', 'c4', 'c5'],
                         'D': ['d2', 'd3', 'd4', 'd5']})
```

In [6]:
```
# df_left 출력
df_left
```

Out[6]:

	KEY	A	B
0	k0	a0	b0
1	k1	a1	b1
2	k2	a2	b2
3	k3	a3	b3

In [7]:
```
# df_right 출력
df_right
```

Out[7]:

	KEY	C	D
0	k2	c2	d2
1	k3	c3	d3
2	k4	c4	d4
3	k5	c5	d5

In [8]:
```
# inner join
# 공통인 KEY 값만 불러온다.
pd.merge(df_left, df_right, how = 'inner')
```

Out[8]:

	KEY	A	B	C	D
0	k2	a2	b2	c2	d2
1	k3	a3	b3	c3	d3

In [9]:
```
# left outer join
# inner join 할 수 없는 값도 불러오고, NaN 값을 도출
# inner join의 경우 df_left와 df_right의 순서는 상관없지만, 아래의 경우는 순서 고려
pd.merge(df_left, df_right, how = 'left')
```

Out[9]:

	KEY	A	B	C	D
0	k0	a0	b0	NaN	NaN
1	k1	a1	b1	NaN	NaN
2	k2	a2	b2	c2	d2
3	k3	a3	b3	c3	d3

In [10]:
```
#right outer join
pd.merge(df_left, df_right, how = 'right')
```

Out[10]:

	KEY	A	B	C	D
0	k2	a2	b2	c2	d2
1	k3	a3	b3	c3	d3
2	k4	NaN	NaN	c4	d4
3	k5	NaN	NaN	c5	d5

In [11]:
```
# fully outer join
pd.merge(df_left, df_right, how = 'outer')
```

Out[11]:

	KEY	A	B	C	D
0	k0	a0	b0	NaN	NaN
1	k1	a1	b1	NaN	NaN
2	k2	a2	b2	c2	d2
3	k3	a3	b3	c3	d3
4	k4	NaN	NaN	c4	d4
5	k5	NaN	NaN	c5	d5

In [12]:
```
# 실습
# 아래 2개의 DataFrame을 K1과 K2 컬럼으로 outer merge
df_left2 = pd.DataFrame({'K1': ['k0', 'k1', 'k2', 'k3'],
                         'A': ['a0', 'a1', 'a2', 'a3'],
                         'B': ['b0', 'b1', 'b2', 'b3']})
df_right2 = pd.DataFrame({'K2': ['k2', 'k3', 'k4', 'k5'],
                          'C': ['c2', 'c3', 'c4', 'c5'],
                          'D': ['d2', 'd3', 'd4', 'd5']})
```

In [13]:
```
# 실습
pd.merge(df_left2, df_right2,
         left_on = 'K1', right_on = 'K2',
         how = 'outer')
```

Out[13]:

	K1	A	B	K2	C	D
0	k0	a0	b0	NaN	NaN	NaN
1	k1	a1	b1	NaN	NaN	NaN
2	k2	a2	b2	k2	c2	d2
3	k3	a3	b3	k3	c3	d3
4	NaN	NaN	NaN	k4	c4	d4
5	NaN	NaN	NaN	k5	c5	d5

In [14]:
```
import os

os.getcwd()
```

Out[14]:
```
'/Users/sehee/Desktop/python/2. 실습 파일'
```

In [ ]:
```
# directory 변경
# os.chdir("C:'파일경로'")
```

In [16]:
```
# 실제 데이터 실습
# 데이터 읽어오기
# 데이터 파일: 'data/movielens/users.dat'
# 컬럼명 = ['사용자아이디','성별','연령','직업','지역']
users = pd.read_csv("data/movielens/users.dat",
            sep = "::",
            names = ['사용자아이디', '성별', '연령', '직업', '지역'])
```

```
/Users/sehee/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:
7: ParserWarning: Falling back to the 'python' engine because the 'c' engine
does not support regex separators (separators > 1 char and different from '\s
+' are interpreted as regex); you can avoid this warning by specifying engine
='python'.
  import sys
```

In [17]:
```
# 03_Quiz 마지막 문제에 대한 부가 설명
users['성별'].value_counts()
```

Out[17]:
```
M    4331
F    1709
Name: 성별, dtype: int64
```

In [18]:
```
# head()
users.head(10)
```

Out[18]:

	사용자아이디	성별	연령	직업	지역
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455
5	6	F	50	9	55117
6	7	M	35	1	06810
7	8	M	25	12	11413
8	9	M	25	17	61614
9	10	F	35	1	95370

In [19]:
```
users.shape
```

Out[19]:
```
(6040, 5)
```

In [20]:
```
# 평점 데이터 읽어오기
# 데이터 파일 : data/movielens/ratings.dat
# 컬럼명들은 ['사용자아이디', '영화아이디', '평점', '타임스탬프']
ratings = pd.read_csv("data/movielens/ratings.dat",
            sep = "::",
            names = ['사용자아이디', '영화아이디', '평점', '타임스탬프'])
```

```
/Users/sehee/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:
6: ParserWarning: Falling back to the 'python' engine because the 'c' engine
does not support regex separators (separators > 1 char and different from '\s
+' are interpreted as regex); you can avoid this warning by specifying engine
='python'.
```

In [21]:
```
# head()
ratings.head()
```

Out[21]:

	사용자아이디	영화아이디	평점	타임스탬프
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291

In [22]:
```
ratings.shape
```

Out[22]:
```
(1000209, 4)
```

In [23]:
```
# 사용자와 평점 데이터 합치기(inner)
pd.merge(users, ratings, how = 'inner')
```

Out[23]:

	사용자아이디	성별	연령	직업	지역	영화아이디	평점	타임스탬프
0	1	F	1	10	48067	1193	5	978300760
1	1	F	1	10	48067	661	3	978302109
2	1	F	1	10	48067	914	3	978301968
3	1	F	1	10	48067	3408	4	978300275
4	1	F	1	10	48067	2355	5	978824291
...	...	...	...	...	...	...	...	...
1000204	6040	M	25	6	11106	1091	1	956716541
1000205	6040	M	25	6	11106	1094	5	956704887
1000206	6040	M	25	6	11106	562	5	956704746
1000207	6040	M	25	6	11106	1096	4	956715648
1000208	6040	M	25	6	11106	1097	4	956715569

1000209 rows × 8 columns

In [24]:
```
# 사용자와 평점 데이터 합치기(left)
pd.merge(users, ratings, how = 'left')
```

Out[24]:

	사용자아이디	성별	연령	직업	지역	영화아이디	평점	타임스탬프
0	1	F	1	10	48067	1193	5	978300760
1	1	F	1	10	48067	661	3	978302109
2	1	F	1	10	48067	914	3	978301968
3	1	F	1	10	48067	3408	4	978300275
4	1	F	1	10	48067	2355	5	978824291
...	...	...	...	...	...	...	...	...
1000204	6040	M	25	6	11106	1091	1	956716541
1000205	6040	M	25	6	11106	1094	5	956704887
1000206	6040	M	25	6	11106	562	5	956704746
1000207	6040	M	25	6	11106	1096	4	956715648
1000208	6040	M	25	6	11106	1097	4	956715569

1000209 rows × 8 columns

In [25]:
```
# users에는 저장된 사용자지만, ratings에는 없는 사용자 찾기
# isnull()
data2 = pd.merge(users, ratings, how = 'inner')
data2['사용자아이디'].isnull()
```

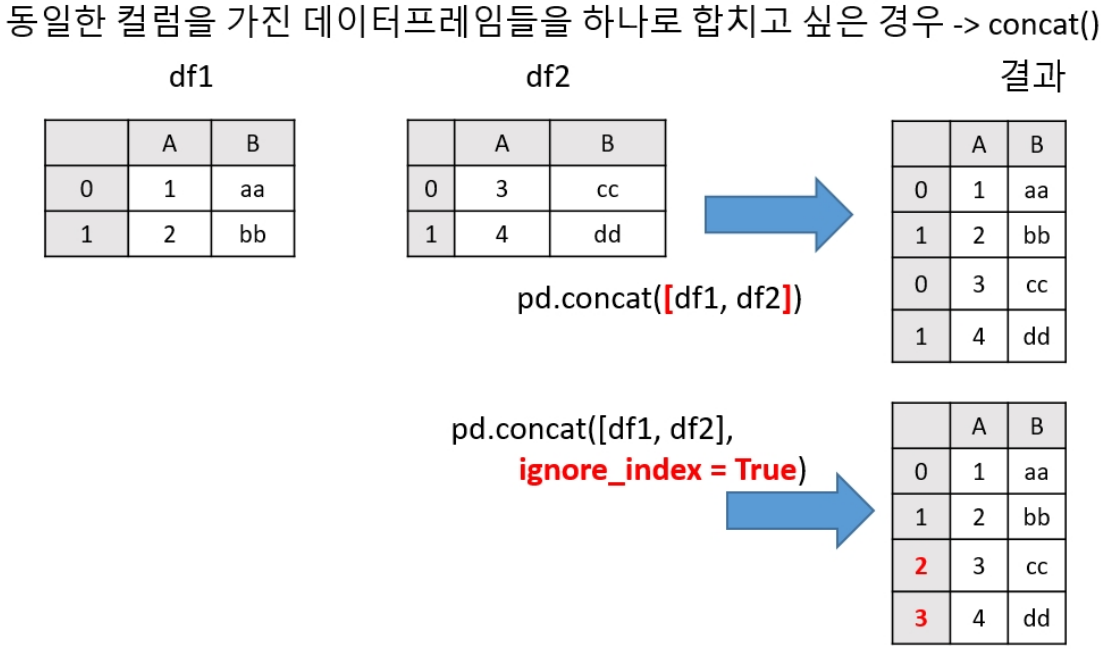
Out[25]:
```
0          False
1          False
2          False
3          False
4          False
           ...
1000204    False
1000205    False
1000206    False
1000207    False
1000208    False
Name: 사용자아이디, Length: 1000209, dtype: bool
```

In [26]:
```
# 여기에서는 Null 값이 없다. True가 없다는 뜻. 있다면 drop 함수를 사용하여 지워준다.
data2[data2['사용자아이디'].isnull()]
```

Out[26]:

사용자아이디	성별	연령	직업	지역	영화아이디	평점	타임스탬프

pandas.concat(axis = 0)



In [27]:
```
df1 = pd.DataFrame([[1, 'aa'], [2, 'bb']],
                   columns = list('AB'))

df2 = pd.DataFrame([[3, 'cc'], [4, 'dd']],
                   columns = list('AB'))
```

In [28]:
```
df1
```

Out[28]:

	A	B
0	1	aa
1	2	bb

In [29]:
```
df2
```

Out[29]:

	A	B
0	3	cc
1	4	dd

In [30]:
```
# df1과 df2 concat 수행하기
# 0이면 행, 1이면 열로 달라붙는다. (axis = 1)
# merge는 어떤 기준, concat는 사용자 정의에 따라 붙여준다.
pd.concat([df1, df2], axis = 0)
```

Out[30]:

	A	B
0	1	aa
1	2	bb
0	3	cc
1	4	dd

In [31]:
```
# 실제 데이터 실습
#  data/NC Dinos.xlsx 파일에서 2013, 2014, 2015시즌 정보를 읽어서
# 각각 NC13, NC14, NC15에 저장
NC = pd.read_excel("data/NC Dinos.xlsx",
                   sheet_name = None)   # 어떤 것도 지정하지 않고 모두 불러온다.
```

In [32]:
```
NC
```

Out[32]:
```
OrderedDict([('2013',
                   Unnamed: 0  선수명  팀명   경기   타석   타수   안타  홈런  득점  타
점  볼넷   삼진  도루  BABIP  \
                0           0  모창민  NC  108  436  395  109  12  57  51  37
68  16  0.307
                1           1  이호준  NC  126  508  442  123  20  46  87  60
109   2  0.324
                2           2  김종호  NC  128  546  465  129   0  72  22  57
100  50  0.352
                3           3  나성범  NC  104  458  404   98  14  55  64  33
95  12  0.279
                4           4  조영훈  NC  120  426  380  107   6  38  39  39
56   4  0.316
                5           5  이현곤  NC   91  161  139   38   0  10   9  16
14   2  0.304
                6           6  이상호  NC  102  138  125   31   0  26  13   9
21  24  0.298
                7           7  강진성  NC    3    3    2    1   0   1   0   1
0   0  0.500
                8           8  조평호  NC   26   86   70   21   2  12   7   6
```

In [33]:
```
NC13, NC14, NC15 = NC.values()
```

In [34]:
```
# 필요없는 컬럼 제거
NC13.drop(['Unnamed: 0'], axis = 1, inplace = True)
NC14.drop(['Unnamed: 0'], axis = 1, inplace = True)
NC15.drop(['Unnamed: 0'], axis = 1, inplace = True)
```

In [35]:
```
# 연도 column 추가
NC13['연도'] = 2013
NC14['연도'] = 2014
NC15['연도'] = 2015
```

In [36]:
```
NC14
```

Out[36]:

	선수명	팀명	경기	타석	타수	안타	홈런	득점	타점	볼넷	삼진	도루	BABIP	타율	출루율	장타율	OPS	wOBA
0	테임즈	NC	125	514	443	152	37	95	121	58	99	11	0.367	0.343	0.422	0.688	1.110	0.456
1	나성범	NC	123	536	477	157	30	88	101	43	128	14	0.397	0.329	0.400	0.597	0.997	0.424
2	박민우	NC	118	491	416	124	1	87	40	56	89	50	0.373	0.298	0.392	0.399	0.791	0.365
3	손시헌	NC	97	361	307	90	5	39	39	34	53	2	0.331	0.293	0.368	0.414	0.782	0.349
4	지석훈	NC	114	238	212	58	6	26	34	16	46	1	0.323	0.274	0.340	0.462	0.802	0.352
5	이호준	NC	122	500	424	115	23	59	78	67	104	2	0.305	0.271	0.371	0.481	0.852	0.369
6	권희동	NC	101	252	221	63	7	39	36	25	43	6	0.324	0.285	0.363	0.443	0.806	0.353
7	모창민	NC	122	468	419	110	16	62	72	37	82	14	0.289	0.263	0.320	0.413	0.733	0.319
8	이종욱	NC	124	495	438	126	6	73	79	40	60	15	0.313	0.288	0.343	0.411	0.754	0.332
9	김준완	NC	6	5	4	2	0	1	0	1	1	1	0.667	0.500	0.600	0.500	1.100	0.509
10	최재원	NC	2	2	1	0	0	1	0	1	1	0	-	0.000	0.500	0.000	0.500	0.363
11	마낙길	NC	2	2	1	0	0	1	0	1	0	0	0.000	0.000	0.500	0.000	0.500	0.363
12	조영훈	NC	92	124	111	29	6	15	22	10	26	1	0.288	0.261	0.325	0.459	0.784	0.337
13	김태우	NC	4	5	5	1	0	0	0	0	3	0	0.500	0.200	0.200	0.400	0.600	0.244
14	박명환*	NC	2	1	1	0	0	0	0	0	1	0	-	0.000	0.000	0.000	0.000	0.000
15	조평호	NC	2	2	2	0	0	0	0	0	0	0	0.000	0.000	0.000	0.000	0.000	0.000
16	강민국	NC	6	3	3	0	0	0	0	0	0	0	0.000	0.000	0.000	0.000	0.000	0.000

	선수명	팀명	경기	타석	타수	안타	홈런	득점	타점	볼넷	삼진	도루	BABIP	타율	출루율	장타율	OPS	wOBA
17	허준	NC	28	36	29	5	0	2	1	5	10	0	0.263	0.172	0.294	0.276	0.570	0.268
18	김성욱	NC	26	26	23	4	1	6	1	3	9	1	0.231	0.174	0.269	0.348	0.617	0.272
19	노진혁	NC	25	16	16	3	1	3	2	0	7	0	0.250	0.188	0.188	0.375	0.563	0.231

In [37]:
```
# pd.concat 수행
# 행으로 합치기
NC = pd.concat([NC13, NC14, NC15], axis = 0)
```

In [38]:
```
NC
```

Out[38]:

	선수명	팀명	경기	타석	타수	안타	홈런	득점	타점	볼넷	삼진	도루	BABIP	타율	출루율	장타율	OPS	wOBA
0	모창민	NC	108	436	395	109	12	57	51	37	68	16	0.307	0.276	0.339	0.443	0.782	0.353
1	이호준	NC	126	508	442	123	20	46	87	60	109	2	0.324	0.278	0.362	0.475	0.837	0.373
2	김종호	NC	128	546	465	129	0	72	22	57	100	50	0.352	0.277	0.376	0.333	0.709	0.339
3	나성범	NC	104	458	404	98	14	55	64	33	95	12	0.279	0.243	0.319	0.416	0.735	0.329
4	조영	NC	120	426	380	107	6	38	39	39	56	4	0.316	0.282	0.350	0.413	0.763	0.348

In [39]:
```
NC.shape
```

Out[39]:
```
(60, 20)
```

pandas.concat(axis = 1)

서로 다른 컬럼을 가진 데이터프레임들을 인덱스 라벨의 값을 기준으로 합치고 싶은 경우
-> **concat([df1, df2,...], axis = 1)**

df1

	A	B
aa	1	10
bb	2	20
cc	3	30

df2

	C	D
aa	3	xx
bb	4	yy

결과

	A	B	C	D
aa	1	10	3	xx
bb	2	20	4	yy
cc	3	30	NaN	NaN

In [41]:
```
# 샘플 데이터 생성
df1 = pd.DataFrame([[1,10],[2,20], [3,30]],
                   columns = list('AB'),
                   index = ['aa','bb', 'cc'])
df2 = pd.DataFrame([[3,'xx'],[4,'yy']],
                   columns = list('CD'),
                   index = ['aa','bb'])
```

In [43]:
```
# concat with axis = 1 수행
pd.concat([df1, df2], axis = 1).fillna('-')
```

```
/Users/sehee/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:
2: FutureWarning: Sorting because non-concatenation axis is not aligned. A fu
ture version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.
```

Out[43]:

	A	B	C	D
aa	1	10	3	xx
bb	2	20	4	yy
cc	3	30	-	-

실습. 로우 인덱스는 '선수명'으로, 컬럼 인덱스는 '연도'로 구성하여,

NC 다이노스 선수들의 홈런 기록을 연도별로 볼 수 있도록 데이터프레임을 구성하시오.

	2013	2014	2015
강구성	0	-	0
강민국	-	0	0
강진성	0	-	-
권희동	-	7	-
김동건	1	-	-
김성욱	0	1	-
김종찬	0	-	-
김종호	0	-	4
김준완	-	0	0

In [46]:
```
# set_index() : index 설정하기
# inplace: 변수에 담아주는 역할
NC13.set_index('선수명', inplace = True)
NC14.set_index('선수명', inplace = True)
NC15.set_index('선수명', inplace = True)
```

In [53]:
```
NC = pd.concat([NC13['홈런'], NC14['홈런'], NC15['홈런']], axis = 1, sort = True).
NC.columns = [2013, 2014, 2015]
NC
```

Out[53]:

	2013	2014	2015
강구성	0	-	0
강민국	-	0	0
강진성	0	-	-
권희동	-	7	-
김동건	1	-	-
김성욱	0	1	-
김종찬	0	-	-
김종호	0	-	4
김준완	-	0	0
김태군	-	-	6
김태우	-	0	-
김태진	-	-	0
나성범	14	30	28
노진혁	-	1	-
마낙길	0	0	-
모창민	12	16	6
박명환*	-	0	-
박민우	0	1	3
박정준	4	-	0
손시헌	-	5	13
용덕한	-	-	0
이상호	0	-	-
이승호	0	-	-
이종욱	-	6	5
이창섭	0	-	0
이현곤	0	-	-
이호준	20	23	24
조영훈	6	6	8
조평호	2	0	1
지석훈	3	6	11
차화준	1	-	-
최재원	-	0	2
테임즈	-	37	47
허준	-	0	-