Name: Thomas Lin

Andrew: ttlin

Assignment 5 - Recommender System

1. A Recommender System Trial

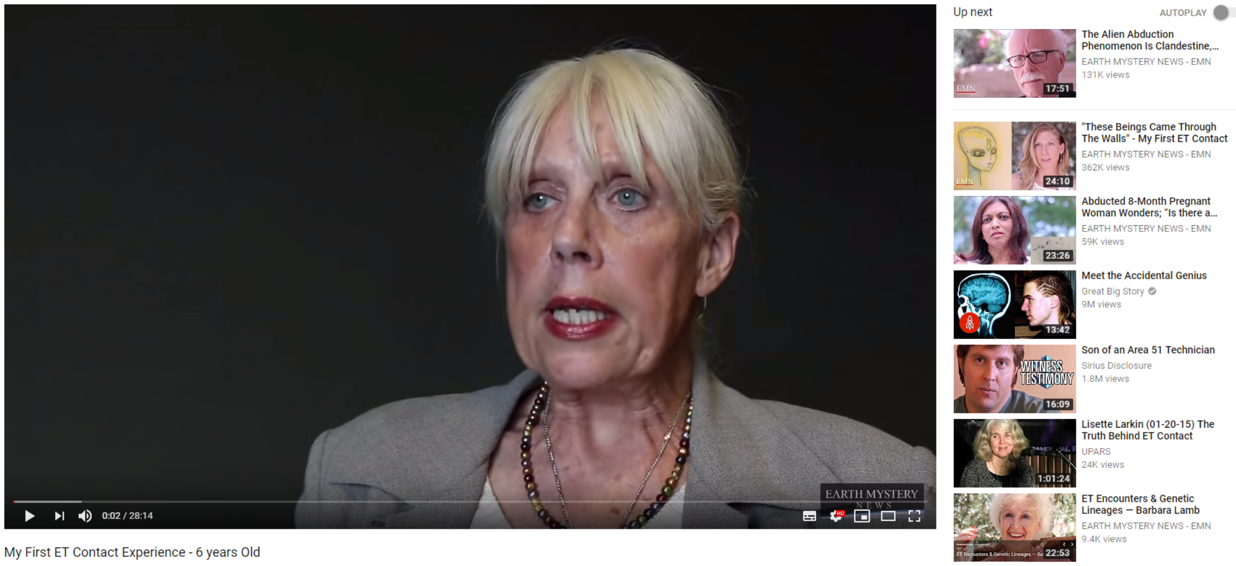
I decided to go down the rabbithole of Aliens Abductions, whether they are real or not. I started with "An Alien Abducted My Family and I Can Prove It! | This Morning", and followed the top recommended list 10 times.

For each video you choose, record a screenshot of the video preview in your document.

1. An Alien Abducted My Family and I Can Prove It! | This Morning

2. The Woman Who Claims She Was Abducted By Aliens | Studio 10

3. My First ET Contact Experience - 6 Years Old



My First ET Contact Experience - 6 years Old

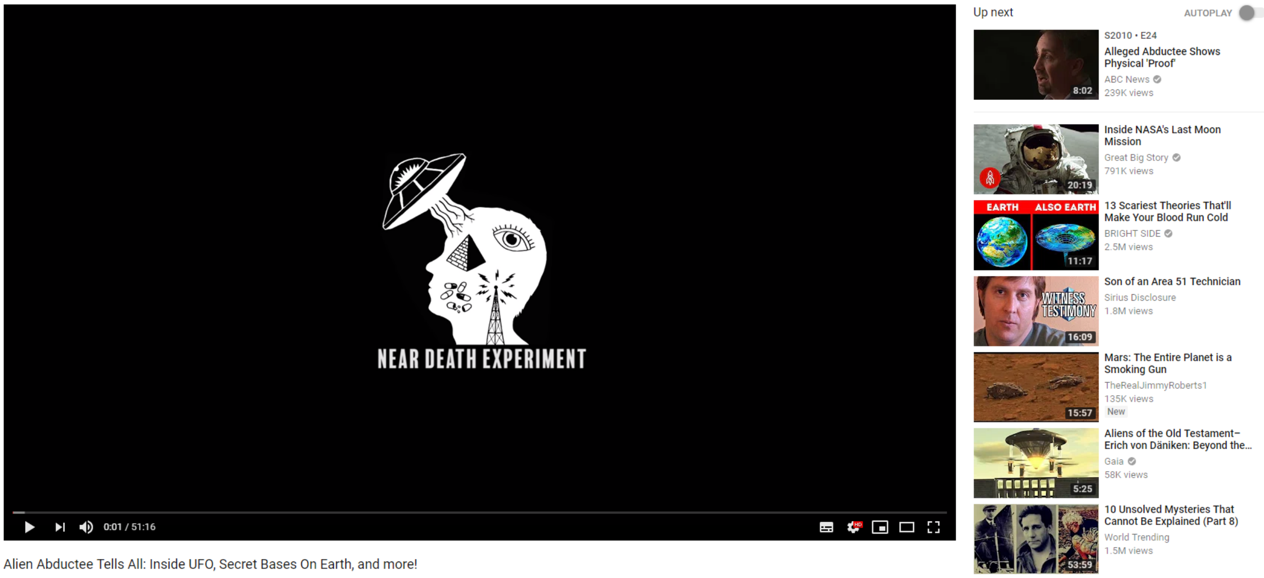
4. The Alien Abduction Phenomenon Is Clandestine, Davidi Jacobs Speaks Out

The Alien Abduction Phenomenon Is Clandestine, David Jacobs Speaks Out

5. Dr. David Jacobs - The Alien Agenda on Ron James' Bigger Questions

Dr. David Jacobs - The Alien Agenda on Ron James' Bigger Questions.

6. Alien Abductee Tells All: Inside UFO, Secret Bases On Earth, and more!



7. Alleged Abductee Shows Physical 'Proof'

8. Alien Abductee Tells All: Inside UFO, Secret Bases On Earth, and more!

9. Son of An Area 51 Technician

10. 5 Incredible Lost Photos Found On Missing Cameras

What do you notice about the videos? How does the topic change over time? Do you notice progressions toward more extreme views like those we read about?

I notice that on some level, the videos get progressively more extreme to an extent. From the first video, which is just proof, we go into more detail onto the insides of UFOs and the alien agendas. However, towards the end, one video is repeated and the videos get less extreme and less relevant. This may be because the library on alien abductions were not that popular, or that they did not believe the videos were worth recommending due to other user's dislike of them. In any case, for the most part, the videos seemed to get more extreme, but not the extent that the articles stated.

2. Creating Your Own Recommender System

So first, we add 10 new movies to our dataset:

In [20]:

```
import numpy as np
import pandas as pd

# Reading readers file
ratings = pd.read_csv('ratings.csv', sep=',', encoding='latin-1', usecols=['userId', 'movieId', 'rati
                                                                                                    
# Reading movies file
movies = pd.read_csv('movies.csv', sep=',', encoding='latin-1', usecols=['movieId', 'title', 'genres'
movies.head(10)
```

Out[20]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
5	6	Heat (1995)	Action|Crime|Thriller
6	7	Sabrina (1995)	Comedy|Romance
7	8	Tom and Huck (1995)	Adventure|Children
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller

In [21]:

```
#inserting 10 movies
new_movies = pd.DataFrame({
    'movieId': [131264, 131265, 131266, 131267, 131268, 131269, 131270, 131271, 131272, 131273],
    'title': ['Avengers: Infinity War(2018)', 'Mission Impossible: Fallout(2018)', 'Isle Of Dogs(2018
    'genres': ['Action|Adventure|Fantasy', 'Action|Adventure|Thriller', 'Animation|Adventure|Comedy',
})

movies = movies.append(new_movies, ignore_index = True)
movies.tail(10)
```

Out[21]:

	movieId	title	genres
27278	131264	Avengers: Infinity War(2018)	Action|Adventure|Fantasy
27279	131265	Mission Impossible: Fallout(2018)	Action|Adventure|Thriller
27280	131266	Isle Of Dogs(2018)	Animation|Adventure|Comedy
27281	131267	Logan(2017)	Action|Drama|Sci-Fi
27282	131268	Ready Player One(2018)	Action|Adventure|Sci-Fi
27283	131269	Love, Simon(2018)	Comedy|Drama|Romance
27284	131270	Game Night (2018)	Action|Comedy|Crime
27285	131271	The Incredibles 2(2018)	Animation|Action|Adventure
27286	131272	Coco (2017)	Animation|Adventure|Comedy
27287	131273	It (2017)	Horror|Thriller

In [22]:

```
#inserting 20 ratings
#my user id is 138494

new_ratings = pd.DataFrame({
    'userId': [138494,138494,138494,138494,138494,138494,138494,138494,138494,138494,138494,138494,13
    'movieId': [131264,131265,131266,131267,131268,131269,131270,131271,131272,131273,8961,356,2959,2
    'rating': [4,2,2.5,5,3.5,3.5,3,4.5,4,4,4,5,4,3.5,5,3.5,3,4.5,5,2]
})

ratings = ratings.drop(['timestamp'],axis= 1)
ratings = ratings.append(new_ratings, ignore_index = True)

ratings.tail(20)
```

Out[22]:

	userId	movieId	rating
20000263	138494	131264	4.0
20000264	138494	131265	2.0
20000265	138494	131266	2.5
20000266	138494	131267	5.0
20000267	138494	131268	3.5
20000268	138494	131269	3.5
20000269	138494	131270	3.0
20000270	138494	131271	4.5
20000271	138494	131272	4.0
20000272	138494	131273	4.0
20000273	138494	8961	4.0
20000274	138494	356	5.0
20000275	138494	2959	4.0
20000276	138494	2918	3.5
20000277	138494	48385	5.0
20000278	138494	1704	3.5
20000279	138494	2	3.0
20000280	138494	1907	4.5
20000281	138494	26999	5.0
20000282	138494	924	2.0

In [23]:

```python
#we then combine all of the genres, movies, and ratings
#full_movies = pd.concat([movies,genre_list], axis = 1)
dataset = pd.merge(movies, ratings)
dataset = dataset.dropna()

#find the mean of the ratings for each movie
ratings_mean = (dataset.groupby('title'))['title','rating'].mean()
ratings_mean['title'] = ratings_mean.index
ratings_mean = pd.DataFrame({'title': ratings_mean['title'], 'mean rating': ratings_mean['rating']})

#find the total number of ratings for each movie
ratings_total = dataset.groupby('title').size()
ratings_total = pd.DataFrame({'title':ratings_total.index,'total ratings': ratings_total.values})

#combine them all to create a dataset
dataset_clean = pd.merge(ratings_mean, pd.merge(movies,ratings_total)).sort_values(by = 'total rating
dataset_clean
```

```
C:\Users\thoma\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2963: FutureWarning: 'titl
e' is both an index level and a column label.
Defaulting to column, but this will raise an ambiguity error in a future version
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[23]:

	title	mean rating	movieId	genres	total ratings
18848	Pulp Fiction (1994)	4.174231	296	Comedy|Crime|Drama|Thriller	67310
8728	Forrest Gump (1994)	4.029015	356	Comedy|Drama|Romance|War	66173
20804	Shawshank Redemption, The (1994)	4.446990	318	Crime|Drama	63366
21019	Silence of the Lambs, The (1991)	4.177057	593	Crime|Horror|Thriller	63299
12663	Jurassic Park (1993)	3.664741	480	Action|Adventure|Sci-Fi|Thriller	59715
21896	Star Wars: Episode IV - A New Hope (1977)	4.190672	260	Action|Adventure|Sci-Fi	54502
3512	Braveheart (1995)	4.042534	110	Action|Drama|War	53769
23008	Terminator 2: Judgment Day (1991)	3.931954	589	Action|Sci-Fi	52244
15206	Matrix, The (1999)	4.187186	2571	Action|Sci-Fi|Thriller	51334
20342	Schindler's List (1993)	4.310175	527	Drama|War	50054
24285	Toy Story (1995)	3.921240	1	Adventure|Animation|Children|Comedy|Fantasy	49695
9030	Fugitive, The (1993)	3.985690	457	Thriller	49581
1460	Apollo 13 (1995)	3.868598	150	Adventure|Drama|IMAX	47777
11861	Independence Day (a.k.a. ID4) (1996)	3.370962	780	Action|Adventure|Sci-Fi|Thriller	47048
24993	Usual Suspects, The (1995)	4.334372	50	Crime|Mystery|Thriller	47006
21898	Star Wars: Episode VI - Return of the Jedi (1983)	4.004622	1210	Action|Adventure|Sci-Fi	46839
2192	Batman (1989)	3.402365	592	Action|Crime|Thriller	46054
21897	Star Wars: Episode V - The Empire Strikes Back...	4.188202	1196	Action|Adventure|Sci-Fi	45313
1108	American Beauty (1999)	4.155934	2858	Comedy|Drama	44987
24612	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	3.898055	32	Mystery|Sci-Fi|Thriller	44980
5838	Dances with Wolves (1990)	3.728465	590	Adventure|Drama|Western	44208
19096	Raiders of the Lost Ark (Indiana Jones and the...	4.219009	1198	Action|Adventure	43295
8082	Fargo (1996)	4.112359	608	Comedy|Crime|Drama|Thriller	43272
20609	Seven (a.k.a. Se7en) (1995)	4.053493	47	Mystery|Thriller	43249
24526	True Lies (1994)	3.491149	380	Action|Adventure|Comedy|Romance|Thriller	43159
828	Aladdin (1992)	3.680829	114240	Adventure|Animation|Children|Comedy|Fantasy	41849
827	Aladdin (1992)	3.680829	588	Adventure|Animation|Children|Comedy|Musical	41849
21707	Speed (1994)	3.493203	377	Action|Romance|Thriller	41562
1934	Back to the Future (1985)	3.924142	1270	Adventure|Comedy|Sci-Fi	41426

	title	mean rating	movieId	genres	total ratings
9600	Godfather, The (1972)	4.364732	858	Crime|Drama	41355
...	...	...	...	...	...
21412	Soap Girl (2002)	3.500000	5905	Drama|Romance	1
21411	So Young (Zhi wo men zhong jiang shi qu de qin...	3.500000	106336	Drama	1
21406	So Much So Fast (2006)	3.500000	113432	Documentary	1
21402	So Evil, So Young (1961)	2.000000	112467	Drama	1
7574	Enola Gay and the Atomic Bombing of Japan (1995)	3.500000	100050	Documentary	1
7580	Enter Arsene Lupin (1944)	0.500000	117716	Crime|Drama	1
16028	Mr. Denning Drives North (1952)	3.500000	121735	Crime|Drama|Mystery	1
21438	Soldier of Fortune (1955)	2.500000	116547	Adventure|Crime|Drama|Romance|Thriller	1
7506	En rachÃ¢chant (1982)	2.000000	126024	(no genres listed)	1
21443	Soldier's Sweetheart, A (1998)	3.500000	48067	Drama|War	1
7461	Emergency Squad (1974)	3.000000	128379	Action|Thriller	1
21501	Something in the Wind (1947)	3.000000	51691	Comedy|Musical	1
7436	Ellen DeGeneres: The Beginning (2000)	4.000000	118892	Comedy	1
21496	Something Wild (1961)	3.000000	49769	Drama	1
21487	Somersault (1965)	4.000000	120777	Drama	1
7450	Elvis and Me (1988)	2.500000	121403	Drama|Romance	1
21484	Someone's Gaze (2013)	4.000000	121302	Animation	1
21480	Someone Like Him (Einer wie Bruno) (2011)	3.500000	107561	Comedy|Drama	1
7457	Embracing (1992)	4.000000	116959	Documentary	1
7478	Emotion (1966)	3.500000	106565	Comedy|Horror	1
21444	Soldier's Tale, A (1989)	3.000000	114369	Drama|Romance|War	1
7481	Emperor Waltz, The (1948)	3.000000	58282	Comedy|Musical|Romance	1
21458	Solstice (1994)	2.500000	79575	Drama	1
21457	Solomon Northup's Odyssey (1984)	3.000000	119878	(no genres listed)	1
7494	Empire of Silver (Bai yin di guo) (2009)	1.500000	111752	Drama|Romance	1
7500	Employees Leaving the LumiÃ¨re Factory (1895)	4.000000	120869	Documentary	1
21451	Solo (2013)	3.000000	107651	Mystery|Thriller	1
21450	Solitude of Prime Numbers, The (2010)	3.500000	101068	Drama	1
7505	En pÃ¥ miljonen (1995)	1.500000	126741	(no genres listed)	1
26754	è²□å□3D (2012)	1.500000	130640	Horror	1

26755 rows × 5 columns

In [24]:

```python
# This is a simple recommendation for the highest rated movies that are relatively popular
# It is useful for predicting

simp_pred = dataset_clean.drop(dataset_clean[dataset_clean["total ratings"] < 1000].index).sort_value

simp_pred
```

Out[24]:

	title	mean rating	movieId	genres	total ratings
20804	Shawshank Redemption, The (1994)	4.446990	318	Crime|Drama	63366
9600	Godfather, The (1972)	4.364732	858	Crime|Drama	41355
24993	Usual Suspects, The (1995)	4.334372	50	Crime|Mystery|Thriller	47006
20342	Schindler's List (1993)	4.310175	527	Drama|War	50054
9601	Godfather: Part II, The (1974)	4.275641	1221	Crime|Drama	27398
20625	Seven Samurai (Shichinin no samurai) (1954)	4.274180	2019	Action|Adventure|Drama	11611
19228	Rear Window (1954)	4.271334	904	Mystery|Thriller	17449
2076	Band of Brothers (2001)	4.263182	7502	Action|Drama|War	4305
4242	Casablanca (1942)	4.258327	912	Drama|Romance	24349
22445	Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	4.256935	922	Drama|Film-Noir|Romance	6525
17345	One Flew Over the Cuckoo's Nest (1975)	4.248079	1193	Drama	29932
6984	Dr. Strangelove or: How I Learned to Stop Worr...	4.247287	750	Comedy|War	23220
23760	Third Man, The (1949)	4.246002	1212	Film-Noir|Mystery|Thriller	6565
4854	City of God (Cidade de Deus) (2002)	4.235410	6016	Action|Adventure|Crime|Drama|Thriller	12937
14092	Lives of Others, The (Das leben der Anderen) (...	4.234790	44555	Drama|Romance|Thriller	5720
16979	North by Northwest (1959)	4.233538	908	Action|Adventure|Mystery|Romance|Thriller	15627
17904	Paths of Glory (1957)	4.232623	1178	Drama|War	3568
8262	Fight Club (1999)	4.227117	2959	Action|Crime|Drama|Thriller	40107
6907	Double Indemnity (1944)	4.224282	3435	Crime|Drama|Film-Noir	4909
79	12 Angry Men (1957)	4.224138	1203	Drama	12934
5923	Dark Knight, The (2008)	4.220129	58559	Action|Crime|Drama|IMAX	20438
19096	Raiders of the Lost Ark (Indiana Jones and the...	4.219009	1198	Action|Adventure	43295
26503	Yojimbo (1961)	4.211717	3030	Action|Adventure	3559
2730	Big Sleep, The (1946)	4.207361	1284	Crime|Film-Noir|Mystery	5529
922	All About Eve (1950)	4.204103	926	Drama	4826
21755	Spirited Away (Sen to Chihiro no kamikakushi) ...	4.203810	5618	Adventure|Animation|Fantasy	13466
4674	Chinatown (1974)	4.199673	1252	Crime|Film-Noir|Mystery|Thriller	15310
17040	Notorious (1946)	4.197790	930	Film-Noir|Romance|Thriller	4932
1090	Amelie (Fabuleux destin d'AmÃ©lie Poulain, Le)...	4.197072	4973	Comedy|Romance	24349
14594	M (1931)	4.193171	1260	Crime|Film-Noir|Thriller	4232
...	...	...	...	...	...
18439	Poltergeist III (1988)	2.006775	1996	Horror|Thriller	1107
22490	Superman IV: The Quest for Peace (1987)	2.000895	2643	Action|Adventure|Sci-Fi	3352
14236	Look Who's Talking Too (1990)	1.996738	7701	Comedy|Romance	1533
8918	Friday the 13th Part V: A New Beginning (1985)	1.990967	1978	Horror	1107
3317	Book of Shadows: Blair Witch 2 (2000)	1.987456	3973	Crime|Horror|Mystery|Thriller	1435
8920	Friday the 13th Part VII: The New Blood (1988)	1.979980	1980	Horror	1024
9891	Grease 2 (1982)	1.963760	1381	Comedy|Musical|Romance	3670
14235	Look Who's Talking Now (1993)	1.947820	5452	Children|Comedy|Romance	2156
15519	Mighty Morphin Power Rangers: The Movie (1995)	1.929002	181	Action|Children	2655

	title	mean rating	movieId	genres	total ratings
10184	Halloween III: Season of the Witch (1982)	1.917925	1984	Horror	1060
7813	Exorcist II: The Heretic (1977)	1.917319	1998	Horror	1022
4344	Catwoman (2004)	1.917193	8666	Action|Crime|Fantasy	1425
21708	Speed 2: Cruise Control (1997)	1.912317	1556	Action|Romance|Thriller	5326
18420	Police Academy 5: Assignment: Miami Beach (1988)	1.888089	2382	Comedy|Crime	2292
18406	PokÃ©mon the Movie 2000 (2000)	1.870215	3799	Animation|Children	1071
12349	Jaws: The Revenge (1987)	1.869478	4124	Horror|Thriller	1245
22240	Street Fighter (1994)	1.867222	393	Action|Adventure|Fantasy	2041
12348	Jaws 3-D (1983)	1.847328	1389	Action|Horror	1965
12802	Kazaam (1996)	1.827577	810	Children|Comedy|Fantasy	1882
8525	Flintstones in Viva Rock Vegas, The (2000)	1.826586	3564	Children|Comedy	1482
10952	Home Alone 3 (1997)	1.819985	1707	Children|Comedy	2697
7167	Dumb and Dumberer: When Harry Met Lloyd (2003)	1.796218	6482	Comedy	1428
18421	Police Academy 6: City Under Siege (1989)	1.794896	2383	Comedy|Crime	2155
24908	Universal Soldier: The Return (1999)	1.791195	2807	Action|Sci-Fi	1238
21724	Spice World (1997)	1.770316	1760	Comedy	2658
22107	Stop! Or My Mom Will Shoot (1992)	1.768392	3268	Action|Comedy	1835
8921	Friday the 13th Part VIII: Jason Takes Manhatt...	1.760067	1981	Horror	1192
18741	Problem Child 2 (1991)	1.760000	2799	Comedy	1125
1882	Baby Geniuses (1999)	1.703002	2555	Comedy	1399
2241	Battlefield Earth (2000)	1.600554	3593	Action|Sci-Fi	3973

3163 rows × 5 columns

In [25]:

```python
#This is a content specific recommender that weighs the movies that you like and ranks them according

my_ratings = dataset.drop(dataset[dataset["userId"] != 138494].index)

movies_watched = []
for movie in my_ratings.title:
    movies_watched.append(movie)

def generate_user_data():
    user_data = {
        '(no genres listed)': 0,
        'Action': 0,
        'Adventure': 0,
        'Animation': 0,
        'Children': 0,
        'Comedy': 0,
        'Crime': 0,
        'Documentary': 0,
        'Drama': 0,
        'Fantasy': 0,
        'Film-Noir': 0,
        'Horror': 0,
        'IMAX': 0,
        'Musical': 0,
        'Mystery': 0,
        'Romance': 0,
        'Sci-Fi': 0,
        'Thriller': 0,
        'War': 0,
        'Western': 0
    }

    user_num = {
        '(no genres listed)': 0,
        'Action': 0,
        'Adventure': 0,
        'Animation': 0,
        'Children': 0,
        'Comedy': 0,
        'Crime': 0,
        'Documentary': 0,
        'Drama': 0,
        'Fantasy': 0,
        'Film-Noir': 0,
        'Horror': 0,
        'IMAX': 0,
        'Musical': 0,
        'Mystery': 0,
        'Romance': 0,
        'Sci-Fi': 0,
        'Thriller': 0,
        'War': 0,
        'Western': 0
    }

    for index, movie in my_ratings.iterrows():
        split = movie["genres"].split("|")
        for text in split:
            user_data[text] += movie["rating"]/5
            user_num[text]+=1

    if(user_num["(no genres listed)"]>0):
        user_data["(no genres listed)"]/=user_num["(no genres listed)"]
    if(user_num["Action"]>0):
        user_data["Action"]/=user_num["Action"]
    if(user_num["Adventure"]>0):
        user_data["Adventure"]/=user_num["Adventure"]
    if(user_num["Animation"]>0):
        user_data["Animation"]/=user_num["Animation"]
    if(user_num["Children"]>0):
        user_data["Children"]/=user_num["Children"]
```

```python
    if(user_num["Comedy"]>0):
        user_data["Comedy"]/=user_num["Comedy"]
    if(user_num["Crime"]>0):
        user_data["Crime"]/=user_num["Crime"]
    if(user_num["Documentary"]>0):
        user_data["Documentary"]/=user_num["Documentary"]
    if(user_num["Drama"]>0):
        user_data["Drama"]/=user_num["Drama"]
    if(user_num["Fantasy"]>0):
        user_data["Fantasy"]/=user_num["Fantasy"]
    if(user_num["Film-Noir"]>0):
        user_data["Film-Noir"]/=user_num["Film-Noir"]
    if(user_num["Horror"]>0):
        user_data["Horror"]/=user_num["Horror"]
    if(user_num["IMAX"]>0):
        user_data["IMAX"]/=user_num["IMAX"]
    if(user_num["Musical"]>0):
        user_data["Musical"]/=user_num["Musical"]
    if(user_num["Mystery"]>0):
        user_data["Mystery"]/=user_num["Mystery"]
    if(user_num["Romance"]>0):
        user_data["Romance"]/=user_num["Romance"]
    if(user_num["Sci-Fi"]>0):
        user_data["Sci-Fi"]/=user_num["Sci-Fi"]
    if(user_num["Thriller"]>0):
        user_data["Thriller"]/=user_num["Thriller"]
    if(user_num["War"]>0):
        user_data["War"]/=user_num["War"]
    if(user_num["Western"]>0):
        user_data["Western"]/=user_num["Western"]
    return user_data

user_data = generate_user_data()

user_data
```

Out[25]:

```
{'(no genres listed)': 0,
 'Action': 0.75,
 'Adventure': 0.7090909090909091,
 'Animation': 0.8166666666666668,
 'Children': 0.825,
 'Comedy': 0.7777777777777777,
 'Crime': 0.7,
 'Documentary': 0,
 'Drama': 0.7857142857142857,
 'Fantasy': 0.7,
 'Film-Noir': 0,
 'Horror': 0.8,
 'IMAX': 0,
 'Musical': 0.95,
 'Mystery': 0,
 'Romance': 0.86,
 'Sci-Fi': 0.6999999999999998,
 'Thriller': 0.6666666666666666,
 'War': 1.0,
 'Western': 0}
```

In [26]:

```
#generating recommendations
def recommend(user_data):
    personal_list = pd.DataFrame({
        'personal_score' : []
    })
    recommendations = pd.concat([simp_pred,personal_list], axis = 1)

    for index, movie in recommendations.iterrows():
        personal_data = pd.DataFrame({
            'personal_score' : [0]
        })
        personal_total = 0
        genre_total = 0
        split = movie['genres'].split("|")
        for text in split:
            personal_total += user_data[text]
            genre_total += 1
        recommendations.at[index, 'personal_score'] = (personal_total/genre_total)*movie['mean rating

    recommendations = recommendations.sort_values(by = 'personal_score',ascending= False)

    for movie in movies_watched:
        recommendations = recommendations.drop(recommendations[recommendations["title"] == movie].ind

    return recommendations

recommend(user_data).head(10)
```

Out[26]:

	title	mean rating	movieId	genres	total ratings	personal_score
19962	Run Silent Run Deep (1958)	3.901608	2670	War	1306	3.901608
20342	Schindler's List (1993)	4.310175	527	Drama|War	50054	3.848371
17904	Paths of Glory (1957)	4.232623	1178	Drama|War	3568	3.779128
6984	Dr. Strangelove or: How I Learned to Stop Worr...	4.247287	750	Comedy|War	23220	3.775366
7147	Duck Soup (1933)	4.129646	1256	Comedy|Musical|War	5569	3.754919
19141	Ran (1985)	4.173611	1217	Drama|War	4824	3.726438
9843	Grand Illusion (La grande illusion) (1937)	4.150878	3134	Drama|War	1594	3.706141
9244	General, The (1926)	4.162770	3022	Comedy|War	2224	3.700240
2544	Best Years of Our Lives, The (1946)	4.114776	1939	Drama|War	1834	3.673908
21842	Stalag 17 (1953)	4.105514	3196	Drama|War	2829	3.665638

3. Augmenting and Testing Your Recommender System

How well do you think it does at predicting movies that you'd like? How would you improve it?

Right now, it works by looking at the genres of the movies I liked, and weighting the other movies based on the genres as well as their overall rating. By doing so, I get a 'score' that takes into account whether the movie is within my genre and if other people liked it. With this initial test, it seems like they recommend Drama and War movies because in my initial 20 ratings, I had only one rating with War, and I gave it a 5. Therefore, the system thinks I like War movies a lot.

Currently, my recommendations look like this:

1. Run Silent Run deep (1958)
2. Schindler's List (1993)
3. Paths of Glory (1957)
4. Dr Strangelove...
5. Duck Soup(1933)

I will now add Run Silent Run Deep and run the recommendation system again to generate new recommendations, and repeat

1. Add Run Silent Run Deep

In [27]:

```
#inserting Run Silent Run Deep

new_ratings = pd.DataFrame({
    'userId': [138494],
    'movieId': [2670],
    'rating': [4.5]
})

ratings = ratings.append(new_ratings, ignore_index = True)
dataset = pd.merge(movies, ratings)
dataset = dataset.dropna()
my_ratings = dataset.drop(dataset[dataset["userId"] != 138494].index)

movies_watched = []
for movie in my_ratings.title:
    movies_watched.append(movie)

user_data = generate_user_data()
recommend(user_data).head(10)
```

Out[27]:

	title	mean rating	movieId	genres	total ratings	personal_score
20342	Schindler's List (1993)	4.310175	527	Drama|War	50054	3.740616
7147	Duck Soup (1933)	4.129646	1256	Comedy|Musical|War	5569	3.686092
17904	Paths of Glory (1957)	4.232623	1178	Drama|War	3568	3.673312
6984	Dr. Strangelove or: How I Learned to Stop Worr...	4.247287	750	Comedy|War	23220	3.669184
19141	Ran (1985)	4.173611	1217	Drama|War	4824	3.622098
9843	Grand Illusion (La grande illusion) (1937)	4.150878	3134	Drama|War	1594	3.602369
9244	General, The (1926)	4.162770	3022	Comedy|War	2224	3.596171
2544	Best Years of Our Lives, The (1946)	4.114776	1939	Drama|War	1834	3.571038
21842	Stalag 17 (1953)	4.105514	3196	Drama|War	2829	3.563000
15302	Meet Me in St. Louis (1944)	3.737912	918	Musical	1820	3.551016

2. Add Schindler's List

In [28]:

```python
new_ratings = pd.DataFrame({
    'userId': [138494],
    'movieId': [527],
    'rating': [4.5]
})

ratings = ratings.append(new_ratings, ignore_index = True)
dataset = pd.merge(movies, ratings)
dataset = dataset.dropna()
my_ratings = dataset.drop(dataset[dataset["userId"] != 138494].index)

movies_watched = []
for movie in my_ratings.title:
     movies_watched.append(movie)

user_data = generate_user_data()

#This is the updated user data
recommend(user_data).head(10)
```

Out[28]:

	title	mean rating	movieId	genres	total ratings	personal_score
17904	Paths of Glory (1957)	4.232623	1178	Drama|War	3568	3.668274
7147	Duck Soup (1933)	4.129646	1256	Comedy|Musical|War	5569	3.663149
6984	Dr. Strangelove or: How I Learned to Stop Worr...	4.247287	750	Comedy|War	23220	3.633790
19141	Ran (1985)	4.173611	1217	Drama|War	4824	3.617130
9843	Grand Illusion (La grande illusion) (1937)	4.150878	3134	Drama|War	1594	3.597428
2544	Best Years of Our Lives, The (1946)	4.114776	1939	Drama|War	1834	3.566140
9244	General, The (1926)	4.162770	3022	Comedy|War	2224	3.561481
21842	Stalag 17 (1953)	4.105514	3196	Drama|War	2829	3.558112
15302	Meet Me in St. Louis (1944)	3.737912	918	Musical	1820	3.551016
12953	Killing Fields, The (1984)	4.090070	1299	Drama|War	7150	3.544727

3. Add Paths of Glory

In [29]:

```python
new_ratings = pd.DataFrame({
    'userId': [138494],
    'movieId': [1178],
    'rating': [5]
})

ratings = ratings.append(new_ratings, ignore_index = True)
dataset = pd.merge(movies, ratings)
dataset = dataset.dropna()
my_ratings = dataset.drop(dataset[dataset["userId"] != 138494].index)

movies_watched = []
for movie in my_ratings.title:
     movies_watched.append(movie)

user_data = generate_user_data()

#This is the updated user data
recommend(user_data).head(10)
```

Out[29]:

	title	mean rating	movieId	genres	total ratings	personal_score
19141	Ran (1985)	4.173611	1217	Drama|War	4824	3.698283
7147	Duck Soup (1933)	4.129646	1256	Comedy|Musical|War	5569	3.686092
9843	Grand Illusion (La grande illusion) (1937)	4.150878	3134	Drama|War	1594	3.678139
6984	Dr. Strangelove or: How I Learned to Stop Worr...	4.247287	750	Comedy|War	23220	3.669184
2544	Best Years of Our Lives, The (1946)	4.114776	1939	Drama|War	1834	3.646149
21842	Stalag 17 (1953)	4.105514	3196	Drama|War	2829	3.637942
12953	Killing Fields, The (1984)	4.090070	1299	Drama|War	7150	3.624256
2226	Battle of Algiers, The (La battaglia di Algeri...	4.069467	26131	Drama|War	1238	3.606000
9244	General, The (1926)	4.162770	3022	Comedy|War	2224	3.596171
12739	Kagemusha (1980)	4.054764	3091	Drama|War	1333	3.592971

4. Add Ran

In [30]:

```python
new_ratings = pd.DataFrame({
    'userId': [138494],
    'movieId': [1217],
    'rating': [5]
})

ratings = ratings.append(new_ratings, ignore_index = True)
dataset = pd.merge(movies, ratings)
dataset = dataset.dropna()
my_ratings = dataset.drop(dataset[dataset["userId"] != 138494].index)

movies_watched = []
for movie in my_ratings.title:
     movies_watched.append(movie)

user_data = generate_user_data()

#This is the updated user data
recommend(user_data).head(10)
```

Out[30]:

	title	mean rating	movieId	genres	total ratings	personal_score
9843	Grand Illusion (La grande illusion) (1937)	4.150878	3134	Drama|War	1594	3.735790
2544	Best Years of Our Lives, The (1946)	4.114776	1939	Drama|War	1834	3.703299
7147	Duck Soup (1933)	4.129646	1256	Comedy|Musical|War	5569	3.699857
21842	Stalag 17 (1953)	4.105514	3196	Drama|War	2829	3.694963
6984	Dr. Strangelove or: How I Learned to Stop Worr...	4.247287	750	Comedy|War	23220	3.690420
12953	Killing Fields, The (1984)	4.090070	1299	Drama|War	7150	3.681063
2226	Battle of Algiers, The (La battaglia di Algeri...	4.069467	26131	Drama|War	1238	3.662520
12739	Kagemusha (1980)	4.054764	3091	Drama|War	1333	3.649287
18176	Pianist, The (2002)	4.053923	5995	Drama|War	10515	3.648531
17922	Patton (1970)	4.044256	1272	Drama|War	8157	3.639831

5. Add Grand Illusion

In [31]:

```python
new_ratings = pd.DataFrame({
    'userId': [138494],
    'movieId': [3134],
    'rating': [5]
})

ratings = ratings.append(new_ratings, ignore_index = True)
dataset = pd.merge(movies, ratings)
dataset = dataset.dropna()
my_ratings = dataset.drop(dataset[dataset["userId"] != 138494].index)

movies_watched = []
for movie in my_ratings.title:
     movies_watched.append(movie)

user_data = generate_user_data()

#This is the updated user data
recommend(user_data).head(10)
```

Out[31]:

	title	mean rating	movieId	genres	total ratings	personal_score
2544	Best Years of Our Lives, The (1946)	4.114776	1939	Drama|War	1834	3.746940
21842	Stalag 17 (1953)	4.105514	3196	Drama|War	2829	3.738506
12953	Killing Fields, The (1984)	4.090070	1299	Drama|War	7150	3.724442
7147	Duck Soup (1933)	4.129646	1256	Comedy|Musical|War	5569	3.709034
2226	Battle of Algiers, The (La battaglia di Algeri...	4.069467	26131	Drama|War	1238	3.705681
6984	Dr. Strangelove or: How I Learned to Stop Worr...	4.247287	750	Comedy|War	23220	3.704578
12739	Kagemusha (1980)	4.054764	3091	Drama|War	1333	3.692292
18176	Pianist, The (2002)	4.053923	5995	Drama|War	10515	3.691527
17922	Patton (1970)	4.044256	1272	Drama|War	8157	3.682724
11092	Hotel Rwanda (2004)	4.040221	30749	Drama|War	8416	3.679050

6. Add The Best Years of Our Lives

In [32]:

```
new_ratings = pd.DataFrame({
    'userId': [138494],
    'movieId': [1939],
    'rating': [5]
})

ratings = ratings.append(new_ratings, ignore_index = True)
dataset = pd.merge(movies, ratings)
dataset = dataset.dropna()
my_ratings = dataset.drop(dataset[dataset["userId"] != 138494].index)

movies_watched = []
for movie in my_ratings.title:
     movies_watched.append(movie)

user_data = generate_user_data()

#This is the updated user data
recommend(user_data).head(10)
```

Out[32]:

	title	mean rating	movieId	genres	total ratings	personal_score
21842	Stalag 17 (1953)	4.105514	3196	Drama|War	2829	3.773163
12953	Killing Fields, The (1984)	4.090070	1299	Drama|War	7150	3.758969
2226	Battle of Algiers, The (La battaglia di Algeri...	4.069467	26131	Drama|War	1238	3.740034
12739	Kagemusha (1980)	4.054764	3091	Drama|War	1333	3.726521
18176	Pianist, The (2002)	4.053923	5995	Drama|War	10515	3.725748
17922	Patton (1970)	4.044256	1272	Drama|War	8157	3.716864
7147	Duck Soup (1933)	4.129646	1256	Comedy|Musical|War	5569	3.715589
6984	Dr. Strangelove or: How I Learned to Stop Worr...	4.247287	750	Comedy|War	23220	3.714691
11092	Hotel Rwanda (2004)	4.040221	30749	Drama|War	8416	3.713155
3529	Breaker Morant (1980)	4.033463	3811	Drama|War	1793	3.706945

7. Add Stalag 17

In [33]:

```python
new_ratings = pd.DataFrame({
    'userId': [138494],
    'movieId': [3196],
    'rating': [5]
})

ratings = ratings.append(new_ratings, ignore_index = True)
dataset = pd.merge(movies, ratings)
dataset = dataset.dropna()
my_ratings = dataset.drop(dataset[dataset["userId"] != 138494].index)

movies_watched = []
for movie in my_ratings.title:
     movies_watched.append(movie)

user_data = generate_user_data()

#This is the updated user data
recommend(user_data).head(10)
```

Out[33]:

	title	mean rating	movieId	genres	total ratings	personal_score
12953	Killing Fields, The (1984)	4.090070	1299	Drama|War	7150	3.787247
2226	Battle of Algiers, The (La battaglia di Algeri...	4.069467	26131	Drama|War	1238	3.768170
12739	Kagemusha (1980)	4.054764	3091	Drama|War	1333	3.754555
18176	Pianist, The (2002)	4.053923	5995	Drama|War	10515	3.753777
17922	Patton (1970)	4.044256	1272	Drama|War	8157	3.744826
11092	Hotel Rwanda (2004)	4.040221	30749	Drama|War	8416	3.741089
3529	Breaker Morant (1980)	4.033463	3811	Drama|War	1793	3.734832
9038	Full Metal Jacket (1987)	4.033180	1222	Drama|War	21926	3.734569
17345	One Flew Over the Cuckoo's Nest (1975)	4.248079	1193	Drama	29932	3.725238
6951	Downfall (Untergang, Der) (2004)	4.022386	31410	Drama|War	3998	3.724575

8. Add The Killing Fields

In [34]:

```
new_ratings = pd.DataFrame({
    'userId': [138494],
    'movieId': [1299],
    'rating': [5]
})

ratings = ratings.append(new_ratings, ignore_index = True)
dataset = pd.merge(movies, ratings)
dataset = dataset.dropna()
my_ratings = dataset.drop(dataset[dataset["userId"] != 138494].index)

movies_watched = []
for movie in my_ratings.title:
     movies_watched.append(movie)

user_data = generate_user_data()

#This is the updated user data
recommend(user_data).head(10)
```

Out[34]:

	title	mean rating	movieId	genres	total ratings	personal_score
2226	Battle of Algiers, The (La battaglia di Algeri...	4.069467	26131	Drama|War	1238	3.791710
12739	Kagemusha (1980)	4.054764	3091	Drama|War	1333	3.778010
18176	Pianist, The (2002)	4.053923	5995	Drama|War	10515	3.777227
17922	Patton (1970)	4.044256	1272	Drama|War	8157	3.768220
11092	Hotel Rwanda (2004)	4.040221	30749	Drama|War	8416	3.764460
17345	One Flew Over the Cuckoo's Nest (1975)	4.248079	1193	Drama	29932	3.762584
3529	Breaker Morant (1980)	4.033463	3811	Drama|War	1793	3.758164
9038	Full Metal Jacket (1987)	4.033180	1222	Drama|War	21926	3.757899
6951	Downfall (Untergang, Der) (2004)	4.022386	31410	Drama|War	3998	3.747842
79	12 Angry Men (1957)	4.224138	1203	Drama	12934	3.741379

9. Add The Battle of Algiers

In [35]:

```python
new_ratings = pd.DataFrame({
    'userId': [138494],
    'movieId': [26131],
    'rating': [5]
})

ratings = ratings.append(new_ratings, ignore_index = True)
dataset = pd.merge(movies, ratings)
dataset = dataset.dropna()
my_ratings = dataset.drop(dataset[dataset["userId"] != 138494].index)

movies_watched = []
for movie in my_ratings.title:
     movies_watched.append(movie)

user_data = generate_user_data()

#This is the updated user data
recommend(user_data).head(10)
```

Out[35]:

	title	mean rating	movieId	genres	total ratings	personal_score
12739	Kagemusha (1980)	4.054764	3091	Drama|War	1333	3.797962
18176	Pianist, The (2002)	4.053923	5995	Drama|War	10515	3.797175
17345	One Flew Over the Cuckoo's Nest (1975)	4.248079	1193	Drama	29932	3.794951
17922	Patton (1970)	4.044256	1272	Drama|War	8157	3.788120
11092	Hotel Rwanda (2004)	4.040221	30749	Drama|War	8416	3.784340
3529	Breaker Morant (1980)	4.033463	3811	Drama|War	1793	3.778011
9038	Full Metal Jacket (1987)	4.033180	1222	Drama|War	21926	3.777745
79	12 Angry Men (1957)	4.224138	1203	Drama	12934	3.773563
6951	Downfall (Untergang, Der) (2004)	4.022386	31410	Drama|War	3998	3.767635
922	All About Eve (1950)	4.204103	926	Drama	4826	3.755665

10. Kagemusha

In [36]:

```
new_ratings = pd.DataFrame({
    'userId': [138494],
    'movieId': [3091],
    'rating': [5]
})

ratings = ratings.append(new_ratings, ignore_index = True)
dataset = pd.merge(movies, ratings)
dataset = dataset.dropna()
my_ratings = dataset.drop(dataset[dataset["userId"] != 138494].index)

movies_watched = []
for movie in my_ratings.title:
     movies_watched.append(movie)

user_data = generate_user_data()

#This is the updated user data
recommend(user_data).head(10)
```

Out[36]:

	title	mean rating	movieId	genres	total ratings	personal_score
17345	One Flew Over the Cuckoo's Nest (1975)	4.248079	1193	Drama	29932	3.823271
18176	Pianist, The (2002)	4.053923	5995	Drama|War	10515	3.814373
17922	Patton (1970)	4.044256	1272	Drama|War	8157	3.805278
79	12 Angry Men (1957)	4.224138	1203	Drama	12934	3.801724
11092	Hotel Rwanda (2004)	4.040221	30749	Drama|War	8416	3.801481
3529	Breaker Morant (1980)	4.033463	3811	Drama|War	1793	3.795122
9038	Full Metal Jacket (1987)	4.033180	1222	Drama|War	21926	3.794856
6951	Downfall (Untergang, Der) (2004)	4.022386	31410	Drama|War	3998	3.784700
922	All About Eve (1950)	4.204103	926	Drama	4826	3.783692
24083	To Kill a Mockingbird (1962)	4.188943	1207	Drama	14769	3.770049

We have now inserted our top recommended movie into our watched list with high recommendations for each one. Since at the start, we had only one movie of War, and this was rated highly, this is reflected in our recommendations. As we continue to rate war movies highly, and most war movies are paired with drama, our final recommendations are entirely drama and war. If we started with a more complete dataset of over 20 movies, this recommendation algorithm might be more effective in giving you recommendations other than singling out one genre. Below we have the list of movies we have "watched" at the end and out final user data.

In [37]:

```
for i in range(30):
    print(str(i+1) + ". "+ movies_watched[i])
```

```
1. Jumanji (1995)
2. Forrest Gump (1994)
3. Schindler's List (1993)
4. 2001: A Space Odyssey (1968)
5. Paths of Glory (1957)
6. Ran (1985)
7. Killing Fields, The (1984)
8. Good Will Hunting (1997)
9. Mulan (1998)
10. Best Years of Our Lives, The (1946)
11. Run Silent Run Deep (1958)
12. Ferris Bueller's Day Off (1986)
13. Fight Club (1999)
14. Kagemusha (1980)
15. Grand Illusion (La grande illusion) (1937)
16. Stalag 17 (1953)
17. Incredibles, The (2004)
18. Battle of Algiers, The (La battaglia di Algeri) (1966)
19. Lion King II: Simba's Pride, The (1998)
20. Borat: Cultural Learnings of America for Make Benefit Glorious Nation of Kazakhstan (2006)
21. Avengers: Infinity War(2018)
22. Mission Impossible: Fallout(2018)
23. Isle Of Dogs(2018)
24. Logan(2017)
25. Ready Player One(2018)
26. Love, Simon(2018)
27. Game Night (2018)
28. The Incredibles 2(2018)
29. Coco (2017)
30. It (2017)
```

In [38]:

```
user_data
```

Out[38]:

```
{'(no genres listed)': 0,
 'Action': 0.75,
 'Adventure': 0.7090909090909091,
 'Animation': 0.8166666666666668,
 'Children': 0.825,
 'Comedy': 0.7777777777777777,
 'Crime': 0.7,
 'Documentary': 0,
 'Drama': 0.9,
 'Fantasy': 0.7,
 'Film-Noir': 0,
 'Horror': 0.8,
 'IMAX': 0,
 'Musical': 0.95,
 'Mystery': 0,
 'Romance': 0.86,
 'Sci-Fi': 0.6999999999999998,
 'Thriller': 0.6666666666666666,
 'War': 0.9818181818181819,
 'Western': 0}
```

A Recommender System for a New Dataset

We will now adapt and train a recommender algorithm for a new dataset that we find. We choose the dataset of Games Reviews, and will be using genres and ratings to build this recommender system once more.

In [39]:

```
xl_file = pd.ExcelFile('gamedata.xlsx')

dfs = {sheet_name: xl_file.parse(sheet_name)
          for sheet_name in xl_file.sheet_names}
```

In [40]:

```python
games = dfs["Sheet1"]
gameId = pd.DataFrame({
    'gameId' : []
})
games = pd.concat([gameId,games], axis = 1)
i = 1

for index, game in games.iterrows():
    games.at[index, 'gameId'] = i
    i+=1

games.head(10)
```

Out[40]:

	gameId	Game	Platform	Score	Genre
0	1.0	Wolfenstein: The New Order	Xbox One	7.8	Shooter
1	2.0	Mario Kart 8	Wii U	9.0	Racing, Action
2	3.0	Sportsfriends	PlayStation 3	8.7	Action, Compilation
3	4.0	Sportsfriends	PlayStation 4	8.7	Action, Compilation
4	5.0	Sportsfriends	PC	8.7	Action, Compilation
5	6.0	Super TIME Force	Xbox One	7.5	Shooter
6	7.0	Super TIME Force	Xbox 360	7.5	Shooter
7	8.0	The Walking Dead: Season Two -- Episode 3: In ...	PC	9.0	Adventure
8	9.0	Borderlands 2	PlayStation Vita	5.4	Shooter, RPG
9	10.0	Bound by Flame	PC	7.0	Action, RPG

In [41]:

```
#inserting 15 ratings
#my user id is 1

my_game_ratings = pd.DataFrame({
    'userId': [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
    'gameId': [2,362,363,2490,2491,3766,2793,171,504,1566,2343,3185,4149,4143,539],
    'rating': [7,8,8,9,9,10,7,6,10,7,7,8,6,8,8]
})

game_dataset = pd.merge(games, my_game_ratings)
game_dataset = game_dataset.dropna()
my_game_ratings = game_dataset.drop(game_dataset[game_dataset["userId"] != 1].index)


my_game_ratings.head(15)
```

Out[41]:

	gameId	Game	Platform	Score	Genre	userId	rating
0	2.0	Mario Kart 8	Wii U	9.0	Racing, Action	1	7
1	171.0	The Walking Dead: A Telltale Game Series -- Se...	PlayStation Vita	8.0	Adventure	1	6
2	362.0	Pokemon Y	Nintendo 3DS	9.0	RPG	1	8
3	363.0	Pokemon X	Nintendo 3DS	9.0	RPG	1	8
4	504.0	Dota 2	PC	9.4	RPG	1	10
5	539.0	Sid Meier's Civilization V: Brave New World	PC	9.4	Strategy	1	8
6	1566.0	Tetris	Nintendo 3DS	9.0	Puzzle	1	7
7	2343.0	Portal 2	PC	9.5	Shooter	1	7
8	2490.0	Pokemon Black Version	Nintendo DS	9.0	RPG	1	9
9	2491.0	Pokemon White Version	Nintendo DS	9.0	RPG	1	9
10	2793.0	Call of Duty: Black Ops	PC	8.5	Shooter	1	7
11	3185.0	Monster Rancher DS	Nintendo DS	5.5	Battle	1	8
12	3766.0	Pokemon SoulSilver Version	Nintendo DS	8.5	RPG	1	10
13	4143.0	Assassin's Creed: Bloodlines	PlayStation Portable	6.9	Action	1	8
14	4149.0	The Sims 3: World Adventures	PC	8.3	Simulation	1	6

In [42]:

```
#This is a content specific recommender that weighs the movies that you like and ranks them according

games_played = []
for game in my_game_ratings.gameId:
    games_played.append(game)

def generate_user_game_data():
    user_data = {
        'Shooter' : 0,
        'Racing': 0,
        'Action': 0,
        'Compilation': 0,
        'Adventure': 0,
        'RPG': 0,
        'Fighting': 0,
        'Strategy': 0,
        'Platformer': 0,
        'Simulation': 0,
        'Sports': 0,
        'Party': 0,
        'None': 0,
        'Baseball': 0,
        'Board': 0,
        'Card': 0,
        'Battle': 0,
        'First-Person': 0,
        'Puzzle': 0,
        'Music': 0,
        'Other': 0,
        'Flight': 0,
        'Pinball': 0,
        'Wrestling': 0,
        'Adult': 0,
        'Hunting': 0,
        'Educational': 0,
        'Word Games': 0,
        'Productivity': 0,
        'Trivia': 0,
        'Virtual Pet': 0,
        'Casino': 0,
        'Editor': 0
    }

    user_num = {
        'Shooter' : 0,
        'Racing': 0,
        'Action': 0,
        'Compilation': 0,
        'Adventure': 0,
        'RPG': 0,
        'Fighting': 0,
        'Strategy': 0,
        'Platformer': 0,
        'Simulation': 0,
        'Sports': 0,
        'Party': 0,
        'None': 0,
        'Baseball': 0,
        'Board': 0,
        'Card': 0,
        'Battle': 0,
        'First-Person': 0,
        'Puzzle': 0,
        'Music': 0,
        'Other': 0,
        'Flight': 0,
        'Pinball': 0,
        'Wrestling': 0,
        'Adult': 0,
        'Hunting': 0,
        'Educational': 0,
```

```
        'Word Games': 0,
        'Productivity': 0,
        'Trivia': 0,
        'Virtual Pet': 0,
        'Casino': 0,
        'Editor': 0
    }

    for index, game in my_game_ratings.iterrows():
        split = game["Genre"].split(", ")
        for text in split:
            user_data[text] += game["Score"]/10
            user_num[text]+=1

    if(user_num["Shooter"]>0):
        user_data["Shooter"]/=user_num["Shooter"]
    if(user_num["Racing"]>0):
        user_data["Racing"]/=user_num["Racing"]
    if(user_num["Action"]>0):
        user_data["Action"]/=user_num["Action"]
    if(user_num["Compilation"]>0):
        user_data["Compilation"]/=user_num["Compilation"]
    if(user_num["Adventure"]>0):
        user_data["Adventure"]/=user_num["Adventure"]
    if(user_num["RPG"]>0):
        user_data["RPG"]/=user_num["RPG"]
    if(user_num["Fighting"]>0):
        user_data["Fighting"]/=user_num["Fighting"]
    if(user_num["Strategy"]>0):
        user_data["Strategy"]/=user_num["Strategy"]
    if(user_num["Platformer"]>0):
        user_data["Platformer"]/=user_num["Platformer"]
    if(user_num["Simulation"]>0):
        user_data["Simulation"]/=user_num["Simulation"]
    if(user_num["Sports"]>0):
        user_data["Sports"]/=user_num["Sports"]
    if(user_num["Party"]>0):
        user_data["Party"]/=user_num["Party"]
    if(user_num["None"]>0):
        user_data["None"]/=user_num["None"]
    if(user_num["Baseball"]>0):
        user_data["Baseball"]/=user_num["Baseball"]
    if(user_num["Board"]>0):
        user_data["Board"]/=user_num["Board"]
    if(user_num["Card"]>0):
        user_data["Card"]/=user_num["Card"]
    if(user_num["Battle"]>0):
        user_data["Battle"]/=user_num["Battle"]
    if(user_num["First-Person"]>0):
        user_data["First-Person"]/=user_num["First-Person"]
    if(user_num["Puzzle"]>0):
        user_data["Puzzle"]/=user_num["Puzzle"]
    if(user_num["Music"]>0):
        user_data["Music"]/=user_num["Music"]
    if(user_num["Other"]>0):
        user_data["Other"]/=user_num["Other"]
    if(user_num["Flight"]>0):
        user_data["Flight"]/=user_num["Flight"]
    if(user_num["Pinball"]>0):
        user_data["Pinball"]/=user_num["Pinball"]
    if(user_num["Wrestling"]>0):
        user_data["Wrestling"]/=user_num["Wrestling"]
    if(user_num["Adult"]>0):
        user_data["Adult"]/=user_num["Adult"]
    if(user_num["Hunting"]>0):
        user_data["Hunting"]/=user_num["Hunting"]
    if(user_num["Educational"]>0):
        user_data["Educational"]/=user_num["Educational"]
    if(user_num["Word Games"]>0):
        user_data["Word Games"]/=user_num["Word Games"]
    if(user_num["Productivity"]>0):
        user_data["Productivity"]/=user_num["Productivity"]
```

```python
    if(user_num["Trivia"]>0):
        user_data["Trivia"]/=user_num["Trivia"]
    if(user_num["Virtual Pet"]>0):
        user_data["Virtual Pet"]/=user_num["Virtual Pet"]
    if(user_num["Casino"]>0):
        user_data["Casino"]/=user_num["Casino"]
    if(user_num["Editor"]>0):
        user_data["Editor"]/=user_num["Editor"]

    return user_data

user_game_data = generate_user_game_data()

user_game_data
```

Out[42]:

```
{'Shooter': 0.8999999999999999,
 'Racing': 0.9,
 'Action': 0.795,
 'Compilation': 0,
 'Adventure': 0.8,
 'RPG': 0.8983333333333333,
 'Fighting': 0,
 'Strategy': 0.9400000000000001,
 'Platformer': 0,
 'Simulation': 0.8300000000000001,
 'Sports': 0,
 'Party': 0,
 'None': 0,
 'Baseball': 0,
 'Board': 0,
 'Card': 0,
 'Battle': 0.55,
 'First-Person': 0,
 'Puzzle': 0.9,
 'Music': 0,
 'Other': 0,
 'Flight': 0,
 'Pinball': 0,
 'Wrestling': 0,
 'Adult': 0,
 'Hunting': 0,
 'Educational': 0,
 'Word Games': 0,
 'Productivity': 0,
 'Trivia': 0,
 'Virtual Pet': 0,
 'Casino': 0,
 'Editor': 0}
```

In [43]:

```python
#generating recommendations
def recommend_game(user_data):
    personal_list = pd.DataFrame({
        'personal_score' : []
    })
    recommendations = pd.concat([games,personal_list], axis = 1)

    for index, game in recommendations.iterrows():
        personal_data = pd.DataFrame({
            'personal_score' : [0]
        })
        personal_total = 0
        genre_total = 0
        split = game['Genre'].split(", ")
        for text in split:
            personal_total += user_data[text]
            genre_total += 1
        recommendations.at[index, 'personal_score'] = (personal_total/genre_total)*game['Score']

    recommendations = recommendations.sort_values(by = 'personal_score',ascending= False)

    for game in games_played:
        recommendations = recommendations.drop(recommendations[recommendations["Game"] == game].index

    return recommendations

recommend_game(user_game_data).head(10)
```

Out[43]:

	gameId	Game	Platform	Score	Genre	personal_score
14628	14629.0	Advance Wars	Game Boy Advance	9.9	Strategy	9.306000
14862	14863.0	Black & White	PC	9.7	Strategy	9.118000
16577	16578.0	Shanghai	Lynx	10.0	Puzzle	9.000000
16548	16549.0	Checkered Flag	Lynx	10.0	Racing	9.000000
9210	9211.0	Tornado Mania	Wireless	10.0	Puzzle	9.000000
16305	16306.0	Pokemon Yellow: Special Pikachu Edition	Game Boy	10.0	RPG	8.983333
14695	14696.0	Dragon Warrior III	Game Boy Color	10.0	RPG	8.983333
15359	15360.0	Pokemon Gold Version	Game Boy Color	10.0	RPG	8.983333
15358	15359.0	Pokemon Silver Version	Game Boy Color	10.0	RPG	8.983333
16627	16628.0	Pokemon Red Version	Game Boy	10.0	RPG	8.983333

For your dataset, what are good features for making recommendations? How well does the recommender work now? What would you do to improve it?

Since we are using games, genres are also applicable to make recommendations. It works pretty well, becuase a few of the games recommended to me were games I have played before and enjoyed. However, since these games are cross-platform, there are some games that I do not own the platform for, or do not like the platform for. The recommender system could keep track of which platforms the user has used before and adjust it's recommendations accordingly