빅데이터 실습

7주차 3차시

데이터 실전 분석 - 영화 평점 분석 [2]

01

영화 평점 분석 실습

보고 싶은 영화 찾기 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

◎ 전체 평점의 개수가 500개 이상인 영화만 대상으로 함

[실습 #3] 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

전체 평점의 개수가 500개 이상인 영화만 대상으로 함.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

[실습 #4] 연령대 별로 영화 평점 분석하기

연령대(10대 미만, 10대, 20대, ...50대) 컬럼을 추가한 후, 영화별 연령대별 영화평점 구하기

In [ ]:

보고 싶은 영화 찾기 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

◎ 전체 평점의 개수가 500개 이상인 영화만 대상으로 함

✔ 영화별 성별 평점 정보를 ex3에 저장

Run Code

아하는 영화 찾기

- 여성 평점이 4.0 이상이고 여성 평점의 개수가 500개 이상인 영화

In [25]:
```
ex1 = data.pivot_table(index = '영화제목', columns = '성별',
                       aggfunc = ['mean', 'count'], values = '평점')
```

In [30]:
```
여성인기영화 = ex1[(ex1[('mean','F')]>= 4.0) & (ex1[('count','F')]>= 500)]
```

In [31]:
```
여성인기영화
```

Out[31]:

	mean		count	
성별	F	M	F	M
영화제목				
American Beauty (1999)	4.238901	4.347301	946.0	2482.0
Being John Malkovich (1999)	4.159930	4.113636	569.0	1672.0
Braveheart (1995)	4.016484	4.297839	546.0	1897.0
Casablanca (1942)	4.300990	4.461340	505.0	1164.0

보고 싶은 영화 찾기 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

◎ 전체 평점의 개수가 500개 이상인 영화 선택

- ✔ 여성 평점 개수와 남성 평점 개수의 합이 500이상인 영화 선택
- ✔ 조건 색인 수행

	mean		count	
성별	F	M	F	M
영화제목				
10 Things I Hate About You (1999)	3.646552	3.311966	232.0	468.0
101 Dalmatians (1961)	3.791444	3.500000	187.0	378.0
12 Angry Men (1957)	4.184397	4.328421	141.0	475.0
13th Warrior, The (1999)	3.112000	3.168000	125.0	625.0
20,000 Leagues Under the Sea (1954)	3.670103	3.709205	97.0	478.0
...	...	...	...	...
X-Files: Fight the Future, The (1998)	3.489474	3.493797	190.0	806.0
X-Men (2000)	3.682310	3.851702	277.0	1234.0
You've Got Mail (1998)	3.542424	3.275591	330.0	508.0
Young Frankenstein (1974)	4.289963	4.239177	269.0	924.0

보고 싶은 영화 찾기 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

◎ 전체 평점의 개수가 500개 이상인 영화 선택

✔ ex3_500에 저장

습 #3] 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

전체 평점의 개수가 500개 이상인 영화만 대상으로 함.

In [58]:
```
ex3 = data.pivot_table(index = '영화제목', columns = '성별',
                       aggfunc = ['mean', 'count'], values = '평점')
```

In [63]:
```
ex3_500 = ex3[ex3[('count','F')] + ex3[('count','M')] >= 500]
```

In [ ]:

In [ ]:

[실습 #4] 연령대 별로 영화 평점 분석하기

연령대(10대 미만, 10대, 20대, ...50대) 컬럼을 추가한 후, 영화별 연령대별 영화평점 구하기

In [ ]:

보고 싶은 영화 찾기 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

◎ ex3_500을 대상으로 남자와 여자의 호불호가 크게 갈리는 영화 찾기

✔ 남자평점과 여자평점의 차이를 계산하여 차이가 많이 나는 것을 선택

```
In [58]: ex3 = data.pivot_table(index = '영화제목', columns = '성별',
                                aggfunc = ['mean', 'count'], values = '평점')
```

```
In [71]: ex3_500 = ex3[ex3[('count','F')] + ex3[('count','M')] >= 500]
```

```
In [73]: ex3_500['mean','F'] - ex3_500['mean','M']
```

```
Out[73]: 영화제목
10 Things I Hate About You (1999)          0.334586
101 Dalmatians (1961)                      0.291444
12 Angry Men (1957)                       -0.144024
13th Warrior, The (1999)                  -0.056000
20,000 Leagues Under the Sea (1954)       -0.039102
                                              ...
X-Files: Fight the Future, The (1998)     -0.004323
X-Men (2000)                              -0.169391
You've Got Mail (1998)                     0.266834
Young Frankenstein (1974)                  0.050785
Young Guns (1988)                         -0.053825
Length: 618, dtype: float64
```

보고 싶은 영화 찾기 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

◎ ex3_500을 대상으로 남자와 여자의 호불호가 크게 갈리는 영화 찾기

✔ abs( ) 함수를 통해 차이를 절댓값으로 표현

```
In [58]: ex3 = data.pivot_table(index = '영화제목', columns = '성별',
                                aggfunc = ['mean', 'count'], values = '평점')

In [71]: ex3_500 = ex3[ex3[('count','F')] + ex3[('count','M')] >= 500]

In [73]: abex3_500['mean','F'] - ex3_500['mean','M']
```

```
Out[73]: 영화제목
10 Things I Hate About You (1999)          0.334586
101 Dalmatians (1961)                      0.291444
12 Angry Men (1957)                       -0.144024
13th Warrior, The (1999)                  -0.056000
20,000 Leagues Under the Sea (1954)       -0.039102
                                              ...
X-Files: Fight the Future, The (1998)     -0.004323
X-Men (2000)                              -0.169391
You've Got Mail (1998)                     0.266834
Young Frankenstein (1974)                  0.050785
Young Guns (1988)                         -0.053825
Length: 618, dtype: float64
```

보고 싶은 영화 찾기 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

◎ ex3_500을 대상으로 남자와 여자의 호불호가 크게 갈리는 영화 찾기

✔ 평점 차이의 절댓값을 평점차이 컬럼으로 추가

```
In [71]: ex3_500 = ex3[ex3[('count','F')] + ex3[('count','M')] >= 500]
```

```
In [75]: ex3_500['평점평균']= abs(ex3_500['mean','F'] - ex3_500['mean','M'])
```

```
<ipython-input-75-cf2f11d931ab>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ex3_500['평점평균']= abs(ex3_500['mean','F'] - ex3_500['mean','M'])
```

```
In [76]: ex3_500
```

Out[76]:

	mean		count		평점평균
성별	F	M	F	M	
영화제목					
10 Things I Hate About You (1999)	3.646552	3.311966	232.0	468.0	0.334586

보고 싶은 영화 찾기 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

◎ ex3_500을 대상으로 남자와 여자의 호불호가 크게 갈리는 영화 찾기

✔ SettingWithCopyWarning : loc를 사용하여 접근하라는 의미

In [80]:
```
ex3_500 = ex3[ex3[('count','F')] + ex3[('count','M')] >= 500]
```

In [81]:
```
ex3_500['평점차이']= abs(ex3_500['mean','F'] - ex3_500['mean','M'])
```

```
<ipython-input-81-31cfdaa03488>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ex3_500['평점차이']= abs(ex3_500['mean','F'] - ex3_500['mean','M'])
```

In [82]:
```
ex3_500
```

Out[82]:

	mean		count		평점차이
성별	F	M	F	M	
영화제목					
10 Things I Hate About You (1999)	3.646552	3.311966	232.0	468.0	0.334586

보고 싶은 영화 찾기 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

◎ ex3_500을 대상으로 남자와 여자의 호불호가 크게 갈리는 영화 찾기

✔ 3군데 모두 .loc를 넣고 :;를 넣어줌

Run Code

```
x = '영화제목', columns = '성별',
'mean', 'count'], values = '평점')
```

In [80]:
```
ex3_500 = ex3[ex3[('count','F')] + ex3[('count','M')] >= 500]
```

In [84]:
```
_500['평점차이']= abs(ex3_500['mean','F'] - ex3_500['mean','M'])
500.loc[:, '평점차이']= abs(ex3_500.loc[:, ('mean','F')] - ex3_500.loc[:, ('mean','M')]
```

```
C:WProgramDataWAnaconda3WlibWsite-packagesWpandasWcoreWindexing.py:1745: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(ilocs[0], value)
```

In [ ]:
```
ex3_500
```

[실습 #4] 연령대 별로 영화 평점 분석하기

보고 싶은 영화 찾기 | 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

◎ 호불호가 크게 갈리는 영화 10개 찾기

✔ 평점의 차이가 큰 10개를 찾기 위해 sort_values( )로 정렬

```
In [80]: ex3_500 = ex3[ex3[('count','F')] + ex3[('count','M')] >= 500]

In [85]: #ex3_500['평점차이']= abs(ex3_500['mean','F'] - ex3_500['mean','M'])
         ex3_500.loc[:, '평점차이']= W
             abs(ex3_500.loc[:, ('mean','F')] - ex3_500.loc[:, ('mean','M')])

In [87]: ex3_500.sort_values(by = '평점차이', ascending = False)
```

Out[87]:

	mean		count		평점차이
성별	F	M	F	M	
영화제목					
Dirty Dancing (1987)	3.790378	2.959596	291.0	396.0	0.830782
Good, The Bad and The Ugly, The (1966)	3.494949	4.221300	99.0	723.0	0.726351
Dumb & Dumber (1994)	2.697987	3.336595	149.0	511.0	0.638608
Evil Dead II (Dead By Dawn) (1987)	3.297297	3.909283	74.0	474.0	0.611985

보고 싶은 영화 찾기 남자와 여자의 호불호가 크게 갈리는 영화 10개 찾기

◎ 호불호가 크게 갈리는 영화 10개 찾기

✔ nlargest( ) 사용가능

Run Markdown

```
점차이']= abs(ex3_500['mean','F'] - ex3_500['mean','M'])
:, '평점차이']= W
    abs(ex3_500.loc[:, ('mean','F')] - ex3_500.loc[:, ('mean','M')])
```

In [*]:
```
#ex3_500.sort_values(by = '평점차이', ascending = False).head(10)
ex3_500.nlargest(10, '평점차이')
```

Out[89]:

	mean		count		평점차이
성별	F	M	F	M	
영화제목					
Dirty Dancing (1987)	3.790378	2.959596	291.0	396.0	0.830782
Good, The Bad and The Ugly, The (1966)	3.494949	4.221300	99.0	723.0	0.726351
Dumb & Dumber (1994)	2.697987	3.336595	149.0	511.0	0.638608
Evil Dead II (Dead By Dawn) (1987)	3.297297	3.909283	74.0	474.0	0.611985
Grease (1978)	3.975265	3.367041	283.0	534.0	0.608224
Caddyshack (1980)	3.396135	3.969737	207.0	760.0	0.573602
Animal House (1978)	3.628906	4.167192	256.0	951.0	0.538286

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ data에는 연령 정보만 있고, 연령대 정보는 없음

✔ 연령 정보에서 10살 단위로 구분하여 연령대라는 새로운 columns을 만듦

[실습 #4] 연령대 별로 영화 평점 분석하기

연령대(10대 미만, 10대, 20대, ...50대) 컬럼을 추가한 후, 영화별 연령대별 영화평점 구하기

In [90]:
```
data
```

Out[90]:

	사용자아이디	성별	연령	직업	지역	영화아이디	평점	타임스탬프	영화제목	장르
0	1	F	1	10	48067	1193	5	978300760	One Flew Over the Cuckoo's Nest (1975)	Drama
1	2	M	56	16	70072	1193	5	978298413	One Flew Over the Cuckoo's Nest (1975)	Drama
2	12	M	25	12	32793	1193	4	978220179	One Flew Over the Cuckoo's Nest (1975)	Drama

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령 columns의 값 분포 확인

✔ describe( ) 활용

(10대 미만, 10대, 20대, ...50대) 컬럼을 추가한 후, 영화별 연령대별 영화평점 구하기

In [91]:
```
# 연령 컬럼의 값 분포 확인
data.describe()
```

Out[91]:

	사용자아이디	연령	직업	영화아이디	평점	타임스탬프
count	1.000209e+06	1.000209e+06	1.000209e+06	1.000209e+06	1.000209e+06	1.000209e+06
mean	3.024512e+03	2.973831e+01	8.036138e+00	1.865540e+03	3.581564e+00	9.722437e+08
std	1.728413e+03	1.175198e+01	6.531336e+00	1.096041e+03	1.117102e+00	1.215256e+07
min	1.000000e+00	1.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00	9.567039e+08
25%	1.506000e+03	2.500000e+01	2.000000e+00	1.030000e+03	3.000000e+00	9.653026e+08
50%	3.070000e+03	2.500000e+01	7.000000e+00	1.835000e+03	4.000000e+00	9.730180e+08
75%	4.476000e+03	3.500000e+01	1.400000e+01	2.770000e+03	4.000000e+00	9.752209e+08
max	6.040000e+03	5.600000e+01	2.000000e+01	3.952000e+03	5.000000e+00	1.046455e+09

In [ ]:

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령 columns의 값 분포 확인

Run Code

연령대(10대 미만, 10대, 20대, ...50대) 컬럼을 추가한 후, 영화별 연령대별 영화평점 구하기

In [91]:
```
# 연령 컬럼의 값 분포 확인
data.describe()
```

Out[91]:

	사용자아이디	연령	직업	영화아이디	평점	타임스탬프
count	1.000209e+06	1.000209e+06	1.000209e+06	1.000209e+06	1.000209e+06	1.000209e+06
mean	3.024512e+03	2.973831e+01	8.036138e+00	1.865540e+03	3.581564e+00	9.722437e+08
std	1.728413e+03	1.175198e+01	6.531336e+00	1.096041e+03	1.117102e+00	1.215256e+07
min	1.000000e+00	1.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00	9.567039e+08
25%	1.506000e+03	2.500000e+01	2.000000e+00	1.030000e+03	3.000000e+00	9.653026e+08
50%	3.070000e+03	2.500000e+01	7.000000e+00	1.835000e+03	4.000000e+00	9.730180e+08
75%	4.476000e+03	3.500000e+01	1.400000e+01	2.770000e+03	4.000000e+00	9.752209e+08
max	6.040000e+03	5.600000e+01	2.000000e+01	3.952000e+03	5.000000e+00	1.046455e+09

In [ ]:

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령 columns의 값 분포 확인

✔ 10대 미만, 10대, 20대, 30대, 40대, 50대 이상

	직업	영화아이디	평점	타임스탬프
count	1.000209e+06	1.000209e+06	1.000209e+06	1.000209e+06
mean	3.024512e+03	2.973831e+01	8.036138e+00	1.865540e+03	3.581564e+00	9.722437e+08
std	1.728413e+03	1.175198e+01	6.531336e+00	1.096041e+03	1.117102e+00	1.215256e+07
min	1.000000e+00	1.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00	9.567039e+08
25%	1.506000e+03	2.500000e+01	2.000000e+00	1.030000e+03	3.000000e+00	9.653026e+08
50%	3.070000e+03	2.500000e+01	7.000000e+00	1.835000e+03	4.000000e+00	9.730180e+08
75%	4.476000e+03	3.500000e+01	1.400000e+01	2.770000e+03	4.000000e+00	9.752209e+08
max	6.040000e+03	5.600000e+01	2.000000e+01	3.952000e+03	5.000000e+00	1.046455e+09

In [ ]:

In [ ]:

In [ ]:

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ① 함수 사용

✔ generate_ages(x) : 연령값을 인자 x로 받아 연령대를 리턴하는 함수
('10대 미만'~'50대 이상')

						램프
						+06
mean	3.024512e+03	2.973831e+01	8.036138e+00	1.865540e+03	3.581564e+00	9.722437e+08
std	1.728413e+03	1.175198e+01	6.531336e+00	1.096041e+03	1.117102e+00	1.215256e+07
min	1.000000e+00	1.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00	9.567039e+08
25%	1.506000e+03	2.500000e+01	2.000000e+00	1.030000e+03	3.000000e+00	9.653026e+08
50%	3.070000e+03	2.500000e+01	7.000000e+00	1.835000e+03	4.000000e+00	9.730180e+08
75%	4.476000e+03	3.500000e+01	1.400000e+01	2.770000e+03	4.000000e+00	9.752209e+08
max	6.040000e+03	5.600000e+01	2.000000e+01	3.952000e+03	5.000000e+00	1.046455e+09

In [ ]:
```
# 1. 연령대 컬럼 추가 - 함수 사용
```

In [ ]:
```
def generate_ages(x):
    
```

In [ ]:

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ① 함수 사용

✔ x가 30보다 크고 40보다 작으면 30대

			1.400000e+01	2.770000e+03	4.000000e+00	9.752209e+08
max	6.040000e+03	5.600000e+01	2.000000e+01	3.952000e+03	5.000000e+00	1.046455e+09

In [ ]:
```
# 1. 연령대 컬럼 추가 - 함수 사용
```

In [ ]:
```
def generate_ages(x):
    if x < 10:
        return '10대 미만'
    eilf < 20:
        return '10대'
    eilf < 30:
        return '20대'
    eilf < 40:
        return '10대'
    eilf < 20:
        return '10대'
```

In [ ]:

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ① 함수 사용

✔ x가 40보다 크고 50보다 작으면 40대

			1.400000e+01	2.770000e+03	4.000000e+00	9.752209e+08
max	6.040000e+03	5.600000e+01	2.000000e+01	3.952000e+03	5.000000e+00	1.046455e+09

In [ ]:
```
# 1. 연령대 컬럼 추가 - 함수 사용
```

In [ ]:
```
def generate_ages(x):
    if x < 10:
        return '10대 미만'
    eilf < 20:
        return '10대'
    eilf < 30:
        return '20대'
    eilf < 40:
        return '30대'
    eilf < 50:
        return '40대'

    디
```

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ① 함수 사용

✔ 그 외에는 50대 이상

Run Code

	476000e+03	3.500000e+01	1.400000e+01	2.770000e+03	4.000000e+00	9.752209e+08
max	6.040000e+03	5.600000e+01	2.000000e+01	3.952000e+03	5.000000e+00	1.046455e+09

In [ ]:
```
# 1. 연령대 컬럼 추가 - 함수 사용
```

In [ ]:
```
def generate_ages(x):
    if x < 10:
        return '10대 미만'
    eilf < 20:
        return '10대'
    eilf < 30:
        return '20대'
    eilf < 40:
        return '30대'
    eilf < 50:
        return '40대'

    else:
        return
```

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ① 함수 사용

✔ generate_ages(35)를 실행하면 30대 출력

```
        return '10대'
    elif x < 30:
        return '20대'
    elif x < 40:
        return '30대'
    elif x < 50:
        return '40대'
    else:
        return '50대 이상'
```

In [95]:
```
generate_ages(35)
```

Out[95]: '30대'

In [ ]:

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ① 함수 사용

✔ generate_ages(43)를 실행하면 40대 출력

```
        return '10대'
    elif x < 30:
        return '20대'
    elif x < 40:
        return '30대'
    elif x < 50:
        return '40대'
    else:
        return '50대 이상'
```

In [96]:
```
generate_ages(43)
```

Out[96]: '40대'

In [ ]:

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ① 함수 사용

✔ generate_ages()를 연령 컬럼에 적용

```
        return '10대'
    elif x < 30:
        return '20대'
    elif x < 40:
        return '30대'
    elif x < 50:
        return '40대'
    else:
        return '50대 이상'
```

In [97]:
```
data.연령
```

Out[97]:

0	1
1	56
2	25
3	25
4	50
	..
1000204	18
1000205	35
1000206	18
1000207	18

보고 싶은 영화 찾기 | 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ① 함수 사용

✔ generate_ages( )에 data.연령을 전달하는 경우 에러 발생

주의하기

✔ generate_ages()는 인자가 숫자값이어야 하지만,
Data.연령은 Series이기 때문에 데이터 타입이 맞지 않음

```
[94]: def generate_ages(x):
          if x < 10:
              return '10대 미만'
          elif x < 20:
              return '10대'
          elif x < 30:
              return '20대'
          elif x < 40:
              return '30대'
          elif x < 50:
              return '40대'
          else:
              return '50대 이상'
```

```
In [98]: generate_ages(data.연령)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-98-ca9f92b06b08> in <module>
----> 1 generate_ages(data.연령)
```

보고 싶은 영화 찾기 | 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ① 함수 사용

✔ 함수의 결과를 연령대 컬럼으로 추가

```
        return '10대 미만'
    elif x < 20:
        return '10대'
    elif x < 30:
        return '20대'
    elif x < 40:
        return '30대'
    elif x < 50:
        return '40대'
    else:
        return '50대 이상'
```

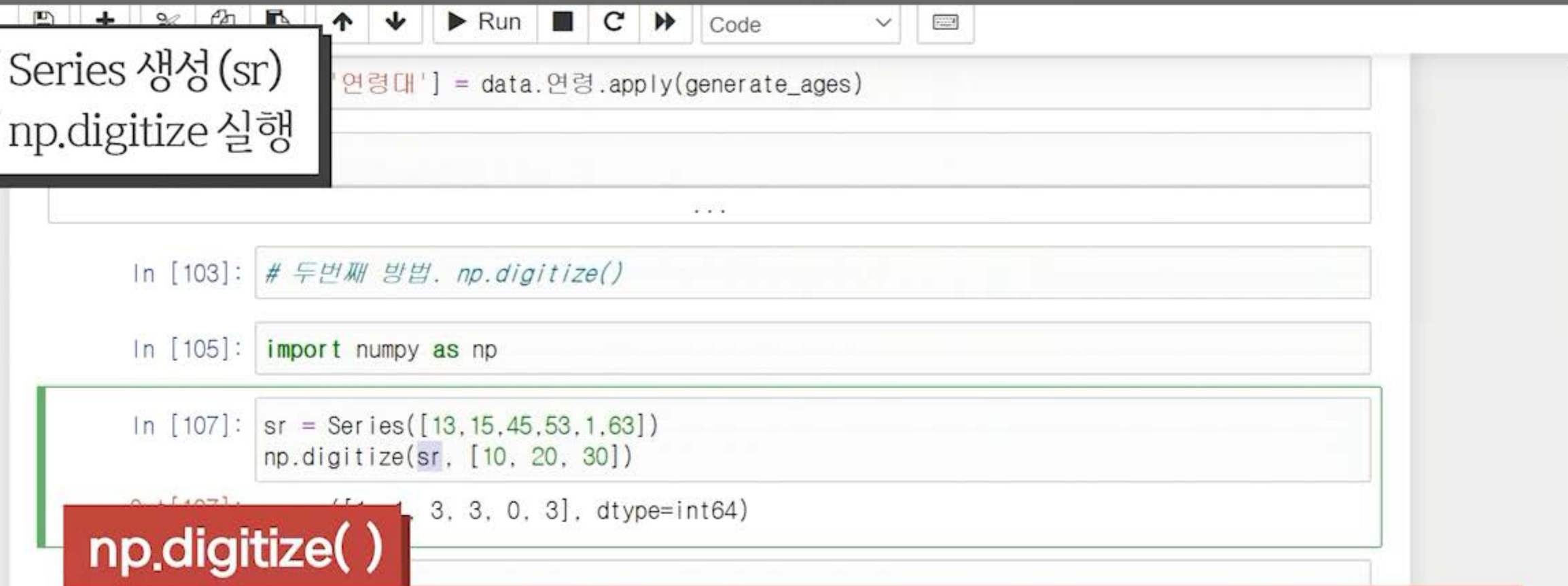
In [101]:
```
data['연령대'] = data.연령.apply(generate_ages)
```

In [ ]:

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ② np.digitize( )

- ✔ Series 생성 (sr)
- ✔ np.digitize 실행



np.digitize( )

- ✔ Series에 해당하는 값들을 두 번째 인자값으로 구분하여 특정 숫자값으로 변환함
- ✔ 구분하는 기준은 두 번째 인자값

보고 싶은 영화 찾기 | 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ② np.digitize( )

- ✔ np.digitize(data.연령, 구분할 기준)
- ✔ 0부터 5까지 맵핑하여 값을 변환함

```
In [105]: import numpy as np

In [107]: sr = Series([13,15,45,53,1,63])
          np.digitize(sr, [10, 20, 30])

Out[107]: array([1, 1, 3, 3, 0, 3], dtype=int64)

In [108]: np.digitize(data.연령, [10, 20, 30, 40, 50])

Out[108]: array([0, 5, 2, ..., 1, 1, 2], dtype=int64)

In [ ]:
```

~9	→	0
10~19	→	1
20~29	→	2
30~39	→	3
40~49	→	4
50~	→	5

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령대 columns 추가하는 방법 - ② np.digitize( )

✔ 새로운 연령대2 columns으로 할당

Run Code

...

In [103]:
```
# 두번째 방법. np.digitize()
```

In [105]:
```
import numpy as np
```

In [107]:
```
sr = Series([13,15,45,53,1,63])
np.digitize(sr, [10, 20, 30])
```

Out[107]:
```
array([1, 1, 3, 3, 0, 3], dtype=int64)
```

In [109]:
```
data['연령대2'] = np.digitize(data.연령, [10, 20, 30, 40, 50])
```

In [ ]:
```
ㅇㅁㅅ
```

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ map( )

✔ 0, 1, 2, 3, 4, 5을 10대 미만, 10대, 20대, 30대, 40대, 50대 이상으로 맵핑하여 변환

In [105]:
```
import numpy as np
```

In [107]:
```
sr = Series([13,15,45,53,1,63])
np.digitize(sr, [10, 20, 30])
```

Out[107]:
```
array([1, 1, 3, 3, 0, 3], dtype=int64)
```

In [109]:
```
data['연령대2'] = np.digitize(data.연령, [10, 20, 30, 40, 50])
```

In [110]:
```
data
```

Out[110]:

	사용자아이디	성별	연령	직업	지역	영화아이디	평점	타임스탬프	영화제목	장르	연령대	연령대2
0	1	F	1	10	48067	1193	5	978300760	One Flew Over the Cuckoo's Nest (1975)	Drama	10대 미만	0
												50

보고 싶은 영화 찾기 | 연령대 별로 영화 평점 분석하기

◎ map( )

- ✔ 1:1로 값을 치환할 때 사용
- ✔ 0~5 값이 원하는 문자열로 변환됨

```
ze(data.연령, [10, 20, 30, 40, 50])
```

```
data.연령대2.map({
    0: '10대 미만',
    1: '10대',
    2: '20대',
    3: '30대',
    4: '40대',
    5: '50대 이상'
})
```

Out[111]:

```
0           10대 미만
1           50대 이상
2              20대
3              20대
4           50대 이상
              ...
1000204        10대
1000205        30대
1000206        10대
1000207        10대
1000208        20대
Name: 연령대2, Length: 1000209, dtype: object
```

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ map( )

✔ data['연령대2']로 다시 할당

									Cuckoo's Nest (1975)		이상	이상
...	...	...	...	...	...	...	...	...	...	...	...	...
000204	5949	M	18	17	47901	2198	5	958846401	Modulations (1998)	Documentary	10대	10대
000205	5675	M	35	14	30030	2703	3	976029116	Broken Vessels (1998)	Drama	30대	30대
000206	5780	M	18	17	92886	2845	1	958153068	White Boys (1999)	Drama	10대	10대
000207	5851	F	18	20	55410	3607	5	957756608	One Little Indian (1973)	Comedy|Drama|Western	10대	10대
000208	5938	M	25	1	35401	2909	4	957273353	Five Wives, Three Secretaries and Me (1998)	Documentary	20대	20대

연령대2가 동일하게 맵핑된 값으로 변환된 것을 확인해 볼 수 있습니다.

In [ ]:

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 그룹 집계

- ✔ data.pivot_table 사용
- ✔ index = 영화제목, columns = 연령대, aggfunc = mean, values = 평점

										Drama	10대 10대
000207	5851	F	18	20	55410	3607	5	957756608	Indian (1973)	Comedy|Drama|Western	10대 10대
000208	5938	M	25	1	35401	2909	4	957273353	Five Wives, Three Secretaries and Me (1998)	Documentary	20대 20대

000209 rows × 12 columns

In [ ]:
```
# 그룹 집계
data.pivot_table (index = '영화제목', columns = '연령대',
                  aggfunc = 'mean', values = '평점')
```

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 그룹 집계

✔ 연령별평점으로 저장

Run Code

		F	18	20	55410	3607	5	957756608	Indian (1973)	Comedy|Drama|Western	대	대
000208	5938	M	25	1	35401	2909	4	957273353	Five Wives, Three Secretaries and Me (1998)	Documentary	20 대	20 대

000209 rows × 12 columns

In [116]:
```
# 그룹 집계
연령별평점 = data.pivot_table (index = '영화제목', columns = '연령대',
                    aggfunc = 'mean', values = '평점')
```

In [117]:
```
연령별평점
```

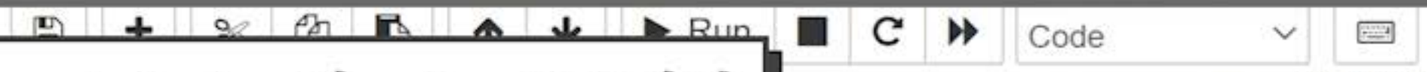
Out[117]:

연령대	10대	10대 미만	20대	30대	40대	50대 이상
영화제목						
$1,000,000 Duck (1971)	3.000000	NaN	3.090909	3.133333	2.000000	2.750000
'Night Mother (1986)	4.666667	2.000000	3.423077	2.904762	3.833333	3.750000

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 그룹 집계

- ✔ sort_index의 axis = 1로 정렬
- ✔ 10대와 10대 미만의 정렬이 되지 않음



35401	2909	4	957273353	Five Wives, Three Secretaries and Me (1998)		Documentary	20대	20대

In [116]:
```
# 그룹 집계
연령별평점 = data.pivot_table (index = '영화제목', columns = '연령대',
                    aggfunc = 'mean', values = '평점')
```

In [118]:
```
연령별평점.sort_index(axis = 1)
```

Out[118]:

연령대	10대	10대 미만	20대	30대	40대	50대 이상
영화제목						
$1,000,000 Duck (1971)	3.000000	NaN	3.090909	3.133333	2.000000	2.750000
'Night Mother (1986)	4.666667	2.000000	3.423077	2.904762	3.833333	3.750000
'Til There Was You (1997)	2.500000	3.500000	2.666667	2.900000	2.333333	2.600000
'burbs, The (1989)	3.244444	4.500000	2.652174	2.818182	2.545455	3.100000

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ 연령별평점에 저장

```
연령별평점 = data.pivot_table (index = '영화제목', columns = '연령대',
                    aggfunc = 'mean', values = '평점')
```

In [120]:
```
연령별평점 = 연령별평점[['10대 미만','10대','20대','30대','40대','50대 이상']]
```

In [121]:
```
연령별평점
```

Out[121]:

연령대	10대 미만	10대	20대	30대	40대	50대 이상
영화제목						
$1,000,000 Duck (1971)	NaN	3.000000	3.090909	3.133333	2.000000	2.750000
'Night Mother (1986)	2.000000	4.666667	3.423077	2.904762	3.833333	3.750000
'Til There Was You (1997)	3.500000	2.500000	2.666667	2.900000	2.333333	2.600000
'burbs, The (1989)	4.500000	3.244444	2.652174	2.818182	2.545455	3.100000
...And Justice for All (1979)	3.000000	3.428571	3.724138	3.657143	4.100000	3.674419
...	...	...	...	...	...	...
Zed & Two Noughts, A (1985)	1.000000	3.000000	3.375000	3.777778	4.000000	3.000000
Zero Effect (1998)	4.125000	3.883333	3.715278	3.608696	3.764706	3.769231

보고 싶은 영화 찾기 연령대 별로 영화 평점 분석하기

◎ NaN : 값이 없는 경우

✔ filllna함수를사용하여 특정 값(-)으로 넣어줌

Out[122]:

연령대	10대 미만	10대	20대	30대	40대	50대 이상
영화제목						
$1,000,000 Duck (1971)	-	3	3.09091	3.13333	2	2.75
'Night Mother (1986)	2	4.66667	3.42308	2.90476	3.83333	3.75
'Til There Was You (1997)	3.5	2.5	2.66667	2.9	2.33333	2.6
'burbs, The (1989)	4.5	3.24444	2.65217	2.81818	2.54545	3.1
...And Justice for All (1979)	3	3.42857	3.72414	3.65714	4.1	3.67442
...	...	...	...	...	...	...
Zed & Two Noughts, A (1985)	1	3	3.375	3.77778	4	3
Zero Effect (1998)	4.125	3.88333	3.71528	3.6087	3.76471	3.76923
Zero Kelvin (Kjærlighetens kjøtere) (1995)	-	-	-	3.5	-	-
Zeus and Roxanne (1997)	1.5	2.5	2.83333	3.5	1	-
eXistenZ (1999)	3.14286	3.28916	3.23497	3.36486	3.22222	3.10345