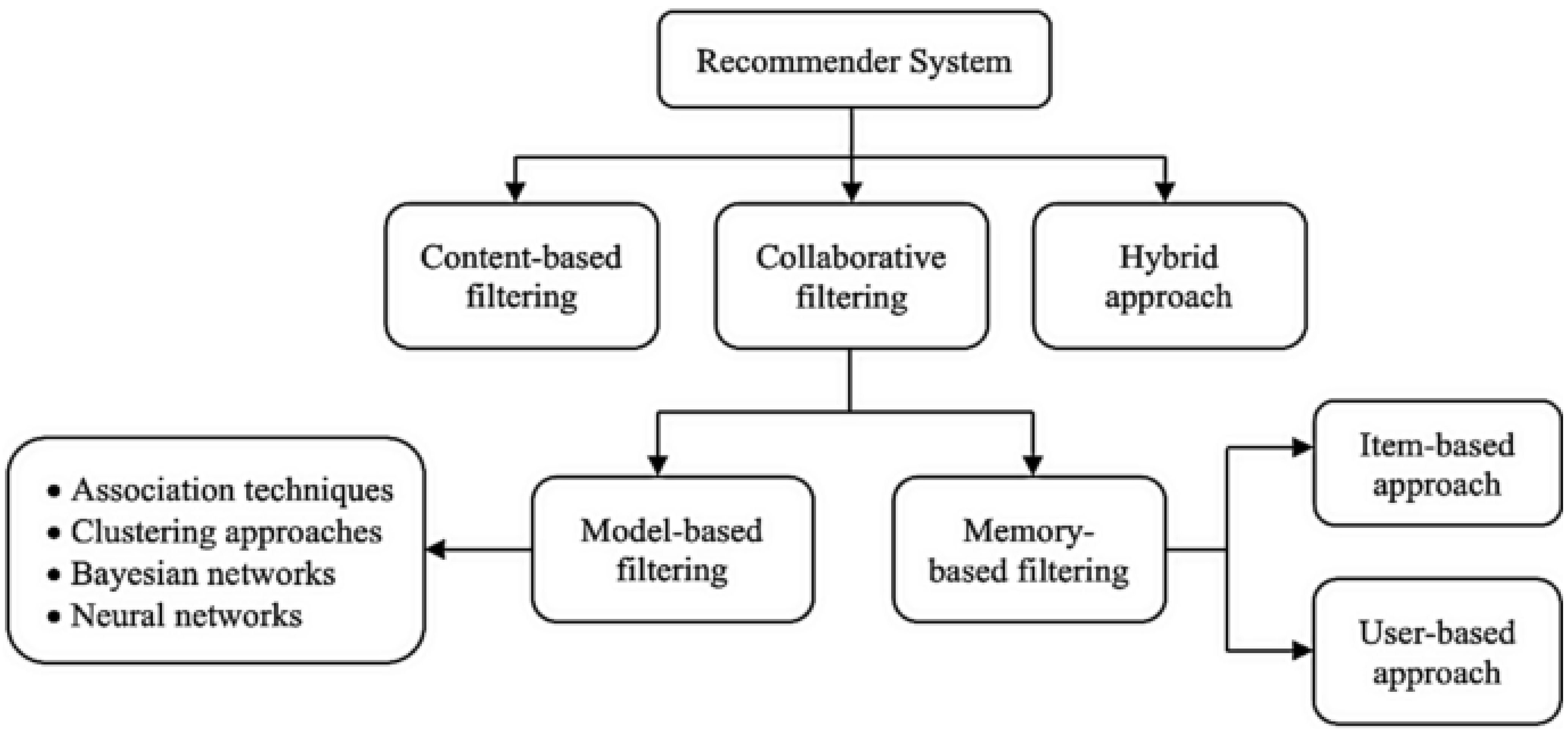
Recommender Systems using Python

What is Recommender System

Recommender System | Recommendation Engine | Recommendation System is basically an algorithm which uses big data to suggest or recommend additiona products o consumers. These Systems are based on

- past purchases
- search history
- demography
- many other factors

Types of Recommender System



Aim

In this notebook, we will focus on providing a basic recommendation system by suggesting items that are most similar to a particular item, in this case, movies i.e., Action and Comedy. It is a basic recommendation system, to describe it more accurately, it is a recommendation of movies that are most similar to user's movie choice.

Getting Started

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

Get the Data

let's first get the review dataset:

In [2]:
```
column_names = ['user_id', 'item_id', 'rating', 'timestamp']
review_dataset = pd.read_csv('Review.data', sep='\t', names=column_names)
```

Now let's get the movie title dataset:

In [3]:
```
movie_title_dataset = pd.read_csv("Movie_Id_Titles")
movie_title_dataset.head()
```

Out[3]:

	item_id	title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)

We can merge them together:

In [4]:
```
df = pd.merge(review_dataset,movie_title_dataset,on='item_id')
df.head()
```

Out[4]:

	user_id	item_id	rating	timestamp	title
0	0	50	5	881250949	Star Wars (1977)
1	290	50	5	880473582	Star Wars (1977)
2	79	50	4	891271545	Star Wars (1977)
3	2	50	5	888552084	Star Wars (1977)
4	8	50	5	879362124	Star Wars (1977)

EDA

Let's explore the data a bit and get a look at some of the best rated movies.

In [5]:
```
#visualization imports

import matplotlib.pyplot as plt
import seaborn as sns
```

```
sns.set_style('white')
%matplotlib inline
```

1. **Let's create a ratings dataframe with average rating and number of ratings:**

In [6]:
```
#average rating
df.groupby('title')['rating'].mean().sort_values(ascending=False).head()
```

Out[6]:
```
title
They Made Me a Criminal (1939)                5.0
Marlene Dietrich: Shadow and Light (1996)     5.0
Saint of Fort Washington, The (1993)          5.0
Someone Else's America (1995)                 5.0
Star Kid (1997)                               5.0
Name: rating, dtype: float64
```

In [7]:
```
#number of ratings
df.groupby('title')['rating'].count().sort_values(ascending=False).head()
```

Out[7]:
```
title
Star Wars (1977)             584
Contact (1997)               509
Fargo (1996)                 508
Return of the Jedi (1983)    507
Liar Liar (1997)             485
Name: rating, dtype: int64
```

In [8]:
```
ratings_data = pd.DataFrame(df.groupby('title')['rating'].mean())
ratings_data.head()
```

Out[8]:

title	rating
'Til There Was You (1997)	2.333333
1-900 (1994)	2.600000
101 Dalmatians (1996)	2.908257
12 Angry Men (1957)	4.344000
187 (1997)	3.024390

1. **Now set the number of ratings column:**

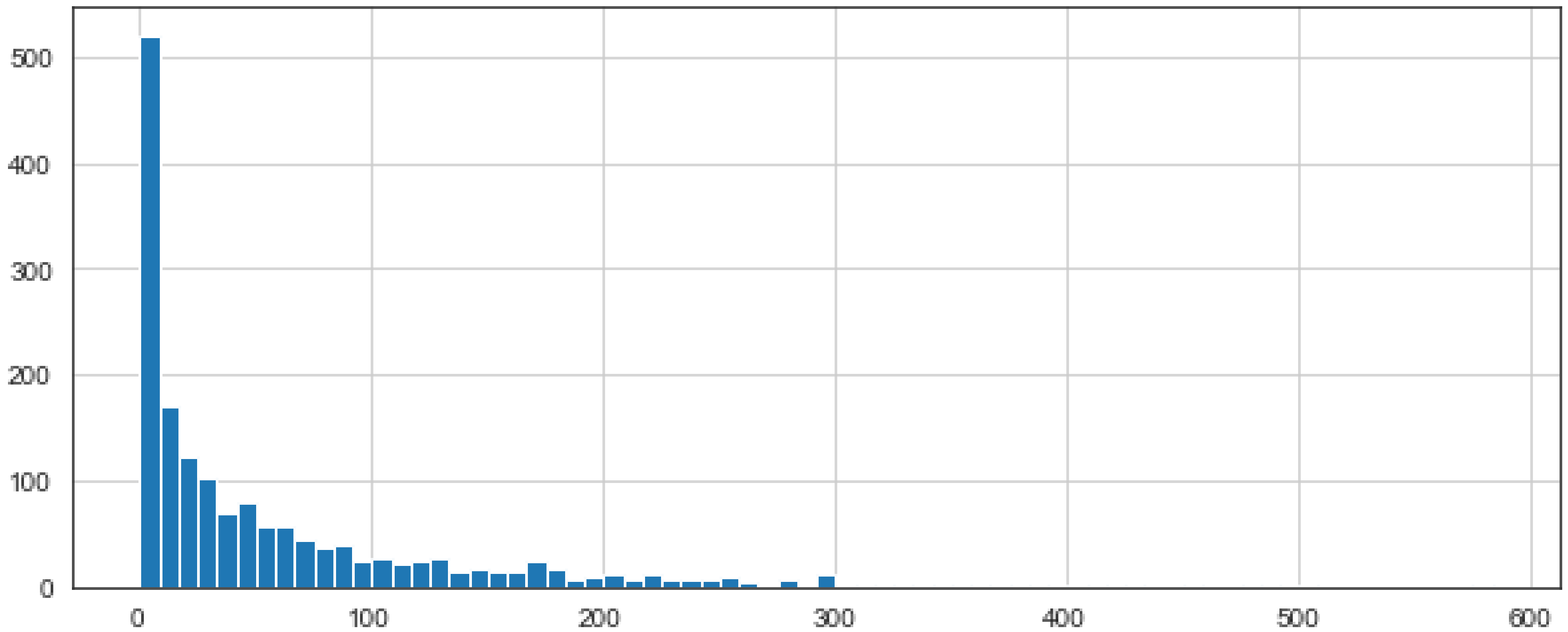
In [9]:
```
ratings_data['num of ratings'] = pd.DataFrame(df.groupby('title')['rating'].count())
ratings_data.head()
```

Out[9]:

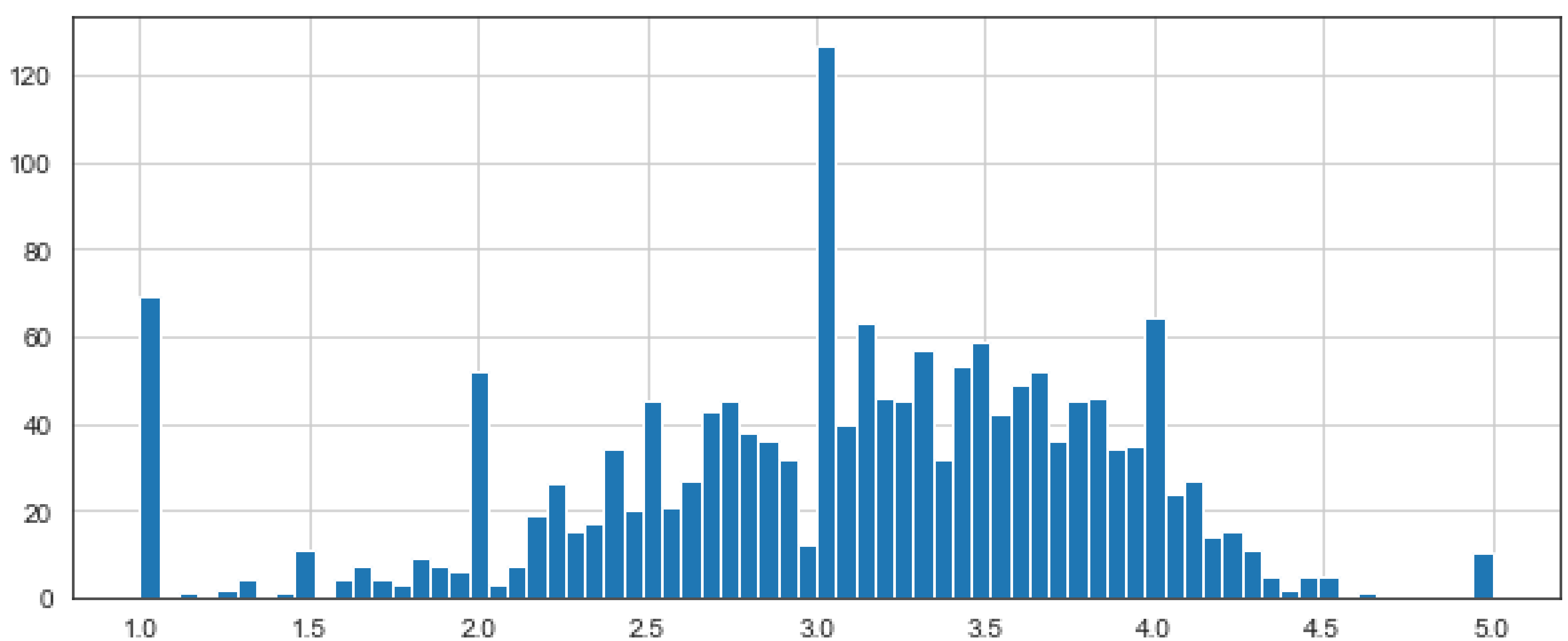
title	rating	num of ratings
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41

1. **Visualizing data with histograms:**

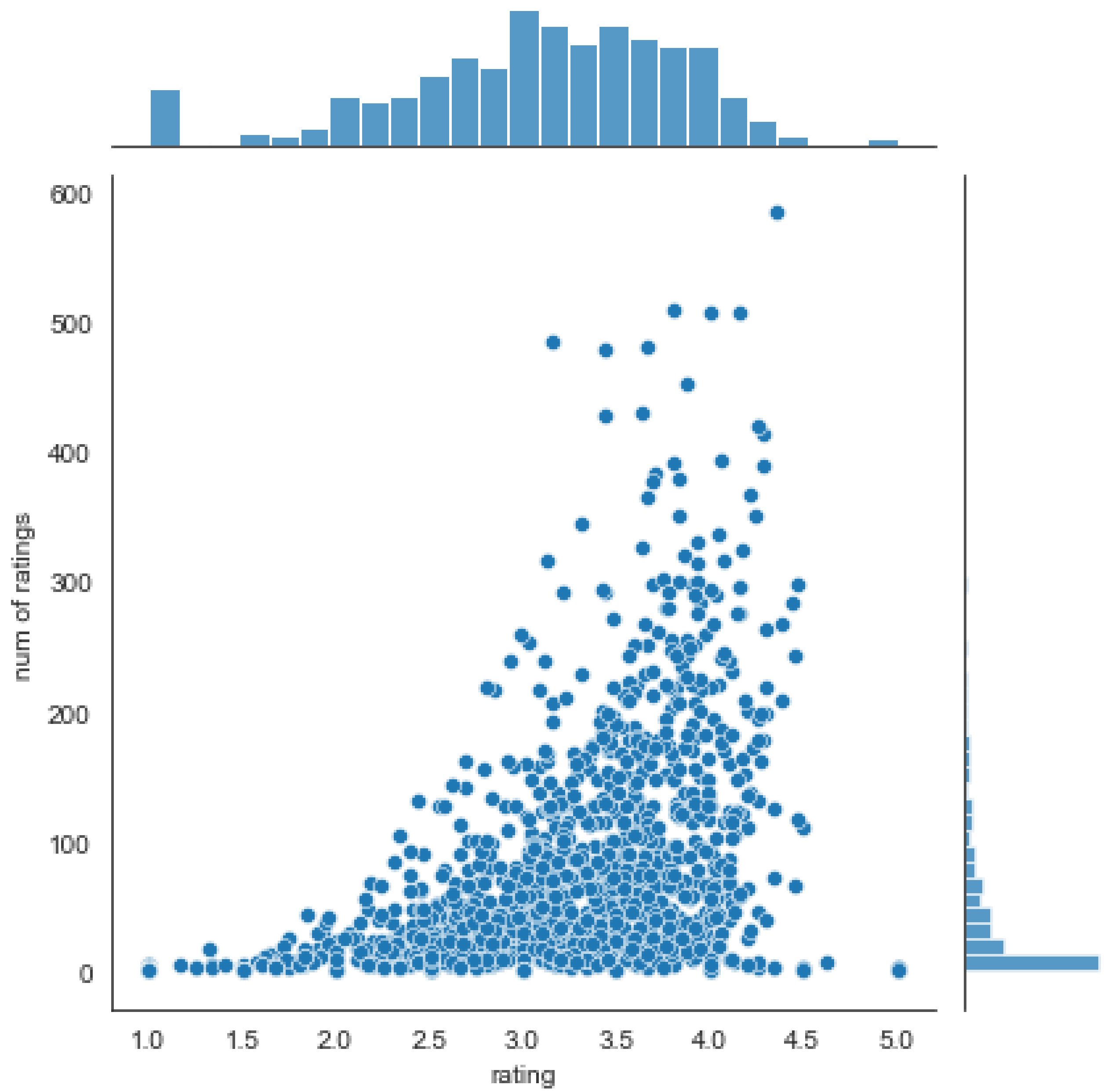
In [10]:
```
plt.figure(figsize=(10,4))
ratings_data['num of ratings'].hist(bins=70);
```



In [11]:
```
plt.figure(figsize=(10,4))
ratings_data['rating'].hist(bins=70);
```



In [12]:
```
sns.jointplot(x='rating',y='num of ratings',data=ratings_data, alpha= 1);
```



Okay! Now that we have a general idea of what the data looks like, let's move on to creating a simple recommendation system:

Recommending Similar Movies

Now let's create a matrix that has the user ids on one access and the movie title on another axis. Each cell will then consist of the rating the user gave to that movie.

Note: *there will be a lot of NaN values, because most people have not seen most of the movies.*

In [13]:
```
#creating movie matrix
moviemat = df.pivot_table(index='user_id',columns='title',values='rating')
moviemat.head()
```

Out[13]:

title	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)	39 Steps, The (1935)	...	Yankee Zulu (1994)	Year of the Horse (1997)	You So Crazy (1994)	Young Frankenstein (1974)	Young Guns (1988)	Y (
user_id																	
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	
1	NaN	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN	NaN	...	NaN	NaN	NaN	5.0	3.0	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	

5 rows × 1664 columns

Most rated movie:

In [14]:
```
ratings_data.sort_values('num of ratings',ascending=False).head(10)
```

Out[14]:

title	rating	num of ratings
Star Wars (1977)	4.359589	584
Contact (1997)	3.803536	509
Fargo (1996)	4.155512	508
Return of the Jedi (1983)	4.007890	507
Liar Liar (1997)	3.156701	485
English Patient, The (1996)	3.656965	481
Scream (1996)	3.441423	478
Toy Story (1995)	3.878319	452
Air Force One (1997)	3.631090	431
Independence Day (ID4) (1996)	3.438228	429

Let's choose two movies: starwars, a sci-fi movie. And Liar Liar, a comedy.

In [15]:

```
ratings_data.head()
```

Out[15]:

title	rating	num of ratings
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41

Now let's grab the user ratings for those two movies from movie matrix:

In [16]:
```
starwars_user_ratings = moviemat['Star Wars (1977)']  #sci-fi movie
liarliar_user_ratings = moviemat['Liar Liar (1997)']  #comedy
starwars_user_ratings.head()
```

Out[16]:
```
user_id
0    5.0
1    5.0
2    5.0
3    NaN
4    5.0
Name: Star Wars (1977), dtype: float64
```

We can then use corrwith() method to get correlations between two pandas series:

In [17]:
```
similar_to_starwars = moviemat.corrwith(starwars_user_ratings)
similar_to_liarliar = moviemat.corrwith(liarliar_user_ratings);
```

Let's clean this by removing NaN values and using a DataFrame instead of a series:

In [18]:
```
corr_starwars = pd.DataFrame(similar_to_starwars,columns=['Correlation'])
corr_starwars.dropna(inplace=True)
corr_starwars.head()
```

Out[18]:

title	Correlation
'Til There Was You (1997)	0.872872
1-900 (1994)	-0.645497
101 Dalmatians (1996)	0.211132
12 Angry Men (1957)	0.184289
187 (1997)	0.027398

Now if we sort the dataframe by correlation, we should get the most similar movies, however note that we get some results that don't really make sense. This is because there are a lot of movies only watched once by users who also watched star wars (it was the most popular movie).

In [19]:
```
#highest correlated movies to star wars

corr_starwars.sort_values('Correlation',ascending=False).head(10)
```

Out[19]:

	Correlation
title	
Hollow Reed (1996)	1.0
Commandments (1997)	1.0
Cosi (1996)	1.0
No Escape (1994)	1.0
Stripes (1981)	1.0
Star Wars (1977)	1.0
Man of the Year (1995)	1.0
Beans of Egypt, Maine, The (1994)	1.0
Old Lady Who Walked in the Sea, The (Vieille qui marchait dans la mer, La) (1991)	1.0
Outlaw, The (1943)	1.0

Let's fix this by filtering out movies that have less than 100 reviews (this value was chosen based off the histogram from earlier).

In [20]:
```
#on the basis of corr + Number of ratings
corr_starwars = corr_starwars.join(ratings_data['num of ratings'])
corr_starwars.head()
```

Out[20]:

title	Correlation	num of ratings
'Til There Was You (1997)	0.872872	9
1-900 (1994)	-0.645497	5
101 Dalmatians (1996)	0.211132	109
12 Angry Men (1957)	0.184289	125
187 (1997)	0.027398	41

Now sort the values and notice how the titles make a lot more sense:

In [21]:

```
#filter results where ratings > 100
corr_starwars[corr_starwars['num of ratings']>100].sort_values('Correlation',ascending=False).head()
```

Out[21]:

title	Correlation	num of ratings
Star Wars (1977)	1.000000	584
Empire Strikes Back, The (1980)	0.748353	368
Return of the Jedi (1983)	0.672556	507
Raiders of the Lost Ark (1981)	0.536117	420
Austin Powers: International Man of Mystery (1997)	0.377433	130

Now the same for the comedy Liar Liar:

In [22]:

```
#similarly
corr_liarliar = pd.DataFrame(similar_to_liarliar,columns=['Correlation'])
corr_liarliar.dropna(inplace=True)
```

```python
corr_liarliar = corr_liarliar.join(ratings_data['num of ratings'])
corr_liarliar[corr_liarliar['num of ratings']>100].sort_values('Correlation',ascending=False).head()
```

Out[22]:

title	Correlation	num of ratings
Liar Liar (1997)	1.000000	485
Batman Forever (1995)	0.516968	114
Mask, The (1994)	0.484650	129
Down Periscope (1996)	0.472681	101
Con Air (1997)	0.469828	137

Conclusion:

Recommender system which is content based accurately recommends sci-fi and comedy movies to the interested users respectively.