Download the files and study the structure. Then create a simple content-based recommendation system. If this data proves to be too big for processing, you can use this data Download this data, which is contains a small sample of user, items, and ratings.

If you use the smaller dataset, note that the recommendation function, as written in class, assumes that user-id corresponds to the row (minus one) in the user-item matrix. For example, if there are 500 users, they would have user-ids 1-500. This is not the case for the book rating data–user-ids vary a lot. As such, passing in "1" to the recommendation function does not actually retrieve recommendations for user with user-id=1. This is okay for this assignment. In other words, feel free to use "user-ids" in the range [1, 500] as arguments for your recommendation function, even if those user-ids are not in the users_small.csv file. If you'd like, it should be fairly straightforward to create a mapping between original user-id and row, but this is not necessary and would require you to modify the example code from class.

Steps to follow:

- Convert each book title, author, publisher into appropriate representations (dataframe/matrix)
- Create user-book rating matrix
- Normalize the ratings
- Predict the items

Finally, recommend a list of 10 books to a particular user based on their profile. Describe the process you followed for building this recommender system in up to a page. Then provide results (top-10 books) for three different users.

```
import pandas as pd
import numpy as np
```

```
# Step-1: Extract relevant data about user, movies, and ratings.
users = pd.read_csv('C:/Users/Fortu/Downloads/Aut2024/Info 376/assignment8_small/users_small.csv'
```

```
users.head(5)
```

	Location	Age
User-ID		
254	minneapolis, minnesota, usa	24.0
3363	knoxville, tennessee, usa	29.0
4017	new orleans, louisiana, usa	48.0
6242	calgary, alberta, canada	NaN
6251	wahiawa, hawaii, usa	32.0

```
users.shape
```

```
(500, 3)
```

```
books = pd.read_csv('C:/Users/Fortu/Downloads/Aut2024/Info 376/assignment8_small/books_small.csv'
```

```
books.head(5)
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
0	1841721522	New Vegetarian: Bold and Beautiful Recipes for...	Celia Brooks Brown	2001	Ryland Peters & Small Ltd
1	042518630X	Purity in Death	J.D. Robb	2002	Berkley Publishing Group
2	042511774X	Breathing Lessons	Anne Tyler	1994	Berkley Publishing Group
3	1853260053	Tess of the D'Urbervilles (Wordsworth Classics)	Thomas Hardy	1997	NTC/Contemporary Publishing Company
4	1558531025	Life's Little Instruction Book (Life's Little ...	H. Jackson Brown	1991	Thomas Nelson

```
books.shape
```

```
(1000, 5)
```

```
ratings = pd.read_csv('C:/Users/Fortu/Downloads/Aut2024/Info 376/assignment8_small/ratings_small.
```

```
ratings.head(5)
```

	User-ID	ISBN	Book-Rating
0	277427	002542730X	10
1	277427	006092988X	0
2	277427	006440188X	0
3	277427	014029628X	0
4	277427	014100018X	0

```
ratings.shape
```

```
(16978, 3)
```

```
# Construct user-movie rating matrix by doing pivot (it's like a join operation in databases)
u_b = ratings.pivot(index='User-ID', columns='ISBN', values='Book-Rating').fillna(0)
```

```
u_b.head(5)
```

ISBN	000649840X	002026478X	002542730X	006000438X	006000925X	006001203X	006008460X	006015957X
User-ID								
254	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3363	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4017	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0
6242	0.0	0.0	0.0	6.0	0.0	0.0	0.0	0.0
6251	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 964 columns

```
u_b.shape
```

```
(500, 964)
```

```
# Step-3: normalize ratings.
data = u_b.values
user_mean = np.mean(data, axis=1)
normalizedRating = data - user_mean.reshape(-1, 1)
```

```
normalizedRating
```

```
array([[ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [-0.03112033, -0.03112033, -0.03112033, ..., -0.03112033,
        -0.03112033, -0.03112033],
       [-0.14522822, -0.14522822, -0.14522822, ..., -0.14522822,
        -0.14522822, -0.14522822],
       ...,
       [-0.05186722, -0.05186722, -0.05186722, ..., -0.05186722,
        -0.05186722, -0.05186722],
       [-0.03838174, -0.03838174,  9.96161826, ..., -0.03838174,
        -0.03838174, -0.03838174],
       [-0.01244813, -0.01244813, -0.01244813, ..., -0.01244813,
        -0.01244813, -0.01244813]])
```

```
# Step-4: SVD (matrix factorization)
from scipy.sparse.linalg import svds
U, sigma, Vt = svds(normalizedRating, k=100)
```

```
U.shape
```

```
(500, 100)
```

```
sigma.shape
```

```
(100,)
```

```
Vt.shape
```

```
(100, 964)
```

```
sigma
```

```
array([ 26.43195204,  26.68504208,  26.74594046,  26.85117423,
        26.99996835,  27.0561371 ,  27.19895426,  27.33607412,
        27.59199551,  27.7089803 ,  27.89938401,  28.0263381 ,
        28.32697977,  28.42236875,  28.53531896,  28.654801  ,
        28.80970331,  28.92024512,  29.02761977,  29.25160491,
        29.46863514,  29.72484796,  29.90955785,  30.04479147,
        30.14579294,  30.33305962,  30.51093501,  30.73324233,
        30.86331269,  31.02187005,  31.24752709,  31.37998539,
        31.46450462,  31.55712966,  31.79066699,  32.17806829,
        32.3050293 ,  32.55163507,  32.72159516,  32.82105312,
        32.9308962 ,  33.18670547,  33.44946591,  33.68172438,
        33.86224072,  33.9396138 ,  34.1140978 ,  34.41220363,
        34.61691381,  34.74153984,  35.09233114,  35.3171815 ,
        35.45731077,  35.61641697,  35.90460137,  36.3093148 ,
        36.62574802,  36.99878055,  37.20192566,  37.48929607,
        37.85786914,  37.96619493,  38.26472801,  38.29785826,
        38.75754013,  39.34198743,  39.5986264 ,  39.92380484,
        40.11812941,  40.9791503 ,  41.0161435 ,  41.23922786,
        41.85753838,  42.2689375 ,  42.82210931,  43.0640142 ,
        43.6223951 ,  43.88903917,  44.24126528,  44.71193125,
        45.61687121,  46.15308101,  47.10780158,  47.70809467,
        47.97369875,  49.46171733,  50.03802372,  51.57132807,
        52.30295579,  54.08688333,  55.83017075,  57.0415622 ,
        58.40339825,  60.02248578,  63.24717078,  68.26293201,
        71.80912297,  83.25928575,  84.83702201, 122.05769804])
```

```
# Step-5: get the diagonal entries of sigma (singular values)
sigma = np.diag(sigma)
```

```
sigma
```

```
array([[ 26.43195204,   0.        ,   0.        , ...,   0.        ,
          0.        ,   0.        ],
       [  0.        ,  26.68504208,   0.        , ...,   0.        ,
          0.        ,   0.        ],
       [  0.        ,   0.        ,  26.74594046, ...,   0.        ,
          0.        ,   0.        ],
       ...,
       [  0.        ,   0.        ,   0.        , ...,  83.25928575,
          0.        ,   0.        ],
       [  0.        ,   0.        ,   0.        , ...,   0.        ,
         84.83702201,   0.        ],
       [  0.        ,   0.        ,   0.        , ...,   0.        ,
          0.        , 122.05769804]])
```

```
sigma.shape
```

```
(100, 100)
```

```
# Step-6: reshape and renormalize ratings using weighted/important components/factors
all_user_ratings = np.dot(np.dot(U,sigma), Vt) + user_mean.reshape(-1,1)
```

```
all_user_ratings
```

```
array([[-4.15020960e-16, -6.38715961e-16, -7.83784419e-17, ...,
        -2.84218837e-16,  4.68750696e-16,  4.60132069e-17],
       [ 3.94638067e-01,  1.02567291e-01, -1.61408076e-01, ...,
         6.56347940e-02, -3.17873755e-02,  1.13543060e-02],
       [-8.67669831e-02,  6.21203269e-01, -4.41078942e-01, ...,
        -1.41546026e-02,  6.80171126e-03,  4.86485363e-03],
       ...,
       [ 1.49452756e-01, -2.74330296e-02,  3.51373188e-01, ...,
         2.38504573e-02, -9.01244560e-02, -1.00170246e-02],
       [ 7.66655476e-03,  9.21588105e-02,  6.86688901e+00, ...,
         3.09078466e-02,  7.77026440e-02,  6.59656321e-02],
       [-8.52392218e-02,  7.58323659e-02, -1.33296922e-01, ...,
         1.87824537e-02, -8.19790438e-02, -2.71880584e-02]])
```

```
# Step-7: construct a dataframe with all these user-movie ratings predictions.
preds = pd.DataFrame(all_user_ratings, columns=u_b.columns)
```

```
preds.head(5)
```

ISBN	000649840X	002026478X	002542730X	006000438X	006000925X	006001203X	006008460X	006015957X
0	-4.150210e-16	-6.387160e-16	-7.837844e-17	-7.513176e-16	-2.380380e-16	1.523474e-16	-5.476965e-16	4.601321e-17
1	3.946381e-01	1.025673e-01	-1.614081e-01	-2.199430e-01	-5.954132e-02	9.328870e-02	-1.330330e-01	1.135431e-02
2	-8.676698e-02	6.212033e-01	-4.410789e-01	8.740804e+00	4.162594e-01	1.846466e-01	-2.487635e-01	4.864854e-03
3	-4.961971e-02	-3.297501e-01	3.528934e-01	4.065511e+00	-1.657962e-02	-3.849007e-01	2.392250e-01	1.399301e-02
4	-6.268908e-01	2.071848e-01	6.170809e-01	2.011453e-01	-8.195056e-02	3.711519e-03	3.428405e-01	2.126359e-02
...	...	...	...	...	...	...	...	...
495	3.029857e-02	-2.489660e-01	-7.388572e-02	-3.973263e-01	3.202351e-01	1.746246e-01	1.074263e-01	5.018372e-03
496	-6.120009e-02	2.329013e-04	-2.831157e-02	-5.537408e-02	1.835343e-02	7.062503e-02	2.276990e-02	3.805583e-02
497	1.494528e-01	-2.743303e-02	3.513732e-01	-6.861142e-01	-5.201962e-02	1.460536e-01	-3.534755e-01	-1.001702e-02
498	7.666555e-03	9.215881e-02	6.866889e+00	-1.607331e-02	7.188330e-02	1.484760e-01	3.667571e-01	6.596563e-02
499	-8.523922e-02	7.583237e-02	-1.332969e-01	-6.168739e-02	-4.910069e-02	-1.516259e-01	-1.477271e-01	-2.718806e-02

500 rows × 964 columns

```
preds.shape
```

```
(500, 964)
```

```
preds.to_csv('C:/Users/Fortu/Downloads/Aut2024/Info 376/assignment8_small/predictions.csv', index=
```

```
# Given a matrix of predictions for book ratings, produce recommendations for a given user
def recommend_books(predictions_df, userId, users_df, books_df, original_ratings_df, num_recs):
  # Get and sort the user's predictions
  user_row_number = userId - 1  # userId starts at 1, not 0
  sorted_user_predictions = predictions_df.iloc[user_row_number].sort_values(ascending=False)
```

```
    # Get the user's data and merge in the book information
    user_data = original_ratings_df[original_ratings_df['User-ID'] == users.index[user_row_number]]
    user_full = (user_data.merge(books_df, how='left', left_on='ISBN', right_on='ISBN')
           .sort_values(['Book-Rating'], ascending=False))
    print('User {0} has already rated {1} books.'.format(userId, user_full.shape[0]))
    print('Recommending highest {0} predicted ratings books not already rated.'.format(num_recs))

    # Recommend the highest predicted rating books that the user hasn't seen yet.
    recommendations = (books_df[~books_df['ISBN'].isin(user_full['ISBN'])]
                .merge(pd.DataFrame(sorted_user_predictions).reset_index(), how='left', left_on='ISBI
                .rename(columns={user_row_number: 'Predictions'})
                .sort_values('Predictions', ascending=False)
                .iloc[:num_recs, :-1])
    return user_full, recommendations
```

```
# What we want to do
already_rated, recommendations = recommend_books(preds, 1, users, books, ratings, 10)
```

```
User 1 has already rated 16 books.
Recommending highest 10 predicted ratings books not already rated.
```

```
already_rated.head(10)
```

	User-ID	ISBN	Book-Rating	Book-Title	Book-Author	Year-Of-Publication	Publisher
0	254	006000438X	0	The Death of Vishnu: A Novel	Manil Suri	2002	Perennial
1	254	014028009X	0	Bridget Jones's Diary	Helen Fielding	1999	Penguin Books
2	254	014100018X	0	Chocolat	Joanne Harris	2000	Penguin Books
3	254	038533348X	0	Cat's Cradle	Kurt Vonnegut	1998	Delta
4	254	038542017X	0	Like Water for Chocolate : A Novel in Monthly ...	LAURA ESQUIVEL	1995	Anchor
5	254	043935806X	0	Harry Potter and the Order of the Phoenix (Boo...	J. K. Rowling	2003	Scholastic
6	254	044018293X	0	Suffer the Children	John Saul	1986	Dell
7	254	051510521X	0	Bitter Sweet	LaVyrle Spencer	1995	Jove Books
8	254	051512317X	0	Rising Tides	Nora Roberts	2001	Jove Books
9	254	051513287X	0	Face the Fire (Three Sisters Island Trilogy)	Nora Roberts	2002	Jove Books

```
recommendations.head(10)
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
679	044098761X	A Wind in the Door (Laurel Leaf Books)	Madeleine L'Engle	1976	Laure Leaf
409	044022103X	One True Thing	Anna Quindlen	1995	Dell
382	1558745718	Chicken Soup for the Pet Lover's Soul (Chicken...	Jack Canfield	1998	Health Communications
431	080411109X	The Hundred Secret Senses	Amy Tan	1996	Ivy Books
223	1576737330	The Prayer of Jabez: Breaking Through to the B...	Bruce Wilkinson	2000	Multnomah
337	034542705X	The Man Who Listens to Horses	Monty Roberts	1998	Ballantine Books
192	067102423X	Bag of Bones	Stephen King	1999	Pocket
331	044022392X	The Keys to the Street: A Novel of Suspense	Ruth Rendell	1997	Dell
334	067151699X	The HIDDEN LIFE OF DOGS	Elizabeth Marshall Thomas	1995	Pocket
198	1573225517	High Fidelity	Nick Hornby	1996	Riverhead Books