(ABOUT DATASET)
*This dataset describes 5-star rating from [MovieLens](http://movielens.org), a movie recommendation service. It contains 100836 ratings
and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018.

*Users were selected at random for inclusion. All selected users had rated at least 20 movies. Each user is represented by an id, and no other information is provided.

*The data are contained in the files `links.csv`, `movies.csv`, `ratings.csv` and `tags.csv`.

user_id- the ID of the user who rated the movie
item_id- the ID of the movie
rating- user gave the rating to the movie, between 1 and 5
timestamp- the time the movie was rated
title- the title of the movie
==========CODE(with explanation under comment lines)==========
//The first thing we need to do is import libraries pandas and numpy//

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

//Warnings library is for the alerts that are triggered during the process of importing a module//

```
import warnings
warnings.filterwarnings('ignore')
```

//Now we will load the dataset using pandas read_csv()//

```
df = pd.read_csv('C:\\Users\\dell\\Desktop\\links.csv',
names=['user_id','item_id','rating','titmestamp'])
df = pd.read_csv('C:\\Users\\dell\\Desktop\\ratings.csv',
names=['user_id','item_id','rating','titmestamp'])
df = pd.read_csv('C:\\Users\\dell\\Desktop\\tags.csv',
names=['user_id','item_id','rating','titmestamp'])
```

//Now we will check the head of the data, the data we are dealing with//

```
df.head()
```

//Now we will load the titles of the movie instead of dealing with the IDs//

```
movie_titles = pd.read_csv('C:\\Users\\dell\\Desktop\\Movie_Titles.csv')
movie_titles.head()
```

//We can merge these datasets on this column as item_id columns are same that has to be merged//

```
df = pd.merge(df, Movie_Titles, on='item_id')
df.head()
```

//getting the brief description of our dataset//

```
df.describe()
```

```
//We will use groupby functionality of pandas here, by grouping the title column
and calculate the mean to get the average rating//

ratings = pd.DataFrame(df.groupby('title')['rating'].mean())
ratings.head()

//We use the count function to compute the number of ratings each movie got//

ratings['number_of_ratings'] = df.groupby('title')['rating'].count()
ratings.head()

//Now we will plot a histogram and visualize the distribution of ratings//

import matplotlib.pyplot as plt
%matplotlib inline
ratings['rating'].hist(bins=50)
ratings['number_of_ratings'].hist(bins=60)

//We will now check the relation between number of ratings and rating of a
movie by using a seaborn library and will plot a scatter plot and for this
jointplot is used//

import seaborn as sns
sns.jointplot(x='rating', y='number_of_ratings', data=ratings)

//Now we will convert our dataset into a matrix, we will use pivot_table() pandas
utility to create the matrix//

make_movie = df.pivot_table(index='user_id', columns='title', values='rating')
make_movie.head()

//Examples taken based on user's watching history//

example_user_rating = make_movie['Air Force One (1997)']
contact_user_rating = make_movie['Contact (1997)']

//we have the dataframes of the user rating that they gave to the two movies//

example_user_rating.head()
contact_user_rating.head()

//This indicates the strong similarity and correlation between the two movie
examples//

similar_as_example=make_movie.corrwith(example_user_rating)
similar_as_example.head()
similar_to_contact = make_movie.corrwith(contact_user_rating)
similar_to_contact.head()

//here we drop the null values as there are many missing and unuseful values,
correlation

results are transformed into dataframes//
corr_contact = pd.DataFrame(similar_to_contact, columns=['Correlation'])
corr_contact.dropna(inplace=True)
corr_contact.head()
corr_EXPL = pd.DataFrame(similar_as_example, columns=['correlation'])
corr_EXPL.dropna(inplace=True)
corr_EXPL.head()
```

```
//We will now join the two dataframes//

corr_EXPL = corr_EXPL.join(ratings['number_of_ratings'])
corr_contact = corr_contact.join(ratings['number_of_ratings'])
corr_EXPL .head()
corr_contact.head()

//now these will be sorted by the correlation column//

corr_EXPL[corr_EXPL['number_of_ratings'] > 100].sort_values(by='correlation',
ascending=False).head(10)

//now same for the other example//

corr_contact[corr_contact['number_of_ratings'] > 100].sort_values(by='Correlation',
ascending=False).head(10)

================================================================================
======================
```