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

# Reading ratings file
# Ignore the timestamp column
ratings = pd.read_csv('ratings.csv', sep='\t', encoding='latin-1', usecols=['user_id', 'movie_id', 'rating'])

# Reading users file
users = pd.read_csv('users.csv', sep='\t', encoding='latin-1', usecols=['user_id', 'gender', 'zipcode', 'age_desc', 'occ_desc'])

# Reading movies file
movies = pd.read_csv('movies.csv', sep='\t', encoding='latin-1', usecols=['movie_id', 'title', 'genres'])
```

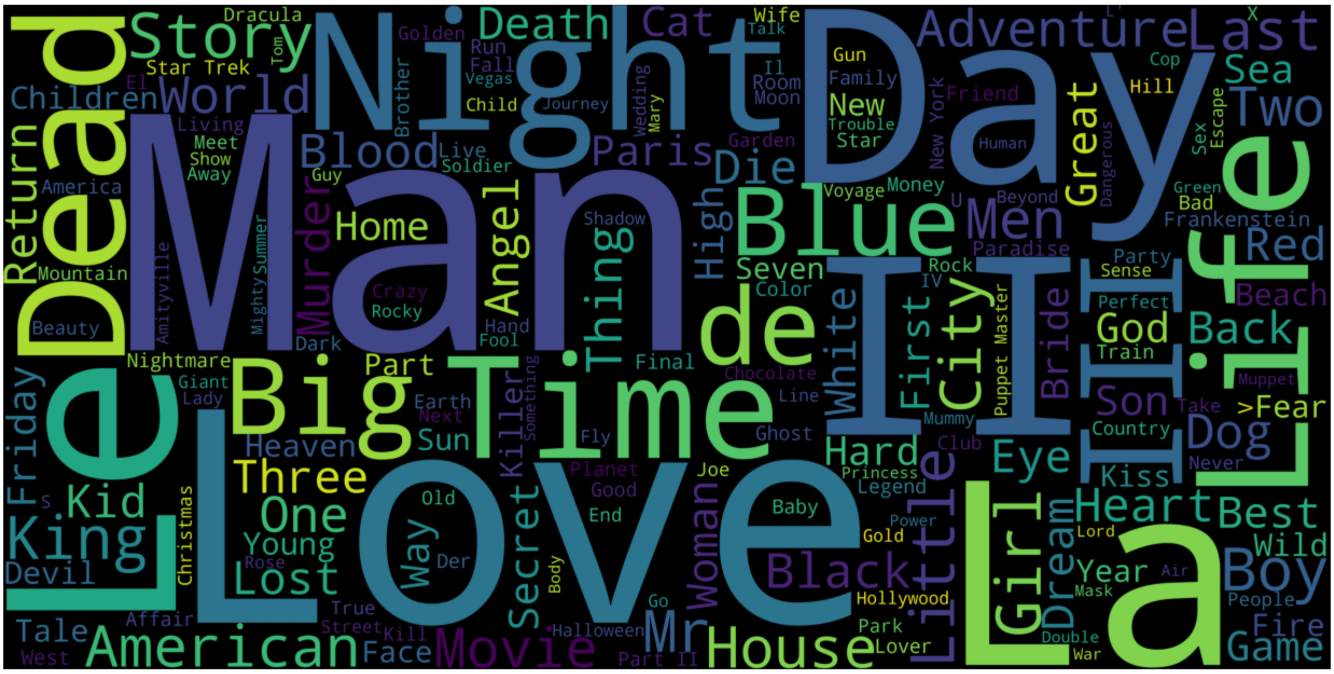
```
# Check the top 5 rows
print(users.head())

# Check the file info
print(users.info())
```

```
   user_id gender zipcode  age_desc              occ_desc
0        1      F   48067  Under 18          K-12 student
1        2      M   70072       56+         self-employed
2        3      M   55117     25-34             scientist
3        4      M   02460     45-49  executive/managerial
4        5      M   55455     25-34                writer
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6040 entries, 0 to 6039
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   user_id   6040 non-null   int64
 1   gender    6040 non-null   object
 2   zipcode   6040 non-null   object
 3   age_desc  6040 non-null   object
 4   occ_desc  6040 non-null   object
dtypes: int64(1), object(4)
memory usage: 236.1+ KB
None
```

```
# Check the top 5 rows
print(movies.head())

# Check the file info
print(movies.info())
```

```
   movie_id                               title                        genres
0         1                    Toy Story (1995)   Animation|Children's|Comedy
1         2                      Jumanji (1995)  Adventure|Children's|Fantasy
2         3             Grumpier Old Men (1995)                Comedy|Romance
3         4            Waiting to Exhale (1995)                  Comedy|Drama
4         5  Father of the Bride Part II (1995)                        Comedy
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3883 entries, 0 to 3882
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   movie_id  3883 non-null   int64
 1   title     3883 non-null   object
 2   genres    3883 non-null   object
dtypes: int64(1), object(2)
memory usage: 91.1+ KB
None
```

```
# Data Exploration
%matplotlib inline
import wordcloud
from wordcloud import WordCloud, STOPWORDS

# Create a wordcloud of the movie titles
movies['title'] = movies['title'].fillna("").astype('str')
title_corpus = ' '.join(movies['title'])
title_wordcloud = WordCloud(stopwords=STOPWORDS, background_color='black', height=2000, width=4000).generate(title_corpus)

# Plot the wordcloud
plt.figure(figsize=(16,8))
plt.imshow(title_wordcloud)
plt.axis('off')
plt.show()
```




```
# Get summary statistics of rating
ratings['rating'].describe()
```

```
count    611137.000000
mean          3.574809
std           1.120023
min           1.000000
25%           3.000000
50%           4.000000
75%           4.000000
max           5.000000
Name: rating, dtype: float64
```

```
# Import seaborn library
import seaborn as sns
sns.set_style('whitegrid')
sns.set(font_scale=1.5)
%matplotlib inline

# Display distribution of rating
sns.distplot(ratings['rating'].fillna(ratings['rating'].median()))
```

```
<ipython-input-5-e3c9e7783721>:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
# Join all 3 files into one dataframe
dataset = pd.merge(pd.merge(movies, ratings),users)
# Display 20 movies with highest ratings
dataset[['title','genres','rating']].sort_values('rating', ascending=False).head(20)
```

| | title | genres | rating |
|---|---|---|---|
| 0 | Toy Story (1995) | Animation\|Children's\|Comedy | 5 |
| 476171 | Exorcist, The (1973) | Horror | 5 |
| 513939 | Modern Times (1936) | Comedy | 5 |
| 104651 | This Is Spinal Tap (1984) | Comedy\|Drama\|Musical | 5 |
| 444376 | Big Night (1996) | Drama | 5 |
| 104653 | Being There (1979) | Comedy | 5 |
| 260498 | Schindler's List (1993) | Drama\|War | 5 |
| 322106 | Trainspotting (1996) | Drama | 5 |
| 557585 | Taxi Driver (1976) | Drama\|Thriller | 5 |
| 104657 | Field of Dreams (1989) | Drama | 5 |
| 187136 | Garden of Finzi-Contini, The (Giardino dei Fin... | Drama | 5 |
| 104660 | Birds, The (1963) | Horror | 5 |
| 104661 | Cape Fear (1962) | Film-Noir\|Thriller | 5 |
| 187135 | Paradise Lost: The Child Murders at Robin Hood... | Documentary | 5 |
| 104663 | Omen, The (1976) | Horror | 5 |
| 187134 | Sling Blade (1996) | Drama\|Thriller | 5 |
| 557584 | Braveheart (1995) | Action\|Drama\|War | 5 |
| 513938 | Double Indemnity (1944) | Crime\|Film-Noir | 5 |
| 104648 | Heathers (1989) | Comedy | 5 |
| 104647 | Big Sleep, The (1946) | Film-Noir\|Mystery | 5 |

```
# Make a census of the genre keywords
genre_labels = set()
for s in movies['genres'].str.split('|').values:
    genre_labels = genre_labels.union(set(s))

# Function that counts the number of times each of the genre keywords appear
def count_word(dataset, ref_col, census):
    keyword_count = dict()
    for s in census:
        keyword_count[s] = 0
    for census_keywords in dataset[ref_col].str.split('|'):
        if type(census_keywords) == float and pd.isnull(census_keywords):
            continue
        for s in [s for s in census_keywords if s in census]:
            if pd.notnull(s):
                keyword_count[s] += 1
    #______________________________________________________________________
    # convert the dictionary in a list to sort the keywords by frequency
    keyword_occurences = []
    for k,v in keyword_count.items():
        keyword_occurences.append([k,v])
    keyword_occurences.sort(key = lambda x:x[1], reverse = True)
    return keyword_occurences, keyword_count

# Calling this function gives access to a list of genre keywords which are sorted by decreasing frequency
keyword_occurences, dum = count_word(movies, 'genres', genre_labels)
keyword_occurences[:5]
```

```
[['Drama', 1603],
 ['Comedy', 1200],
 ['Action', 503],
```

```
 ['Thriller', 492],
 ['Romance', 471]]
```

```
#ContentBased
# Break up the big genre string into a string array
movies['genres'] = movies['genres'].str.split('|')
# Convert genres to string value
movies['genres'] = movies['genres'].fillna("").astype('str')
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0.0, stop_words='english')
tfidf_matrix = tf.fit_transform(movies['genres'])
tfidf_matrix.shape
```

```
(3883, 127)
```

```
from sklearn.metrics.pairwise import linear_kernel
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
cosine_sim[:4, :4]
```

```
array([[1.        , 0.14193614, 0.09010857, 0.1056164 ],
       [0.14193614, 1.        , 0.        , 0.        ],
       [0.09010857, 0.        , 1.        , 0.1719888 ],
       [0.1056164 , 0.        , 0.1719888 , 1.        ]])
```

```
# Create two user-item matrices, one for training and another for testing
train_data_matrix = train_data[['user_id', 'movie_id', 'rating']].values
test_data_matrix = test_data[['user_id', 'movie_id', 'rating']].values

# Check their shape
print(train_data_matrix.shape)
print(test_data_matrix.shape)
```

```
# Build a 1-dimensional array with movie titles
titles = movies['title']
indices = pd.Series(movies.index, index=movies['title'])

# Function that get movie recommendations based on the cosine similarity score of movie genres
def genre_recommendations(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:21]
    movie_indices = [i[0] for i in sim_scores]
    return titles.iloc[movie_indices]
```

```
genre_recommendations('Good Will Hunting (1997)').head(20)
```

```
25                                       Othello (1995)
26                                  Now and Then (1995)
29     Shanghai Triad (Yao a yao yao dao waipo qiao) ...
30                              Dangerous Minds (1995)
35                             Dead Man Walking (1995)
39                     Cry, the Beloved Country (1995)
42                                  Restoration (1995)
52                                     Lamerica (1994)
54                                      Georgia (1995)
56                         Home for the Holidays (1995)
61                             Mr. Holland's Opus (1995)
66                                     Two Bits (1995)
77                           Crossing Guard, The (1995)
79          White Balloon, The (Badkonake Sefid ) (1995)
81                        Antonia's Line (Antonia) (1995)
82        Once Upon a Time... When We Were Colored (1995)
89                      Journey of August King, The (1995)
92                              Beautiful Girls (1996)
95                              Hate (Haine, La) (1995)
112                            Margaret's Museum (1995)
Name: title, dtype: object
```

```
genre_recommendations('Toy Story (1995)').head(20)
```

```
1050              Aladdin and the King of Thieves (1996)
2072                            American Tail, An (1986)
2073          American Tail: Fievel Goes West, An (1991)
2285                           Rugrats Movie, The (1998)
2286                                Bug's Life, A (1998)
3045                                  Toy Story 2 (1999)
```

```
3542                                Saludos Amigos (1943)
3682                                   Chicken Run (2000)
3685        Adventures of Rocky and Bullwinkle, The (2000)
236                                 Goofy Movie, A (1995)
12                                           Balto (1995)
241                               Gumby: The Movie (1995)
310                             Swan Princess, The (1994)
592                                      Pinocchio (1940)
612                                Aristocats, The (1970)
700                              Oliver & Company (1988)
876      Land Before Time III: The Time of the Great Gi...
1010          Winnie the Pooh and the Blustery Day (1968)
1012                         Sword in the Stone, The (1963)
1020                        Fox and the Hound, The (1981)
Name: title, dtype: object
```

```
genre_recommendations('Saving Private Ryan (1998)').head(20)
```

```
461            Heaven & Earth (1993)
1204        Full Metal Jacket (1987)
1214     Boat, The (Das Boot) (1981)
1222                    Glory (1989)
1545                G.I. Jane (1997)
1959      Saving Private Ryan (1998)
2358       Thin Red Line, The (1998)
2993        Longest Day, The (1962)
3559            Flying Tigers (1942)
3574    Fighting Seabees, The (1944)
3585    Guns of Navarone, The (1961)
3684             Patriot, The (2000)
40                Richard III (1995)
153            Beyond Rangoon (1995)
332         Walking Dead, The (1995)
523          Schindler's List (1993)
641        Courage Under Fire (1996)
967          Nothing Personal (1995)
979           Michael Collins (1996)
1074                  Platoon (1986)
Name: title, dtype: object
```