pandas 불러오기

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

users.data 불러오기

In [2]:
```
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:
3: 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'.
  This is separate from the ipykernel package so we can avoid doing imports u
ntil
```

ratings.data 불러오기

In [3]:
```
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:
3: 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'.
  This is separate from the ipykernel package so we can avoid doing imports u
ntil
```

users.dat와 ratings.dat를 inner join 해주세요

In [4]:
```
df_inner = pd.merge(users, ratings, how = 'inner')
df_inner
```

Out[4]:

	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

inner join한 데이터의 처음 10개 행을 출력해주세요

In [5]:
```
df_inner.head(10)
```

Out[5]:

	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
5	1	F	1	10	48067	1197	3	978302268
6	1	F	1	10	48067	1287	5	978302039
7	1	F	1	10	48067	2804	5	978300719
8	1	F	1	10	48067	594	4	978302268
9	1	F	1	10	48067	919	4	978301368

타임스탬프 column을 삭제해주세요

In [6]:
```
df_inner.drop(['타임스탬프'], axis = 1, inplace = True)
df_inner.head()
```

Out[6]:

	사용자아이디	성별	연령	직업	지역	영화아이디	평점
0	1	F	1	10	48067	1193	5
1	1	F	1	10	48067	661	3
2	1	F	1	10	48067	914	3
3	1	F	1	10	48067	3408	4
4	1	F	1	10	48067	2355	5

총 사용자가 몇명이고 영화에 가장 많은 평점을 부여한 5명을 찾아주세요

In [7]:
```
# 모범답안: 총 사용자 명수
len(df_inner['사용자아이디'].value_counts())
```

Out[7]:
```
6040
```

In [9]:
```
# 모범답안: 영화에 가장 많은 평점을 부여한 5명
# 자동 내림차순 정렬
df_inner['사용자아이디'].value_counts().head(5).index
```

Out[9]:
```
Int64Index([4169, 1680, 4277, 1941, 1181], dtype='int64')
```

In [8]:
```
# sehee: 특정 열에 대한 중복 여부 판단
df_inner.duplicated(subset = ['사용자아이디'])
```

Out[8]:
```
0          False
1           True
2           True
3           True
4           True
           ...
1000204     True
1000205     True
1000206     True
1000207     True
1000208     True
Length: 1000209, dtype: bool
```

In [9]:
```
# sehee: dictionary length
len(join)
```

Out[9]:
```
1000209
```

In [ ]:
```
# sehee: (참고)총 사용자 수 - for문 컴파일 처리 소요 시간 증가
total_users = 0

for i in range (len(join)):
    if (join.duplicated(subset = ['사용자아이디'])[i] == False):
        total_users += 1

    else:
        continue

print('The total number of users is ', total_users)
```

In [10]:
```
# sehee: 총 사용자 수 - value count

user_dic = {}

users = join['사용자아이디']   # users = join.loc[:, '사용자아이디']
for i in users:
    if i in user_dic.keys():
        user_dic[i] += 1
    else:
        user_dic[i] = 1

print('The total number of users is ', len(user_dic), '.')
```

```
The total number of users is  6040 .
```

In [40]:
```
# sehee: 동일한 사용자 아이디 수
user_dic.values()
```

Out[40]:
```
dict_values([53, 129, 51, 21, 198, 71, 31, 139, 106, 401, 137, 23, 108, 25, 2
01, 35, 211, 305, 255, 24, 22, 297, 304, 136, 85, 400, 70, 107, 108, 43, 119,
48, 391, 164, 198, 351, 53, 100, 62, 96, 25, 231, 24, 193, 297, 41, 22, 598,
108, 43, 40, 79, 684, 40, 25, 67, 64, 437, 213, 70, 36, 498, 98, 27, 121, 26,
64, 72, 65, 54, 29, 43, 255, 43, 175, 87, 39, 140, 31, 48, 86, 118, 99, 31, 3
9, 48, 59, 68, 21, 225, 44, 430, 220, 21, 99, 81, 154, 20, 107, 76, 106, 132,
115, 46, 61, 47, 121, 37, 81, 80, 92, 60, 68, 98, 38, 86, 505, 224, 105, 63,
72, 57, 608, 23, 71, 51, 174, 22, 158, 135, 295, 89, 170, 183, 70, 378, 201,
65, 245, 55, 23, 47, 68, 32, 39, 426, 187, 624, 592, 232, 471, 24, 26, 44, 9
6, 148, 427, 22, 36, 20, 297, 107, 514, 26, 110, 410, 58, 26, 552, 87, 24, 2
3, 561, 97, 317, 78, 44, 115, 110, 59, 301, 83, 101, 84, 51, 122, 417, 71, 3
1, 153, 30, 525, 242, 59, 822, 32, 22, 379, 402, 26, 87, 670, 127, 446, 153,
30, 23, 184, 25, 110, 35, 110, 22, 166, 34, 794, 20, 86, 36, 102, 29, 80, 25
6, 204, 508, 21, 28, 31, 212, 179, 97, 90, 53, 29, 440, 62, 166, 258, 71, 14
3, 57, 273, 33, 85, 764, 29, 31, 34, 110, 20, 73, 60, 53, 23, 244, 95, 53, 14
5, 139, 154, 354, 106, 92, 480, 98, 50, 131, 261, 30, 126, 312, 482, 187, 14
8, 89, 23, 22, 198, 47, 50, 111, 29, 27, 471, 139, 43, 103, 56, 74, 33, 102,
41, 317, 49, 191, 97, 89, 27, 104, 249, 357, 788, 209, 27, 157, 193, 182, 55
7, 123, 223, 20, 57, 26, 293, 22, 77, 21, 21, 721, 21, 222, 40, 118, 76, 43,
```

In [11]:
```
# sehee: 영화 평점 빈도수가 가장 높은 사용자 5명

user_list = list(user_dic.values())
user_list.sort(reverse = True)

for i in range(5):
    print('The ', i+1, 'th most rated user of the movie is ', user_list[i], '.
```

```
The  1 th most rated user of the movie is  2314 .
The  2 th most rated user of the movie is  1850 .
The  3 th most rated user of the movie is  1743 .
The  4 th most rated user of the movie is  1595 .
The  5 th most rated user of the movie is  1521 .
```

각 직업마다의 영화 평균 평점을 구한 데이터프레임을 만들어주세요

In [2]:
```
# 아래와 같은 데이터 프레임 결과를 만들어주세요
# 코드 작성은 아래 셀에서 해주세요
```

Out[2]:

	직업	평점
0	0	3.537544
1	1	3.576642
2	2	3.573081
3	3	3.656516
4	4	3.536793
5	5	3.537529
6	6	3.661578
7	7	3.599772
8	8	3.466741
9	9	3.656589
10	10	3.532675
11	11	3.617371
12	12	3.654001
13	13	3.781736
14	14	3.618481
15	15	3.689774
16	16	3.596575
17	17	3.613574
18	18	3.530117
19	19	3.414050
20	20	3.497392

In [15]:
```
# 모범답안: 각 직업마다의 영화 평균 평점을 구한 데이터프레임
job = df_inner['직업'].value_counts().sort_index().index
score = []

for i in job:
    temp = df_inner[df_inner.직업 == i]['평점'].mean()
    score.append(temp)

pd.DataFrame({'직업': job, '평점': score})
```

Out[15]:

	직업	평점
0	0	3.537544
1	1	3.576642
2	2	3.573081
3	3	3.656516
4	4	3.536793
5	5	3.537529
6	6	3.661578
7	7	3.599772
8	8	3.466741
9	9	3.656589
10	10	3.532675
11	11	3.617371
12	12	3.654001
13	13	3.781736
14	14	3.618481
15	15	3.689774
16	16	3.596575
17	17	3.613574
18	18	3.530117
19	19	3.414050
20	20	3.497392

In [11]:
```
# 모범답안 - 참고
df_inner[df_inner.직업 == 0]['평점'].mean()
```

Out[11]:
```
3.5375443489988427
```

In [51]:
```
# sehee: 직업별 영화의 평균 평점을 구한 DataFrame

job_num = 0
#job_dic = {}
grade_dic = {}
avg_dic = {}

job = join['직업']   # job = join.loc[:, '직업']
grades = join['평점']   # grades = join.loc[:, '평점']

for i in job:
    if i in grade_dic.keys():
#         job_dic[0] += 1
        grade_dic[i] += i
    else:
#         job_dic[0] = 1
        job_num += 1
        grade_dic[i] = i


    avg_dic[i] = grade_dic[i] / job_num

avg_dic
```

Out[51]:
```
{10: 11090.47619047619,
 16: 35063.619047619046,
 15: 16393.571428571428,
 7: 35141.666666666664,
 20: 57520.95238095238,
 9: 4862.142857142857,
 1: 4064.3333333333335,
 12: 32693.714285714286,
 17: 58946.28571428572,
 0: 0.0,
 3: 4517.571428571428,
 14: 32739.333333333332,
 4: 24958.47619047619,
 11: 10771.095238095239,
 8: 1030.857142857143,
 19: 13484.57142857143,
 2: 4768.380952380952,
 18: 10359.42857142857,
 5: 5202.380952380952,
 13: 8514.380952380952,
 6: 10630.0}
```

In [35]:
```
# sehee
grade_dic.values
```

Out[35]:
```
<function dict.values>
```

영화 id 특정 번호 사이의 평점 평균 구해보기(함수를 통해서)

between_rating_mean이라는 함수를 만들어서 아래와 같은 출력 값이 나와야 합니다!

밑에 3.51127값은 영화 id가 10~20(20포함)인 평점의 평균을 구한 값입니다

In [3]:
```
# 여기 있는 셀은 건드리지 마세요!
# 아래 셀에서 작성해주세요
between_rating_mean(10, 20)
```

Out[3]:
```
3.5112717992343683
```

In [14]:
```
df_inner.shape[0]
```

Out[14]:
```
1000209
```

In [21]:
```
# 모범답안: 영화 id 특정 번호 사이의 평점 평균
def between_rating_mean(num1, num2):
    movie = range(num1, num2 + 1)
    score = 0
    num = 0

    for i in movie:
        temp = df_inner[df_inner.영화아이디 == i]
        score += temp['평점'].sum()
        num += temp.shape[0]

    return score/num
```

In [22]:
```
# 모범답안: 영화 id 특정 번호 사이의 평점 평균
between_rating_mean(10, 20)
```

Out[22]:
```
3.5112717992343683
```

In [45]:
```
# sehee: 영화 아이디가 10부터 20까지인 평점의 평균

movie_num = 0
grade_num = 0
avg10_num = 0

movie_id = join['영화아이디']
grades10 = join['평점']

for i in movie_id:
    movie_num += 1
    grade_num += i

    avg10_num = grade_num / movie_num

avg10_num
```

Out[45]:
```
1865.5398981612843
```