Recommendations_with_IBM

November 23, 2021

1 Recommendations with IBM

In this notebook, you will be putting your recommendation skills to use on real data from the IBM Watson Studio platform.

You may either submit your notebook through the workspace here, or you may work from your local machine and submit through the next page. Either way assure that your code passes the project RUBRIC. **Please save regularly.**

By following the table of contents, you will build out a number of different methods for making recommendations that can be used for different situations.

1.1 Table of Contents

At the end of the notebook, you will find directions for how to submit your work. Let's get started by importing the necessary libraries and reading in the data.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import project_tests as t
        import pickle

        %matplotlib inline

        df = pd.read_csv('data/user-item-interactions.csv')
        df_content = pd.read_csv('data/articles_community.csv')
        del df['Unnamed: 0']
        del df_content['Unnamed: 0']

        # Show df to get an idea of the data
        df.head()
```

```
Out[1]:    article_id                                              title  \
        0      1430.0  using pixiedust for fast, flexible, and easier...
        1      1314.0       healthcare python streaming application demo
        2      1429.0         use deep learning for image classification
        3      1338.0          ml optimization using cognitive assistant
```

```
4      1276.0          deploy your python model as a restful api

                                         email
0  ef5f11f77ba020cd36e1105a00ab868bbdbf7fe7
1  083cbdfa93c8444beaa4c5f5e0f5f9198e4f9e0b
2  b96a4f2e92d8572034b1e9b28f9ac673765cd074
3  06485706b34a5c9bf2a0ecdac41daf7e7654ceb7
4  f01220c46fc92c6e6b161b1849de11faacd7ccb2
```

In [3]:
```
df.shape[0]
```

Out[3]:
```
45993
```

In [4]:
```
df_content.shape[0]
```

Out[4]:
```
1056
```

In [29]:
```
# Show df_content to get an idea of the data
df_content.head()
```

Out[29]:
```
                                            doc_body  \
0  Skip navigation Sign in SearchLoading...\r\n\r...
1  No Free Hunch Navigation * kaggle.com\r\n\r\n ...
2   * Login\r\n * Sign Up\r\n\r\n * Learning Pat...
3  DATALAYER: HIGH THROUGHPUT, LOW LATENCY AT SCA...
4  Skip navigation Sign in SearchLoading...\r\n\r...

                                     doc_description  \
0  Detect bad readings in real time using Python ...
1  See the forest, see the trees. Here lies the c...
2  Heres this weeks news in Data Science and Bi...
3  Learn how distributed DBs solve the problem of...
4  This video demonstrates the power of IBM DataS...

                                       doc_full_name doc_status  article_id
0  Detect Malfunctioning IoT Sensors with Streami...       Live           0
1  Communicating data science: A guide to present...       Live           1
2         This Week in Data Science (April 18, 2017)       Live           2
3  DataLayer Conference: Boost the performance of...       Live           3
4      Analyze NY Restaurant data using Spark in DSX       Live           4
```

1.1.1 Part I : Exploratory Data Analysis

Use the dictionary and cells below to provide some insight into the descriptive statistics of the data.

1. What is the distribution of how many articles a user interacts with in the dataset? Provide a visual and descriptive statistics to assist with giving a look at the number of times each user interacts with an article.

a. Exploratory analysis about "number of users interact with an article"

- Number of times each user interacts with an article

```
In [6]: df.groupby(['article_id','email']).size()
```

```
Out[6]: article_id  email
        0.0         2841916b462a2b89d36f4f95ca2d1f42559a5788    1
                    384255292a8223e84f05ca1e1deaa450c993e148    3
                    451a9a4a4cb1cc4e5f38d04e8859cc3fb275cc66    1
                    74ca1ae8b034f7fad73a54d55fb1f58747f00493    1
                    8bd0afc488016810c287ac4ec844895d570b0af4    1
                    a60b7e945a8f2114d5dfbdd53182ad1d526534e2    1
                    ad06c765d31179e56f309438367ecb30e1059620    1
                    ca7d48adf2c7394ed5a8776de959fa8047e43d4b    1
                    db8ac9b2f552db35750239ada8bfcb59b3ae48c0    1
                    df722d3aac72766b93d4a65d8b4ac084a968d684    1
                    e667c9a1cd56368dfa2f4b974ab2d848585552d7    1
                    e6ed9e15addba353fe3c1f36d865a63fa254b9cc    1
        2.0         0246d11c827f90850ce7062e9554c9d5eeb30027    1
                    0286bfe26356436658cf4b29b232f0700f0bb9ce    2
                    12815feeacc6f27dff5b3441a54418d2d51001ef    1
                    12bb8a9740400ced27ae5a7d4c990ac3b7e3c77d    1
                    15a1660b6450e064200f1272d9b3d049cf8cf5f1    1
                    1d74fc07ef225ff993b9f80dfba85a6bd2bd55b8    1
                    249d60fc4edda28cd8fd76f549ecc43259e07038    1
                    26b8f921fac7a4d81f2749d64c10020491281545    1
                    2b6c0f514c2f2b04ad3c4583407dccd0810469ee    3
                    2f5c7feae533ce046f2cb16fb3a29fe00528ed66    1
                    3427a5a4065625363e28ac8e85a57a9436010e9c    3
                    387f29d1e6f4360fa1a2c9607edfa184520bd716    1
                    3e9be703aad3a99412af09cdefcdf28fe5ff2a32    1
                    40222b846f3cef9a645dfb34fc15f7c1c244e393    1
                    40a942b88fbc6b891eb335e12fbc589870098153    1
                    449235d86e9d80f808112130da55d08bfb41703d    1
                    4594c0796c9f32915d3f5c05c2cc5378ffcd40b4    1
                    497935037e41a94d2ae02488d098c7abda9a30bc    1
                                                                ..
        1440.0      a4ce0a47da79499778f8cf4c94fd5cd0fde39002    1
                    aaaf1a153d6b2f025b275c68324b6dd6d15f22ff    2
                    e685741240520687a02b033e21938ddf3acdab7f    2
        1441.0      3c3b60c3fb094373d6aa9420b2f8c08cd6a23354    1
                    3cdbb321cad01f39848fcde8288109a73ee7febb    2
                    436337957e43e0b1db33a58e87971319214d03a9    2
                    6cce7568da5452718e1a3702edffac34a8da74ec    1
                    8c37a3959fb30f349adff02cc545907dafd41b2a    1
                    d5843ed71361c87b364f578f20a48101289d60f9    1
        1442.0      0bd8ddb0c5cec4623a6bb663592747ad55477680    1
                    21026417853cd181c161ae20651318978bf177b0    1
                    5ad53e0336bcd4aef788745048a451f9c383bf81    1
```

```
                66fd330648798030d5fe39b4ea5b8f61e618eb6e     1
1443.0          32c368e390424c9326f736d32725e1e167abcbe4     2
                3d840ffab77365a1b2e5ccd464bd439a347d6105     1
                4ac1eb6a4d2dfbd4e18042c0908de9d3dd7c39b0     2
                4accff186b3c4fb061b11ad6ae1920556bd68382     2
                6cce7568da5452718e1a3702edffac34a8da74ec     1
                8a7982ea3a9d4fe4fd7389e0a94a539ccc81a7bc     1
                9676b33c28cfc8f8dea534ebfd26ec140f8e442f     5
                c20e17b171ff2a34b4d684b4fe3af3be7e0700f3     2
                d53071c9f495307a7a2b9d3619f7ec6e8721e1b7     1
                d5843ed71361c87b364f578f20a48101289d60f9     1
                ecee962fad63a7828319b3aa0a6557f94fc5691e     1
                fa19d1470e496ac4a7bd0eeb07d17b2b3a2f9e30     3
1444.0          6cce7568da5452718e1a3702edffac34a8da74ec     1
                c45f9495a76bf95d2633444817f1be8205ad542d     1
                d313c83ab3ed388ba16042a6cd33fce57d6a9e9a     1
                d5843ed71361c87b364f578f20a48101289d60f9     1
                fd824fc62b4753107e3db7704cd9e8a4a1c961f1     1
Length: 33669, dtype: int64
```

- Number of users interact with an article

In [7]:
```
article_interact_unique = df.groupby(['article_id'])['email'].nunique()
article_interact_unique
```

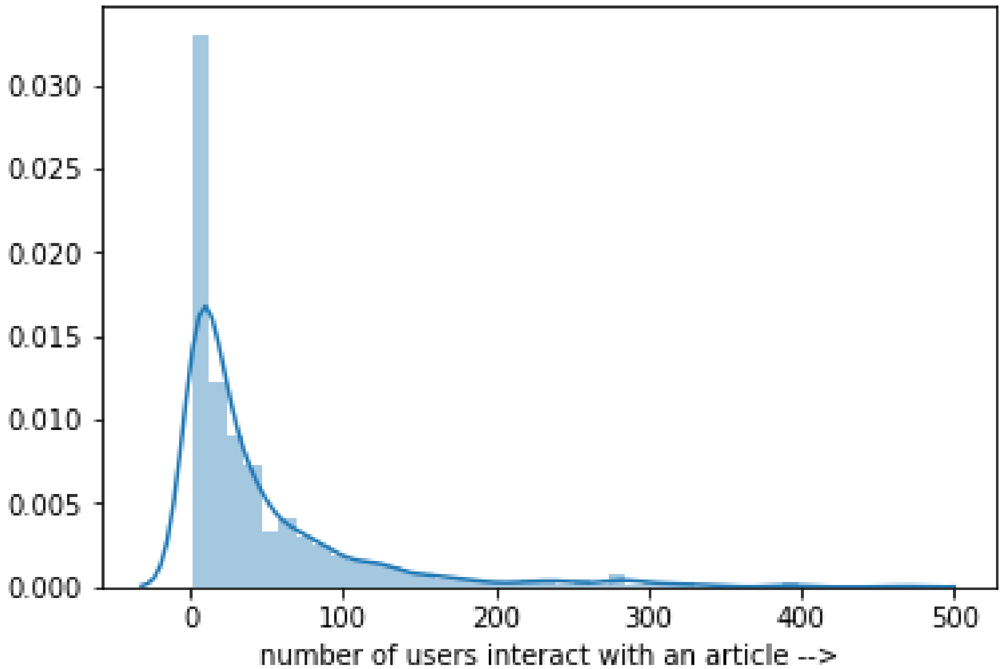
Out[7]:
```
article_id
0.0        12
2.0        44
4.0        13
8.0        82
9.0        10
12.0       99
14.0       89
15.0       26
16.0       56
18.0       68
20.0      186
25.0       15
26.0       80
28.0       39
29.0       41
30.0       17
32.0       60
33.0      109
34.0       86
36.0       18
39.0       59
40.0       64
```

```
43.0      299
48.0       11
50.0       69
51.0      107
53.0       93
54.0       20
57.0      128
58.0       11
          ...
1412.0     19
1414.0      4
1415.0     10
1416.0     73
1418.0     41
1419.0      6
1420.0     94
1421.0      3
1422.0    105
1423.0    102
1424.0    115
1425.0     57
1426.0     96
1427.0    308
1428.0     91
1429.0    397
1430.0    237
1431.0    320
1432.0    232
1433.0     86
1434.0     36
1435.0     75
1436.0    282
1437.0    127
1439.0     43
1440.0      8
1441.0      6
1442.0      4
1443.0     12
1444.0      5
Name: email, Length: 714, dtype: int64
```

- The distribution of "how many user interact with an article"

In [9]:
```
x2 = pd.Series(article_interact_unique, name="number of users interact with an article -
sns.distplot(x2)
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1f266cd470>
```



b. Exploratory analysis about "number of articles a user interacts with"

- Number of times a user interacts with articles

```
In [10]: df.groupby(['email','article_id']).size()
```

```
Out[10]: email                                     article_id
         0000b6387a0366322d7fbfc6434af145adf7fed1  43.0          2
                                                   124.0         1
                                                   173.0         1
                                                   288.0         1
                                                   349.0         1
                                                   618.0         1
                                                   732.0         1
                                                   1162.0        1
                                                   1232.0        1
                                                   1314.0        1
                                                   1337.0        1
                                                   1354.0        1
         001055fc0bb67f71e8fa17002342b256a30254cd  124.0         1
                                                   254.0         1
                                                   390.0         1
                                                   1386.0        1
         00148e4911c7e04eeff8def7bbbdaf1c59c2c621  258.0         1
                                                   932.0         1
                                                   1386.0        1
```

```
001a852ecbd6cc12ab77a785efa137b2646505fe  232.0          1
                                          349.0          1
                                          593.0          1
                                          957.0          1
                                          1364.0         2
001fc95b90da5c3cb12c501d201a915e4f093290  379.0          1
                                          1364.0         1
0042719415c4fca7d30bd2d4e9d17c5fc570de13  20.0           1
                                          1060.0         1
00772abe2d0b269b2336fc27f0f4d7cb1d2b65d7  732.0          1
                                          1427.0         2
                                                        ..
ffe3d0543c9046d35c2ee3724ea9d774dff98a32  617.0          1
                                          701.0          1
                                          727.0          1
                                          782.0          1
                                          784.0          1
                                          878.0          1
                                          943.0          1
                                          986.0          1
                                          1047.0         1
                                          1162.0         1
                                          1165.0         1
                                          1314.0         2
                                          1360.0         2
                                          1386.0         1
                                          1422.0         3
                                          1425.0         1
                                          1427.0         1
fff9fc3ec67bd18ed57a34ed1e67410942c4cd81  116.0          1
                                          232.0          1
                                          268.0          2
                                          525.0          1
                                          684.0          3
                                          962.0          1
                                          1431.0         1
fffb93a166547448a0ff0232558118d59395fecd  329.0          1
                                          981.0          1
                                          1304.0         1
                                          1305.0         8
                                          1430.0         1
                                          1437.0         1
Length: 33669, dtype: int64
```

- Number of articles a user interacts with.

In [11]:
```
user_interact_unique = df.groupby(['email'])['article_id'].nunique()
user_interact_unique
```

```
Out[11]: email
0000b6387a0366322d7fbfc6434af145adf7fed1    12
001055fc0bb67f71e8fa17002342b256a30254cd     4
00148e4911c7e04eeff8def7bbbdaf1c59c2c621     3
001a852ecbd6cc12ab77a785efa137b2646505fe     5
001fc95b90da5c3cb12c501d201a915e4f093290     2
0042719415c4fca7d30bd2d4e9d17c5fc570de13     2
00772abe2d0b269b2336fc27f0f4d7cb1d2b65d7     2
008ba1d5b4ebf54babf516a2d5aa43e184865da5    10
008ca24b82c41d513b3799d09ae276d37f92ce72     1
008dfc7a327b5186244caec48e0ab61610a0c660    10
009af4e0537378bf8e8caf0ad0e2994f954d822e     1
00bda305223d05f6df5d77de41abd2a0c7d895fe     4
00c2d5190e8c6b821b0e3848bf56f6e47e428994     3
00ced21f957bbcee5edf7b107b2bd05628b04774     4
00d9337ecd5f70fba1c4c7a78e21b3532e0112c4     1
00e524e4f13137a6fac54f9c71d7769c6507ecde     8
00f8341cbecd6af00ba8c78b3bb6ec49adf83248     2
00f946b14100f0605fa25089437ee9486378872c     1
01041260c97ab9221d923b0a2c525437f148d589     2
0108ce3220657a9a89a85bdec959b0f2976dd51c     3
011455e91a24c1fb815a4deac6b6eaf5ad16819e     9
01198c58d684d79c9026abe355cfb532cb524dc5     1
011ae4de07ffb332b0f51c155a35c23c80294962    29
011fcfb582be9534e9a275336f7e7c3717100381     4
0129dfcdb701b6e1d309934be6393004c6683a2d    12
01327bbc4fd7bfe8ad62e599453d2876b928e725     3
01455f0ab0a5a22a93d94ad35f6e78431aa90625     6
014dedab269f1453c647598c92a3fa37b39eed97     2
014e4fe6e6c5eb3fe5ca0b16c16fb4599df6375c     1
01560f88312a91894d254e6406c25df19f0ad5e8     9
                                            ..
fe5396e3762c36767c9c915f7ed1731691d7e4b4     1
fe5480ff15f0ac51eeb2314a192351f168d7aad7     1
fe56a49b62752708ed2f6e30677c57881f7b78d1    10
fe5885b80e91be887510a0b6dd04e011178d6364     3
fe5f9d7528518e00b0a73c7a3994afc335496961     3
fe66aa534c7824eca663b84b99a437a98a9b026e     2
fe69c72c964a8346dbc7763309c4e07d818d360f     2
fe88d1f683f308b32fb3d7554f007cc55cc48df5     1
fe8c1cb974e39d8ea8c005044e927b3f0de8acd0     2
fe90d98b0287090fe8e653bafba6ed3eff19331e     1
fe9327be39fd457df70e83d3fc8cba9b8b3f95b1     1
feaea388105a4ccc48795b191bbf0c26a23b1356     4
fef0c6be3a2ed226e1fb8a811b0ee68a389f6f3c    11
fef28e45f7217026b2684d1783a2e18b061bdffb     3
fef3bc88def1aa787c99957ded7d5b2c0edc040e     3
ff27ffd93e21154b8a9cf2722f2cc0f75dc39eff     1
```

```
ff288722b76eba5209cdbf9158c6dfbf229b9129     1
ff452614b91f4c9bd965150b1a82e7bf18f59334     2
ff4d3e1c359cfbb73bcae07fa1eb62c45da2b161     3
ff55d0c0b2a4f56aae87c2a21afb7070ab34383d     1
ff6e82c763fe2443643e48a03e239eb635f406dc    13
ff7a0f59ba022102ad22981141a7182c4d8273c3     5
ff833869969184d86f870f98405e7988eccc2309     9
ff979e07f9d906a32ba35a9b75fd9585f6306dbc    15
ffaefa3a1bc2d074d9a14c9924d4e67a46c35410     1
ffc6cfa435937ca0df967b44e9178439d04e3537     1
ffc96f8fbb35aac4cb0029332b0fc78e7766bb5d     2
ffe3d0543c9046d35c2ee3724ea9d774dff98a32    27
fff9fc3ec67bd18ed57a34ed1e67410942c4cd81     7
fffb93a166547448a0ff0232558118d59395fecd     6
Name: article_id, Length: 5148, dtype: int64
```

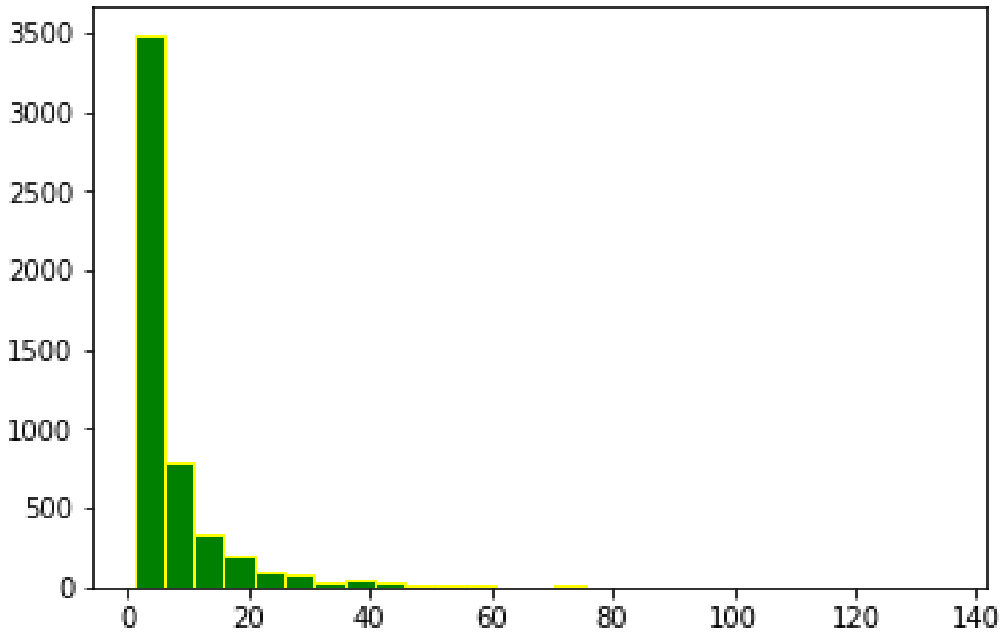
- Visualize the distribution of how many articles a user interacts with

In [12]:
```
user_interact_unique.max()
```

Out[12]:
```
135
```

In [13]:
```
# data visualization
plt.hist(user_interact_unique, bins = 27, ec="yellow", fc="green") #bins =27 for bin_si
```

Out[13]:
```
(array([  3.48900000e+03,   7.84000000e+02,   3.38000000e+02,
          1.92000000e+02,   1.02000000e+02,   8.10000000e+01,
          3.30000000e+01,   4.00000000e+01,   2.30000000e+01,
          1.20000000e+01,   1.30000000e+01,   1.50000000e+01,
          4.00000000e+00,   4.00000000e+00,   1.00000000e+01,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   5.00000000e+00,   1.00000000e+00,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   0.00000000e+00,   2.00000000e+00]),
 array([   1.        ,    5.96296296,   10.92592593,   15.88888889,
          20.85185185,   25.81481481,   30.77777778,   35.74074074,
          40.7037037 ,   45.66666667,   50.62962963,   55.59259259,
          60.55555556,   65.51851852,   70.48148148,   75.44444444,
          80.40740741,   85.37037037,   90.33333333,   95.2962963 ,
         100.25925926,  105.22222222,  110.18518519,  115.14814815,
         120.11111111,  125.07407407,  130.03703704,  135.        ]),
 <a list of 27 Patch objects>)
```



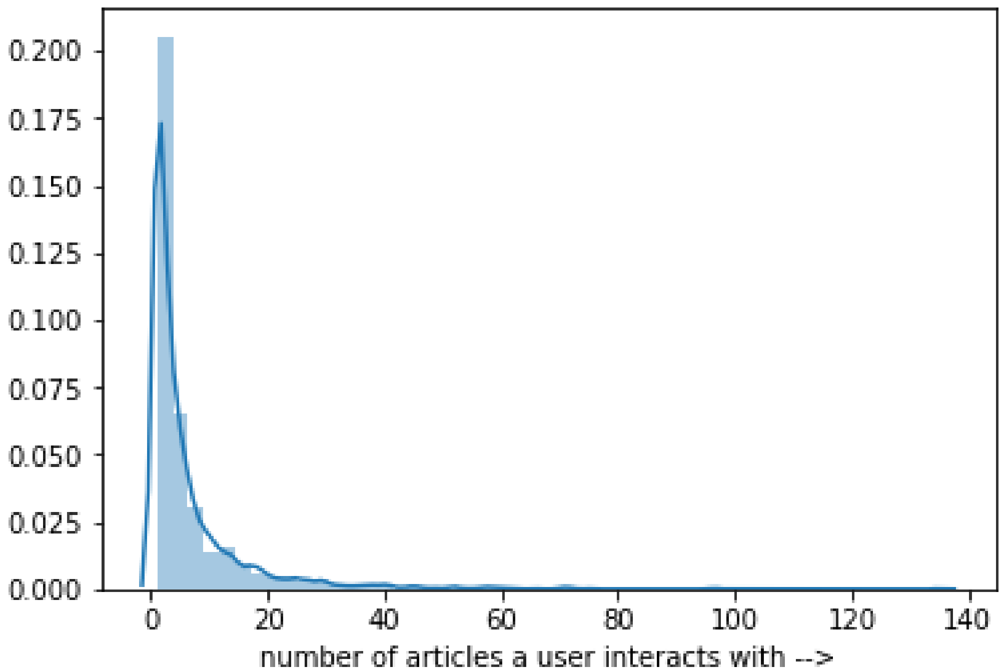
using Seaborn for seeing better distribution

```
In [14]: sns.__version__ # don't support histplot, only can support distplot

Out[14]: '0.8.1'

In [15]: x1 = pd.Series(user_interact_unique, name="number of articles a user interacts with -->
         sns.distplot(x1)

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1f245fdcc0>
```



- Cumulative distribution statistics of "how many articles a user interacts with" (use Mathplotlib because seaborn ver 0.8.1 doesnot support histplot)

```
In [16]: plt.hist(user_interact_unique, bins = 100, ec="yellow", fc="green", cumulative = True)
```

```
Out[16]: (array([ 2309.,  2833.,  3489.,  3735.,  3912.,  4159.,  4273.,  4353.,
                  4495.,  4564.,  4611.,  4693.,  4738.,  4776.,  4822.,  4844.,
                  4861.,  4905.,  4920.,  4940.,  4971.,  4986.,  4993.,  5008.,
                  5013.,  5019.,  5034.,  5043.,  5049.,  5067.,  5069.,  5071.,
                  5082.,  5087.,  5088.,  5093.,  5094.,  5096.,  5104.,  5106.,
                  5107.,  5112.,  5118.,  5120.,  5124.,  5125.,  5126.,  5126.,
                  5128.,  5128.,  5128.,  5130.,  5138.,  5138.,  5138.,  5140.,
                  5140.,  5140.,  5140.,  5140.,  5140.,  5140.,  5140.,  5140.,
                  5140.,  5140.,  5140.,  5140.,  5140.,  5140.,  5142.,  5144.,
                  5144.,  5145.,  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,
                  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,
                  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,  5146.,
                  5146.,  5146.,  5146.,  5148.]),
          array([   1.  ,    2.34,    3.68,    5.02,    6.36,    7.7 ,    9.04,
                   10.38,   11.72,   13.06,   14.4 ,   15.74,   17.08,   18.42,
                   19.76,   21.1 ,   22.44,   23.78,   25.12,   26.46,   27.8 ,
                   29.14,   30.48,   31.82,   33.16,   34.5 ,   35.84,   37.18,
                   38.52,   39.86,   41.2 ,   42.54,   43.88,   45.22,   46.56,
                   47.9 ,   49.24,   50.58,   51.92,   53.26,   54.6 ,   55.94,
                   57.28,   58.62,   59.96,   61.3 ,   62.64,   63.98,   65.32,
```

```
         66.66,   68.  ,   69.34,   70.68,   72.02,   73.36,   74.7 ,
         76.04,   77.38,   78.72,   80.06,   81.4 ,   82.74,   84.08,
         85.42,   86.76,   88.1 ,   89.44,   90.78,   92.12,   93.46,
         94.8 ,   96.14,   97.48,   98.82,  100.16,  101.5 ,  102.84,
        104.18,  105.52,  106.86,  108.2 ,  109.54,  110.88,  112.22,
        113.56,  114.9 ,  116.24,  117.58,  118.92,  120.26,  121.6 ,
        122.94,  124.28,  125.62,  126.96,  128.3 ,  129.64,  130.98,
        132.32,  133.66,  135.  ]),
<a list of 100 Patch objects>)
```

- Calculate the descriptive parameters: the median value of "how many articles a user interacts with" distribution

In [17]:
```
median_val = np.median(user_interact_unique)
median_val
```

Out[17]: 3.0

- Calculate the descriptive parameters: the max value of "how many articles a user interacts with"

In [18]:
```
df.groupby(['email']).size().max()
```

Out[18]: 364

- Fill in the descriptive parameters

In [19]:
```
# Fill in the median and maximum number of user_article interactios below

median_val = 3.0 # 50% of individuals interact with 3.0 number of articles or fewer.
max_views_by_user = 364 # The maximum number of user-article interactions by any 1 user
```

2. Explore and remove duplicate articles from the **df_content** dataframe.

In [20]:
```
# Check the number of items in df_content (articles)
df_content.shape[0]
```

Out[20]:
```
1056
```

In [21]:
```
# Find and explore duplicate articles
df_content[df_content.duplicated(subset=['article_id'], keep='first') == True]
```

Out[21]:
```
                                              doc_body  \
365  Follow Sign in / Sign up Home About Insight Da...
692  Homepage Follow Sign in / Sign up Homepage * H...
761  Homepage Follow Sign in Get started Homepage *...
970  This video shows you how to construct queries ...
971  Homepage Follow Sign in Get started * Home\r\n...

                                       doc_description  \
365  During the seven-week Insight Data Engineering...
692  One of the earliest documented catalogs was co...
761  Todays world of data science leverages data f...
970  This video shows you how to construct queries ...
971  If you are like most data scientists, you are ...

                                         doc_full_name doc_status  article_id
365                       Graph-based machine learning       Live          50
692  How smart catalogs can turn the big data flood...       Live         221
761  Using Apache Spark as a parallel processing fr...       Live         398
970                              Use the Primary Index       Live         577
971  Self-service data preparation with IBM Data Re...       Live         232
```

In [38]:
```
# Remove any rows that have the same article_id - only keep the first
df_content_drop = df_content.drop_duplicates(subset=['article_id'], keep='first') #drop
```

In [23]:
```
df_content_drop.shape[0] #check the number of articles after drop the duplicated rows
```

Out[23]:
```
1051
```

3. Use the cells below to find:
a. The number of unique articles that have an interaction with a user.
b. The number of unique articles in the dataset (whether they have any interactions or not). **c.** The number of unique users in the dataset. (excluding null values) **d.** The number of user-article interactions in the dataset.

```
In [24]: # The number of unique articles that have at least one interaction
         article_interact_unique.shape[0]
```

```
Out[24]: 714
```

```
In [25]: # The number of unique articles on the IBM platform
         df_content_drop.shape[0]
```

```
Out[25]: 1051
```

```
In [26]: # The number of unique users
         user_interact_unique.shape[0]
```

```
Out[26]: 5148
```

```
In [27]: # The number of user-article interactions
         df.shape[0]
```

```
Out[27]: 45993
```

```
In [28]: unique_articles = 714 # The number of unique articles that have at least one interactio
         total_articles = 1051 # The number of unique articles on the IBM platform
         unique_users = 5148 # The number of unique users
         user_article_interactions = 45993 # The number of user-article interactions
```

4. Use the cells below to find the most viewed **article_id**, as well as how often it was viewed. After talking to the company leaders, the `email_mapper` function was deemed a reasonable way to map users to ids. There were a small number of null values, and it was found that all of these null values likely belonged to a single user (which is how they are stored using the function below).

```
In [29]: # The most viewed article in the dataset was viewed how many times?
         df.groupby(['article_id']).size().max()
```

```
Out[29]: 937
```

```
In [30]: article_interactions = df.groupby(['article_id']).size()
         article_interactions
```

```
Out[30]: article_id
         0.0        14
         2.0        58
         4.0        13
         8.0        85
         9.0        10
         12.0      157
         14.0       89
         15.0       26
         16.0       61
         18.0       78
         20.0      249
```

25.0	15
26.0	89
28.0	42
29.0	75
30.0	17
32.0	64
33.0	141
34.0	93
36.0	18
39.0	68
40.0	70
43.0	460
48.0	11
50.0	89
51.0	124
53.0	115
54.0	20
57.0	140
58.0	11
	...
1412.0	25
1414.0	4
1415.0	11
1416.0	102
1418.0	43
1419.0	6
1420.0	113
1421.0	3
1422.0	163
1423.0	155
1424.0	131
1425.0	71
1426.0	138
1427.0	643
1428.0	120
1429.0	937
1430.0	336
1431.0	671
1432.0	340
1433.0	108
1434.0	42
1435.0	120
1436.0	481
1437.0	218
1439.0	59
1440.0	10
1441.0	8
1442.0	4

```
1443.0     22
1444.0      5
Length: 714, dtype: int64
```

In [31]:
```
article_interactions[article_interactions == article_interactions.max()]
```

Out[31]:
```
article_id
1429.0    937
dtype: int64
```

In [32]:
```
most_viewed_article_id = '1429.0' # The most viewed article in the dataset as a string
max_views = 937 # The most viewed article in the dataset was viewed how many times?
```

In [2]:
```
## No need to change the code here - this will be helpful for later parts of the noteboo
# Run this cell to map the user email to a user_id column and remove the email column

def email_mapper():
    coded_dict = dict()
    cter = 1
    email_encoded = []

    for val in df['email']:
        if val not in coded_dict:
            coded_dict[val] = cter
            cter+=1

        email_encoded.append(coded_dict[val])
    return email_encoded

email_encoded = email_mapper()
del df['email']
df['user_id'] = email_encoded

# show header
df.head()
```

Out[2]:

	article_id	title	user_id
0	1430.0	using pixiedust for fast, flexible, and easier...	1
1	1314.0	healthcare python streaming application demo	2
2	1429.0	use deep learning for image classification	3
3	1338.0	ml optimization using cognitive assistant	4
4	1276.0	deploy your python model as a restful api	5

In [34]:
```
## If you stored all your results in the variable names above,
## you shouldn't need to change anything in this cell

sol_1_dict = {
    '`50% of individuals have _____ or fewer interactions.`': median_val,
    '`The total number of user-article interactions in the dataset is ______.`': user_a
```

```
    '`The maximum number of user-article interactions by any 1 user is ______.`': max_v
    '`The most viewed article in the dataset was viewed _____ times.`': max_views,
    '`The article_id of the most viewed article is ______.`': most_viewed_article_id,
    '`The number of unique articles that have at least 1 rating ______.`': unique_artic
    '`The number of unique users in the dataset is ______`': unique_users,
    '`The number of unique articles on the IBM platform`': total_articles
}

# Test your dictionary against the solution
t.sol_1_test(sol_1_dict)
```

```
It looks like you have everything right here! Nice job!
```

1.1.2 Part II: Rank-Based Recommendations

Unlike in the earlier lessons, we don't actually have ratings for whether a user liked an article or not. We only know that a user has interacted with an article. In these cases, the popularity of an article can really only be based on how often an article was interacted with.

1. Fill in the function below to return the **n** top articles ordered with most interactions as the top. Test your function using the tests below.

- Draft the functions with n = 10

In [2]:
```
df.groupby(['article_id']).size()
```

Out[2]:
```
article_id
0.0       14
2.0       58
4.0       13
8.0       85
9.0       10
12.0     157
14.0      89
15.0      26
16.0      61
18.0      78
20.0     249
25.0      15
26.0      89
28.0      42
29.0      75
30.0      17
32.0      64
33.0     141
34.0      93
36.0      18
39.0      68
40.0      70
```

```
43.0      460
48.0       11
50.0       89
51.0      124
53.0      115
54.0       20
57.0      140
58.0       11
          ...
1412.0     25
1414.0      4
1415.0     11
1416.0    102
1418.0     43
1419.0      6
1420.0    113
1421.0      3
1422.0    163
1423.0    155
1424.0    131
1425.0     71
1426.0    138
1427.0    643
1428.0    120
1429.0    937
1430.0    336
1431.0    671
1432.0    340
1433.0    108
1434.0     42
1435.0    120
1436.0    481
1437.0    218
1439.0     59
1440.0     10
1441.0      8
1442.0      4
1443.0     22
1444.0      5
Length: 714, dtype: int64
```

In [3]:
```
article_interactions_nlargest = df.groupby(['article_id']).size().nlargest(10)
article_interactions_nlargest
```

Out[3]:
```
article_id
1429.0    937
1330.0    927
1431.0    671
```

```
1427.0    643
1364.0    627
1314.0    614
1293.0    572
1170.0    565
1162.0    512
1304.0    483
dtype: int64
```

```
In [11]: article_interactions_nlargest.index
```

```
Out[11]: Float64Index([1429.0, 1330.0, 1431.0, 1427.0, 1364.0, 1314.0, 1293.0, 1170.0,
                       1162.0, 1304.0],
                      dtype='float64', name='article_id')
```

```
In [19]: article_interactions_nlargest.index.map(str)
```

```
Out[19]: Index(['1429.0', '1330.0', '1431.0', '1427.0', '1364.0', '1314.0', '1293.0',
                '1170.0', '1162.0', '1304.0'],
               dtype='object', name='article_id')
```

```
In [9]: article_interactions_nlargest.index.map(str).tolist()
```

```
Out[9]: ['1429.0',
         '1330.0',
         '1431.0',
         '1427.0',
         '1364.0',
         '1314.0',
         '1293.0',
         '1170.0',
         '1162.0',
         '1304.0']
```

```
In [96]: top_articles_index = article_interactions_nlargest.index.tolist()
         top_articles_index
```

```
Out[96]: [1429.0,
          1330.0,
          1431.0,
          1427.0,
          1364.0,
          1314.0,
          1293.0,
          1170.0,
          1162.0,
          1304.0]
```

```
In [79]: df.head()
```

```
Out[79]:    article_id                                              title  \
         0      1430.0  using pixiedust for fast, flexible, and easier...
         1      1314.0       healthcare python streaming application demo
         2      1429.0         use deep learning for image classification
         3      1338.0          ml optimization using cognitive assistant
         4      1276.0          deploy your python model as a restful api

                                               email
         0  ef5f11f77ba020cd36e1105a00ab868bbdbf7fe7
         1  083cbdfa93c8444beaa4c5f5e0f5f9198e4f9e0b
         2  b96a4f2e92d8572034b1e9b28f9ac673765cd074
         3  06485706b34a5c9bf2a0ecdac41daf7e7654ceb7
         4  f01220c46fc92c6e6b161b1849de11faacd7ccb2
```

```
In [103]: df_drop = df[['article_id','title']]
          df_drop.head()
```

```
Out[103]:    article_id                                              title
          0      1430.0  using pixiedust for fast, flexible, and easier...
          1      1314.0       healthcare python streaming application demo
          2      1429.0         use deep learning for image classification
          3      1338.0          ml optimization using cognitive assistant
          4      1276.0          deploy your python model as a restful api
```

```
In [104]: df_drop = df_drop.drop_duplicates(subset=['article_id'], keep='first')
          df_drop.head()
```

```
Out[104]:    article_id                                              title
          0      1430.0  using pixiedust for fast, flexible, and easier...
          1      1314.0       healthcare python streaming application demo
          2      1429.0         use deep learning for image classification
          3      1338.0          ml optimization using cognitive assistant
          4      1276.0          deploy your python model as a restful api
```

```
In [89]: df_drop.shape[0]
```

```
Out[89]: 696
```

```
In [97]: df_drop[df_drop['article_id'].isin(top_articles_index)]
```

```
Out[97]:      article_id                                              title
         1        1314.0       healthcare python streaming application demo
         2        1429.0         use deep learning for image classification
         14       1170.0           apache spark lab, part 1: basic concepts
         29       1364.0  predicting churn with the spss random tree alg...
         31       1162.0              analyze energy consumption in buildings
         37       1431.0                         visualize car data with brunel
         42       1427.0  use xgboost, scikit-learn & ibm watson machine...
         56       1304.0  gosales transactions for logistic regression m...
         66       1330.0        insights from new york car accident reports
         154      1293.0  finding optimal locations of new store using d...
```

```
In [25]: top_articles_id = df.groupby(['article_id']).size().nlargest(10).index.map(str).tolist(
         top_articles_id
```

```
Out[25]: ['1429.0',
          '1330.0',
          '1431.0',
          '1427.0',
          '1364.0',
          '1314.0',
          '1293.0',
          '1170.0',
          '1162.0',
          '1304.0']
```

- Fill in the functions

```
In [3]: def get_top_articles(n, df=df):
            '''
            INPUT:
            n - (int) the number of top articles to return
            df - (pandas dataframe) df as defined at the top of the notebook

            OUTPUT:
            top_articles - (list) A list of the top 'n' article titles

            '''
            # Your code here
            article_interactions_nlargest = df.groupby(['article_id']).size().nlargest(n)
            top_articles_index = article_interactions_nlargest.index.tolist()
            df_drop = df[['article_id','title']]
            df_drop = df_drop.drop_duplicates(subset=['article_id'], keep='first')
            top_articles = df_drop[df_drop['article_id'].isin(top_articles_index)]['title']

            return top_articles # Return the top article titles from df (not df_content)

        def get_top_article_ids(n, df=df):
            '''
            INPUT:
            n - (int) the number of top articles to return
            df - (pandas dataframe) df as defined at the top of the notebook

            OUTPUT:
            top_articles - (list) A list of the top 'n' article titles

            '''
            # Your code here
            top_articles = df.groupby(['article_id']).size().nlargest(n).index.map(str).tolist()

            return top_articles # Return the top article ids
```

```
In [4]: print(get_top_articles(10))
        print(get_top_article_ids(10))
```

```
1             healthcare python streaming application demo
2               use deep learning for image classification
14                 apache spark lab, part 1: basic concepts
29     predicting churn with the spss random tree alg...
31                  analyze energy consumption in buildings
37                           visualize car data with brunel
42     use xgboost, scikit-learn & ibm watson machine...
56     gosales transactions for logistic regression m...
66              insights from new york car accident reports
154    finding optimal locations of new store using d...
Name: title, dtype: object
['1429.0', '1330.0', '1431.0', '1427.0', '1364.0', '1314.0', '1293.0', '1170.0', '1162.0', '1304
```

```
In [5]: # Test your function by returning the top 5, 10, and 20 articles
        top_5 = get_top_articles(5)
        top_10 = get_top_articles(10)
        top_20 = get_top_articles(20)

        # Test each of your three lists from above
        t.sol_2_test(get_top_articles)
```

```
Your top_5 looks like the solution list! Nice job.
Your top_10 looks like the solution list! Nice job.
Your top_20 looks like the solution list! Nice job.
```

1.1.3 Part III: User-User Based Collaborative Filtering

1. Use the function below to reformat the **df** dataframe to be shaped with users as the rows and articles as the columns.

- Each **user** should only appear in each **row** once.
- Each **article** should only show up in one **column**.
- **If a user has interacted with an article, then place a 1 where the user-row meets for that article-column**. It does not matter how many times a user has interacted with the article, all entries where a user has interacted with an article should be a 1.
- **If a user has not interacted with an item, then place a zero where the user-row meets for that article-column**.

Use the tests to make sure the basic structure of your matrix matches what is expected by the solution.

a. Draft the function

In [8]: df.head()

```
Out[8]:    article_id                                              title  user_id
        0      1430.0  using pixiedust for fast, flexible, and easier...        1
        1      1314.0       healthcare python streaming application demo        2
        2      1429.0         use deep learning for image classification        3
        3      1338.0          ml optimization using cognitive assistant        4
        4      1276.0          deploy your python model as a restful api        5
```

In [10]: df['article_id'].shape[0]

Out[10]: 45993

```
In [17]: df_article_col = df.drop_duplicates(subset=['article_id'], keep='first')['article_id']
         df_article_col.head()
```

```
Out[17]: 0    1430.0
         1    1314.0
         2    1429.0
         3    1338.0
         4    1276.0
         Name: article_id, dtype: float64
```

In [16]: df_article_col.shape[0]

Out[16]: 714

```
In [19]: df_user_row = df.drop_duplicates(subset=['user_id'], keep='first')['user_id']
         df_user_row.head()
```

```
Out[19]: 0    1
         1    2
         2    3
         3    4
         4    5
         Name: user_id, dtype: int64
```

In [20]: df_user_row.shape[0]

Out[20]: 5149

```
In [34]: interactions = df.groupby(['user_id','article_id']).size()
         interactions.head()
```

```
Out[34]: user_id  article_id
         1        43.0          1
                  109.0         1
                  151.0         1
                  268.0         1
                  310.0         2
         dtype: int64
```

```
In [37]: df_interactions = df.groupby(['user_id','article_id']).size().index.to_frame()
         df_interactions.head()
```

Out[37]:

user_id	article_id	user_id	article_id
1	43.0	1	43.0
	109.0	1	109.0
	151.0	1	151.0
	268.0	1	268.0
	310.0	1	310.0

```
In [38]: df_interactions['interaction'] = interactions
         df_interactions
```

Out[38]:

user_id	article_id	user_id	article_id	interaction
1	43.0	1	43.0	1
	109.0	1	109.0	1
	151.0	1	151.0	1
	268.0	1	268.0	1
	310.0	1	310.0	2
	329.0	1	329.0	1
	346.0	1	346.0	1
	390.0	1	390.0	1
	494.0	1	494.0	1
	525.0	1	525.0	1
	585.0	1	585.0	2
	626.0	1	626.0	1
	668.0	1	668.0	2
	732.0	1	732.0	1
	768.0	1	768.0	1
	910.0	1	910.0	1
	968.0	1	968.0	1
	981.0	1	981.0	1
	1052.0	1	1052.0	2
	1170.0	1	1170.0	2
	1183.0	1	1183.0	2
	1185.0	1	1185.0	2
	1232.0	1	1232.0	1
	1293.0	1	1293.0	1
	1305.0	1	1305.0	1
	1363.0	1	1363.0	2
	1368.0	1	1368.0	1
	1391.0	1	1391.0	1
	1400.0	1	1400.0	1
	1406.0	1	1406.0	2
...		...	...	...
5143	485.0	5143	485.0	1

```
            495.0         5143      495.0           1
            588.0         5143      588.0           2
            1324.0        5143     1324.0           1
            1330.0        5143     1330.0           2
            1343.0        5143     1343.0           1
            1354.0        5143     1354.0           1
            1360.0        5143     1360.0           1
            1398.0        5143     1398.0           3
            1400.0        5143     1400.0           3
            1409.0        5143     1409.0           1
            1430.0        5143     1430.0           2
            1431.0        5143     1431.0           1
            1436.0        5143     1436.0           1
    5144    270.0         5144      270.0           1
    5145    20.0          5145       20.0           1
            138.0         5145      138.0           1
            962.0         5145      962.0           2
            1165.0        5145     1165.0           1
            1305.0        5145     1305.0           1
    5146    142.0         5146      142.0           1
            1125.0        5146     1125.0           1
            1157.0        5146     1157.0           1
            1282.0        5146     1282.0           1
            1324.0        5146     1324.0           3
            1394.0        5146     1394.0           1
            1416.0        5146     1416.0           1
    5147    233.0         5147      233.0           1
    5148    1160.0        5148     1160.0           1
    5149    16.0          5149       16.0           1

[33682 rows x 3 columns]
```

In [43]:
```
user_article_matrix = df_interactions.pivot(*df_interactions.columns).fillna(0)
user_article_matrix
```

Out[43]:
```
article_id  0.0     2.0     4.0     8.0     9.0     12.0    14.0    15.0    \
user_id
1              0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
2              0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
3              0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0
4              0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
5              0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
6              0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
7              0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
8              0.0     0.0     0.0     0.0     0.0     0.0     1.0     0.0
9              0.0     0.0     0.0     0.0     0.0     0.0     1.0     0.0
10             0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
11             0.0     0.0     0.0     0.0     0.0     2.0     0.0     0.0
```

12	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
22	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
23	0.0	3.0	0.0	0.0	0.0	7.0	1.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
5120	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5121	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5122	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5123	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5124	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5125	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5126	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5127	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5128	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5129	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5130	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5131	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5132	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5133	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5134	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5135	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5136	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5137	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5138	0.0	0.0	0.0	0.0	0.0	2.0	1.0	0.0
5139	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
5140	0.0	3.0	0.0	0.0	0.0	1.0	0.0	0.0
5141	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5142	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
5143	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5144	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5145	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5146	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5147	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
5148          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5149          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0

article_id  16.0   18.0    ...   1434.0 1435.0 1436.0 1437.0 1439.0  \
user_id                    ...
1             0.0    0.0   ...      0.0    0.0    1.0    0.0    1.0
2             0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
3             0.0    0.0   ...      0.0    0.0    1.0    0.0    0.0
4             0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
5             0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
6             0.0    0.0   ...      0.0    0.0    1.0    0.0    0.0
7             0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
8             0.0    0.0   ...      0.0    0.0    4.0    0.0    0.0
9             1.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
10            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
11            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
12            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
13            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
14            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
15            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
16            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
17            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
18            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
19            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
20            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
21            0.0    0.0   ...      0.0    0.0    2.0    3.0    0.0
22            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
23            1.0    0.0   ...      0.0    0.0    6.0    0.0    1.0
24            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
25            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
26            0.0    0.0   ...      0.0    0.0    4.0    0.0    0.0
27            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
28            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
29            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
30            0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
...           ...    ...   ...      ...    ...    ...    ...    ...
5120          0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
5121          0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
5122          0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
5123          0.0    0.0   ...      0.0    0.0    1.0    1.0    0.0
5124          0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
5125          0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
5126          0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
5127          0.0    0.0   ...      1.0    0.0    0.0    0.0    0.0
5128          0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
5129          0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
5130          0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
5131          0.0    0.0   ...      0.0    0.0    0.0    0.0    0.0
```

```
5132         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5133         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5134         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5135         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5136         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5137         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5138         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5139         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5140         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5141         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5142         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5143         0.0     0.0   ...     0.0     0.0     1.0     0.0     0.0
5144         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5145         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5146         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5147         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5148         0.0     0.0   ...     0.0     0.0     0.0     0.0     0.0
5149         1.0     0.0   ...     0.0     0.0     0.0     0.0     0.0

article_id  1440.0  1441.0  1442.0  1443.0  1444.0
user_id
1              0.0     0.0     0.0     0.0     0.0
2              0.0     0.0     0.0     0.0     0.0
3              0.0     0.0     0.0     0.0     0.0
4              0.0     0.0     0.0     0.0     0.0
5              0.0     0.0     0.0     0.0     0.0
6              0.0     0.0     0.0     0.0     0.0
7              0.0     0.0     0.0     0.0     0.0
8              0.0     0.0     0.0     0.0     0.0
9              0.0     0.0     0.0     0.0     0.0
10             0.0     0.0     0.0     0.0     0.0
11             0.0     0.0     0.0     0.0     0.0
12             0.0     0.0     0.0     0.0     0.0
13             0.0     0.0     0.0     0.0     0.0
14             0.0     0.0     0.0     0.0     0.0
15             0.0     0.0     0.0     0.0     0.0
16             0.0     0.0     0.0     0.0     0.0
17             0.0     0.0     0.0     0.0     0.0
18             0.0     0.0     0.0     0.0     0.0
19             0.0     0.0     0.0     0.0     0.0
20             0.0     0.0     0.0     0.0     0.0
21             0.0     0.0     0.0     0.0     0.0
22             0.0     0.0     0.0     0.0     0.0
23             0.0     0.0     0.0     0.0     0.0
24             1.0     0.0     0.0     0.0     0.0
25             0.0     0.0     0.0     0.0     0.0
26             0.0     0.0     0.0     0.0     0.0
27             0.0     0.0     0.0     0.0     0.0
```

```
28            0.0    0.0    0.0    0.0    0.0
29            0.0    0.0    0.0    0.0    0.0
30            0.0    0.0    0.0    0.0    0.0
...           ...    ...    ...    ...    ...
5120          0.0    0.0    0.0    0.0    0.0
5121          0.0    0.0    0.0    0.0    0.0
5122          0.0    0.0    0.0    0.0    0.0
5123          0.0    0.0    0.0    0.0    0.0
5124          0.0    0.0    0.0    0.0    0.0
5125          0.0    0.0    0.0    0.0    0.0
5126          0.0    0.0    0.0    0.0    0.0
5127          0.0    0.0    0.0    0.0    0.0
5128          0.0    0.0    0.0    0.0    0.0
5129          0.0    0.0    0.0    0.0    0.0
5130          0.0    0.0    0.0    0.0    0.0
5131          0.0    0.0    0.0    0.0    0.0
5132          0.0    0.0    0.0    0.0    0.0
5133          0.0    0.0    0.0    0.0    0.0
5134          0.0    0.0    0.0    0.0    0.0
5135          0.0    0.0    0.0    0.0    0.0
5136          0.0    0.0    0.0    0.0    0.0
5137          0.0    0.0    0.0    0.0    0.0
5138          0.0    0.0    0.0    0.0    0.0
5139          0.0    0.0    0.0    0.0    0.0
5140          0.0    0.0    0.0    0.0    0.0
5141          0.0    0.0    0.0    0.0    0.0
5142          0.0    0.0    0.0    0.0    0.0
5143          0.0    0.0    0.0    0.0    0.0
5144          0.0    0.0    0.0    0.0    0.0
5145          0.0    0.0    0.0    0.0    0.0
5146          0.0    0.0    0.0    0.0    0.0
5147          0.0    0.0    0.0    0.0    0.0
5148          0.0    0.0    0.0    0.0    0.0
5149          0.0    0.0    0.0    0.0    0.0

[5149 rows x 714 columns]
```

In [45]:
```
user_article_matrix [user_article_matrix > 1] = 1
user_article_matrix
```

Out[45]:
```
article_id  0.0    2.0    4.0    8.0    9.0    12.0   14.0   15.0   \
user_id
1             0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
2             0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
3             0.0    0.0    0.0    0.0    0.0    1.0    0.0    0.0
4             0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5             0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
6             0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
```

7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
22	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
23	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
5120	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5121	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5122	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5123	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5124	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5125	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5126	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5127	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5128	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5129	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5130	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5131	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5132	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5133	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5134	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5135	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5136	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5137	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5138	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0
5139	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
5140	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
5141	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5142	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

```
5143          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5144          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5145          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5146          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5147          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5148          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5149          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0

article_id  16.0    18.0    ...    1434.0  1435.0  1436.0  1437.0  1439.0  \
user_id                     ...
1             0.0    0.0    ...       0.0     0.0     1.0     0.0     1.0
2             0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
3             0.0    0.0    ...       0.0     0.0     1.0     0.0     0.0
4             0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
5             0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
6             0.0    0.0    ...       0.0     0.0     1.0     0.0     0.0
7             0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
8             0.0    0.0    ...       0.0     0.0     1.0     0.0     0.0
9             1.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
10            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
11            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
12            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
13            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
14            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
15            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
16            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
17            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
18            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
19            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
20            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
21            0.0    0.0    ...       0.0     0.0     1.0     1.0     0.0
22            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
23            1.0    0.0    ...       0.0     0.0     1.0     0.0     1.0
24            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
25            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
26            0.0    0.0    ...       0.0     0.0     1.0     0.0     0.0
27            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
28            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
29            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
30            0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
...           ...    ...    ...       ...     ...     ...     ...     ...
5120          0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
5121          0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
5122          0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
5123          0.0    0.0    ...       0.0     0.0     1.0     1.0     0.0
5124          0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
5125          0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
5126          0.0    0.0    ...       0.0     0.0     0.0     0.0     0.0
```

```
5127           0.0     0.0    ...      1.0     0.0     0.0     0.0     0.0
5128           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5129           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5130           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5131           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5132           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5133           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5134           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5135           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5136           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5137           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5138           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5139           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5140           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5141           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5142           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5143           0.0     0.0    ...      0.0     0.0     1.0     0.0     0.0
5144           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5145           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5146           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5147           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5148           0.0     0.0    ...      0.0     0.0     0.0     0.0     0.0
5149           1.0     0.0    ...      0.0     0.0     0.0     0.0     0.0

article_id  1440.0  1441.0  1442.0  1443.0  1444.0
user_id
1              0.0     0.0     0.0     0.0     0.0
2              0.0     0.0     0.0     0.0     0.0
3              0.0     0.0     0.0     0.0     0.0
4              0.0     0.0     0.0     0.0     0.0
5              0.0     0.0     0.0     0.0     0.0
6              0.0     0.0     0.0     0.0     0.0
7              0.0     0.0     0.0     0.0     0.0
8              0.0     0.0     0.0     0.0     0.0
9              0.0     0.0     0.0     0.0     0.0
10             0.0     0.0     0.0     0.0     0.0
11             0.0     0.0     0.0     0.0     0.0
12             0.0     0.0     0.0     0.0     0.0
13             0.0     0.0     0.0     0.0     0.0
14             0.0     0.0     0.0     0.0     0.0
15             0.0     0.0     0.0     0.0     0.0
16             0.0     0.0     0.0     0.0     0.0
17             0.0     0.0     0.0     0.0     0.0
18             0.0     0.0     0.0     0.0     0.0
19             0.0     0.0     0.0     0.0     0.0
20             0.0     0.0     0.0     0.0     0.0
21             0.0     0.0     0.0     0.0     0.0
22             0.0     0.0     0.0     0.0     0.0
```

```
23           0.0   0.0   0.0   0.0   0.0
24           1.0   0.0   0.0   0.0   0.0
25           0.0   0.0   0.0   0.0   0.0
26           0.0   0.0   0.0   0.0   0.0
27           0.0   0.0   0.0   0.0   0.0
28           0.0   0.0   0.0   0.0   0.0
29           0.0   0.0   0.0   0.0   0.0
30           0.0   0.0   0.0   0.0   0.0
...          ...   ...   ...   ...   ...
5120         0.0   0.0   0.0   0.0   0.0
5121         0.0   0.0   0.0   0.0   0.0
5122         0.0   0.0   0.0   0.0   0.0
5123         0.0   0.0   0.0   0.0   0.0
5124         0.0   0.0   0.0   0.0   0.0
5125         0.0   0.0   0.0   0.0   0.0
5126         0.0   0.0   0.0   0.0   0.0
5127         0.0   0.0   0.0   0.0   0.0
5128         0.0   0.0   0.0   0.0   0.0
5129         0.0   0.0   0.0   0.0   0.0
5130         0.0   0.0   0.0   0.0   0.0
5131         0.0   0.0   0.0   0.0   0.0
5132         0.0   0.0   0.0   0.0   0.0
5133         0.0   0.0   0.0   0.0   0.0
5134         0.0   0.0   0.0   0.0   0.0
5135         0.0   0.0   0.0   0.0   0.0
5136         0.0   0.0   0.0   0.0   0.0
5137         0.0   0.0   0.0   0.0   0.0
5138         0.0   0.0   0.0   0.0   0.0
5139         0.0   0.0   0.0   0.0   0.0
5140         0.0   0.0   0.0   0.0   0.0
5141         0.0   0.0   0.0   0.0   0.0
5142         0.0   0.0   0.0   0.0   0.0
5143         0.0   0.0   0.0   0.0   0.0
5144         0.0   0.0   0.0   0.0   0.0
5145         0.0   0.0   0.0   0.0   0.0
5146         0.0   0.0   0.0   0.0   0.0
5147         0.0   0.0   0.0   0.0   0.0
5148         0.0   0.0   0.0   0.0   0.0
5149         0.0   0.0   0.0   0.0   0.0

[5149 rows x 714 columns]
```

b. Fill in the function

```
In [4]: # create the user-article matrix with 1's and 0's

        def create_user_item_matrix(df):
            '''
```

```
    INPUT:
    df - pandas dataframe with article_id, title, user_id columns

    OUTPUT:
    user_item - user item matrix

    Description:
    Return a matrix with user ids as rows and article ids on the columns with 1 values u
    an article and a 0 otherwise
    '''
    # Fill in the function here
    interactions = df.groupby(['user_id','article_id']).size()
    df_interactions = df.groupby(['user_id','article_id']).size().index.to_frame()
    df_interactions['interaction'] = interactions
    user_item = df_interactions.pivot(*df_interactions.columns).fillna(0)
    user_item [user_item > 1] = 1

    return user_item # return the user_item matrix

user_item = create_user_item_matrix(df)
```

c. Test the function

```
In [7]: ## Tests: You should just need to run this cell.  Don't change the code.
        assert user_item.shape[0] == 5149, "Oops!  The number of users in the user-article matri
        assert user_item.shape[1] == 714, "Oops!  The number of articles in the user-article mat
        assert user_item.sum(axis=1)[1] == 36, "Oops!  The number of articles seen by user 1 doe
        print("You have passed our quick tests!  Please proceed!")

You have passed our quick tests!  Please proceed!
```

2. Complete the function below which should take a user_id and provide an ordered list of the most similar users to that user (from most similar to least similar). The returned result should not contain the provided user_id, as we know that each user is similar to him/herself. Because the results for each user here are binary, it (perhaps) makes sense to compute similarity as the dot product of two users.

Use the tests to test your function.

a. Draft the function

Computes the similarity of every pair of users based on the dot product

```
In [93]: # check the user_item matrix and its dimensions
         user_item

Out[93]: article_id  0.0     2.0     4.0     8.0     9.0     12.0    14.0    15.0    \
         user_id
         1              0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         2              0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
         3              0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0
```

4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
22	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
23	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
5120	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5121	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5122	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5123	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5124	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5125	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5126	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5127	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5128	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5129	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5130	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5131	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5132	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5133	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5134	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5135	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5136	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5137	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5138	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0
5139	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

```
5140            0.0    1.0    0.0    0.0    0.0    1.0    0.0    0.0
5141            0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5142            0.0    0.0    0.0    0.0    0.0    0.0    1.0    0.0
5143            0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5144            0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5145            0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5146            0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5147            0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5148            0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
5149            0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0

article_id  16.0    18.0    ...    1434.0  1435.0  1436.0  1437.0  1439.0  \
user_id                     ...
1               0.0    0.0  ...       0.0     0.0     1.0     0.0     1.0
2               0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
3               0.0    0.0  ...       0.0     0.0     1.0     0.0     0.0
4               0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
5               0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
6               0.0    0.0  ...       0.0     0.0     1.0     0.0     0.0
7               0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
8               0.0    0.0  ...       0.0     0.0     1.0     0.0     0.0
9               1.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
10              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
11              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
12              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
13              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
14              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
15              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
16              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
17              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
18              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
19              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
20              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
21              0.0    0.0  ...       0.0     0.0     1.0     1.0     0.0
22              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
23              1.0    0.0  ...       0.0     0.0     1.0     0.0     1.0
24              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
25              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
26              0.0    0.0  ...       0.0     0.0     1.0     0.0     0.0
27              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
28              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
29              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
30              0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
...             ...    ...  ...       ...     ...     ...     ...     ...
5120            0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
5121            0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
5122            0.0    0.0  ...       0.0     0.0     0.0     0.0     0.0
5123            0.0    0.0  ...       0.0     0.0     1.0     1.0     0.0
```

```
5124         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5125         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5126         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5127         0.0    0.0   ...    1.0    0.0    0.0    0.0    0.0
5128         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5129         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5130         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5131         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5132         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5133         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5134         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5135         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5136         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5137         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5138         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5139         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5140         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5141         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5142         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5143         0.0    0.0   ...    0.0    0.0    1.0    0.0    0.0
5144         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5145         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5146         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5147         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5148         0.0    0.0   ...    0.0    0.0    0.0    0.0    0.0
5149         1.0    0.0   ...    0.0    0.0    0.0    0.0    0.0

article_id  1440.0  1441.0  1442.0  1443.0  1444.0
user_id
1              0.0     0.0     0.0     0.0     0.0
2              0.0     0.0     0.0     0.0     0.0
3              0.0     0.0     0.0     0.0     0.0
4              0.0     0.0     0.0     0.0     0.0
5              0.0     0.0     0.0     0.0     0.0
6              0.0     0.0     0.0     0.0     0.0
7              0.0     0.0     0.0     0.0     0.0
8              0.0     0.0     0.0     0.0     0.0
9              0.0     0.0     0.0     0.0     0.0
10             0.0     0.0     0.0     0.0     0.0
11             0.0     0.0     0.0     0.0     0.0
12             0.0     0.0     0.0     0.0     0.0
13             0.0     0.0     0.0     0.0     0.0
14             0.0     0.0     0.0     0.0     0.0
15             0.0     0.0     0.0     0.0     0.0
16             0.0     0.0     0.0     0.0     0.0
17             0.0     0.0     0.0     0.0     0.0
18             0.0     0.0     0.0     0.0     0.0
19             0.0     0.0     0.0     0.0     0.0
```

```
20          0.0   0.0   0.0   0.0   0.0
21          0.0   0.0   0.0   0.0   0.0
22          0.0   0.0   0.0   0.0   0.0
23          0.0   0.0   0.0   0.0   0.0
24          1.0   0.0   0.0   0.0   0.0
25          0.0   0.0   0.0   0.0   0.0
26          0.0   0.0   0.0   0.0   0.0
27          0.0   0.0   0.0   0.0   0.0
28          0.0   0.0   0.0   0.0   0.0
29          0.0   0.0   0.0   0.0   0.0
30          0.0   0.0   0.0   0.0   0.0
...         ...   ...   ...   ...   ...
5120        0.0   0.0   0.0   0.0   0.0
5121        0.0   0.0   0.0   0.0   0.0
5122        0.0   0.0   0.0   0.0   0.0
5123        0.0   0.0   0.0   0.0   0.0
5124        0.0   0.0   0.0   0.0   0.0
5125        0.0   0.0   0.0   0.0   0.0
5126        0.0   0.0   0.0   0.0   0.0
5127        0.0   0.0   0.0   0.0   0.0
5128        0.0   0.0   0.0   0.0   0.0
5129        0.0   0.0   0.0   0.0   0.0
5130        0.0   0.0   0.0   0.0   0.0
5131        0.0   0.0   0.0   0.0   0.0
5132        0.0   0.0   0.0   0.0   0.0
5133        0.0   0.0   0.0   0.0   0.0
5134        0.0   0.0   0.0   0.0   0.0
5135        0.0   0.0   0.0   0.0   0.0
5136        0.0   0.0   0.0   0.0   0.0
5137        0.0   0.0   0.0   0.0   0.0
5138        0.0   0.0   0.0   0.0   0.0
5139        0.0   0.0   0.0   0.0   0.0
5140        0.0   0.0   0.0   0.0   0.0
5141        0.0   0.0   0.0   0.0   0.0
5142        0.0   0.0   0.0   0.0   0.0
5143        0.0   0.0   0.0   0.0   0.0
5144        0.0   0.0   0.0   0.0   0.0
5145        0.0   0.0   0.0   0.0   0.0
5146        0.0   0.0   0.0   0.0   0.0
5147        0.0   0.0   0.0   0.0   0.0
5148        0.0   0.0   0.0   0.0   0.0
5149        0.0   0.0   0.0   0.0   0.0

[5149 rows x 714 columns]
```

In [94]:
```python
# need to transpose to match the matrix dimension when calculate the dot product
user_item.transpose()
```

Out[94]:
```
user_id    1    2    3    4    5    6    7    8    9    10   ...  \
```

article_id											...
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
12.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
14.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	...
15.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
16.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...
18.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
20.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...
25.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
26.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
28.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...
29.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
30.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
32.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...
33.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
34.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...
36.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
39.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
40.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...
43.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
48.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
50.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
51.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
53.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
54.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
57.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...
58.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
...	...	...	...	...	...	...	...	...	...	...	...
1412.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1414.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1415.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1416.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1418.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1419.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1420.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1421.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1422.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...
1423.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1424.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1425.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1426.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1427.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	...
1428.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...
1429.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	...

1430.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...
1431.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...
1432.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	...
1433.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1434.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1435.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1436.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	...
1437.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1439.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1440.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1441.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1442.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1443.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1444.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

user_id	5140	5141	5142	5143	5144	5145	5146	5147	5148	5149
article_id										
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
18.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
25.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
29.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
32.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
33.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
34.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
36.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
39.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
40.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
43.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
48.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
51.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
53.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
54.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
57.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
58.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...

```
1412.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1414.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1415.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1416.0        0.0   0.0   0.0   0.0   0.0   0.0   1.0   0.0   0.0   0.0
1418.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1419.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1420.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1421.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1422.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1423.0        1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1424.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1425.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1426.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1427.0        1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1428.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1429.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1430.0        1.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0
1431.0        0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0
1432.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1433.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1434.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1435.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1436.0        0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0
1437.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1439.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1440.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1441.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1442.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1443.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1444.0        0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[714 rows x 5149 columns]
```

In [171]:
```
# set the owner's id and get the interaction data of this user
selected_user_id = 3
useri = user_item[user_item.index == selected_user_id]
useri
```

Out[171]:
```
article_id  0.0     2.0     4.0     8.0     9.0     12.0    14.0    15.0    \
user_id
3              0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0

article_id  16.0    18.0     ...    1434.0  1435.0  1436.0  1437.0  1439.0  \
user_id                      ...
3              0.0     0.0   ...       0.0     0.0     1.0     0.0     0.0

article_id  1440.0  1441.0  1442.0  1443.0  1444.0
user_id
```

```
3               0.0     0.0     0.0     0.0     0.0

[1 rows x 714 columns]
```

In [102]:
```
# use the dot product to calculate the similarity of other users to the owner
useri.dot(user_item.transpose())
```

Out[102]:
```
user_id  1     2     3     4     5     6     7     8     9     10    ...   \
user_id                                                            ...
3         6.0   1.0  40.0   5.0   1.0   7.0   1.0   5.0   2.0   5.0 ...

user_id  5140  5141  5142  5143  5144  5145  5146  5147  5148  5149
user_id
3         7.0   0.0   0.0   5.0   0.0   2.0   0.0   0.0   0.0   0.0

[1 rows x 5149 columns]
```

Find the way to sort by similarity

In [127]:
```
# cannot sort the pivot matrix directly
useri.dot(user_item.transpose()).sort_values(by = "user_id", axis = 1, ascending = Tru
```

Out[127]:
```
user_id  1     2     3     4     5     6     7     8     9     10    ...   \
user_id                                                            ...
3         6.0   1.0  40.0   5.0   1.0   7.0   1.0   5.0   2.0   5.0 ...

user_id  5140  5141  5142  5143  5144  5145  5146  5147  5148  5149
user_id
3         7.0   0.0   0.0   5.0   0.0   2.0   0.0   0.0   0.0   0.0

[1 rows x 5149 columns]
```

In [146]:
```
# test to use stack, cannot use because of duplicated 'user_id'
useri.dot(user_item.transpose()).stack()
```

Out[146]:
```
user_id  user_id
3        1           6.0
         2           1.0
         3          40.0
         4           5.0
         5           1.0
         6           7.0
         7           1.0
         8           5.0
         9           2.0
         10          5.0
         11         13.0
         12          0.0
         13          2.0
```

```
14          1.0
15          1.0
16          1.0
17          3.0
18          2.0
19          0.0
20          0.0
21         12.0
22          4.0
23         23.0
24          3.0
25          0.0
26          4.0
27          4.0
28         10.0
29          0.0
30          0.0
            ...
5120        0.0
5121        0.0
5122        0.0
5123        4.0
5124        3.0
5125        0.0
5126        0.0
5127        3.0
5128        0.0
5129        1.0
5130        1.0
5131        0.0
5132        0.0
5133        1.0
5134        3.0
5135        0.0
5136        0.0
5137        0.0
5138       13.0
5139        2.0
5140        7.0
5141        0.0
5142        0.0
5143        5.0
5144        0.0
5145        2.0
5146        0.0
5147        0.0
5148        0.0
5149        0.0
```

```
Length: 5149, dtype: float64
```

```
In [169]: # need to to melt the pivot matrix to sort the similarity
          similarity = useri.dot(user_item.transpose()).melt().sort_values(by = 'value', ascendi
          similarity.head()
```

```
Out[169]:       user_id  value
          3352     3353   40.0
          2           3   40.0
          22         23   23.0
          3781     3782   23.0
          3763     3764   17.0
```

Create list of just the user ids

```
In [173]: # get the user_id
          most_similar_users = similarity['user_id']
          most_similar_users.head()
```

```
Out[173]: 3352    3353
          2          3
          22        23
          3781    3782
          3763    3764
          Name: user_id, dtype: int64
```

Remove the own user's id

```
In [177]: # the owner's id
          selected_user_id
```

```
Out[177]: 3
```

```
In [180]: # remove the owner's id
          most_similar_users = most_similar_users[most_similar_users!= selected_user_id]
          most_similar_users
```

```
Out[180]: 3352    3353
          22        23
          3781    3782
          3763    3764
          97        98
          4458    4459
          202      203
          48        49
          3696    3697
          3595    3596
          51        52
          911      912
```

3539	3540
203	204
39	40
4931	4932
241	242
5137	5138
3869	3870
3909	3910
3965	3966
2925	2926
130	131
10	11
3577	3578
4773	4774
764	765
124	125
213	214
4784	4785
	...
2484	2485
2485	2486
2454	2455
2453	2454
2447	2448
2442	2443
2405	2406
2407	2408
2408	2409
2410	2411
2411	2412
2412	2413
2418	2419
2419	2420
2420	2421
2423	2424
2425	2426
2426	2427
2427	2428
2428	2429
2431	2432
2432	2433
2433	2434
2435	2436
2436	2437
2437	2438
2438	2439
2440	2441
2441	2442

```
5148    5149
Name: user_id, Length: 5148, dtype: int64
```

In [189]:
```python
# test the top similar users
most_similar_users.reset_index(drop=True, inplace=True)
most_similar_users[:10]
```

Out[189]:
```
0    3353
1      23
2    3782
3    3764
4      98
5    4459
6     203
7      49
8    3697
9    3596
Name: user_id, dtype: int64
```

In [5]:
```python
def find_similar_users(user_id, user_item=user_item):
    '''
    INPUT:
    user_id - (int) a user_id
    user_item - (pandas dataframe) matrix of users by articles:
                1's when a user has interacted with an article, 0 otherwise

    OUTPUT:
    similar_users - (list) an ordered list where the closest users (largest dot product
                    are listed first

    Description:
    Computes the similarity of every pair of users based on the dot product
    Returns an ordered

    '''
    # compute similarity of each user to the provided user
    useri = user_item[user_item.index == user_id]
    useri.dot(user_item.transpose())

    # sort by similarity
    similarity = useri.dot(user_item.transpose()).melt().sort_values(by = 'value', ascen

    # create list of just the ids
    most_similar_users = similarity['user_id']

    # remove the own user's id
    most_similar_users = most_similar_users[most_similar_users!= user_id]
    most_similar_users.reset_index(drop=True, inplace=True)
```

```
    return most_similar_users # return a list of the users in order from most to least s
```

```
In [9]: # Do a spot check of your function
        print("The 10 most similar users to user 1 are: {}".format(find_similar_users(1)[:10]))
        print("The 5 most similar users to user 3933 are: {}".format(find_similar_users(3933)[:5
        print("The 3 most similar users to user 46 are: {}".format(find_similar_users(46)[:3]))
```

```
The 10 most similar users to user 1 are: 0    3933
1      23
2    3782
3     203
4    4459
5    3870
6     131
7    4201
8      46
9    5041
Name: user_id, dtype: int64
The 5 most similar users to user 3933 are: 0       1
1      23
2    3782
3     203
4    4459
Name: user_id, dtype: int64
The 3 most similar users to user 46 are: 0    4201
1    3782
2      23
Name: user_id, dtype: int64
```

3. Now that you have a function that provides the most similar users to each user, you will want to use these users to find articles you can recommend. Complete the functions below to return the articles you would recommend to each user.

a. Draft the functions

Get article names from article ids

```
In [61]: df.head()
```

Out[61]:

	article_id	title	user_id
0	1430.0	using pixiedust for fast, flexible, and easier...	1
1	1314.0	healthcare python streaming application demo	2
2	1429.0	use deep learning for image classification	3
3	1338.0	ml optimization using cognitive assistant	4
4	1276.0	deploy your python model as a restful api	5

```
In [127]: article_ids_list = ['1430.0', '1276.0', '1429.0']
          article_ids_list = np.array(article_ids_list, dtype=np.float64)
          article_ids_list
```

Out[127]: array([1430., 1276., 1429.])

In [111]:
```
df_article_drop=df.drop_duplicates(subset=['article_id'], keep='first')[['article_id',
df_article_drop.head()
```

Out[111]:

	article_id	title
0	1430.0	using pixiedust for fast, flexible, and easier...
1	1314.0	healthcare python streaming application demo
2	1429.0	use deep learning for image classification
3	1338.0	ml optimization using cognitive assistant
4	1276.0	deploy your python model as a restful api

In [119]:
```
# isin doesn't keep the order of article_ids -> cannot use
article_names = df_article_drop[df_article_drop['article_id'].isin(article_ids_list)==
article_names
```

Out[119]:
```
0    using pixiedust for fast, flexible, and easier...
2           use deep learning for image classification
4            deploy your python model as a restful api
Name: title, dtype: object
```

In [172]:
```
article_names = df_article_drop.set_index('article_id').loc[article_ids_list].reset_in
article_names
```

Out[172]:
```
0    using pixiedust for fast, flexible, and easier...
1            deploy your python model as a restful api
2           use deep learning for image classification
Name: title, dtype: object
```

Get the articles seen by a user

In [175]:
```
selected_user_id = 20
user_article = user_item[user_item.index == selected_user_id]
user_article
```

Out[175]:
```
article_id  0.0     2.0     4.0     8.0     9.0     12.0    14.0    15.0    \
user_id
20             0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0

article_id  16.0    18.0     ...    1434.0  1435.0  1436.0  1437.0  1439.0  \
user_id                      ...
20             0.0     0.0   ...       0.0     0.0     0.0     0.0     0.0

article_id  1440.0  1441.0  1442.0  1443.0  1444.0
user_id
20             0.0     0.0     0.0     0.0     0.0

[1 rows x 714 columns]
```

```
In [176]: user_article=user_article.melt()
          user_article.head()
```

```
Out[176]:    article_id  value
          0         0.0    0.0
          1         2.0    0.0
          2         4.0    0.0
          3         8.0    0.0
          4         9.0    0.0
```

```
In [177]: article_ids_list = user_article[user_article['value'] == 1.0]['article_id']
          #article_ids_list.reset_index(drop=True, inplace=True)
          article_ids_list.head()
```

```
Out[177]: 104     232.0
          347     844.0
          619    1320.0
          Name: article_id, dtype: float64
```

Recommend the articles the user hasn't seen before from the articles of close users

```
In [9]: #find the list of close users
        user_id = 1
        closeness_loop = find_similar_users(user_id, user_item)
        closeness_loop.head(10)
```

```
Out[9]: 0    3933
        1      23
        2    3782
        3     203
        4    4459
        5    3870
        6     131
        7    4201
        8      46
        9    5041
        Name: user_id, dtype: int64
```

```
In [12]: read_article_ids, read_article_names = get_user_articles(user_id, user_item)
```

```
In [13]: read_article_ids
```

```
Out[13]: 0       43.0
         1      109.0
         2      151.0
         3      268.0
         4      310.0
         5      329.0
         6      346.0
```

```
7       390.0
8       494.0
9       525.0
10      585.0
11      626.0
12      668.0
13      732.0
14      768.0
15      910.0
16      968.0
17      981.0
18     1052.0
19     1170.0
20     1183.0
21     1185.0
22     1232.0
23     1293.0
24     1305.0
25     1363.0
26     1368.0
27     1391.0
28     1400.0
29     1406.0
30     1427.0
31     1429.0
32     1430.0
33     1431.0
34     1436.0
35     1439.0
Name: article_id, dtype: float64
```

```
In [45]: m_recs = 10
         user_id = 1
         closeness_loop = find_similar_users(user_id, user_item)
         r = 0
         recs = []
         read_article_ids, read_article_names = get_user_articles(user_id, user_item)
         print(read_article_ids)
         for i in closeness_loop:
             rec_article_ids , rec_article_names = get_user_articles(i, user_item)
             for j in rec_article_ids:
                 if (j not in read_article_ids.tolist()):
                     r = r + 1
                     recs.append(j)
                     if r == m_recs:
                         break
             if r == m_recs:
                 break
```

```
print(recs)
```

```
0       43.0
1      109.0
2      151.0
3      268.0
4      310.0
5      329.0
6      346.0
7      390.0
8      494.0
9      525.0
10     585.0
11     626.0
12     668.0
13     732.0
14     768.0
15     910.0
16     968.0
17     981.0
18    1052.0
19    1170.0
20    1183.0
21    1185.0
22    1232.0
23    1293.0
24    1305.0
25    1363.0
26    1368.0
27    1391.0
28    1400.0
29    1406.0
30    1427.0
31    1429.0
32    1430.0
33    1431.0
34    1436.0
35    1439.0
Name: article_id, dtype: float64
[2.0, 12.0, 14.0, 16.0, 26.0, 28.0, 29.0, 33.0, 50.0, 74.0]
```

In [37]:
```
43.0 in read_article_ids
```

Out[37]: False

In [38]:
```
read_article_ids.tolist()
```

```
Out[38]: [43.0,
          109.0,
```

```
         151.0,
         268.0,
         310.0,
         329.0,
         346.0,
         390.0,
         494.0,
         525.0,
         585.0,
         626.0,
         668.0,
         732.0,
         768.0,
         910.0,
         968.0,
         981.0,
         1052.0,
         1170.0,
         1183.0,
         1185.0,
         1232.0,
         1293.0,
         1305.0,
         1363.0,
         1368.0,
         1391.0,
         1400.0,
         1406.0,
         1427.0,
         1429.0,
         1430.0,
         1431.0,
         1436.0,
         1439.0]
```

b. Fill in the functions

```
In [6]: #def get_article_names(article_ids, df=df):
        def get_article_names(article_ids, df=df):
            '''
            INPUT:
            article_ids - (list) a list of article ids
            df - (pandas dataframe) df as defined at the top of the notebook

            OUTPUT:
            article_names - (list) a list of article names associated with the list of article i
                            (this is identified by the title column)
            '''
```

```
    # Your code here
    article_ids = np.array(article_ids, dtype=np.float64)
    df_article_drop=df.drop_duplicates(subset=['article_id'], keep='first')[['article_id
    #article_names = df_article_drop[df_article_drop['article_id'].isin(article_ids)==Tr
    article_names = df_article_drop.set_index('article_id').loc[article_ids].reset_index
    #article_names.reset_index(drop=True, inplace=True)
    article_names = article_names.tolist()

    return article_names # Return the article names associated with list of article ids


#def get_user_articles(user_id, user_item=user_item):
def get_user_articles(user_id):
    '''
    INPUT:
    user_id - (int) a user id
    user_item - (pandas dataframe) matrix of users by articles:
                1's when a user has interacted with an article, 0 otherwise

    OUTPUT:
    article_ids - (list) a list of the article ids seen by the user
    article_names - (list) a list of article names associated with the list of article i
                    (this is identified by the doc_full_name column in df_content)

    Description:
    Provides a list of the article_ids and article titles that have been seen by a user
    '''
    # Your code here
    user_article_list = user_item[user_item.index == user_id]
    user_article_list = user_article_list.melt()
    article_ids = user_article_list[user_article_list['value'] == 1.0]['article_id']
    #article_ids.reset_index(drop=True, inplace=True)
    article_ids = article_ids.tolist()

    article_ids = [str(x) for x in article_ids]

    article_names = get_article_names(article_ids, df=df)

    return article_ids, article_names # return the ids and names


def user_user_recs(user_id, m=10):
    '''
    INPUT:
    user_id - (int) a user id
    m - (int) the number of recommendations you want for the user

    OUTPUT:
```

```
            recs - (list) a list of recommendations for the user

            Description:
            Loops through the users based on closeness to the input user_id
            For each user - finds articles the user hasn't seen before and provides them as recs
            Does this until m recommendations are found

            Notes:
            Users who are the same closeness are chosen arbitrarily as the 'next' user

            For the user where the number of recommended articles starts below m
            and ends exceeding m, the last items are chosen arbitrarily

            '''
            # Your code here
            closeness_loop = find_similar_users(user_id, user_item)
            r = 0
            recs = []
            #read_article_ids, read_article_names = get_user_articles(user_id, user_item)
            read_article_ids, read_article_names = get_user_articles(user_id)
            for i in closeness_loop:
                #rec_article_ids , rec_article_names = get_user_articles(i, user_item)
                rec_article_ids , rec_article_names = get_user_articles(i)
                for j in rec_article_ids:
                    if (j not in read_article_ids):
                        r = r + 1
                        recs.append(j)
                        if r == m:
                            break
                if r == m:
                    break
            return recs # return your recommendations for this user_id
```

In [7]:
```
# Check Results
article_ids = [1430.0, 1429.0, 1276.0]
#get_article_names(article_ids, df)
get_article_names(article_ids)
```

Out[7]:
```
['using pixiedust for fast, flexible, and easier data analysis and experimentation',
 'use deep learning for image classification',
 'deploy your python model as a restful api']
```

In [8]:
```
# Check Results
#get_user_articles(1, user_item)
get_user_articles(1)
```

Out[8]:
```
(['43.0',
  '109.0',
  '151.0',
```

```
 '268.0',
 '310.0',
 '329.0',
 '346.0',
 '390.0',
 '494.0',
 '525.0',
 '585.0',
 '626.0',
 '668.0',
 '732.0',
 '768.0',
 '910.0',
 '968.0',
 '981.0',
 '1052.0',
 '1170.0',
 '1183.0',
 '1185.0',
 '1232.0',
 '1293.0',
 '1305.0',
 '1363.0',
 '1368.0',
 '1391.0',
 '1400.0',
 '1406.0',
 '1427.0',
 '1429.0',
 '1430.0',
 '1431.0',
 '1436.0',
 '1439.0'],
['deep learning with tensorflow course by big data university',
 'tensorflow quick tips',
 'jupyter notebook tutorial',
 'sector correlations shiny app',
 'time series prediction using recurrent neural networks (lstms)',
 'introduction to market basket analysis in\xa0python',
 'fighting gerrymandering: using data science to draw fairer congressional districts',
 'introducing ibm watson studio ',
 'python for loops explained (python for data science basics #5)',
 'new shiny cheat sheet and video tutorial',
 'tidyverse practice: mapping large european cities',
 'analyze db2 warehouse on cloud data in rstudio in dsx',
 'shiny: a data scientists best friend',
 'rapidly build machine learning flows with dsx',
 'python if statements explained (python for data science basics #4)',
```

```
 'working with ibm cloud object storage in python',
 'shiny 0.13.0',
 'super fast string matching in python',
 'access db2 warehouse on cloud and db2 with python',
 'apache spark lab, part 1: basic concepts',
 'categorize urban density',
 'classify tumors with machine learning',
 'country statistics: life expectancy at birth',
 'finding optimal locations of new store using decision optimization',
 'gosales transactions for naive bayes model',
 'predict loan applicant behavior with tensorflow neural networking',
 'putting a human face on machine learning',
 'sudoku',
 'uci ml repository: chronic kidney disease data set',
 'uci: iris',
 'use xgboost, scikit-learn & ibm watson machine learning apis',
 'use deep learning for image classification',
 'using pixiedust for fast, flexible, and easier data analysis and experimentation',
 'visualize car data with brunel',
 'welcome to pixiedust',
 'working with ibm cloud object storage in r'])
```

In [11]:
```
# Check Results
get_article_names(user_user_recs(1, 10)) # Return 10 recommendations for user 1
```

Out[11]:
```
['this week in data science (april 18, 2017)',
 'timeseries data analysis of iot events by using jupyter notebook',
 'got zip code data? prep it for analytics.  ibm watson data lab  medium',
 'higher-order logistic regression for large datasets',
 'using machine learning to predict parking difficulty',
 'deep forest: towards an alternative to deep neural networks',
 'experience iot with coursera',
 'using brunel in ipython/jupyter notebooks',
 'graph-based machine learning',
 'the 3 kinds of context: machine learning and the art of the frame']
```

In [12]:
```
get_article_names(['1024.0','1176.0', '1305.0', '1314.0', '1422.0', '1427.0'])
```

Out[12]:
```
['using deep learning to reconstruct high-resolution audio',
 'build a python app on the streaming analytics service',
 'gosales transactions for naive bayes model',
 'healthcare python streaming application demo',
 'use r dataframes & ibm watson natural language understanding',
 'use xgboost, scikit-learn & ibm watson machine learning apis']
```

In [13]:
```
get_article_names(['1314.0'])
```

Out[13]:
```
['healthcare python streaming application demo']
```

In [14]:
```
get_article_names(['1320.0', '232.0', '844.0'])
```

```
Out[14]: ['housing (2015): united states demographic measures',
          'self-service data preparation with ibm data refinery',
          'use the cloudant-spark connector in python notebook']
```

```
In [170]: get_user_articles(2)[0]
```

```
Out[170]: [1024.0, 1176.0, 1305.0, 1314.0, 1422.0, 1427.0]
```

```
In [169]: get_user_articles(20)[0]
```

```
Out[169]: [232.0, 844.0, 1320.0]
```

```
In [15]: # Test your functions here - No need to change this code - just run this cell
         assert set(get_article_names(['1024.0', '1176.0', '1305.0', '1314.0', '1422.0', '1427.0
         assert set(get_article_names(['1320.0', '232.0', '844.0'])) == set(['housing (2015): un
         assert set(get_user_articles(20)[0]) == set(['1320.0', '232.0', '844.0'])
         assert set(get_user_articles(20)[1]) == set(['housing (2015): united states demographic
         assert set(get_user_articles(2)[0]) == set(['1024.0', '1176.0', '1305.0', '1314.0', '14
         assert set(get_user_articles(2)[1]) == set(['using deep learning to reconstruct high-re
         print("If this is all you see, you passed all of our tests!  Nice job!")
```

```
If this is all you see, you passed all of our tests!  Nice job!
```

4. Now we are going to improve the consistency of the **user_user_recs** function from above.

- Instead of arbitrarily choosing when we obtain users who are all the same closeness to a given user - choose the users that have the most total article interactions before choosing those with fewer article interactions.
- Instead of arbitrarily choosing articles from the user where the number of recommended articles starts below m and ends exceeding m, choose articles with the articles with the most total interactions before choosing those with fewer total interactions. This ranking should be what would be obtained from the **top_articles** function you wrote earlier.

a. Draft the functions

- The neighbor data frame including the neighbor's similarity and the number of articles viewed by the neighbor

Calculate the neighbor's similarity

```
In [8]: selected_user_id = 3
        useri = user_item[user_item.index == selected_user_id]
        useri
```

```
Out[8]: article_id  0.0     2.0     4.0     8.0     9.0     12.0    14.0    15.0    \
        user_id
        3              0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0
```

```
article_id  16.0    18.0     ...    1434.0  1435.0  1436.0  1437.0  1439.0  \
user_id                      ...
3              0.0     0.0   ...       0.0     0.0     1.0     0.0     0.0

article_id  1440.0  1441.0  1442.0  1443.0  1444.0
user_id
3              0.0     0.0     0.0     0.0     0.0

[1 rows x 714 columns]
```

In [17]:
```
neighbors = useri.dot(user_item.transpose()).melt().sort_values(by = 'user_id', ascendi
neighbors.columns = ['user_id', 'similarity']
neighbors.head()
```

Out[17]:

	user_id	similarity
0	1	6.0
1	2	1.0
2	3	40.0
3	4	5.0
4	5	1.0

Calculate the number of articles viewed by the neighbor

In [16]:
```
df.groupby(['user_id']).size()
```

Out[16]:
```
user_id
1        47
2         6
3        82
4        45
5         5
6        19
7         4
8        82
9        32
10       22
11       35
12       13
13       21
14       28
15       17
16        3
17       35
18        3
19        8
20        3
21      137
22       37
23      364
```

```
24      30
25      10
26      27
27      34
28      42
29       1
30       5
       ...
5120     1
5121     1
5122     1
5123    13
5124    22
5125     1
5126     2
5127    29
5128     4
5129    29
5130     1
5131     1
5132     2
5133     3
5134     9
5135     3
5136     2
5137     2
5138    95
5139    13
5140   101
5141     1
5142     2
5143    25
5144     1
5145     6
5146     9
5147     1
5148     1
5149     1
Length: 5149, dtype: int64
```

```
In [13]: df.groupby(['user_id']).size().tolist()
```

```
Out[13]: [47,
          6,
          82,
          45,
          5,
          19,
```

```
4,
82,
32,
22,
35,
13,
21,
28,
17,
3,
35,
3,
8,
3,
137,
37,
364,
30,
10,
27,
34,
42,
1,
5,
12,
13,
8,
7,
19,
7,
29,
68,
2,
78,
14,
9,
9,
15,
73,
63,
1,
12,
147,
8,
11,
132,
5,
24,
```

```
33,
38,
25,
142,
7,
103,
25,
2,
31,
57,
32,
36,
58,
5,
35,
5,
14,
49,
6,
9,
7,
8,
4,
4,
7,
5,
8,
25,
10,
2,
25,
14,
69,
57,
8,
34,
5,
13,
1,
27,
12,
3,
3,
170,
1,
13,
1,
10,
```

```
35,
7,
8,
2,
19,
3,
7,
25,
35,
6,
68,
20,
31,
2,
3,
15,
6,
21,
33,
22,
2,
1,
59,
39,
1,
1,
47,
2,
145,
25,
20,
30,
82,
5,
7,
3,
8,
26,
1,
2,
4,
10,
39,
7,
8,
6,
26,
7,
```

```
11,
25,
13,
12,
33,
8,
7,
4,
7,
29,
3,
1,
16,
16,
3,
7,
4,
52,
55,
116,
8,
7,
53,
1,
2,
40,
2,
4,
1,
10,
28,
7,
11,
104,
8,
79,
63,
18,
3,
14,
6,
16,
15,
21,
72,
15,
43,
14,
```

```
50,
13,
1,
1,
160,
97,
26,
3,
26,
25,
21,
9,
52,
8,
35,
62,
29,
2,
20,
1,
8,
32,
6,
6,
79,
1,
12,
10,
15,
6,
7,
31,
6,
7,
9,
1,
51,
1,
9,
5,
20,
35,
26,
148,
10,
23,
10,
13,
```

```
2,
8,
94,
3,
49,
25,
6,
2,
3,
60,
1,
2,
4,
21,
24,
41,
26,
3,
2,
22,
20,
45,
2,
3,
21,
5,
84,
6,
2,
4,
9,
1,
1,
29,
13,
1,
6,
4,
13,
2,
33,
91,
4,
80,
7,
15,
2,
17,
```

```
91,
35,
16,
6,
24,
2,
6,
4,
6,
47,
28,
2,
2,
9,
1,
22,
1,
38,
52,
1,
16,
1,
2,
4,
46,
5,
59,
85,
25,
39,
3,
10,
3,
14,
2,
76,
1,
15,
8,
49,
13,
3,
5,
2,
6,
6,
3,
1,
```

```
15,
11,
6,
41,
32,
3,
1,
4,
13,
17,
1,
17,
39,
4,
8,
2,
17,
16,
6,
55,
17,
4,
22,
2,
2,
2,
2,
11,
95,
2,
14,
1,
29,
2,
5,
1,
53,
11,
12,
4,
46,
2,
5,
1,
10,
1,
4,
2,
```

```
12,
1,
4,
1,
69,
5,
1,
1,
7,
1,
1,
20,
27,
38,
8,
10,
4,
2,
50,
12,
12,
1,
22,
8,
6,
3,
5,
60,
16,
53,
17,
7,
4,
7,
4,
5,
4,
5,
2,
4,
5,
12,
4,
19,
2,
1,
5,
29,
```

```
12,
6,
6,
10,
8,
7,
41,
2,
1,
5,
1,
11,
2,
1,
12,
13,
7,
23,
11,
1,
3,
9,
3,
3,
2,
14,
1,
4,
2,
1,
11,
24,
26,
13,
5,
7,
5,
4,
4,
11,
42,
10,
3,
6,
17,
14,
2,
4,
```

```
8,
47,
4,
35,
3,
28,
1,
12,
4,
1,
6,
14,
10,
18,
10,
9,
2,
1,
1,
1,
1,
12,
28,
25,
31,
39,
13,
2,
23,
17,
2,
35,
2,
2,
37,
2,
5,
1,
13,
1,
10,
1,
3,
11,
1,
8,
12,
14,
```

```
45,
37,
2,
38,
10,
4,
2,
27,
1,
1,
15,
3,
5,
11,
4,
2,
6,
3,
1,
6,
35,
1,
23,
25,
1,
13,
17,
2,
13,
3,
10,
1,
6,
20,
2,
11,
1,
4,
4,
1,
23,
5,
1,
4,
1,
2,
2,
24,
```

```
10,
9,
11,
4,
7,
2,
1,
3,
84,
7,
2,
1,
2,
19,
1,
18,
1,
21,
13,
1,
1,
1,
1,
5,
24,
1,
10,
1,
1,
17,
14,
6,
1,
2,
2,
7,
84,
2,
52,
1,
2,
11,
2,
9,
3,
1,
5,
3,
```

```
8,
1,
1,
2,
14,
8,
14,
24,
38,
35,
29,
11,
3,
38,
1,
36,
38,
26,
29,
8,
98,
19,
5,
4,
1,
48,
4,
18,
16,
5,
26,
21,
16,
7,
77,
13,
13,
59,
67,
41,
5,
4,
21,
16,
4,
15,
2,
4,
```

```
6,
8,
18,
12,
10,
8,
8,
12,
24,
52,
34,
3,
3,
10,
55,
29,
4,
79,
26,
18,
14,
44,
1,
8,
8,
2,
7,
18,
23,
2,
8,
8,
6,
36,
19,
6,
45,
3,
1,
6,
8,
8,
21,
35,
31,
7,
2,
64,
```

```
15,
5,
6,
11,
3,
9,
22,
15,
14,
7,
5,
16,
8,
5,
14,
2,
2,
7,
40,
16,
1,
6,
3,
64,
18,
5,
9,
60,
16,
7,
11,
11,
12,
35,
20,
14,
27,
3,
68,
5,
1,
2,
3,
9,
10,
3,
20,
23,
```

```
2,
3,
4,
11,
2,
5,
3,
3,
31,
10,
57,
28,
8,
6,
11,
18,
31,
5,
1,
34,
1,
9,
19,
21,
4,
1,
7,
3,
3,
5,
2,
3,
50,
15,
14,
17,
8,
13,
5,
23,
3,
10,
5,
2,
6,
36,
7,
34,
```

```
18,
4,
21,
5,
1,
16,
53,
4,
3,
11,
19,
3,
3,
7,
4,
1,
2,
11,
6,
19,
15,
16,
31,
4,
16,
2,
4,
7,
22,
2,
1,
27,
8,
8,
3,
8,
3,
42,
1,
18,
2,
2,
7,
8,
1,
4,
2,
11,
```

```
18,
2,
4,
6,
9,
8,
1,
10,
5,
7,
8,
5,
2,
7,
13,
13,
18,
1,
15,
11,
14,
5,
4,
8,
21,
28,
11,
5,
2,
20,
3,
14,
3,
2,
1,
8,
27,
4,
4,
8,
2,
102,
13,
14,
3,
3,
9,
14,
```

```
4,
9,
1,
11,
19,
23,
14,
48,
35,
3,
2,
3,
3,
11,
10,
2,
3,
12,
8,
21,
18,
7,
4,
7,
19,
6,
5,
16,
3,
15,
5,
19,
2,
6,
10,
7,
14,
3,
18,
10,
14,
8,
6,
22,
1,
4,
11,
8,
```

```
 6,
 2,
 16,
 30,
 11,
 13,
 7,
 8,
 9,
 6,
 13,
 9,
 2,
 9,
 11,
 7,
 3,
 7,
 12,
 3,
 19,
 30,
 1,
 5,
 2,
 14,
 2,
 3,
 12,
 8,
 9,
 8,
 6,
 4,
 ...]
```

```
In [19]: neighbors['num_interactions'] = df.groupby(['user_id']).size().tolist()
         neighbors.head()
```

Out[19]:

	user_id	similarity	num_interactions
0	1	6.0	47
1	2	1.0	6
2	3	40.0	82
3	4	5.0	45
4	5	1.0	5

```
In [21]: neighbors = neighbors.sort_values(by = ['similarity','num_interactions'], ascending = F
         neighbors.head()
```

Out[21]:

	user_id	similarity	num_interactions
2	3	40.0	82
3352	3353	40.0	80
22	23	23.0	364
3781	3782	23.0	363
97	98	17.0	170

- Recommendations for the user

In [12]:
```
selected_user_id = 3
closeness_list = get_top_sorted_users(selected_user_id, df, user_item)['user_id'].tolis
```

In [14]:
```
closeness_list
```

Out[14]:
```
[3,
 3353,
 23,
 3782,
 98,
 3764,
 203,
 4459,
 49,
 3697,
 52,
 3596,
 912,
 3540,
 242,
 3910,
 131,
 3870,
 204,
 5138,
 2926,
 40,
 4932,
 11,
 3966,
 21,
 4785,
 619,
 195,
 3578,
 765,
 214,
 125,
 4774,
 4755,
```

```
754,
3141,
4642,
656,
3024,
371,
3784,
290,
28,
2161,
288,
4706,
273,
135,
3621,
186,
696,
2982,
46,
4201,
211,
3485,
807,
334,
3794,
4038,
383,
3801,
4824,
4471,
134,
58,
3740,
184,
4892,
295,
3006,
322,
3622,
4134,
3532,
4293,
395,
113,
4883,
5041,
67,
72,
```

```
126,
3172,
111,
3057,
4161,
1058,
4392,
170,
3169,
60,
5140,
249,
3483,
591,
4277,
223,
3358,
750,
187,
3967,
4934,
1059,
4595,
64,
3856,
3136,
621,
3775,
199,
4618,
4209,
197,
479,
3695,
3877,
1062,
1068,
3264,
56,
538,
4449,
3058,
722,
90,
689,
791,
3938,
215,
```

```
471,
648,
3079,
209,
2989,
6,
4904,
755,
741,
651,
3072,
665,
4484,
38,
669,
3208,
3684,
726,
5057,
4900,
693,
4308,
420,
688,
235,
3486,
5059,
4404,
1,
129,
488,
4206,
268,
3933,
4660,
1355,
3651,
4778,
860,
1897,
3061,
745,
5079,
4323,
536,
712,
3960,
55,
```

```
3783,
4137,
3417,
3968,
4231,
4792,
280,
492,
3636,
4623,
261,
4272,
244,
2908,
4543,
4788,
761,
193,
196,
3878,
3589,
469,
8,
2975,
330,
3197,
256,
418,
668,
2903,
5078,
5080,
88,
169,
362,
173,
2981,
313,
3007,
3238,
409,
4494,
3829,
4,
4145,
262,
670,
2290,
```

```
3741,
145,
3147,
312,
647,
2913,
521,
3614,
4526,
4802,
646,
4241,
69,
640,
794,
3674,
3879,
4167,
65,
220,
347,
3611,
511,
3439,
1163,
3100,
4511,
375,
438,
1240,
3118,
4842,
542,
4142,
697,
4876,
85,
5143,
582,
456,
1384,
10,
733,
1262,
3474,
4491,
3310,
4225,
```

```
3918,
790,
1105,
3520,
227,
545,
3163,
45,
3500,
87,
3818,
321,
3637,
379,
3693,
1040,
251,
3949,
4515,
319,
535,
715,
3898,
700,
3957,
4021,
4497,
3414,
512,
2718,
644,
22,
3802,
5118,
66,
296,
5023,
27,
1026,
2974,
155,
287,
1272,
3421,
1401,
3245,
3553,
3633,
```

```
723,
4088,
970,
3336,
3535,
4200,
649,
896,
3441,
3884,
26,
854,
907,
3812,
110,
152,
510,
1076,
1139,
3835,
4453,
5077,
2953,
3732,
4268,
1063,
1162,
1251,
2994,
3291,
310,
1198,
1261,
3108,
3420,
3710,
4872,
600,
1353,
3082,
114,
239,
900,
3354,
3378,
3765,
3943,
842,
```

```
1386,
500,
658,
706,
751,
3237,
3724,
294,
352,
1310,
3176,
3466,
3043,
4086,
4720,
3240,
3831,
464,
1567,
2288,
2423,
3495,
560,
601,
886,
3296,
4386,
5123,
48,
225,
411,
1215,
3329,
4378,
5105,
4680,
4696,
1426,
4667,
753,
770,
1265,
1302,
76,
1588,
3198,
3402,
714,
```

```
1476,
785,
3570,
168,
4517,
926,
304,
3005,
4725,
346,
445,
176,
324,
1281,
3632,
404,
3408,
3986,
4393,
4943,
4427,
17,
490,
4668,
121,
3211,
3691,
2007,
63,
24,
1048,
1266,
2194,
3212,
4204,
641,
1197,
2790,
3313,
3477,
5127,
181,
509,
786,
2144,
3376,
4697,
5053,
```

```
5086,
94,
4560,
4933,
661,
1137,
3292,
3430,
3754,
3763,
4274,
4768,
82,
208,
252,
323,
558,
1014,
2312,
3442,
3518,
3919,
4107,
4285,
575,
707,
1172,
2955,
4015,
4082,
122,
266,
365,
413,
851,
3124,
4148,
5124,
271,
673,
721,
895,
938,
1110,
1114,
4557,
4877,
1038,
```

```
1548,
1668,
2202,
2487,
2901,
3301,
3305,
3768,
4010,
4071,
5012,
5055,
35,
107,
713,
797,
833,
923,
943,
987,
1084,
1577,
2993,
3412,
3634,
4186,
598,
698,
862,
1082,
1301,
3325,
3338,
3807,
3969,
4098,
4099,
4527,
4915,
5115,
354,
2907,
3366,
4336,
847,
1213,
3225,
4636,
```

```
4992,
5001,
5070,
808,
1156,
1350,
1354,
4319,
5052,
86,
699,
809,
925,
992,
1395,
3042,
32,
100,
285,
666,
2270,
2284,
3117,
3346,
3616,
3618,
4472,
4620,
410,
508,
1264,
1671,
2948,
2983,
3864,
4044,
4949,
4958,
932,
1250,
2186,
2308,
3506,
4577,
83,
406,
609,
683,
```

```
1317,
1478,
3063,
3206,
3213,
3235,
4039,
74,
584,
980,
1003,
1061,
1322,
3514,
3669,
3749,
4085,
5134,
248,
703,
719,
856,
858,
866,
894,
1208,
1291,
2304,
2636,
3017,
4843,
424,
736,
1225,
1408,
1443,
1801,
3326,
3464,
3581,
3824,
3888,
5111,
253,
1331,
1612,
2919,
3488,
```

```
3757,
4119,
4834,
5046,
660,
1359,
2377,
3561,
3956,
4970,
4995,
1345,
5034,
1094,
1635,
3667,
3833,
2205,
639,
4076,
820,
4130,
103,
213,
240,
760,
927,
4367,
4522,
4619,
3970,
4837,
9,
115,
783,
988,
2488,
3247,
3705,
160,
694,
4487,
4502,
305,
4170,
4364,
4820,
403,
```

```
763,
2415,
140,
149,
241,
4275,
4882,
57,
132,
1025,
3195,
3727,
4390,
4588,
54,
299,
470,
607,
638,
4025,
4255,
4685,
557,
814,
924,
2956,
2978,
4073,
4250,
4311,
962,
1051,
1633,
4495,
5013,
13,
120,
662,
798,
825,
2579,
4248,
568,
3093,
3425,
3951,
4097,
4571,
```

```
4599,
5048,
434,
652,
950,
1191,
3713,
4001,
4707,
188,
681,
871,
939,
1223,
2731,
3196,
3671,
4861,
359,
363,
561,
810,
2917,
3284,
3360,
3451,
4026,
4356,
4385,
4553,
4942,
163,
164,
297,
315,
360,
659,
674,
746,
828,
946,
1056,
1325,
1511,
3404,
3876,
4215,
4437,
```

```
292,
676,
734,
889,
948,
1487,
2440,
3081,
3998,
4022,
4049,
4759,
198,
484,
635,
637,
891,
955,
959,
1071,
1303,
2935,
3067,
3244,
3434,
3641,
3889,
4061,
4230,
4409,
4648,
92,
153,
200,
246,
335,
351,
563,
885,
972,
977,
1128,
1337,
1360,
1437,
2279,
3300,
3564,
```

```
4420,
4632,
5066,
5139,
381,
533,
995,
1705,
2931,
3703,
4127,
4149,
4194,
4795,
4947,
4965,
4987,
5058,
51,
344,
370,
757,
758,
778,
840,
890,
1132,
1150,
1271,
1807,
1984,
2235,
2255,
2376,
3243,
3344,
3400,
3542,
3554,
3885,
4110,
4389,
4417,
4658,
4809,
4969,
102,
144,
```

```
326,
442,
501,
953,
958,
1329,
1405,
1499,
1549,
1574,
1685,
2728,
2899,
2962,
3015,
3046,
3341,
3523,
3627,
4510,
4914,
233,
308,
460,
975,
978,
1004,
1023,
1169,
1183,
1300,
2303,
2506,
2941,
3069,
3446,
4045,
4068,
4234,
4363,
4384,
4426,
4652,
4801,
4806,
4921,
4968,
171,
```

```
414,
631,
680,
702,
720,
855,
1170,
1338,
1340,
1673,
1715,
2164,
2416,
2644,
2898,
3107,
3201,
3499,
3715,
3972,
3975,
4087,
4465,
4564,
4815,
5110,
59,
75,
166,
664,
724,
744,
850,
880,
884,
954,
1019,
1057,
1115,
1118,
1129,
1135,
1219,
1238,
1385,
1433,
1455,
1509,
```

```
 1582,
 1608,
 1698,
 1915,
 2342,
 ...]
```

```
In [21]: rec_article_ids_test = ['1430.0','1429.0']
         rec_article_ids_test
```

```
Out[21]: ['1430.0', '1429.0']
```

```
In [18]: df.head()
```

```
Out[18]:    article_id                                              title  user_id
         0      1430.0  using pixiedust for fast, flexible, and easier...        1
         1      1314.0       healthcare python streaming application demo        2
         2      1429.0         use deep learning for image classification        3
         3      1338.0          ml optimization using cognitive assistant        4
         4      1276.0          deploy your python model as a restful api        5
```

```
In [42]: rec_article_ids_test = np.array(rec_article_ids_test, dtype=np.float64)
         count_interactions = df.set_index('article_id').loc[rec_article_ids_test].reset_index(i
         count_interactions.head()
```

```
Out[42]:    article_id                                              title  user_id
         0      1430.0  using pixiedust for fast, flexible, and easier...        1
         1      1430.0  using pixiedust for fast, flexible, and easier...       15
         2      1430.0  using pixiedust for fast, flexible, and easier...       33
         3      1430.0  using pixiedust for fast, flexible, and easier...       41
         4      1430.0  using pixiedust for fast, flexible, and easier...       21
```

```
In [43]: count_interactions = count_interactions.groupby('article_id').size().sort_values(ascend
         count_interactions
```

```
Out[43]: article_id
         1429.0    937
         1430.0    336
         dtype: int64
```

```
In [46]: list_articles_sorted = count_interactions.index
         list_articles_sorted = [str(x) for x in list_articles_sorted]
         list_articles_sorted
```

```
Out[46]: ['1429.0', '1430.0']
```

```
In [48]: # The function to sort the articles descending by the number of interactions
         def get_sorted_articles(article_ids, df=df):
             '''
             INPUT:
```

```
        article_ids - list of articles
        df - (pandas dataframe) df as defined at the top of the notebook

        OUTPUT:
        articles_sorted - list of articles that are sorted from the most interactions to th

        '''
        article_ids = np.array(article_ids, dtype=np.float64)
        count_interactions = df.set_index('article_id').loc[article_ids].reset_index(inplac
        count_interactions = count_interactions.groupby('article_id').size().sort_values(as
        articles_sorted = count_interactions.index
        articles_sorted = [str(x) for x in articles_sorted]

        return articles_sorted
```

```
In [49]: # Test
         get_sorted_articles(rec_article_ids_test, df)
```

```
Out[49]: ['1429.0', '1430.0']
```

```
In [53]: m_recs = 10
         selected_user_id = 3
         closeness_list = get_top_sorted_users(selected_user_id, df, user_item)['user_id'].tolis
         r = 0
         recs = []
         read_article_ids, read_article_names = get_user_articles(selected_user_id)
         print(read_article_ids)
         for i in closeness_list:
             if i == selected_user_id:
                 continue
             rec_article_ids , rec_article_names = get_user_articles(i)
             rec_article_ids = get_sorted_articles(rec_article_ids, df)
             for j in rec_article_ids:
                 if (j not in read_article_ids):
                     r = r + 1
                     recs.append(j)
                     if r == m_recs:
                         break
             if r == m_recs:
                 break
         print(recs)
```

```
['12.0', '20.0', '29.0', '43.0', '50.0', '62.0', '109.0', '116.0', '120.0', '193.0', '213.0', '3
['1427.0', '1364.0', '1170.0', '1162.0', '1304.0', '1393.0', '1185.0', '1160.0', '1354.0', '1368
```

b. Fill in the functions

```
In [54]: def get_top_sorted_users(user_id, df=df, user_item=user_item):
             '''
```

```
    INPUT:
    user_id - (int)
    df - (pandas dataframe) df as defined at the top of the notebook
    user_item - (pandas dataframe) matrix of users by articles:
            1's when a user has interacted with an article, 0 otherwise


    OUTPUT:
    neighbors_df - (pandas dataframe) a dataframe with:
                    neighbor_id - is a neighbor user_id
                    similarity - measure of the similarity of each user to the provided
                    num_interactions - the number of articles viewed by the user - if a

    Other Details - sort the neighbors_df by the similarity and then by number of inter
                    highest of each is higher in the dataframe

    '''
    # Your code here
    user_article_i = user_item[user_item.index == user_id]
    neighbors_df = user_article_i.dot(user_item.transpose()).melt().sort_values(by = 'u
    neighbors_df.columns = ['user_id', 'similarity']
    neighbors_df['num_interactions'] = df.groupby(['user_id']).size().tolist()
    neighbors_df = neighbors_df.sort_values(by = ['similarity','num_interactions'], asc

    return neighbors_df # Return the dataframe specified in the doc_string


def user_user_recs_part2(user_id, m=10):
    '''
    INPUT:
    user_id - (int) a user id
    m - (int) the number of recommendations you want for the user

    OUTPUT:
    recs - (list) a list of recommendations for the user by article id
    rec_names - (list) a list of recommendations for the user by article title

    Description:
    Loops through the users based on closeness to the input user_id
    For each user - finds articles the user hasn't seen before and provides them as rec
    Does this until m recommendations are found

    Notes:
    * Choose the users that have the most total article interactions
    before choosing those with fewer article interactions.

    * Choose articles with the articles with the most total interactions
    before choosing those with fewer total interactions.
```

```
    '''
    # Your code here
    closeness_list = get_top_sorted_users(user_id, df, user_item)['user_id'].tolist()
    r = 0
    recs = []
    read_article_ids, read_article_names = get_user_articles(user_id)

    for i in closeness_list:
        if i == user_id:
            continue
        rec_article_ids , rec_article_names = get_user_articles(i)
        rec_article_ids = get_sorted_articles(rec_article_ids, df)
        for j in rec_article_ids:
            if (j not in read_article_ids):
                r = r + 1
                recs.append(j)
                if r == m:
                    break
        if r == m:
            break

    rec_names = get_article_names(recs, df=df)
    return recs, rec_names
```

In [9]:
```
# Test
get_top_sorted_users(3, df, user_item)
```

Out[9]:

	user_id	similarity	num_interactions
2	3	40.0	82
3352	3353	40.0	80
22	23	23.0	364
3781	3782	23.0	363
97	98	17.0	170
3763	3764	17.0	169
202	203	16.0	160
4458	4459	16.0	158
48	49	16.0	147
3696	3697	16.0	145
51	52	15.0	132
3595	3596	15.0	131
911	912	14.0	102
3539	3540	14.0	101
241	242	13.0	148
3909	3910	13.0	147
130	131	13.0	145
3869	3870	13.0	144
203	204	13.0	97

```
5137      5138         13.0                 95
2925      2926         13.0                 83
39          40         13.0                 78
4931      4932         13.0                 76
10          11         13.0                 35
3965      3966         13.0                 33
20          21         12.0                137
4784      4785         12.0                136
618        619         12.0                 84
194        195         12.0                 72
3577      3578         12.0                 70
...        ...          ...                ...
5055      5056          0.0                  1
5059      5060          0.0                  1
5064      5065          0.0                  1
5067      5068          0.0                  1
5070      5071          0.0                  1
5072      5073          0.0                  1
5075      5076          0.0                  1
5083      5084          0.0                  1
5084      5085          0.0                  1
5086      5087          0.0                  1
5090      5091          0.0                  1
5091      5092          0.0                  1
5097      5098          0.0                  1
5099      5100          0.0                  1
5100      5101          0.0                  1
5103      5104          0.0                  1
5106      5107          0.0                  1
5112      5113          0.0                  1
5115      5116          0.0                  1
5118      5119          0.0                  1
5119      5120          0.0                  1
5120      5121          0.0                  1
5121      5122          0.0                  1
5124      5125          0.0                  1
5130      5131          0.0                  1
5140      5141          0.0                  1
5143      5144          0.0                  1
5146      5147          0.0                  1
5147      5148          0.0                  1
5148      5149          0.0                  1

[5149 rows x 3 columns]
```

In [55]:
```
# Quick spot check - don't change this code - just use it to test your functions
rec_ids, rec_names = user_user_recs_part2(20, 10)
print("The top 10 recommendations for user 20 are the following article ids:")
```

```
print(rec_ids)
print()
print("The top 10 recommendations for user 20 are the following article names:")
print(rec_names)
```

```
The top 10 recommendations for user 20 are the following article ids:
['1330.0', '1427.0', '1364.0', '1170.0', '1162.0', '1304.0', '1351.0', '1160.0', '1354.0', '1368

The top 10 recommendations for user 20 are the following article names:
['insights from new york car accident reports', 'use xgboost, scikit-learn & ibm watson machine
```

5. Use your functions from above to correctly fill in the solutions to the dictionary below. Then test your dictionary against the solution. Provide the code you need to answer each following the comments below.

In [70]:
```
get_top_sorted_users(1, df, user_item).head(2)
```

Out[70]:

	user_id	similarity	num_interactions
0	1	36.0	47
3932	3933	35.0	45

In [67]:
```
get_top_sorted_users(131, df, user_item).head(11)
```

Out[67]:

	user_id	similarity	num_interactions
130	131	75.0	145
3869	3870	74.0	144
3781	3782	39.0	363
22	23	38.0	364
202	203	33.0	160
4458	4459	33.0	158
97	98	29.0	170
3763	3764	29.0	169
48	49	29.0	147
3696	3697	29.0	145
241	242	25.0	148

In [68]:
```
### Tests with a dictionary of results

user1_most_sim = 3933 # Find the user that is most similar to user 1
user131_10th_sim = 242 # Find the 10th most similar user to user 131
```

In [69]:
```
## Dictionary Test Here
sol_5_dict = {
    'The user that is most similar to user 1.': user1_most_sim,
    'The user that is the 10th most similar to user 131': user131_10th_sim
}

t.sol_5_test(sol_5_dict)
```

```
This all looks good!  Nice job!
```

6. If we were given a new user, which of the above functions would you be able to use to make recommendations? Explain. Can you think of a better way we might make recommendations? Use the cell below to explain a better method for new users.

Provide your response here.

7. Using your existing functions, provide the top 10 recommended articles you would provide for the a new user below. You can test your function against our thoughts to make sure we are all on the same page with how we might make a recommendation.

```
In [77]: new_user = 0
         user_user_recs_part2(new_user, 10)

Out[77]: ([], [])
```

The above result is ok because with a new user we cannot calculate the similarity of existing users to the new user.

```
In [79]: get_top_sorted_users(new_user, df, user_item).head()

Out[79]:       user_id  similarity  num_interactions
         22        NaN         NaN               364
         3781      NaN         NaN               363
         97        NaN         NaN               170
         3763      NaN         NaN               169
         202       NaN         NaN               160
```

Thus we should recommend the new user just the articles with the highest interactions.

```
In [90]: articles_new_user = df.groupby('article_id').size().sort_values(ascending = False).head
         articles_new_user

Out[90]: article_id
         1429.0    937
         1330.0    927
         1431.0    671
         1427.0    643
         1364.0    627
         1314.0    614
         1293.0    572
         1170.0    565
         1162.0    512
         1304.0    483
         dtype: int64

In [93]: articles_new_user = article_new_user.index
         articles_new_user = [str(x) for x in articles_new_user]
         articles_new_user
```

Out[93]: ['1429.0',
 '1330.0',
 '1431.0',
 '1427.0',
 '1364.0',
 '1314.0',
 '1293.0',
 '1170.0',
 '1162.0',
 '1304.0']

Or an existing function in Part II can be used to get this article ids list

```
In [98]: get_top_article_ids(10, df)
```

Out[98]: ['1429.0',
 '1330.0',
 '1431.0',
 '1427.0',
 '1364.0',
 '1314.0',
 '1293.0',
 '1170.0',
 '1162.0',
 '1304.0']

Fill in the result

```
In [94]: new_user = '0.0'

         # What would your recommendations be for this new user '0.0'?  As a new user, they have
         # Provide a list of the top 10 article ids you would give to
         new_user_recs = ['1429.0',
          '1330.0',
          '1431.0',
          '1427.0',
          '1364.0',
          '1314.0',
          '1293.0',
          '1170.0',
          '1162.0',
          '1304.0'] # Your recommendations here
```

```
In [95]: assert set(new_user_recs) == set(['1314.0','1429.0','1293.0','1427.0','1162.0','1364.0'

         print("That's right!  Nice job!")
```

That's right! Nice job!

1.1.4 Part IV: Content Based Recommendations (EXTRA - NOT REQUIRED)

Another method we might use to make recommendations is to perform a ranking of the highest ranked articles associated with some term. You might consider content to be the **doc_body**, **doc_description**, or **doc_full_name**. There isn't one way to create a content based recommendation, especially considering that each of these columns hold content related information.

`1.` Use the function body below to create a content based recommender. Since there isn't one right answer for this recommendation tactic, no test functions are provided. Feel free to change the function inputs if you decide you want to try a method that requires more input values. The input values are currently set with one idea in mind that you may use to make content based recommendations. One additional idea is that you might want to choose the most popular recommendations that meet your 'content criteria', but again, there is a lot of flexibility in how you might make these recommendations.

1.1.5 This part is NOT REQUIRED to pass this project. However, you may choose to take this on as an extra way to show off your skills.

```
In [ ]: def make_content_recs():
            '''
            INPUT:

            OUTPUT:

            '''
```

`2.` Now that you have put together your content-based recommendation system, use the cell below to write a summary explaining how your content based recommender works. Do you see any possible improvements that could be made to your function? Is there anything novel about your content based recommender?

1.1.6 This part is NOT REQUIRED to pass this project. However, you may choose to take this on as an extra way to show off your skills.

Write an explanation of your content based recommendation system here.

`3.` Use your content-recommendation system to make recommendations for the below scenarios based on the comments. Again no tests are provided here, because there isn't one right answer that could be used to find these content based recommendations.

1.1.7 This part is NOT REQUIRED to pass this project. However, you may choose to take this on as an extra way to show off your skills.

```
In [ ]: # make recommendations for a brand new user


        # make a recommendations for a user who only has interacted with article id '1427.0'
```

1.1.8 Part V: Matrix Factorization

In this part of the notebook, you will build use matrix factorization to make article recommendations to the users on the IBM Watson Studio platform.

1. You should have already created a **user_item** matrix above in **question 1** of **Part III** above. This first question here will just require that you run the cells to get things set up for the rest of **Part V** of the notebook.

```
In [ ]: # Load the matrix here
        user_item_matrix = pd.read_pickle('user_item_matrix.p')
```

```
In [ ]: # quick look at the matrix
        user_item_matrix.head()
```

2. In this situation, you can use Singular Value Decomposition from numpy on the user-item matrix. Use the cell to perform SVD, and explain why this is different than in the lesson.

```
In [ ]: # Perform SVD on the User-Item Matrix Here

        u, s, vt = # use the built in to get the three matrices
```

Provide your response here.

3. Now for the tricky part, how do we choose the number of latent features to use? Running the below cell, you can see that as the number of latent features increases, we obtain a lower error rate on making predictions for the 1 and 0 values in the user-item matrix. Run the cell below to get an idea of how the accuracy improves as we increase the number of latent features.

```
In [ ]: num_latent_feats = np.arange(10,700+10,20)
        sum_errs = []

        for k in num_latent_feats:
            # restructure with k latent features
            s_new, u_new, vt_new = np.diag(s[:k]), u[:, :k], vt[:k, :]

            # take dot product
            user_item_est = np.around(np.dot(np.dot(u_new, s_new), vt_new))

            # compute error for each prediction to actual value
            diffs = np.subtract(user_item_matrix, user_item_est)

            # total errors and keep track of them
            err = np.sum(np.sum(np.abs(diffs)))
            sum_errs.append(err)


        plt.plot(num_latent_feats, 1 - np.array(sum_errs)/df.shape[0]);
        plt.xlabel('Number of Latent Features');
        plt.ylabel('Accuracy');
        plt.title('Accuracy vs. Number of Latent Features');
```

4. From the above, we can't really be sure how many features to use, because simply having a better way to predict the 1's and 0's of the matrix doesn't exactly give us an indication of if we are

able to make good recommendations. Instead, we might split our dataset into a training and test set of data, as shown in the cell below.

Use the code from question 3 to understand the impact on accuracy of the training and test sets of data with different numbers of latent features. Using the split below:

- How many users can we make predictions for in the test set?
- How many users are we not able to make predictions for because of the cold start problem?
- How many articles can we make predictions for in the test set?
- How many articles are we not able to make predictions for because of the cold start problem?

In [ ]:
```
df_train = df.head(40000)
df_test = df.tail(5993)

def create_test_and_train_user_item(df_train, df_test):
    '''
    INPUT:
    df_train - training dataframe
    df_test - test dataframe

    OUTPUT:
    user_item_train - a user-item matrix of the training dataframe
                      (unique users for each row and unique articles for each column)
    user_item_test - a user-item matrix of the testing dataframe
                    (unique users for each row and unique articles for each column)
    test_idx - all of the test user ids
    test_arts - all of the test article ids

    '''
    # Your code here

    return user_item_train, user_item_test, test_idx, test_arts

user_item_train, user_item_test, test_idx, test_arts = create_test_and_train_user_item(d
```

In [ ]:
```
# Replace the values in the dictionary below
a = 662
b = 574
c = 20
d = 0


sol_4_dict = {
    'How many users can we make predictions for in the test set?': # letter here,
    'How many users in the test set are we not able to make predictions for because of t
    'How many articles can we make predictions for in the test set?': # letter here,
    'How many articles in the test set are we not able to make predictions for because o
```

```
}

t.sol_4_test(sol_4_dict)
```

5. Now use the **user_item_train** dataset from above to find U, S, and V transpose using SVD. Then find the subset of rows in the **user_item_test** dataset that you can predict using this matrix decomposition with different numbers of latent features to see how many features makes sense to keep based on the accuracy on the test data. This will require combining what was done in questions 2 - 4.

Use the cells below to explore how well SVD works towards making predictions for recommendations on the test data.

```
In [ ]: # fit SVD on the user_item_train matrix
        u_train, s_train, vt_train = # fit svd similar to above then use the cells below
```

```
In [ ]: # Use these cells to see how well you can use the training
        # decomposition to predict on test data
```

```
In [ ]:
```

```
In [ ]:
```

6. Use the cell below to comment on the results you found in the previous question. Given the circumstances of your results, discuss what you might do to determine if the recommendations you make with any of the above recommendation systems are an improvement to how users currently find articles?

Your response here.

Extras Using your workbook, you could now save your recommendations for each user, develop a class to make new predictions and update your results, and make a flask app to deploy your results. These tasks are beyond what is required for this project. However, from what you learned in the lessons, you certainly capable of taking these tasks on to improve upon your work here!

1.2 Conclusion

Congratulations! You have reached the end of the Recommendations with IBM project!

Tip: Once you are satisfied with your work here, check over your report to make sure that it is satisfies all the areas of the rubric. You should also probably remove all of the "Tips" like this one so that the presentation is as polished as possible.

1.3 Directions to Submit

Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).

Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.

Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

In [ ]:
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Recommendations_with_IBM.ipynb'])
```