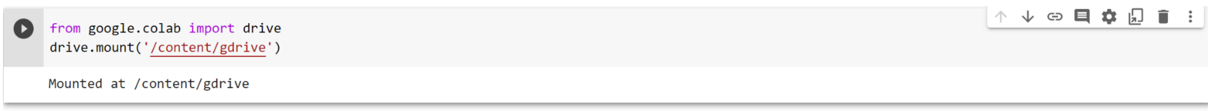
Homework5

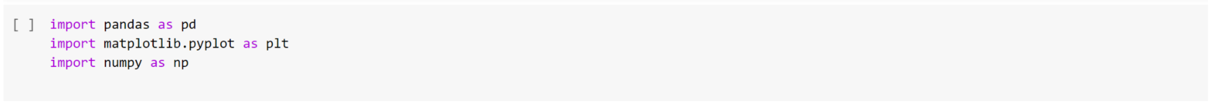
Run the code in today’s lecture with MovieLenz Small dataset. Choose a different movie (any movie) as a base for next recommendations.

Load the data and import modules.

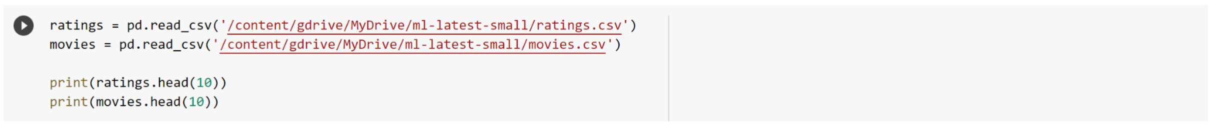


```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```



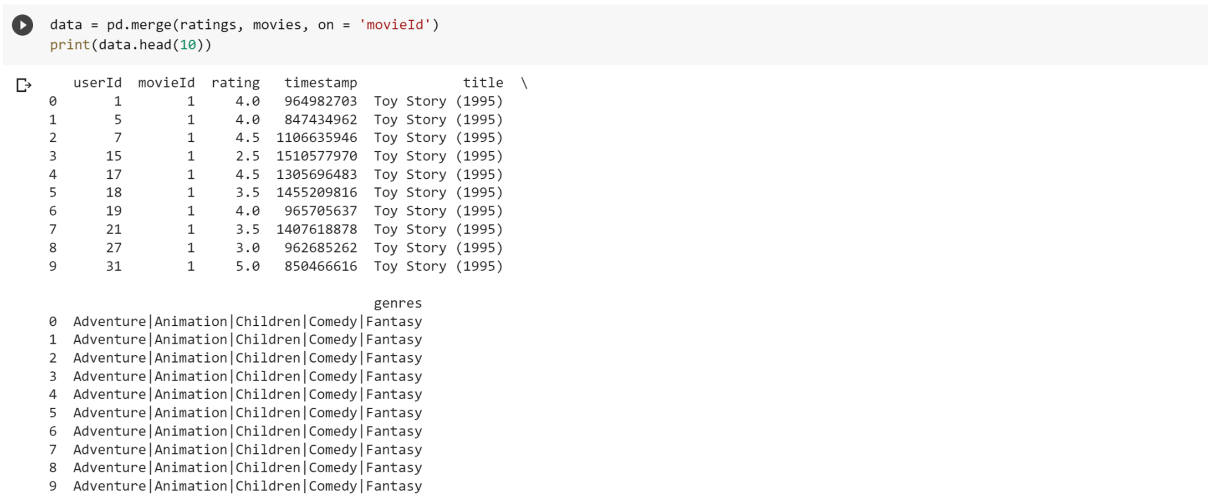
```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    import numpy as np
```



```
ratings = pd.read_csv('/content/gdrive/MyDrive/ml-latest-small/ratings.csv')
movies = pd.read_csv('/content/gdrive/MyDrive/ml-latest-small/movies.csv')

print(ratings.head(10))
print(movies.head(10))
```

Merge ratings and movies data.

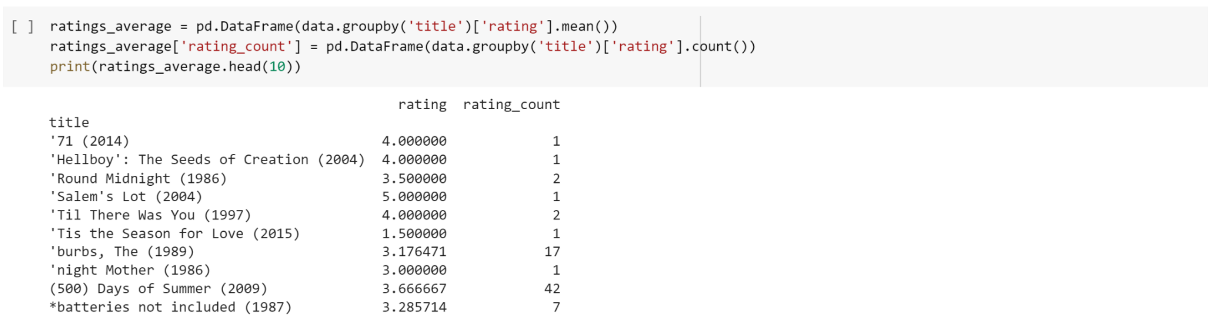


```
data = pd.merge(ratings, movies, on = 'movieId')
print(data.head(10))
```

```
   userId  movieId  rating   timestamp             title  \
0       1        1     4.0   964982703  Toy Story (1995)
1       5        1     4.0   847434962  Toy Story (1995)
2       7        1     4.5  1106635946  Toy Story (1995)
3      15        1     2.5  1510577970  Toy Story (1995)
4      17        1     4.5  1305696483  Toy Story (1995)
5      18        1     3.5  1455209816  Toy Story (1995)
6      19        1     4.0   965705637  Toy Story (1995)
7      21        1     3.5  1407618878  Toy Story (1995)
8      27        1     3.0   962685262  Toy Story (1995)
9      31        1     5.0   850466616  Toy Story (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1  Adventure|Animation|Children|Comedy|Fantasy
2  Adventure|Animation|Children|Comedy|Fantasy
3  Adventure|Animation|Children|Comedy|Fantasy
4  Adventure|Animation|Children|Comedy|Fantasy
5  Adventure|Animation|Children|Comedy|Fantasy
6  Adventure|Animation|Children|Comedy|Fantasy
7  Adventure|Animation|Children|Comedy|Fantasy
8  Adventure|Animation|Children|Comedy|Fantasy
9  Adventure|Animation|Children|Comedy|Fantasy
```

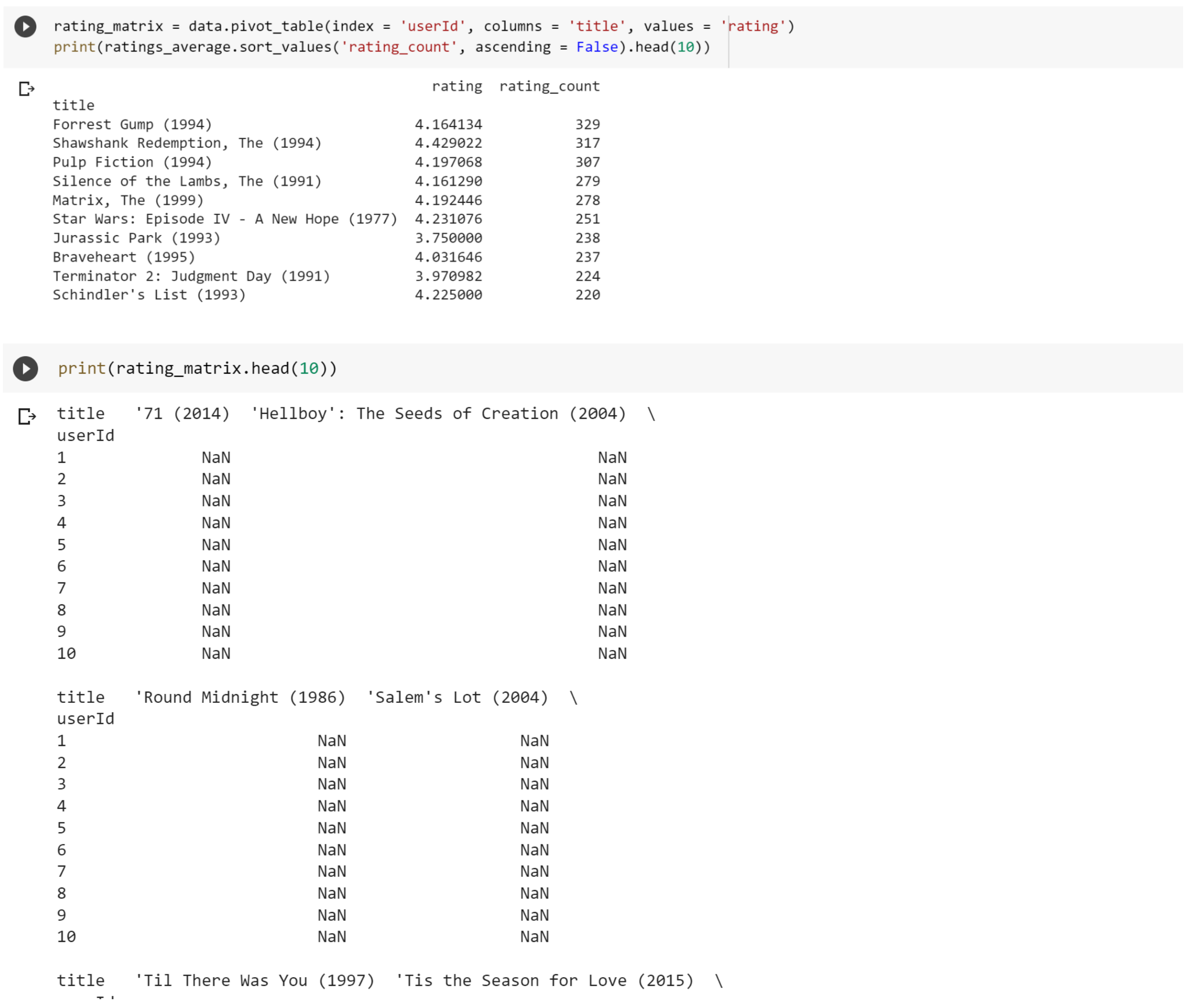
Construct average rating and add ratings count. Then getting familiar with the dataset find highest ratings count.



```
[ ] ratings_average = pd.DataFrame(data.groupby('title')['rating'].mean())
    ratings_average['rating_count'] = pd.DataFrame(data.groupby('title')['rating'].count())
    print(ratings_average.head(10))
```

```
                                         rating  rating_count
title
'71 (2014)                             4.000000             1
'Hellboy': The Seeds of Creation (2004)  4.000000             1
'Round Midnight (1986)                 3.500000             2
'Salem's Lot (2004)                    5.000000             1
'Til There Was You (1997)              4.000000             2
'Tis the Season for Love (2015)        1.500000             1
'burbs, The (1989)                     3.176471            17
'night Mother (1986)                   3.000000             1
(500) Days of Summer (2009)            3.666667            42
*batteries not included (1987)         3.285714             7
```

Turn ratings data into a user-item matrix and show the matrix's information.



```
rating_matrix = data.pivot_table(index = 'userId', columns = 'title', values = 'rating')
print(ratings_average.sort_values('rating_count', ascending = False).head(10))
```

```
                                           rating  rating_count
title
Forrest Gump (1994)                      4.164134           329
Shawshank Redemption, The (1994)         4.429022           317
Pulp Fiction (1994)                      4.197068           307
Silence of the Lambs, The (1991)         4.161290           279
Matrix, The (1999)                       4.192446           278
Star Wars: Episode IV - A New Hope (1977) 4.231076          251
Jurassic Park (1993)                     3.750000           238
Braveheart (1995)                        4.031646           237
Terminator 2: Judgment Day (1991)        3.970982           224
Schindler's List (1993)                  4.225000           220
```

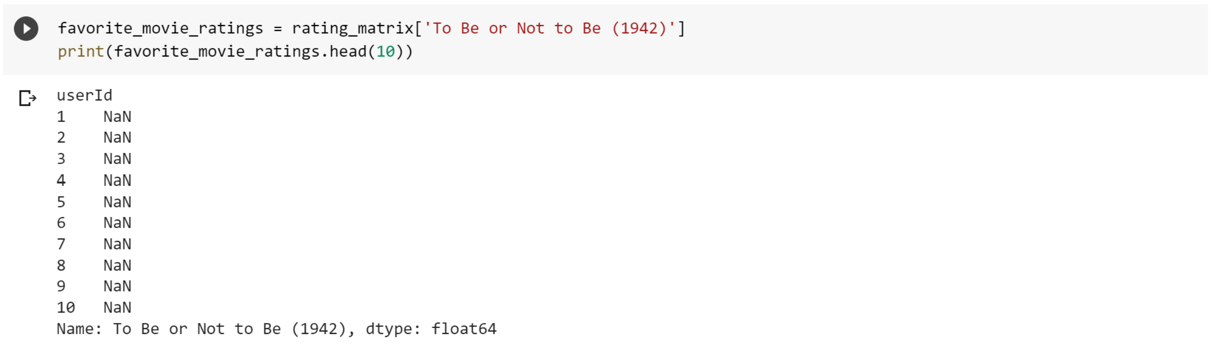
```
print(rating_matrix.head(10))
```

```
title   '71 (2014)  'Hellboy': The Seeds of Creation (2004)  \
userId
1              NaN                                      NaN
2              NaN                                      NaN
3              NaN                                      NaN
4              NaN                                      NaN
5              NaN                                      NaN
6              NaN                                      NaN
7              NaN                                      NaN
8              NaN                                      NaN
9              NaN                                      NaN
10             NaN                                      NaN

title   'Round Midnight (1986)  'Salem's Lot (2004)  \
userId
1                          NaN                  NaN
2                          NaN                  NaN
3                          NaN                  NaN
4                          NaN                  NaN
5                          NaN                  NaN
6                          NaN                  NaN
7                          NaN                  NaN
8                          NaN                  NaN
9                          NaN                  NaN
10                         NaN                  NaN

title   'Til There Was You (1997)  'Tis the Season for Love (2015)  \
```

I selected ‘To Be or Not to Be (1942)’ and tried to find similar movies of it.



```
favorite_movie_ratings = rating_matrix['To Be or Not to Be (1942)']
print(favorite_movie_ratings.head(10))
```

```
userId
1     NaN
2     NaN
3     NaN
4     NaN
5     NaN
6     NaN
7     NaN
8     NaN
9     NaN
10    NaN
Name: To Be or Not to Be (1942), dtype: float64
```

Remove empty values, and add correlation column labels and rating counts. Then see the highest correlation again.

```
[ ] similar_movies = rating_matrix.corrwith(favorite_movie_ratings)

    correlation = pd.DataFrame(similar_movies, columns = ['Correlation'])
    correlation.dropna(inplace = True)
```

```
/usr/local/lib/python3.7/dist-packages/numpy/lib/function_base.py:2683: RuntimeWarning: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.7/dist-packages/numpy/lib/function_base.py:2542: RuntimeWarning: divide by zero encountered in true_divide
  c *= np.true_divide(1, fact)
```

```
correlation = correlation.join(ratings_average['rating_count'])
print(correlation.sort_values('Correlation', ascending = False).head(10))
```

```
                              Correlation  rating_count
title
(500) Days of Summer (2009)           1.0            42
Juno (2007)                           1.0            65
Midnight in Paris (2011)              1.0            25
Meet Me in St. Louis (1944)           1.0            10
Mean Girls (2004)                     1.0            39
Love and Other Drugs (2010)           1.0             7
Love Actually (2003)                  1.0            59
Little Miss Sunshine (2006)           1.0            77
Legally Blonde (2001)                 1.0            64
Lady and the Tramp (1955)             1.0            55
```

Limit only to highly correlated movies with at least 40 rating counts. Then, got the recommendations.

```
[ ] recommendation = correlation[correlation['rating_count'] > 40].sort_values('Correlation', ascending = False)

    print(recommendation.head(10))
```

```
                                                    Correlation  rating_count
title
(500) Days of Summer (2009)                                 1.0            42
Monty Python's Life of Brian (1979)                         1.0            89
Clueless (1995)                                             1.0           104
Hangover, The (2009)                                        1.0            76
Hitch (2005)                                                1.0            45
101 Dalmatians (One Hundred and One Dalmatians)...          1.0            44
Knocked Up (2007)                                           1.0            52
Kung Fu Panda (2008)                                        1.0            54
Lady and the Tramp (1955)                                   1.0            55
Legally Blonde (2001)                                       1.0            64
```

```
recommendation = recommendation.merge(movies, on = 'title')

print(recommendation.head(10))
```

```
                                               title  Correlation  \
0                        (500) Days of Summer (2009)          1.0
1                Monty Python's Life of Brian (1979)          1.0
2                                    Clueless (1995)          1.0
3                               Hangover, The (2009)          1.0
4                                       Hitch (2005)          1.0
5  101 Dalmatians (One Hundred and One Dalmatians...          1.0
6                                  Knocked Up (2007)          1.0
7                                Kung Fu Panda (2008)          1.0
8                          Lady and the Tramp (1955)          1.0
9                              Legally Blonde (2001)          1.0

   rating_count  movieId                                genres
0            42    69757                  Comedy|Drama|Romance
1            89     1080                                Comedy
2           104       39                        Comedy|Romance
3            76    69122                          Comedy|Crime
4            45    31685                        Comedy|Romance
5            44     2085          Adventure|Animation|Children
6            52    52973                  Comedy|Drama|Romance
7            54    59784  Action|Animation|Children|Comedy|IMAX
8            55     2080      Animation|Children|Comedy|Romance
9            64     4447                        Comedy|Romance
```

Results show that each recommendation movie has high correlation scores. In addition, the genres of these recommended movies have in common are all Comedy and Romance.