In [1]:

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

data = pd.read_csv(r"C:\Users\Dell\Documents\movie recommendation\ml-latest-small\ratings.csv")
data.head(10)
```

Out[1]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
5	1	70	3.0	964982400
6	1	101	5.0	964980868
7	1	110	4.0	964982176
8	1	151	5.0	964984041
9	1	157	5.0	964984100

In [2]:

```python
movie_titles_genre = pd.read_csv(r"C:\Users\Dell\Documents\movie recommendation\ml-latest-
small\movies.csv")
movie_titles_genre.head(10)
```

Out[2]:

	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 [3]:

```python
data = data.merge(movie_titles_genre,on='movieId', how='left')
data.head(10)
```

Out[3]:

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	1	3	4.0	964981247	Grumpier Old Men (1995)	Comedy|Romance
2	1	6	4.0	964982224	Heat (1995)	Action|Crime|Thriller
3	1	47	5.0	964983815	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller

	userId	movieId	rating	timestamp	title	genres
4	1	50	5.0	964982931	Usual Suspects, The (1995)	Crime|Mystery|Thriller
5	1	70	3.0	964982400	From Dusk Till Dawn (1996)	Action|Comedy|Horror|Thriller
6	1	101	5.0	964980868	Bottle Rocket (1996)	Adventure|Comedy|Crime|Romance
7	1	110	4.0	964982176	Braveheart (1995)	Action|Drama|War
8	1	151	5.0	964984041	Rob Roy (1995)	Action|Drama|Romance|War
9	1	157	5.0	964984100	Canadian Bacon (1995)	Comedy|War

In [4]:

```
Average_ratings = pd.DataFrame(data.groupby('title')['rating'].mean())
Average_ratings.head(10)
```

Out[4]:

title	rating
'71 (2014)	4.000000
'Hellboy': The Seeds of Creation (2004)	4.000000
'Round Midnight (1986)	3.500000
'Salem's Lot (2004)	5.000000
'Til There Was You (1997)	4.000000
'Tis the Season for Love (2015)	1.500000
'burbs, The (1989)	3.176471
'night Mother (1986)	3.000000
(500) Days of Summer (2009)	3.666667
*batteries not included (1987)	3.285714

In [5]:

```
Average_ratings['Total Ratings'] = pd.DataFrame(data.groupby('title')['rating'].count())
Average_ratings.head(10)
```

Out[5]:

title	rating	Total Ratings
'71 (2014)	4.000000	1
'Hellboy': The Seeds of Creation (2004)	4.000000	1
'Round Midnight (1986)	3.500000	2
'Salem's Lot (2004)	5.000000	1
'Til There Was You (1997)	4.000000	2
'Tis the Season for Love (2015)	1.500000	1
'burbs, The (1989)	3.176471	17
'night Mother (1986)	3.000000	1
(500) Days of Summer (2009)	3.666667	42
*batteries not included (1987)	3.285714	7

In [6]:

```
movie_user = data.pivot_table(index='userId',columns='title',values='rating')
movie_user.head(10)
```

Out[6]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[REC] (2007)	[REC]² (2009)	[REC]³ 3 Génesis (2012)
userId															
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	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
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN

10 rows × 9719 columns

In [7]:

```
correlations = movie_user.corrwith(movie_user['Toy Story (1995)'])
correlations.head()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\function_base.py:2522: RuntimeWarning:
Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar)
C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\function_base.py:2451: RuntimeWarning: divide
by zero encountered in true_divide
  c *= np.true_divide(1, fact)
```

Out[7]:

```
title
'71 (2014)                                NaN
'Hellboy': The Seeds of Creation (2004)   NaN
'Round Midnight (1986)                    NaN
'Salem's Lot (2004)                       NaN
'Til There Was You (1997)                 NaN
dtype: float64
```

In [8]:

```
recommendation = pd.DataFrame(correlations,columns=['Correlation'])
recommendation.dropna(inplace=True)
recommendation = recommendation.join(Average_ratings['Total Ratings'])
recommendation.head()
```

Out[8]:

	Correlation	Total Ratings
title		
'burbs, The (1989)	0.240563	17
(500) Days of Summer (2009)	0.353833	42
*batteries not included (1987)	-0.427425	7
10 Cent Pistol (2015)	1.000000	2
10 Cloverfield Lane (2016)	-0.285732	14

In [9]:

```
recc = recommendation[recommendation['Total Ratings']>100].sort_values('Correlation',ascending=False).reset_index()
```

In [10]:

```
recc = recc.merge(movie_titles_genre,on='title', how='left')
recc.head(10)
```

Out[10]:

	title	Correlation	Total Ratings	movieId	genres
0	Toy Story (1995)	1.000000	215	1	Adventure|Animation|Children|Comedy|Fantasy
1	Incredibles, The (2004)	0.643301	125	8961	Action|Adventure|Animation|Children|Comedy
2	Finding Nemo (2003)	0.618701	141	6377	Adventure|Animation|Children|Comedy
3	Aladdin (1992)	0.611892	183	588	Adventure|Animation|Children|Comedy|Musical
4	Monsters, Inc. (2001)	0.490231	132	4886	Adventure|Animation|Children|Comedy|Fantasy
5	Mrs. Doubtfire (1993)	0.446261	144	500	Comedy|Drama
6	Amelie (Fabuleux destin d'Amélie Poulain, Le) ...	0.438237	120	4973	Comedy|Romance
7	American Pie (1999)	0.420117	103	2706	Comedy|Romance
8	Die Hard: With a Vengeance (1995)	0.410939	144	165	Action|Crime|Thriller
9	E.T. the Extra-Terrestrial (1982)	0.409216	122	1097	Children|Drama|Sci-Fi

In [ ]:

```
Recommender systems are no joke. They have found enterprise application a long time ago by helping all the top players in the
online market place. Amazon, Netflix, Google and many others have been using the technology to curate content and products for
its customers. Amazon recommends products based on your purchase history, user ratings of the product etc. Netflix recommends
movies and TV shows all made possible by highly efficient recommender systems.

MovieLens Dataset

If you are a data aspirant you must definitely be familiar with the MovieLens dataset. It is one of the first go-to datasets
for building a simple recommender system.

We will build a simple Movie Recommendation System using the MovieLens dataset (F. Maxwell Harper and Joseph A. Konstan. 2015.
The MovieLens Datasets: History and Context.ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19.)

The dataset will consist of just over 100,000 ratings applied to over 9,000 movies by approximately 600 users.

Download the dataset from MovieLens.

The data is distributed in four different CSV files which are named as ratings, movies, links and tags. For building this
recommender we will only consider the ratings and the movies datasets.

The ratings dataset consists of 100,836 observations and each observation is a record of the ID for the user who rated the
movie (userId), the ID of the Movie that is rated (movieId), the rating given by the user for that particular movie (rating)
and the time at which the rating was recorded(timestamp).

The movies dataset consists of the ID of the movies(movieId), the corresponding title (title) and genre of each movie(genres).

Building A Simple Movie Recommender
Loading the data
Feature Engineering

Average Rating
```

```
The dataset is a collection of ratings by a number of users for different movies. Let’s find out th
e average rating for each
and every movie in the dataset.

Total Number Of Rating

The rating of a movie is proportional to the total number of ratings it has. Therefore, we will als
o consider the total ratings
cast for each movie.

Building The Recommender

Calculating The Correlation
The above code will create a table where the rows are userIds and the columns represent the movies.
The values of the matrix
represent the rating for each movie by each user.
Now we need to select a movie to test our recommender system. Choose any movie title from the data.
Here, I chose Toy Story
(1995).

To find the correlation value for the movie with all other movies in the data we will pass all the
ratings of the picked movie
to the corrwith method of the Pandas Dataframe. The method computes the pairwise correlation betwee
n rows or columns of a
DataFrame with rows or columns of Series or DataFrame.

Now we will remove all the empty values and merge the total ratings to the correlation table.

Testing The Recommendation System

Let’s filter all the movies with a correlation value to Toy Story (1995) and with at least 100 rati
ngs.

Let’s also merge the movies dataset for verifying the recommendations.

We can see that the top recommendations are pretty good. The movie that has the highest/full correl
ation to Toy Story is
Toy Story itself. The movies such as The Incredibles, Finding Nemo and Alladin show high correlatio
n with Toy Story.
```