DEPARTMENT OF COMPUTER SCIENCE
SHAHEED SUKHDEV COLLEGE OF BUSINESS STUDIES
(UNIVERSITY OF DELHI)

ANALYSIS ON MOVIE LENS DATASET (DATA ANALYSIS AND VISUALIZATION PROJECT REPORT)

SUBMITTED BY:
Shefalika Ghosh (19544)
Niti Tyagi (19522)

PROJECT SUPERVISOR:
Dr. Anamika Gupta, PhD
(Assistant Professor)

DECLARATION

It is hereby certified that the work being presented in the Data Analysis and Visualization Project Report entitled **“Movie Lens, a movie recommendation service"** has been successfully completed under the supervision of Dr. Anamika Gupta, Ph.D. (Assistant Professor, Shaheed Sukhdev College of Business Studies, affiliated to University of Delhi) and is an authentic record of my own work carried out during the academic year 2021-2022.

Shefalika Ghosh

(Roll No: 19544)

This is to certify that the above statement made by the student is correct to the best of my knowledge.

Dr. Anamika Gupta, Ph.D.

(Assistant Professor)

(Project Supervisor)

ACKNOWLEDGEMENT

A perfect finish to any project requires guidance and I was lucky to have that support, bearing, and supervision in every perspective from my instructor. I am using this opportunity to express my gratitude to my professor **Dr. Anamika Gupta** who supported me throughout the course of this Data Analysis and Visualization project. Her aspiring guidance, support, encouragement and enthusiasm during the project work helped me in widening my horizons of knowledge. I am sincerely grateful to her for sharing her honest and illuminating views and experience on a number of issues related to the project.

I would also like to extend my sincere thanks to my project partner **Ms. Niti Tyagi**. This project would not have been possible without her kind support, help and incredible contribution every step of the way.

This acknowledgement will remain incomplete if I fail to express my deep sense of obligation to my parents for their consistent support and encouragement.

ABSTRACT

Data analysis can be described as the process of collecting and organizing data to draw helpful conclusions that support decision-making. Analysis also involves visualization which gives us a clear idea of what the information means by giving it visual context through maps or graphs making the data easier to comprehend and hence, easier to identify trends, patterns, and outliers within large data sets. In our data-rich age, we generate and collect a colossal volume of data every day. The importance of data analysis lies in the fact that analyzed data reveals insights that tells one where to focus your efforts. This saves time, money, effort and gives rise to smarter business decisions. There are several methods and techniques to perform analysis depending on the industry and the aim of the analysis.

This project acts as a platform to give us an introductory understanding to the vast field of data analysis and visualization. For our project we have chosen the Movie Lens Dataset which describes ratings and free-text tagging activities from 'MovieLens', a movie recommendation service (more details on the dataset will follow). The Movie Lens Dataset is often used for the purpose of recommender systems which aim to give personalized movie recommendations based on a user's movie ratings.

The objective of this project is to analyze the MovieLens dataset to gain insight into the history of cinematography. The analysis answer questions related to popular genres, number of users reviewing the movies, average movie ratings and using them to recommend movies to users based on their interest. This document contains the full Python code used for the analysis and visualization of the dataset

Table of contents

1. Declaration 1

2. Acknowledgement 2

3. Abstract 3

4. Table of Contents 4

5. Dataset Description 5-7

6. Analysis and Visualization 8-42

7. Summary 43-44

8. Bibliography 45

Dataset Description

This dataset describes ratings and free-text tagging activities from MovieLens, a movie recommendation service. It contains 20000263 ratings and 465564 tag applications across 27278 movies. This data was created by 138493 users between January 09, 1995 and March 31, 2015.

The dataset was generated on October 17, 2016 with users having been selected at random for inclusion. All selected users had rated at least 20 movies.

The data in the dataset is contained in 2 files: rating.csv, movie.csv. Details of the content and usage of these files is given below.

Dataset Content and Usage

The dataset files are in csv format (comma-separated value) with a single header row. Columns containing commas (,) are escaped using double-quotes ("). These files are encoded as UTF-8.

User Ids

MovieLens users were selected at random for inclusion for inclusion. Their ids have been anonymized. User ids are consistent between ratings.csv and tags.csv files (i.e., the same id refers to the same user across the two files).

Movie Ids

Only movies with at least one rating or tag are included in the dataset. These movie ids are consistent with those used on the MovieLens web site (e.g., id 1 corresponds to the URL https://movielens.org/movies/1). Movie ids are consistent between files rating.csv, tag.csv, movie.csv, and link.csv (i.e., the same id refers to the same movie across these four data files).

Ratings Data File Structure (rating.csv)

All ratings are contained in the file rating.csv. Each line of this file after the header row represents one rating of one movie by one user, and has the following format:
userId,movieId,rating,timestamp

The lines within this file are ordered first by userId, then, within user, by movieId.

Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).

Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

Movies Data File Structure (movie.csv)

Movie information is contained in the file movie.csv. Each line of this file after the header row represents one movie, and has the following format:
movieId,title,genres

Movie titles are entered manually or imported from https://www.themoviedb.org, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles.

Genres are a pipe-separated list, and are selected from the following:

• Action • Adventure • Animation • Children's • Comedy • Crime • Documentary • Drama • Fantasy	• Film-Noir • Horror • Musical • Mystery • Romance • Sci-Fi • Thriller • War • Western • (no genres listed)

<u>Python Libraries</u>

1. Pandas
2. Numpy
3. Matplotlib
4. Seaborn

MovieLens Dataset

Team Members

Niti Tyagi(19522), Shefalika Ghosh(19544)

Context

The datasets describe ratings and free-text tagging activities from MovieLens, a movie recommendation service. It contains 20000263 ratings and 465564 tag applications across 27278 movies. These data were created by 138493 users between January 09, 1995 and March 31, 2015. This dataset was generated on October 17, 2016.

Users were selected at random for inclusion. All selected users rated at least 20 movies where each user is represented by an id - userId

Content

The data is contained in two files: movie.csv, rating.csv

1. **rating.csv** - contains ratings of movies by users:

- userId
- movieId
- rating
- timestamp

1. **movie.csv** - contains movie information:

- movieId
- title
- genres

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [25]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
movie = pd.read_csv("./MovieLens/movie.csv")
print(movie.columns)
movie.head(10)
```

```
Index(['movieId', 'title', 'genres'], dtype='object')
```

Out[3]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy

	movieId	title	genres
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 [5]:

```
movie.describe()
```

Out[5]:

	movieId
count	27278.000000
mean	59855.480570
std	44429.314697
min	1.000000
25%	6931.250000
50%	68068.000000
75%	100293.250000
max	131262.000000

In [4]:

```
rating = pd.read_csv("./MovieLens/rating.csv")
print(rating.columns)
```

```
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

DESCRIPTIVE ANALYSIS

In [5]:

```
rating['rating'].describe()
```

Out[5]:

```
count     2.000026e+07
mean      3.525529e+00
std       1.051989e+00
min       5.000000e-01
25%       3.000000e+00
50%       3.500000e+00
75%       4.000000e+00
max       5.000000e+00
Name: rating, dtype: float64
```

Minimum rating given to any movie: 0.5

Maximum rating given to any movie: 5.0

In [6]:

```
rating = rating.loc[:, ["userId", "movieId", "rating"]]
```

```
rating.head(10)
```

Out[6]:

	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5
2	1	32	3.5
3	1	47	3.5
4	1	50	3.5
5	1	112	3.5
6	1	151	4.0
7	1	223	4.0
8	1	253	4.0
9	1	260	4.0

In [7]:

```
data = pd.merge(movie,rating)
data.head(10)
```

Out[7]:

	movieId	title	genres	userId	rating
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	3	4.0
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	6	5.0
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	8	4.0
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	10	4.0
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	11	4.5
5	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	12	4.0
6	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	13	4.0
7	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	14	4.5
8	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	16	3.0
9	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	19	5.0

In [8]:

```
data.tail(10)
```

Out[8]:

	movieId	title	genres	userId	rating
20000253	131241	Ants in the Pants (2000)	Comedy|Romance	79570	4.0
20000254	131243	Werner - Gekotzt wird später (2003)	Animation|Comedy	79570	4.0
20000255	131248	Brother Bear 2 (2006)	Adventure|Animation|Children|Comedy|Fantasy	79570	4.0
20000256	131250	No More School (2000)	Comedy	79570	4.0
20000257	131252	Forklift Driver Klaus: The First Day on the Jo...	Comedy|Horror	79570	4.0
20000258	131254	Kein Bund für's Leben (2007)	Comedy	79570	4.0
20000259	131256	Feuer, Eis & Dosenbier (2002)	Comedy	79570	4.0
20000260	131258	The Pirates (2014)	Adventure	28906	2.5

	movieId	title	genres	userId	rating
20000261	131260	Rentun Ruusu (2001)	(no genres listed)	65409	3.0
20000262	131262	Innocence (2014)	Adventure|Fantasy|Horror	133047	4.0

In [9]:

```
print("\nTotal NaN at each column in the DataFrame :")
data.isnull().sum()
```

```
Total NaN at each column in the DataFrame :
```

Out[9]:

```
movieId    0
title      0
genres     0
userId     0
rating     0
dtype: int64
```

In [12]:

```
n = data.nunique(axis=0)
print("No.of.unique values in each column :\n", n)
```

```
No.of.unique values in each column :
 movieId     26744
title       26729
genres       1329
userId     138493
rating         10
dtype: int64
```

In [13]:

```
mid = data['movieId'].unique()
print("Unique movie ids in dataset: \n", mid)
```

```
Unique movie ids in dataset:
 [     1      2      3 ... 131258 131260 131262]
```

In [14]:

```
title = data['title'].unique()
print("All movies in dataset: \n", title)
uid = data['userId'].unique()
print("\n\nAll unique user ids in dataset: \n", uid)
```

```
All movies in dataset:
 ['Toy Story (1995)' 'Jumanji (1995)' 'Grumpier Old Men (1995)' ...
 'The Pirates (2014)' 'Rentun Ruusu (2001)' 'Innocence (2014)']


All unique user ids in dataset:
 [    3     6     8 ... 86872 90947 50542]
```

In [15]:

```
ranking = data['rating'].unique()
print("Unique ratings: \n", ranking)
```

```
Unique ratings:
 [4.  5.  4.5 3.  1.  3.5 1.5 2.  2.5 0.5]
```

In [16]:

```
#extracting a subset of columns from original dataset
cols_subset = data.loc[:, ['movieId', 'title', 'userId', 'rating']]
print("Movie dataset without genre column: \n")
cols_subset
```

```
Movie dataset without genre column:
```

Out[16]:

movieId	title	userId	rating

	movieId	title	userId	rating
0	1	Toy Story (1995)	3	4.0
1	1	Toy Story (1995)	6	5.0
2	1	Toy Story (1995)	8	4.0
3	1	Toy Story (1995)	10	4.0
4	1	Toy Story (1995)	11	4.5
...	...	...	...	...
20000258	131254	Kein Bund für's Leben (2007)	79570	4.0
20000259	131256	Feuer, Eis & Dosenbier (2002)	79570	4.0
20000260	131258	The Pirates (2014)	28906	2.5
20000261	131260	Rentun Ruusu (2001)	65409	3.0
20000262	131262	Innocence (2014)	133047	4.0

20000263 rows × 4 columns

1.Display all the ratings given to the movie "Toy Story (1995)" by different users and find average rating received by the movie

In [17]:

```
toyStory = data.loc[:, ['title', 'userId', 'rating']][data.title == 'Toy Story (1995)']
print(toyStory)
print("\nAverage rating received by movie Toy Story (1995):", round(toyStory.rating.mean(), 2))
```

```
                  title  userId  rating
0      Toy Story (1995)       3     4.0
1      Toy Story (1995)       6     5.0
2      Toy Story (1995)       8     4.0
3      Toy Story (1995)      10     4.0
4      Toy Story (1995)      11     4.5
...                 ...     ...     ...
49690  Toy Story (1995)  138483     4.0
49691  Toy Story (1995)  138486     5.0
49692  Toy Story (1995)  138488     3.0
49693  Toy Story (1995)  138491     2.0
49694  Toy Story (1995)  138493     3.5

[49695 rows x 3 columns]

Average rating received by movie Toy Story (1995): 3.92
```

2. Display all movie titles rated by user with userId '741'

In [18]:

```
cols_subset.loc[:, ['title', 'rating']][cols_subset.userId == 741]
```

Out[18]:

	title	rating
258	Toy Story (1995)	5.0
49818	Jumanji (1995)	3.0
72001	Grumpier Old Men (1995)	3.0
87500	Father of the Bride Part II (1995)	4.0
99705	Heat (1995)	3.5
...	...	...

	title	rating
18812161	Good Luck Chuck (2007)	0.5
18818718	Seeker: The Dark Is Rising, The (2007)	5.0
18824066	Elizabeth: The Golden Age (2007)	5.0
18831761	Reservation Road (2007)	4.5
18841602	Saw IV (2007)	0.5

2212 rows × 2 columns

Data cleaning and discretization

Cleaning:

- Extracting years from movie titles and creating a new column 'year_of_release'.
- Replacing missing data from column 'year_of_release' and filling it with valid values.
- Conversion of column 'year_of_release' to integer type for further numerical analysis.

Discretization

Binning of 'year_of_release' column.

Analysis:

- Finding which year range received highest average rating and hence list of movies released during that time period.
- Finding top 5 most popular genres based on average rating values.
- For movie with id-2 find the count for no. of users which have have given the movie a particular rating.

In [19]:

```
movie2 = movie.copy()
movie2.head(5)
```

Out[19]:

	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

Movie release year extraction and conversion to integer type

In [10]:

```
#movie release year extraction
movie['year_of_release'] = movie.title.str[-5:-1]

movie['year_of_release'] = movie['year_of_release'].replace(
['002)', '948)', '965)', 'lon ', '998)', 'piel', '010)', '008)', '929)','001)','poma','986)','007)',
['2002', '1948', '1965', '1993', '1998', '1970', '2010', '2008', '1929','2001','2010','1986','2007',

movie['year_of_release'] = movie['year_of_release'].astype(float)
movie['year_of_release'] = movie['year_of_release'].fillna(0)
```

```
movie['year_of_release'] = movie['year_of_release'].astype(int)
movie.tail(4)
```

Out[10]:

	movieId	title	genres	year_of_release
27274	131256	Feuer, Eis & Dosenbier (2002)	Comedy	2002
27275	131258	The Pirates (2014)	Adventure	2014
27276	131260	Rentun Ruusu (2001)	(no genres listed)	2001
27277	131262	Innocence (2014)	Adventure|Fantasy|Horror	2014

Binning of 'year_of_release' column

In [11]:

```
movie['year_of_release'].describe()
```

Out[11]:

```
count    27278.000000
mean      1989.381516
std         23.333149
min       1891.000000
25%       1976.000000
50%       1998.000000
75%       2008.000000
max       2015.000000
Name: year_of_release, dtype: float64
```

In [12]:

```
bins = [1890,1910,1930,1950,1970,1990,2010,2015]
movie['year_bins'] = pd.cut(movie['year_of_release'], bins)
movie.head(8)
```

Out[12]:

	movieId	title	genres	year_of_release	year_bins
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	(1990, 2010]
1	2	Jumanji (1995)	Adventure|Children|Fantasy	1995	(1990, 2010]
2	3	Grumpier Old Men (1995)	Comedy|Romance	1995	(1990, 2010]
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	1995	(1990, 2010]
4	5	Father of the Bride Part II (1995)	Comedy	1995	(1990, 2010]
5	6	Heat (1995)	Action|Crime|Thriller	1995	(1990, 2010]
6	7	Sabrina (1995)	Comedy|Romance	1995	(1990, 2010]
7	8	Tom and Huck (1995)	Adventure|Children	1995	(1990, 2010]

In [13]:

```
movie.tail(10)
```

Out[13]:

	movieId	title	genres	year_of_release	year_bins
27268	131241	Ants in the Pants (2000)	Comedy|Romance	2000	(1990, 2010]
27269	131243	Werner - Gekotzt wird später (2003)	Animation|Comedy	2003	(1990, 2010]
27270	131248	Brother Bear 2 (2006)	Adventure|Animation|Children|Comedy|Fantasy	2006	(1990, 2010]
27271	131250	No More School (2000)	Comedy	2000	(1990, 2010]

	movieId	title	genres	year_of_release	year_bins
27272	131252	Forklift Driver Klaus: The First Day on the Jo...	Comedy|Horror	2001	(1990, 2010]
27273	131254	Kein Bund für's Leben (2007)	Comedy	2007	(1990, 2010]
27274	131256	Feuer, Eis & Dosenbier (2002)	Comedy	2002	(1990, 2010]
27275	131258	The Pirates (2014)	Adventure	2014	(2010, 2015]
27276	131260	Rentun Ruusu (2001)	(no genres listed)	2001	(1990, 2010]
27277	131262	Innocence (2014)	Adventure|Fantasy|Horror	2014	(2010, 2015]

In [14]:

```
movie[1000:1110]
```

Out[14]:

	movieId	title	genres	year_of_release	year_bins
1000	1019	20,000 Leagues Under the Sea (1954)	Adventure|Drama|Sci-Fi	1954	(1950, 1970]
1001	1020	Cool Runnings (1993)	Comedy	1993	(1990, 2010]
1002	1021	Angels in the Outfield (1994)	Children|Comedy	1994	(1990, 2010]
1003	1022	Cinderella (1950)	Animation|Children|Fantasy|Musical|Romance	1950	(1930, 1950]
1004	1023	Winnie the Pooh and the Blustery Day (1968)	Animation|Children|Musical	1968	(1950, 1970]
...	...	...	...	...	...
1105	1128	Fog, The (1980)	Horror	1980	(1970, 1990]
1106	1129	Escape from New York (1981)	Action|Adventure|Sci-Fi|Thriller	1981	(1970, 1990]
1107	1130	Howling, The (1980)	Horror|Mystery	1980	(1970, 1990]
1108	1131	Jean de Florette (1986)	Drama|Mystery	1986	(1970, 1990]
1109	1132	Manon of the Spring (Manon des sources) (1986)	Drama	1986	(1970, 1990]

110 rows × 5 columns

Total no. of movies released in each of the 7 time frames

In [16]:

```
movie.groupby(movie['year_bins']).count()['movieId']
```

Out[16]:

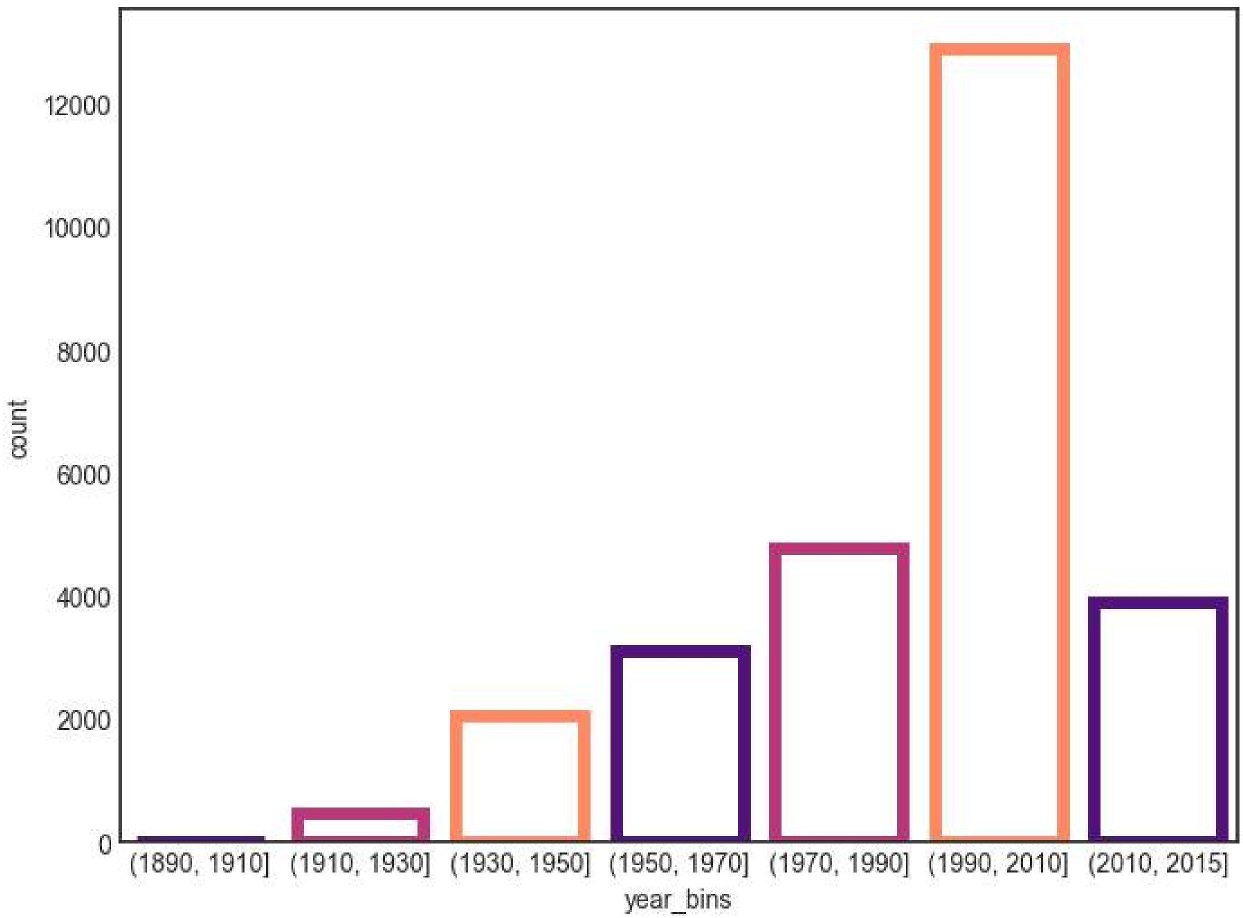
```
year_bins
(1890, 1910]      25
(1910, 1930]     472
(1930, 1950]    2059
```

```
(1950, 1970]     3114
(1970, 1990]     4787
(1990, 2010]    12902
(2010, 2015]     3919
Name: movieId, dtype: int64
```

In [21]:

```
plt.figure(figsize = (8, 6))
plt.style.use('seaborn-white')

ax=sns.countplot(x="year_bins", data=movie, facecolor=(0, 0, 0, 0),
                 linewidth=5,edgecolor=sns.color_palette("magma", 3))
plt.show()
```



Maximum no. of movies are released in the time frame : 1990-2010

EXPLORATORY ANALYSIS

In [22]:

```
rating2 = rating.loc[:, ["movieId", "rating"]]
avg_rating = rating2.groupby('movieId').mean().round(2).reset_index()
avg_rating.head()
```

Out[22]:

	movieId	rating
0	1	3.92
1	2	3.21
2	3	3.15
3	4	2.86

	movieId	rating
4	5	3.06

In [23]:

```
movie_rating = pd.merge(movie,avg_rating)
movie_rating.head(6)
```

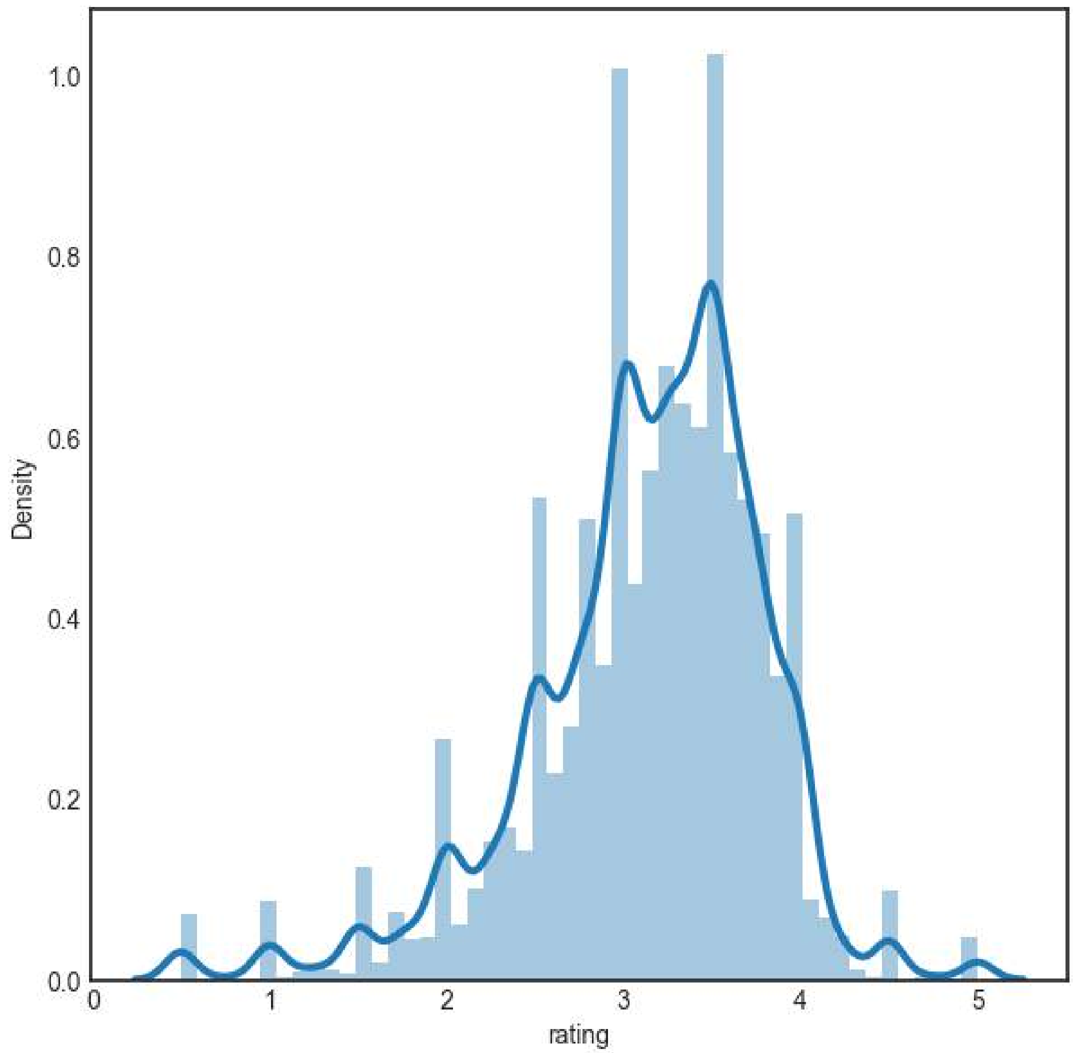
Out[23]:

	movieId	title	genres	year_of_release	year_bins	rating
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	(1990, 2010]	3.92
1	2	Jumanji (1995)	Adventure|Children|Fantasy	1995	(1990, 2010]	3.21
2	3	Grumpier Old Men (1995)	Comedy|Romance	1995	(1990, 2010]	3.15
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	1995	(1990, 2010]	2.86
4	5	Father of the Bride Part II (1995)	Comedy	1995	(1990, 2010]	3.06
5	6	Heat (1995)	Action|Crime|Thriller	1995	(1990, 2010]	3.83

Density Distribution Plot of average_rating

In [26]:

```
plt.figure(figsize = (7, 7))
sns.distplot(movie_rating['rating'],kde=True,kde_kws = {'linewidth': 3}, hist = True).set(ylabel='De
plt.show()
```



The density distribution plot above shows that maximum movies received an average rating between 3 and 4. The bell shaped curve represents a normal distribution.

Categorical column (Remarks)

Let's create a categorical column based on rating (avg_rating)of movies:

1. For ratings between (0,1], the movie is given the remark 'Super-flop'
2. For ratings between (1,2], the movie is given the remark 'Flop'
3. For ratings between (2,3], the movie is given the remark 'Hit'
4. For ratings between (3,4], the movie is given the remark 'Superhit'
5. For ratings between (4,5], the movie is given the remark 'Blockbuster'

In [27]:

```
def create_cat(i):
    if i >= 0 and i <=1:
        return 'Super-flop'
    if i > 1 and i <=2:
        return 'Flop'
    if i > 2 and i <=3:
        return 'Hit'
    if i > 3 and i <=4:
        return 'Superhit'
```

```
    if i > 4 and i <=5:
        return 'Blockbuster'
movie_rating['Remarks'] = movie_rating['rating'].apply(create_cat)
```

In [28]:

```
movie_rating.head(10)
```

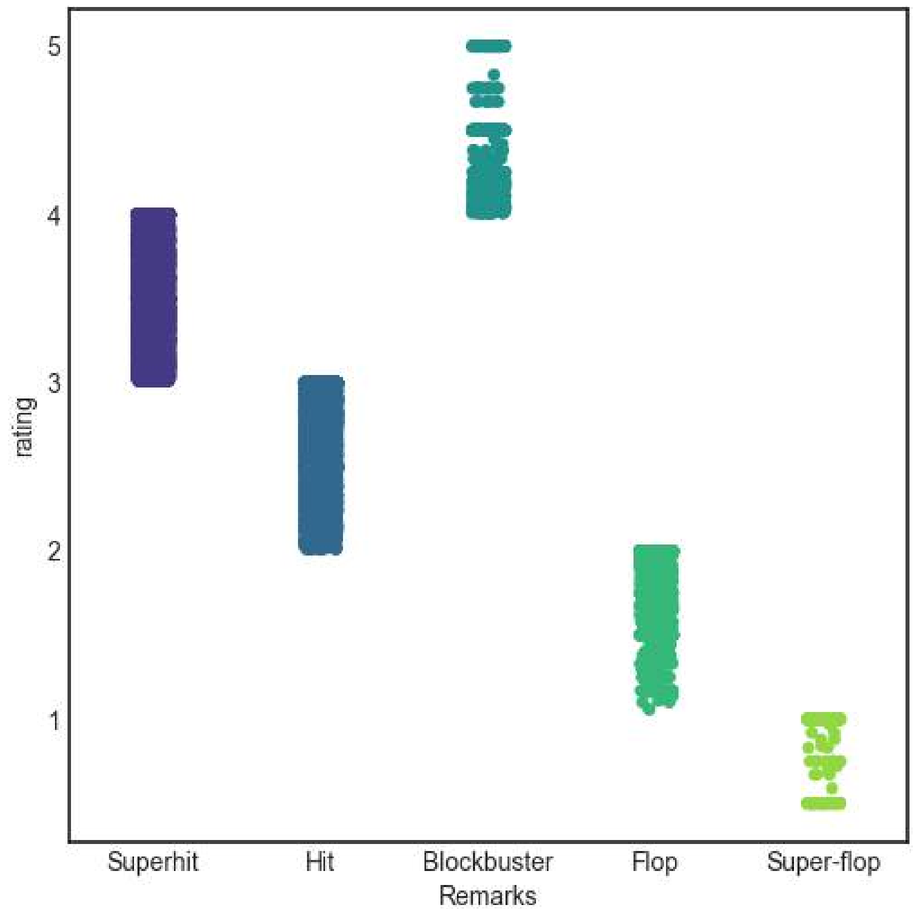
Out[28]:

	movieId	title	genres	year_of_release	year_bins	rating	Remarks
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	(1990, 2010]	3.92	Superhit
1	2	Jumanji (1995)	Adventure|Children|Fantasy	1995	(1990, 2010]	3.21	Superhit
2	3	Grumpier Old Men (1995)	Comedy|Romance	1995	(1990, 2010]	3.15	Superhit
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	1995	(1990, 2010]	2.86	Hit
4	5	Father of the Bride Part II (1995)	Comedy	1995	(1990, 2010]	3.06	Superhit
5	6	Heat (1995)	Action|Crime|Thriller	1995	(1990, 2010]	3.83	Superhit
6	7	Sabrina (1995)	Comedy|Romance	1995	(1990, 2010]	3.37	Superhit
7	8	Tom and Huck (1995)	Adventure|Children	1995	(1990, 2010]	3.14	Superhit
8	9	Sudden Death (1995)	Action	1995	(1990, 2010]	3.00	Hit
9	10	GoldenEye (1995)	Action|Adventure|Thriller	1995	(1990, 2010]	3.43	Superhit

Visualisation of categorical column(Remarks)

In [32]:

```
fig = plt.figure(figsize=(6,6))
sns.stripplot(x="Remarks",y="rating",data=movie_rating,palette="viridis")
plt.show()
```



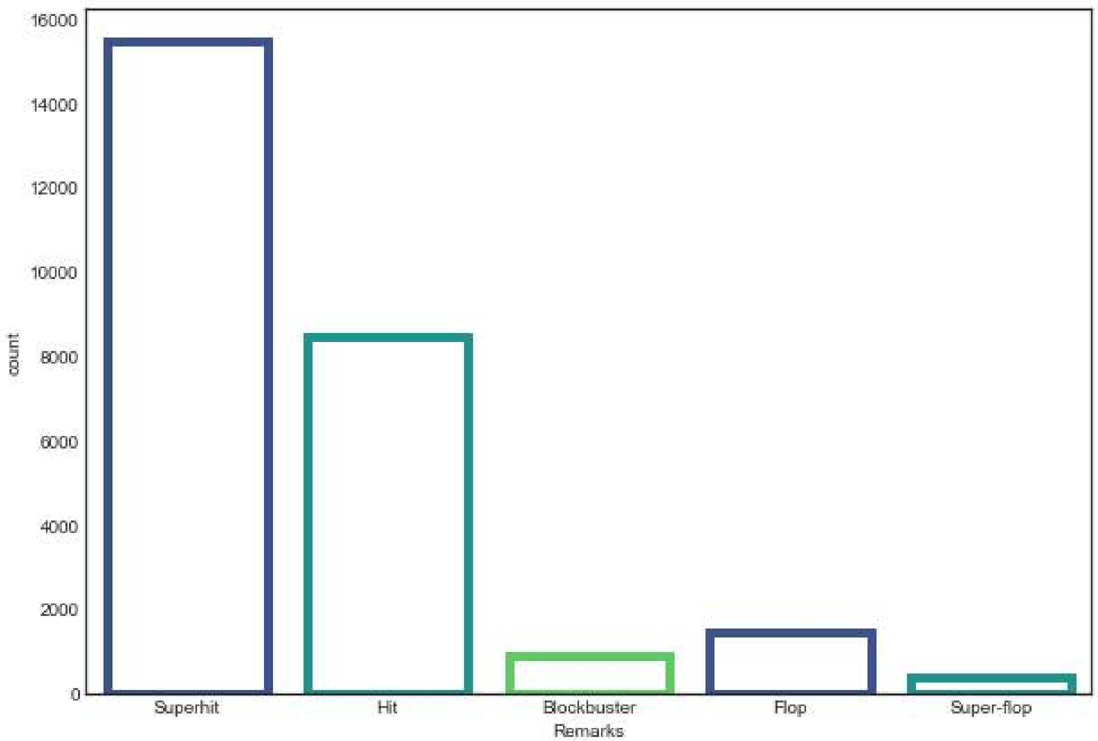
Number of movies based on each remark

In [108…

```
plt.figure(figsize = (10, 7))
plt.style.use('seaborn-white')
sns.countplot(x='Remarks', data=movie_rating, facecolor=(0, 0, 0, 0),
                 linewidth=5,edgecolor=sns.color_palette("viridis", 3))
```

Out[108…

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fa04c81388>
```



Our dataset has maximum no. of Super-Hit movies (movies which have received rating between 3 and 4)

Finding no. of movies released in each genre

In [33]:

```
gen =movie_rating['genres'].value_counts()
print(gen,"\n")
```

```
Drama                                 4416
Comedy                                2251
Documentary                           1879
Comedy|Drama                          1241
Drama|Romance                         1043
                                      ...
Animation|Documentary|War                1
Action|Crime|Drama|Western               1
Action|Comedy|Thriller|Western           1
IMAX                                     1
Animation|Children|Comedy|Western        1
Name: genres, Length: 1329, dtype: int64
```

Top 10 most commonly watched genres.

In [34]:

```
gen[0:10]
```

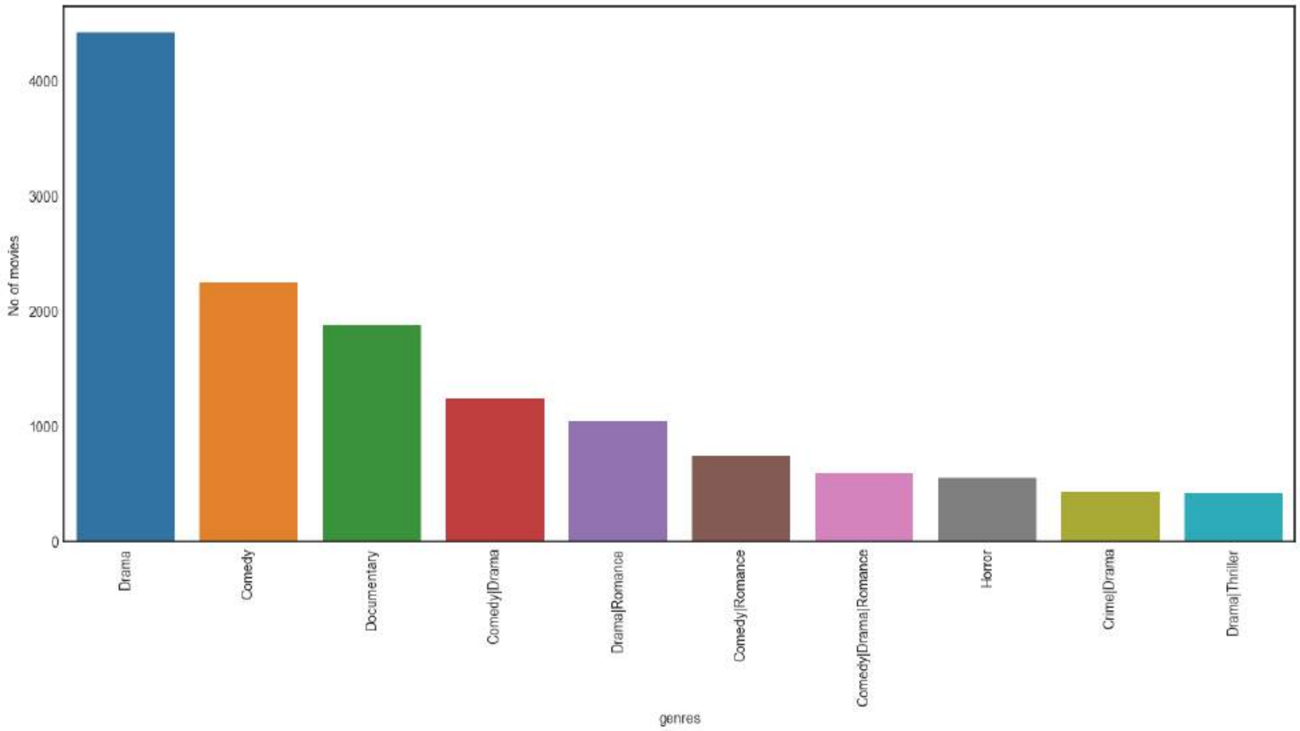
Out[34]:

```
Drama                    4416
Comedy                   2251
Documentary              1879
Comedy|Drama             1241
Drama|Romance            1043
Comedy|Romance            741
Comedy|Drama|Romance      594
Horror                    556
Crime|Drama               435
```

```
Drama|Thriller          421
Name: genres, dtype: int64
```

In [35]:

```
plt.figure(figsize=(16,7))
sns.barplot(x=gen.index[0:10],y=gen[0:10])
plt.xlabel('genres')
plt.ylabel('No of movies')
plt.xticks(rotation=90)
plt.show()
```



Maximum no. of movies are released in genre :- Drama

Creating subset of top 5 most commonly watched genres.

In [36]:

```
sub_set=movie_rating[movie_rating['genres'].isin(['Drama','Comedy','Documentary','Comedy|Drama','Dra
sub_set.head(10)
```

Out[36]:

	movieId	title	genres	year_of_release	year_bins	rating	Remarks
4	5	Father of the Bride Part II (1995)	Comedy	1995	(1990, 2010]	3.06	Superhit
13	14	Nixon (1995)	Drama	1995	(1990, 2010]	3.43	Superhit
16	17	Sense and Sensibility (1995)	Drama|Romance	1995	(1990, 2010]	3.97	Superhit
17	18	Four Rooms (1995)	Comedy	1995	(1990, 2010]	3.37	Superhit
18	19	Ace Ventura: When Nature Calls (1995)	Comedy	1995	(1990, 2010]	2.61	Hit
24	25	Leaving Las Vegas (1995)	Drama|Romance	1995	(1990, 2010]	3.69	Superhit
25	26	Othello (1995)	Drama	1995	(1990, 2010]	3.63	Superhit
27	28	Persuasion (1995)	Drama|Romance	1995	(1990, 2010]	4.06	Blockbuster
30	31	Dangerous Minds (1995)	Drama	1995	(1990, 2010]	3.25	Superhit
34	35	Carrington (1995)	Drama|Romance	1995	(1990, 2010]	3.50	Superhit

Multindex

For the 5 most commonly watched genres find the no. of movies released remark wise

In [37]:

```
sub_set.groupby(['genres'])['Remarks'].value_counts().to_frame()
```

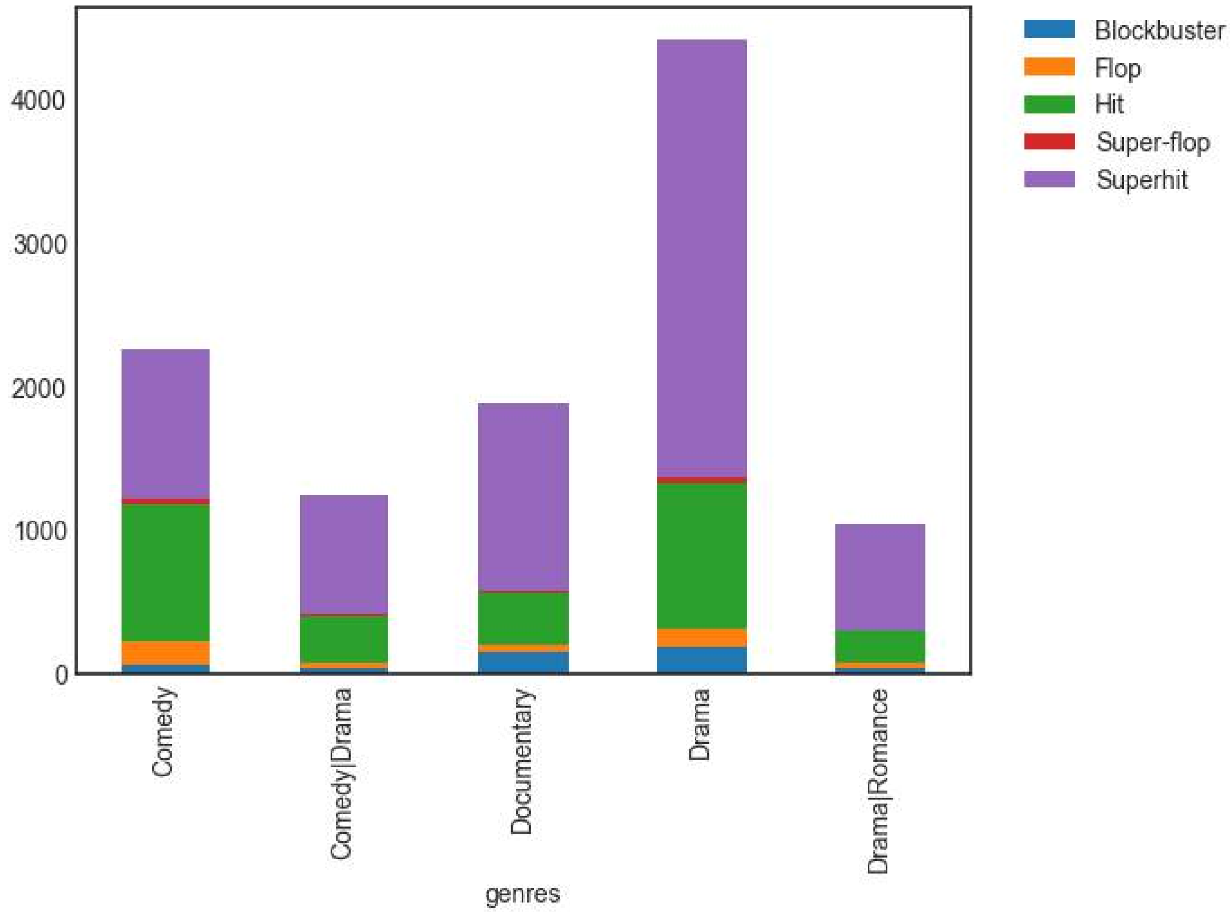
Out[37]:

		Remarks
genres	**Remarks**	
Comedy	**Superhit**	1036
	Hit	952
	Flop	162
	Blockbuster	63
	Super-flop	38
Comedy|Drama	**Superhit**	832
	Hit	330
	Blockbuster	39
	Flop	32
	Super-flop	8
Documentary	**Superhit**	1304
	Hit	363
	Blockbuster	148
	Flop	54
	Super-flop	10
Drama	**Superhit**	3051
	Hit	1008
	Blockbuster	182
	Flop	131
	Super-flop	44
Drama|Romance	**Superhit**	744
	Hit	225
	Flop	39
	Blockbuster	28
	Super-flop	7

In [39]:

```
plt.figure(figsize=(16,8))
sub_set.groupby(["genres","Remarks"]).size().unstack().plot(kind='bar',stacked=True)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.show()
```

<Figure size 1600x800 with 0 Axes>



Maximum no. of superhit and blockbuster movies are of the genre Drama

Plot the temporal trends of the top 5 most commonly watched genre

In [40]:

```
# Drama, Comedy, Documentary, Comedy|Drama, Drama|Romance

frame1 = movie_rating[(movie_rating['year_of_release']>=1890) & (movie_rating['year_of_release']<=19
                      (movie_rating['genres'].isin(['Drama','Comedy','Documentary','Comedy|Drama','D
count1 = frame1['genres'].value_counts()
c1 = dict(count1)

frame2 = movie_rating[(movie_rating['year_of_release']>=1910) & (movie_rating['year_of_release']<=19
                      (movie_rating['genres'].isin(['Drama','Comedy','Documentary','Comedy|Drama','D
count2 = frame2['genres'].value_counts()
c2 = dict(count2)

frame3 = movie_rating[(movie_rating['year_of_release']>=1930) & (movie_rating['year_of_release']<=19
                      (movie_rating['genres'].isin(['Drama','Comedy','Documentary','Comedy|Drama','D
count3 = frame3['genres'].value_counts()
c3 = dict(count3)

frame4 = movie_rating[(movie_rating['year_of_release']>=1950) & (movie_rating['year_of_release']<=19
                      (movie_rating['genres'].isin(['Drama','Comedy','Documentary','Comedy|Drama','D
count4 = frame4['genres'].value_counts()
c4 = dict(count4)

frame5 = movie_rating[(movie_rating['year_of_release']>=1970) & (movie_rating['year_of_release']<=19
                      (movie_rating['genres'].isin(['Drama','Comedy','Documentary','Comedy|Drama','D
count5 = frame5['genres'].value_counts()
c5 = dict(count5)
```

```
frame6 = movie_rating[(movie_rating['year_of_release']>=1990) & (movie_rating['year_of_release']<=20
                      (movie_rating['genres'].isin(['Drama','Comedy','Documentary','Comedy|Drama','D
count6 = frame6['genres'].value_counts()
c6 = dict(count6)

frame7 = movie_rating[(movie_rating['year_of_release']>=2010) & (movie_rating['year_of_release']<=20
                      (movie_rating['genres'].isin(['Drama','Comedy','Documentary','Comedy|Drama','D
count7 = frame7['genres'].value_counts()
c7 = dict(count7)
```

In [41]:

```
data = {
    '1890-1910': c1,
    '1910-1930':c2,
    '1930-1950':c3,
    '1950-1970':c4,
    '1970-1990':c5,
    '1990-2010':c6,
    '2010-2015':c7,
}
genre_trends = pd.DataFrame(data)
genre_trends
```

Out[41]:

	1890-1910	1910-1930	1930-1950	1950-1970	1970-1990	1990-2010	2010-2015
Documentary	5.0	12	14	60	144	1141	656
Comedy	2.0	78	108	217	458	1097	403
Drama	1.0	87	305	586	774	2176	755
Drama|Romance	NaN	34	123	115	142	540	133
Comedy|Drama	NaN	14	54	80	190	725	258

In [42]:

```
genre_trends = genre_trends.reindex(index = ['Drama','Comedy','Documentary','Comedy|Drama','Drama|Ro
genre_trends = genre_trends.rename_axis('Genre').reset_index()
```

In [43]:

```
gen = pd.melt(genre_trends,
        id_vars='Genre',
        value_vars=['1890-1910','1910-1930','1930-1950','1950-1970','1970-1990','1990-2010','2010-201
        var_name='Year_Bins',
        value_name='Frequency')
gen
```

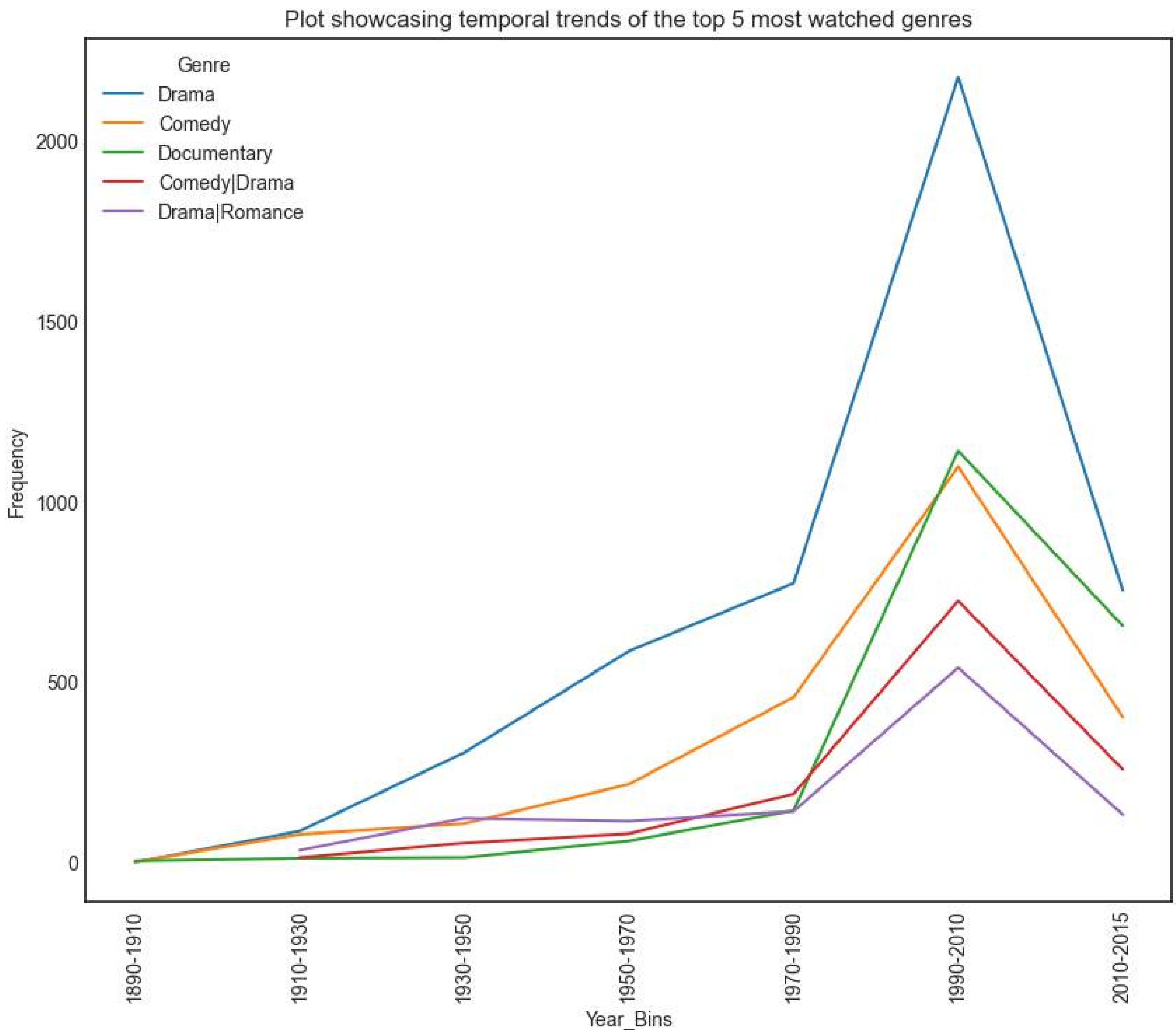
Out[43]:

	Genre	Year_Bins	Frequency
0	Drama	1890-1910	1.0
1	Comedy	1890-1910	2.0
2	Documentary	1890-1910	5.0
3	Comedy|Drama	1890-1910	NaN
4	Drama|Romance	1890-1910	NaN
5	Drama	1910-1930	87.0
6	Comedy	1910-1930	78.0
7	Documentary	1910-1930	12.0
8	Comedy|Drama	1910-1930	14.0

	Genre	Year_Bins	Frequency
9	Drama|Romance	1910-1930	34.0
10	Drama	1930-1950	305.0
11	Comedy	1930-1950	108.0
12	Documentary	1930-1950	14.0
13	Comedy|Drama	1930-1950	54.0
14	Drama|Romance	1930-1950	123.0
15	Drama	1950-1970	586.0
16	Comedy	1950-1970	217.0
17	Documentary	1950-1970	60.0
18	Comedy|Drama	1950-1970	80.0
19	Drama|Romance	1950-1970	115.0
20	Drama	1970-1990	774.0
21	Comedy	1970-1990	458.0
22	Documentary	1970-1990	144.0
23	Comedy|Drama	1970-1990	190.0
24	Drama|Romance	1970-1990	142.0
25	Drama	1990-2010	2176.0
26	Comedy	1990-2010	1097.0
27	Documentary	1990-2010	1141.0
28	Comedy|Drama	1990-2010	725.0
29	Drama|Romance	1990-2010	540.0
30	Drama	2010-2015	755.0
31	Comedy	2010-2015	403.0
32	Documentary	2010-2015	656.0
33	Comedy|Drama	2010-2015	258.0
34	Drama|Romance	2010-2015	133.0

In [44]:

```
fig_dims = (10, 8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.lineplot(x = "Year_Bins", y = "Frequency", data = gen, hue = "Genre", ax=ax)
plt.xticks(rotation=90)
plt.title('Plot showcasing temporal trends of the top 5 most watched genres')
plt.show()
```



Temporal trends of the top 5 most commonly watched genre:-

1- ***Drama*** has maintained its position as the most watched genre from beginning till the end.
2- ***Documentry*** genre started gaining popularity from the year 1970 (notice the sharp rise) reached a peak from 1990-2010 going head to head with ***Comedy***. However, after year 2010, it started losing popularity once again.
3- It is also interesting to note that the ***Romance*** genre did not make an appearance until the year 1910.

Which time frame has released the movies with highest average rating?

In [45]:

```
#mean rating for a time period
popular_year = movie_rating.groupby('year_bins').mean().round(2).reset_index()
popular_year = popular_year.loc[:, ["year_bins", "rating"]]
popular_year
```

Out[45]:

	year_bins	rating
0	(1890, 1910]	3.36
1	(1910, 1930]	3.28
2	(1930, 1950]	3.21

	year_bins	rating
3	(1950, 1970]	3.22
4	(1970, 1990]	3.10
5	(1990, 2010]	3.11
6	(2010, 2015]	3.13

In [46]:

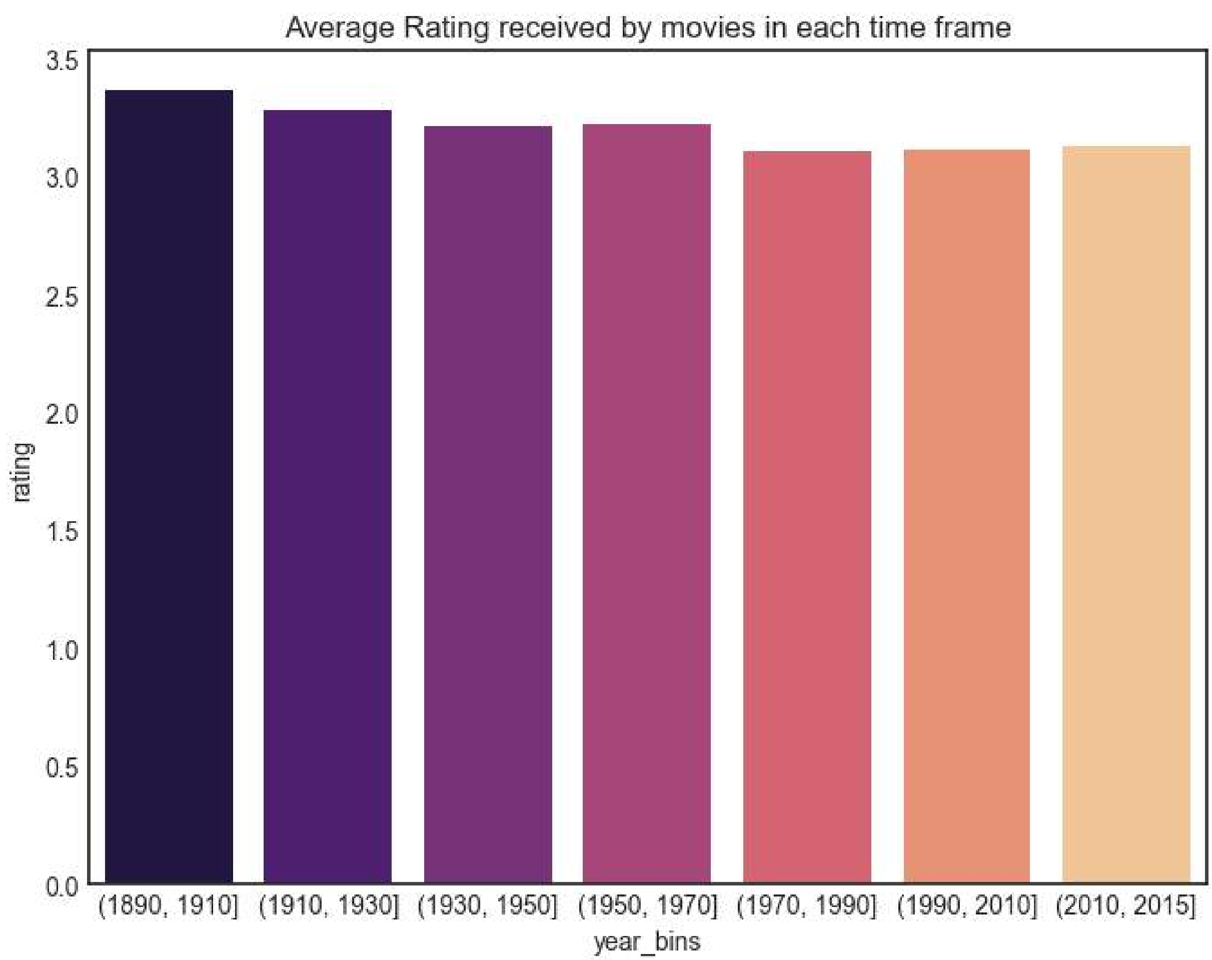
```
maxrating = popular_year['rating'].max()
rslt_df = popular_year[popular_year['rating'] == maxrating]
rslt_df
```

Out[46]:

	year_bins	rating
0	(1890, 1910]	3.36

In [48]:

```
fig = plt.figure(figsize=(8, 6))
sns.barplot(y=popular_year['rating'], x=popular_year['year_bins'], palette="magma")
plt.title('Average Rating received by movies in each time frame')
plt.show()
```



Hence, we find that year 1890-1910 has the highest average rating for movies

List of movies released during 1890-1910 (period that has received the highest average rating)

In [49]:
```
movie90_10=movie_rating[(movie_rating['year_of_release'] > 1890) & (movie_rating['year_of_release']
movie90_10
```

Out[49]:

	movieId	title	genres	year_of_release	year_bins	rating	Remarks
9995	32898	Trip to the Moon, A (Voyage dans la lune, Le) ...	Action|Adventure|Fantasy|Sci-Fi	1902	(1890, 1910]	3.74	Superhit
11460	49389	Great Train Robbery, The (1903)	Crime|Western	1903	(1890, 1910]	3.34	Superhit
16289	82337	Four Heads Are Better Than One (Un homme de tê...	Fantasy	1898	(1890, 1910]	3.75	Superhit
16294	82362	Pyramid of Triboulet, The (La pyramide de Trib...	Fantasy	1899	(1890, 1910]	3.62	Superhit
17597	88674	Edison Kinetoscopic Record of a Sneeze (1894)	Documentary	1894	(1890, 1910]	2.71	Hit
18656	93162	Moscow Clad in Snow (Moscou sous la neige) (1909)	Documentary	1909	(1890, 1910]	2.50	Hit
18815	93865	Frankenstein (1910)	Drama|Horror|Sci-Fi	1910	(1890, 1910]	3.47	Superhit
18934	94431	Ella Lola, a la Trilby (1898)	(no genres listed)	1898	(1890, 1910]	5.00	Blockbuster
18964	94657	Turkish Dance, Ella Lola (1898)	(no genres listed)	1898	(1890, 1910]	5.00	Blockbuster
18980	94737	Boys Diving, Honolulu (1901)	Documentary	1901	(1890, 1910]	5.00	Blockbuster
19032	94951	Dickson Experimental Sound Film (1894)	Musical	1894	(1890, 1910]	3.43	Superhit
19160	95541	Blacksmith Scene (1893)	(no genres listed)	1893	(1890, 1910]	3.38	Superhit
19265	96009	Kiss, The (1896)	Romance	1896	(1890, 1910]	2.93	Hit
20023	98981	Arrival of a Train, The (1896)	Documentary	1896	(1890, 1910]	3.44	Superhit
21821	105776	Trip to Mars, A (1910)	Sci-Fi	1910	(1890, 1910]	2.50	Hit
22808	109524	Woman Always Pays, The (Afgrunden) (Abyss, The...	Drama	1910	(1890, 1910]	4.00	Superhit
23633	113048	Tables Turned on the Gardener (1895)	Comedy	1895	(1890, 1910]	2.25	Hit
23948	114371	Lonely Villa, The (1909)	Crime|Drama	1909	(1890, 1910]	3.00	Hit
24697	117909	The Kiss (1900)	Romance	1900	(1890, 1910]	3.17	Superhit
25164	120803	Those Awful Hats (1909)	Comedy	1909	(1890, 1910]	2.75	Hit
25193	120869	Employees Leaving the Lumière Factory (1895)	Documentary	1895	(1890, 1910]	4.00	Superhit

	movieId	title	genres	year_of_release	year_bins	rating	Remarks
25724	125978	Santa Claus (1898)	Sci-Fi	1898	(1890, 1910]	2.50	Hit
25733	125996	The Black Devil (1905)	Comedy|Fantasy	1905	(1890, 1910]	2.50	Hit
26481	129849	Old Man Drinking a Glass of Beer (1898)	(no genres listed)	1898	(1890, 1910]	3.00	Hit
26482	129851	Dickson Greeting (1891)	(no genres listed)	1891	(1890, 1910]	3.00	Hit

Finding count of movies in most watched genres for the time frame 1890-1910

In [50]:

```
data1=movie90_10.loc[(movie90_10['genres']=='Drama') |(movie90_10['genres']=='Comedy')|(movie90_10['
data1
```

Out[50]:

	movieId	title	genres	year_of_release	year_bins	rating	Remarks
17597	88674	Edison Kinetoscopic Record of a Sneeze (1894)	Documentary	1894	(1890, 1910]	2.71	Hit
18656	93162	Moscow Clad in Snow (Moscou sous la neige) (1909)	Documentary	1909	(1890, 1910]	2.50	Hit
18980	94737	Boys Diving, Honolulu (1901)	Documentary	1901	(1890, 1910]	5.00	Blockbuster
20023	98981	Arrival of a Train, The (1896)	Documentary	1896	(1890, 1910]	3.44	Superhit
22808	109524	Woman Always Pays, The (Afgrunden) (Abyss, The...	Drama	1910	(1890, 1910]	4.00	Superhit
23633	113048	Tables Turned on the Gardener (1895)	Comedy	1895	(1890, 1910]	2.25	Hit
25164	120803	Those Awful Hats (1909)	Comedy	1909	(1890, 1910]	2.75	Hit
25193	120869	Employees Leaving the Lumière Factory (1895)	Documentary	1895	(1890, 1910]	4.00	Superhit

In [103…

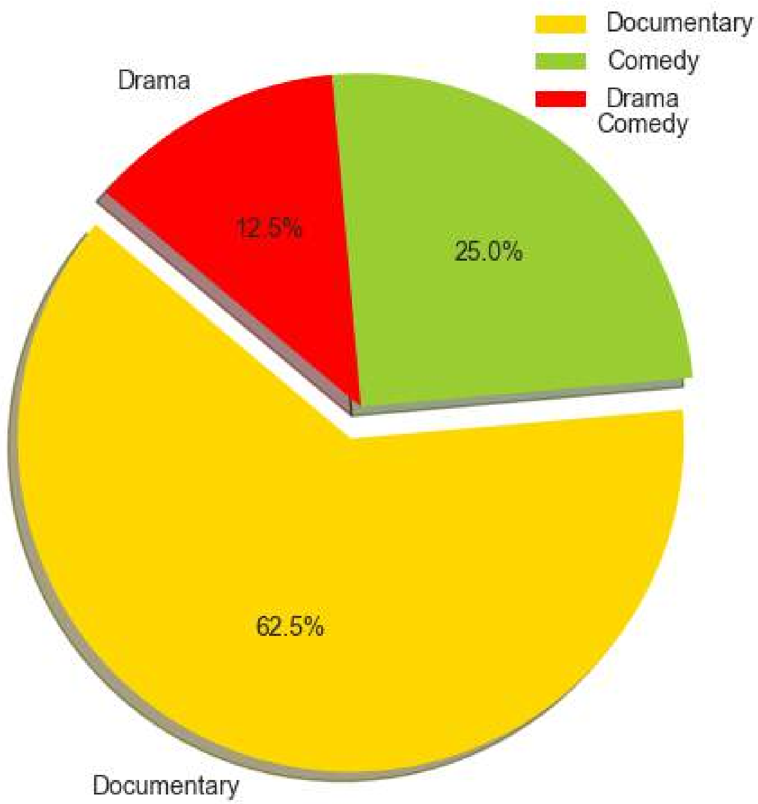
```
data1.groupby('genres')['genres'].count()
```

Out[103…

```
genres
Comedy         2
Documentary    5
Drama          1
Name: genres, dtype: int64
```

In [52]:

```
plt.figure(figsize=(6,6))
colors=['gold', 'yellowgreen','red']
explode = (0.1, 0,0)
plt.pie(list(data1['genres'].value_counts()),labels=list(data1['genres'].value_counts().keys()),autc
        colors=colors,shadow=True,explode= explode,startangle=140)
plt.legend(loc="upper right")
plt.show()
```

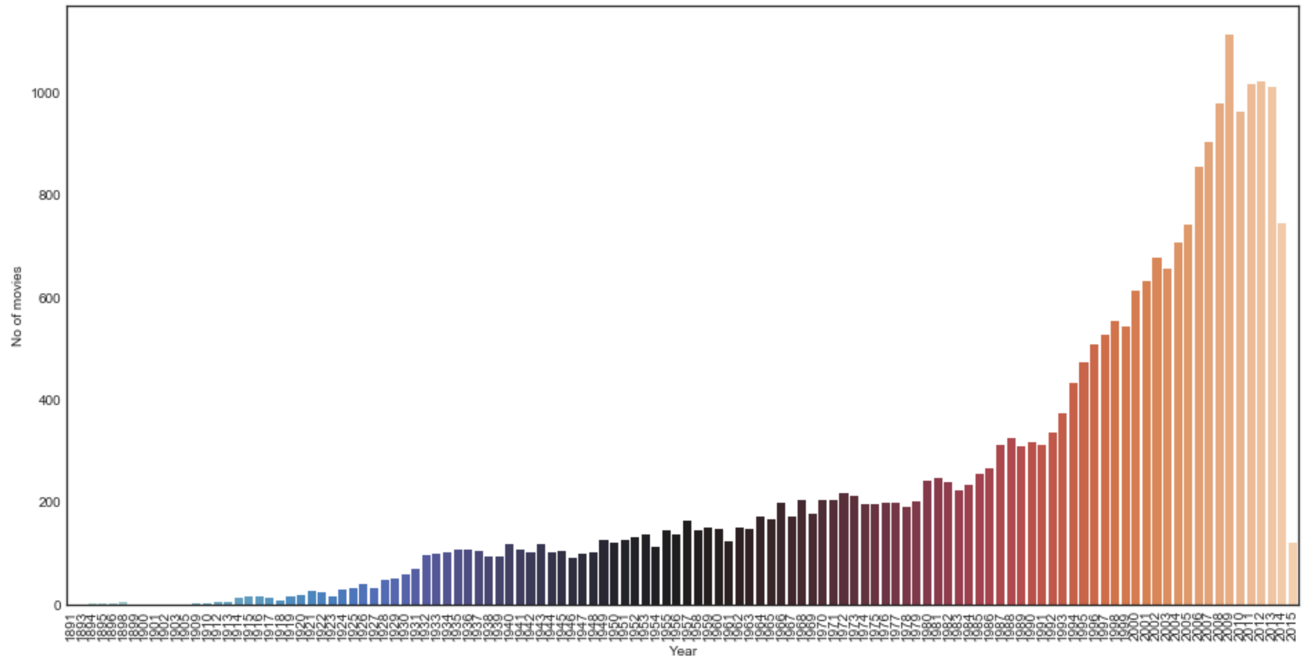


Majority of movies released during 1890-1910 are from genre Documentary

No.of movies Vs Year of release

In [110…

```
plt.figure(figsize=(16,8))
sns.barplot(x=year.index,y=year,palette='icefire')
plt.xticks(rotation=90)
plt.xlabel('Year')
plt.ylabel('No of movies')
plt.show()
```



Which year has the highest number of movie releases ?

In [54]:

```
print("No. of unique years: ",movie.year_of_release.nunique())
year=movie['year_of_release'].value_counts()
print("\nYears with movie release count:")
print(year)
```

```
No. of unique years:  118

Years with movie release count:
2009    1114
2012    1022
2011    1017
2013    1012
2008     980
        ...
1893       1
1901       1
1903       1
1902       1
1891       1
Name: year_of_release, Length: 118, dtype: int64
```

In [55]:

```
print(year.idxmax)
```

```
<bound method Series.idxmax of 2009    1114
2012    1022
2011    1017
2013    1012
2008     980
        ...
1893       1
1901       1
1903       1
1902       1
1891       1
Name: year_of_release, Length: 118, dtype: int64>
```

Maximum number of movies are released in the year 2009 i.e. 1114

Which year has released maximum no. of blockbusters?

In [56]:
```
fam_yr = movie_rating.loc[(movie_rating['Remarks']=='Blockbuster')]
fam_yr
```

Out[56]:

	movieId	title	genres	year_of_release	year_bins	rating	Remarks
27	28	Persuasion (1995)	Drama|Romance	1995	(1990, 2010]	4.06	Blockbuster
46	47	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller	1995	(1990, 2010]	4.05	Blockbuster
49	50	Usual Suspects, The (1995)	Crime|Mystery|Thriller	1995	(1990, 2010]	4.33	Blockbuster
108	110	Braveheart (1995)	Action|Drama|War	1995	(1990, 2010]	4.04	Blockbuster
109	111	Taxi Driver (1976)	Crime|Drama|Thriller	1976	(1970, 1990]	4.11	Blockbuster
...	...	...	...	...	...	...	...
26655	130996	The Beautiful Story (1992)	Adventure|Drama|Fantasy	1992	(1990, 2010]	5.00	Blockbuster
26665	131027	But Forever in My Mind (1999)	Comedy|Drama	1999	(1990, 2010]	4.50	Blockbuster
26667	131050	Stargate SG-1 Children of the Gods - Final Cut...	Adventure|Sci-Fi|Thriller	2009	(1990, 2010]	5.00	Blockbuster
26682	131082	Playground (2009)	(no genres listed)	2009	(1990, 2010]	4.50	Blockbuster
26729	131176	A Second Chance (2014)	Drama	2014	(2010, 2015]	4.50	Blockbuster

911 rows × 7 columns

In [57]:
```
c=fam_yr['year_of_release'].value_counts()
c
```

Out[57]:
```
2009    43
2012    37
2013    36
2011    35
2008    32
        ..
1928     2
1924     1
1901     1
1932     1
1927     1
Name: year_of_release, Length: 93, dtype: int64
```

Maximum no. of blockbusters are released in year 2009 i.e. 43

It can be seen that 2009 is the most popular year. The reason is twofold:

1- 2009 is the year with highest no. of movie releases- 1114.

2- We also witness the maximum number of blockbuster movie releases (43 blockbuster movies) in 2009.

List of movies released in the year 2009

In [58]:
```
movie2009=movie_rating[(movie_rating['year_of_release'] == 2009)]
movie2009
```

Out[58]:

	movieId	title	genres	year_of_release	year_bins	rating	Remarks
12846	60684	Watchmen (2009)	Action|Drama|Mystery|Sci-Fi|Thriller|IMAX	2009	(1990, 2010]	3.73	Superhit
13023	62265	Accidental Husband, The (2009)	Comedy|Romance	2009	(1990, 2010]	2.99	Hit
13091	63072	Road, The (2009)	Adventure|Drama|Thriller	2009	(1990, 2010]	3.61	Superhit
13326	65585	Bride Wars (2009)	Comedy|Romance	2009	(1990, 2010]	2.87	Hit
13329	65601	My Bloody Valentine 3-D (2009)	Horror|Thriller	2009	(1990, 2010]	2.61	Hit
...	...	...	...	...	...	...	...
26678	131074	Mount St. Elias (2009)	Documentary	2009	(1990, 2010]	2.50	Hit
26682	131082	Playground (2009)	(no genres listed)	2009	(1990, 2010]	4.50	Blockbuster
26699	131116	La Première étoile (2009)	Comedy	2009	(1990, 2010]	3.50	Superhit
26721	131160	Oscar and the Lady in Pink (2009)	Drama	2009	(1990, 2010]	4.00	Superhit
26724	131166	WWII IN HD (2009)	(no genres listed)	2009	(1990, 2010]	4.00	Superhit

1102 rows × 7 columns

List of movies released in the year 2009 whose genres are one of Drama, Comedy, Documentary, Comedy|Drama, Drama|Romance (5 most commonly watched genres)

In [59]:
```
m=movie2009.loc[(movie2009['genres']=='Drama') |(movie2009['genres']=='Comedy')|(movie2009['genres']
                (movie2009['genres']=='Comedy|Drama') |(movie2009['genres']=='Drama|Romance')]
m
```

Out[59]:

	movieId	title	genres	year_of_release	year_bins	rating	Remarks
13448	66503	Rally On! (Ralliraita) (2009)	Comedy	2009	(1990, 2010]	2.35	Hit
13450	66509	Funny People (2009)	Comedy|Drama	2009	(1990, 2010]	3.26	Superhit
13462	66581	Still Waiting... (2009)	Comedy	2009	(1990, 2010]	2.60	Hit
13521	67087	I Love You, Man (2009)	Comedy	2009	(1990, 2010]	3.65	Superhit
13567	67607	We Live in Public (2009)	Documentary	2009	(1990, 2010]	3.33	Superhit
...	...	...	...	...	...	...	...
26566	130388	Black Field (2009)	Drama|Romance	2009	(1990, 2010]	0.50	Super-flop
26600	130516	Glowing Stars (2009)	Drama	2009	(1990, 2010]	3.00	Hit

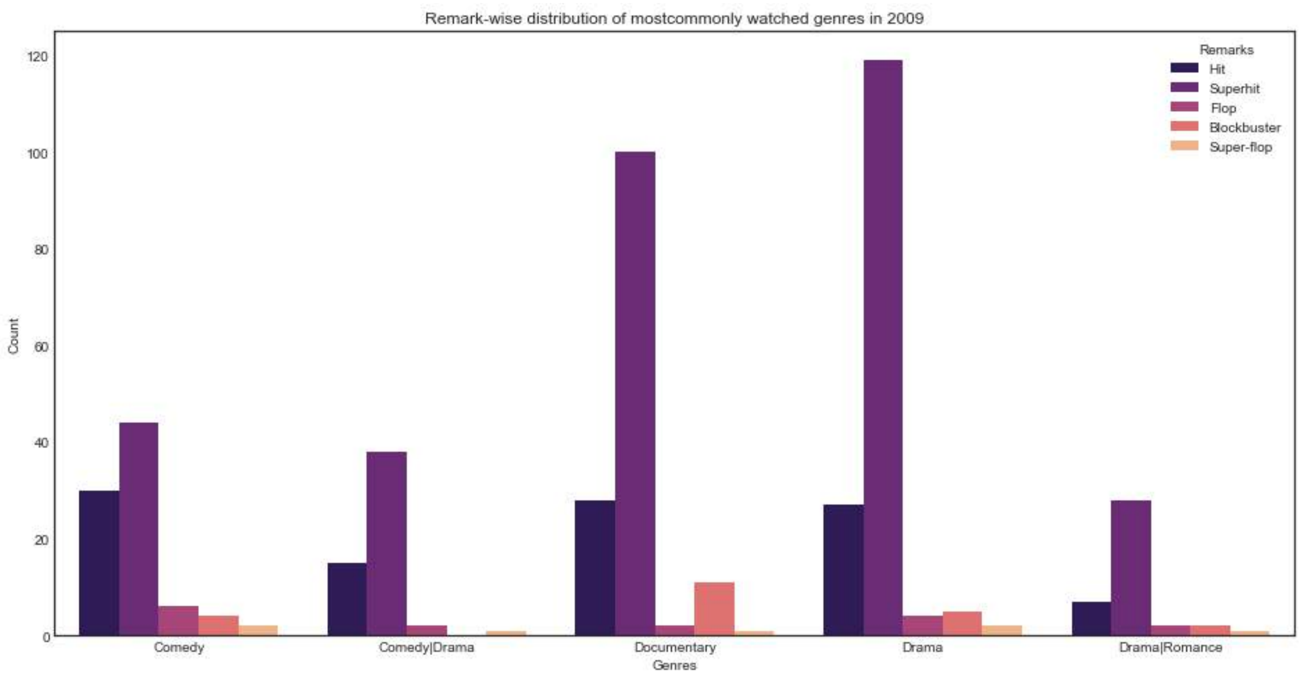
	movieId	title	genres	year_of_release	year_bins	rating	Remarks
26678	131074	Mount St. Elias (2009)	Documentary	2009	(1990, 2010]	2.50	Hit
26699	131116	La Première étoile (2009)	Comedy	2009	(1990, 2010]	3.50	Superhit
26721	131160	Oscar and the Lady in Pink (2009)	Drama	2009	(1990, 2010]	4.00	Superhit

481 rows × 7 columns

Remark-wise distribution of the most commonly watched genres in 2009

In [102…

```
plt.figure(figsize=(16,8))
sns.countplot(x='genres',hue='Remarks', data=m, palette='magma')
plt.xlabel("Genres")
plt.ylabel("Count")
plt.title("Remark-wise distribution of mostcommonly watched genres in 2009")
plt.show()
```



Amongst the top 5 most common genres (Drama, Comedy, Documentary, Comedy|Drama, Drama|Romance) we can see that movies of genres ***Drama*** and ***Documenary*** performed well in the year 2009 too.

Which movie has been rated by maximum no. of viewers (in other words which movie has the most viewership?)

In [61]:

```
usr = rating.groupby('movieId',as_index=False).userId.count()
usr.rename(columns = {'userId':'no_of_users'}, inplace = True)
usr_movie=pd.merge(movie,usr,on='movieId')
usr_movie.sort_values('no_of_users',ascending=False,inplace=True)
usr_movie.head()
```

Out[61]:

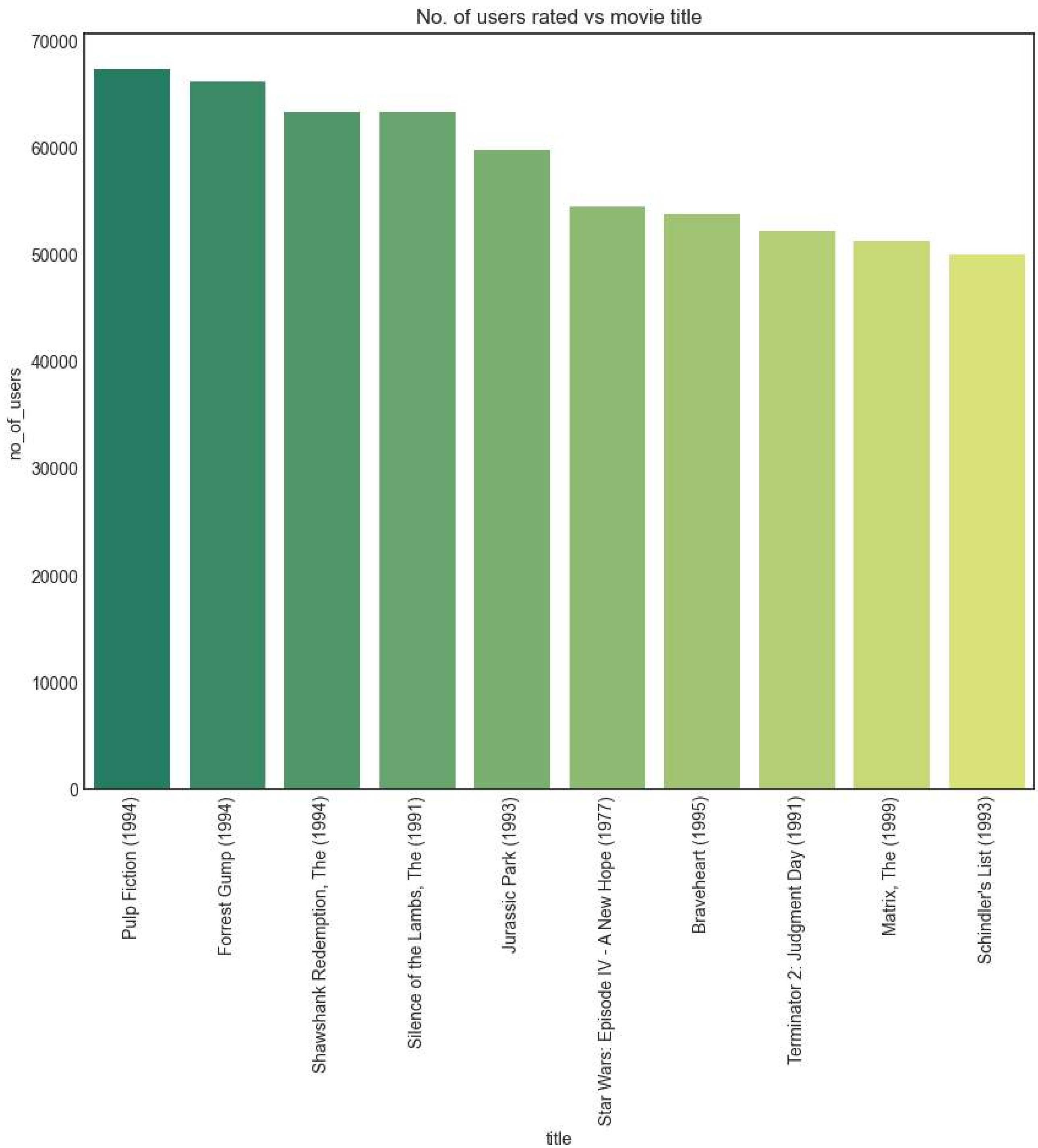
	movieId	title	genres	year_of_release	year_bins	no_of_users

	movieId	title	genres	year_of_release	year_bins	no_of_users
293	296	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller	1994	(1990, 2010]	67310
352	356	Forrest Gump (1994)	Comedy|Drama|Romance|War	1994	(1990, 2010]	66172
315	318	Shawshank Redemption, The (1994)	Crime|Drama	1994	(1990, 2010]	63366
587	593	Silence of the Lambs, The (1991)	Crime|Horror|Thriller	1991	(1990, 2010]	63299
476	480	Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller	1993	(1990, 2010]	59715

Plot - movie title Vs. number of ratings received

In [63]:

```
sorted_rated_df = usr_movie[:10]
fig = plt.figure(figsize=(10, 8))
sns.barplot(x=sorted_rated_df['title'], y=sorted_rated_df['no_of_users'], palette="summer")
plt.xticks(rotation=90)
plt.title('No. of users rated vs movie title')
plt.show()
```



In [64]:

```
movie_rating[movie_rating['movieId']==296].title
```

Out[64]:

```
293    Pulp Fiction (1994)
Name: title, dtype: object
```

Pulp Fiction (1994) has the most viewership/ maximum no. of ratings. Note that the movie falls into the Drama, Comedy genre, proving yet again that Drama and Comedy genres are indeed popular.

In [65]:

```
movie_rating[movie_rating['movieId']==296].Remarks
```

Out[65]:

```
293    Blockbuster
Name: Remarks, dtype: object
```

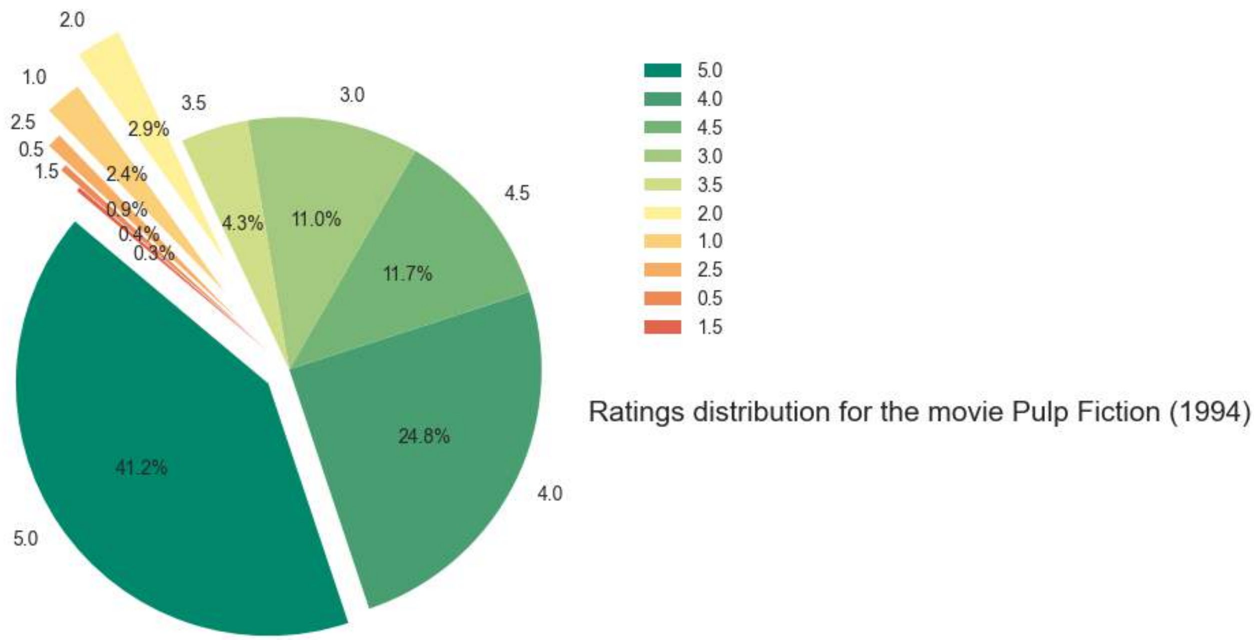
Pulp Fiction (1994) has been categorized as a blockbuster movie.

Let's see the distribution of its ratings using a pie plot

In [66]:
```
pulp_fiction = rating[rating['movieId']==296]
rating_count = pulp_fiction['rating'].value_counts()
```

Out[66]:
```
5.0    27762
4.0    16724
4.5     7867
3.0     7389
3.5     2913
2.0     1951
1.0     1595
2.5      628
0.5      291
1.5      190
Name: rating, dtype: int64
```

In [90]:
```
plt.figure(figsize=(6,6))
colors=['#00876c','#489e71','#75b477','#a2c97f','#d0de8a','#fff199','#fcd07a','#f7ad62','#ef8a54','#
explode = (0.1, 0,0,0,0,0.5,0.4,0.3,0.2,0.1)
plt.pie(list(pulp_fiction['rating'].value_counts()),labels=list(pulp_fiction['rating'].value_counts(
        colors=colors,shadow=False,explode= explode,startangle=140)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.title('Ratings distribution for the movie Pulp Fiction (1994)', x=1.5, y=0.4, fontsize=15)
plt.show()
```



PREDICTIVE ANALYSIS

A basic recommendation system using item-based filtering

Context

Item-item collaborative filtering, or item-based, or item-to-item, is a form of collaborative filtering for recommender systems based on the similarity between items calculated using people's ratings of those items.

Here, we find the correlation between a movie say "movieA" (which has been watched and rated by people) and other movies. The movie having the highest correlation with "movieA" is then recommended.

In [92]:

```
rating.head()
```

Out[92]:

	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5
2	1	32	3.5
3	1	47	3.5
4	1	50	3.5

In [93]:

```
movie2 = pd.read_csv("./MovieLens/movie.csv")
data = pd.merge(movie2, rating)
```

(The number of samples in the data frame is 20 million that is too much. If we try to create a pivot table from this data, Jupyter gives 'ValueError: Unstacked DataFrame is too big, causing int32 overflow'. Hence, for this item based recommendation system we use 1 million data samples)

In [94]:

```
data2 = data.iloc[:1000000,:]
```

Making a pivot table in which rows indicate user Id, columns indicate movie titles and values are ratings given to any particular movie by any particular user -

In [95]:

```
pivot_table = data2.pivot_table(index = ["userId"],columns = ["title"],values = "rating")
pivot_table.head(10)
```

Out[95]:

title	Ace Ventura: When Nature Calls (1995)	Across the Sea of Time (1995)	Amazing Panda Adventure, The (1995)	American President, The (1995)	Angela (1995)	Angels and Insects (1995)	Anne Frank Remembered (1995)	Antonia's Line (Antonia) (1995)	Assassins (1995)	Babe (1995)	...	Unfc
userId												
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
4	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
5	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	...	
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
7	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	...	
8	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
10	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	...	
11	3.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	

10 rows × 146 columns

- Consider a scenario in which a movie "movieA" is watched and rated by people. The question is that which movie do we recommend these people who watched movie "movieA".
- To answer this question we find similarities between "movieA" and other movies (i.e. correlation between "movieA" and other movies).

In [96]:

```
#creating a function recommender() to implement the simple recommendation system
def recommender(movieA):
    mv_watched = pivot_table[movieA]
    movies_similarity = pivot_table.corrwith(mv_watched)  # find correlation between "movieA" and ot
    movies_similarity = movies_similarity.sort_values(ascending=False)
    print(movies_similarity.head())
```

Ex-1) Which movie do we recommend people who have watched the movie - ***Bad Boys (1995)*** ?

In [97]:

```
recommender("Bad Boys (1995)")
```

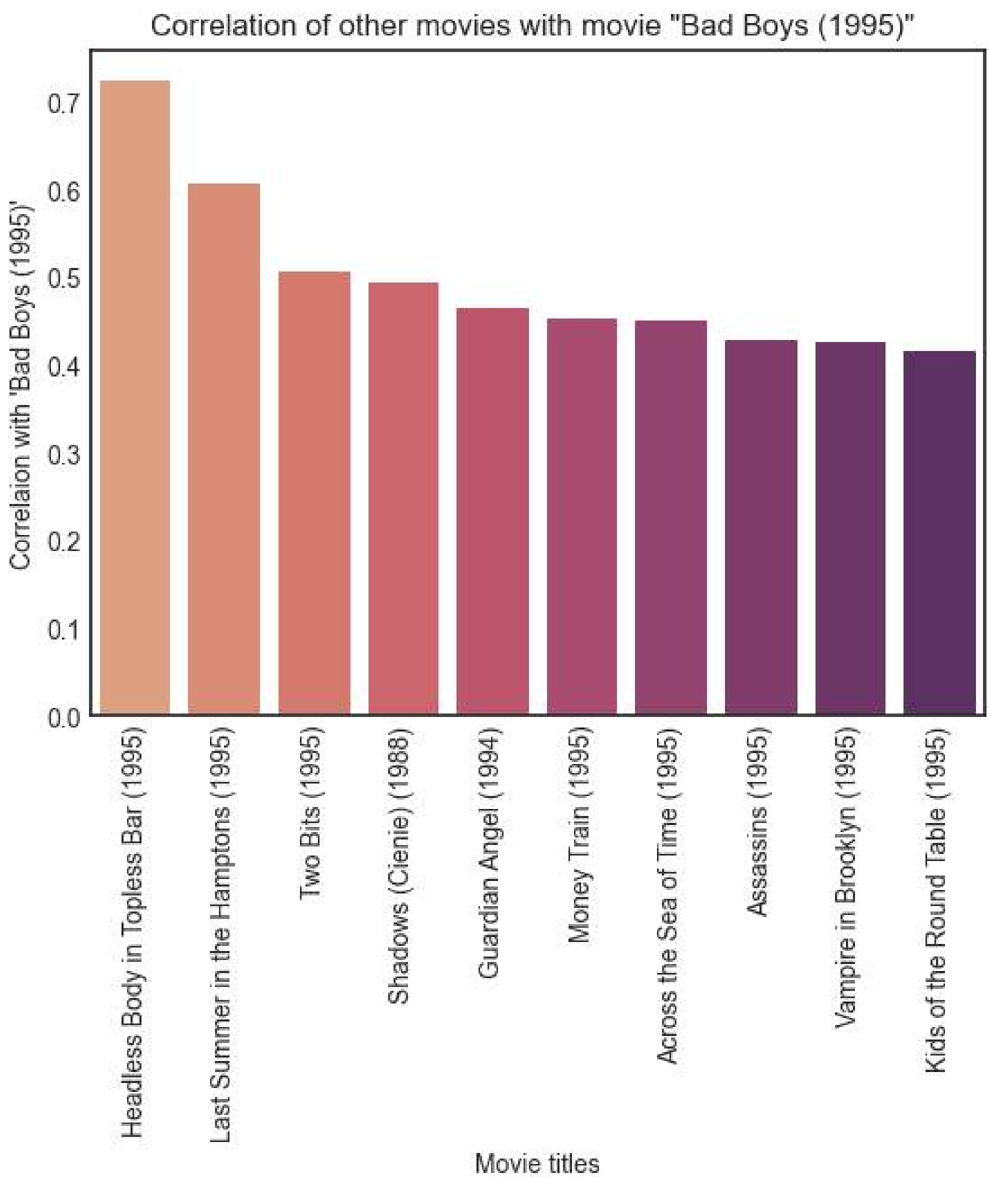
```
title
Bad Boys (1995)                        1.000000
Headless Body in Topless Bar (1995)    0.723747
Last Summer in the Hamptons (1995)     0.607554
Two Bits (1995)                        0.507008
Shadows (Cienie) (1988)                0.494186
dtype: float64
```

We can see that "Headless Body in Topless Bar (1995)" has the highest correlation with "Bad Boys (1995)"

In [98]:

```
mov_watched = pivot_table["Bad Boys (1995)"]
movies_corr = pivot_table.corrwith(mov_watched)

data3 = movies_corr.sort_values(ascending=False)[1:11]
sns.barplot(x=data3.index, y=data3, palette='flare')
plt.xlabel("Movie titles")
plt.ylabel("Correlaion with 'Bad Boys (1995)'")
plt.xticks(rotation=90)
plt.title('Correlation of other movies with movie "Bad Boys (1995)" ')
plt.show()
```



Hence we can see that we need to recommend "Headless Body in Topless Bar (1995)" movie to people who watched "Bad Boys (1995)"

Ex-2) Which movie do we recommend people who have watched the movie - ***Up Close and Personal (1996)*** ?

In [60]:

```
recommender("Up Close and Personal (1996)")
```

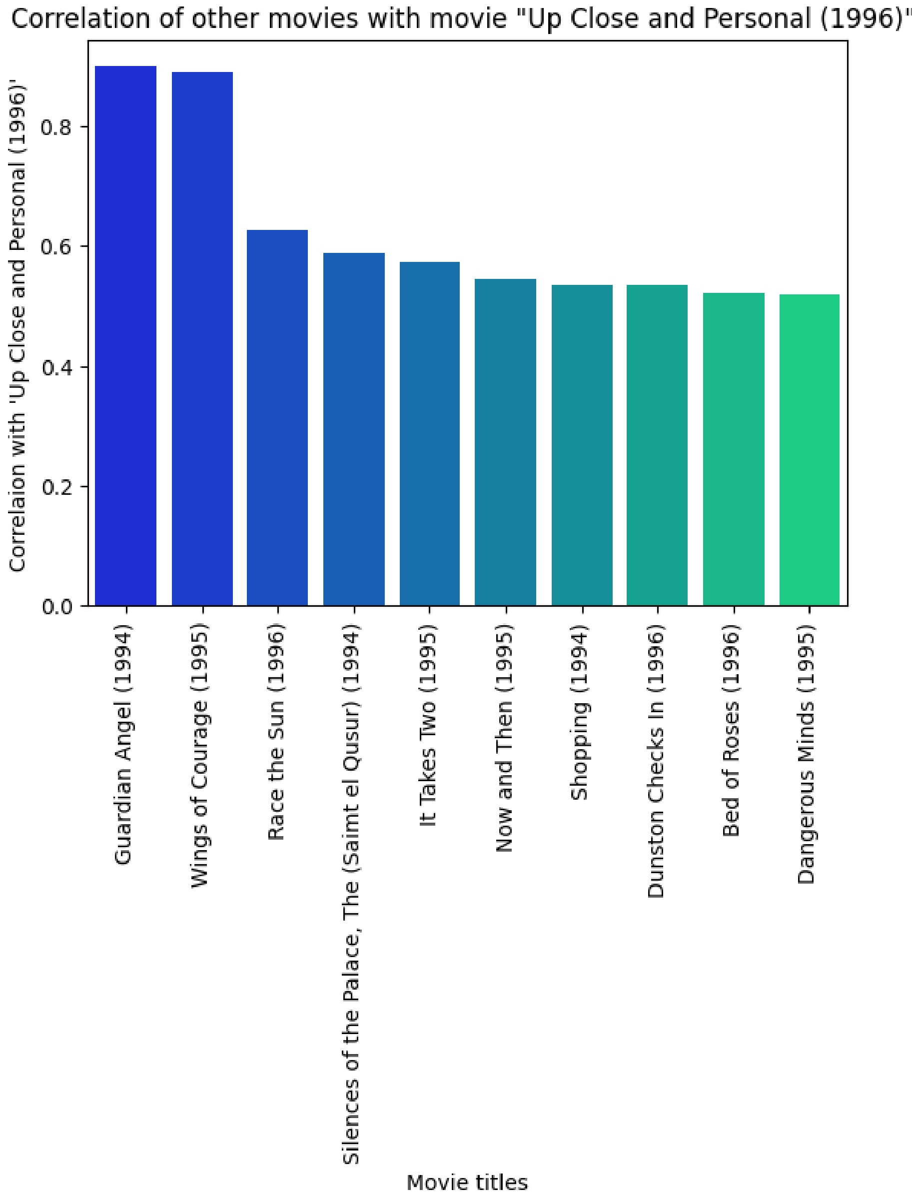
```
title
Up Close and Personal (1996)                             1.000000
Guardian Angel (1994)                                    0.898546
Wings of Courage (1995)                                  0.889001
Race the Sun (1996)                                      0.626593
Silences of the Palace, The (Saimt el Qusur) (1994)     0.589256
dtype: float64
```

We can see that "Guardian Angel (1994)" has the highest correlation with "Up Close and Personal (1996)"

In [100…

```
mov_watched2 = pivot_table["Up Close and Personal (1996)"]
movies_corr2 = pivot_table.corrwith(mov_watched2)
```

```
data4 =movies_corr2.sort_values(ascending=False)[1:11]
sns.barplot(x=data4.index, y=data4, palette='winter')
plt.xlabel("Movie titles")
plt.ylabel("Correlaion with 'Up Close and Personal (1996)'")
plt.xticks(rotation=90)
plt.title('Correlation of other movies with movie "Up Close and Personal (1996)" ')
plt.show()
```



We can see that we need to recommend "Guardian Angel (1994)" movie to people who watched "Up Close and Personal (1996)"

SUMMARY OF RESULTS

1) DESCRIPTIVE ANALYSIS

We find that:

1. Maximum rating given to any movie is 5; minimum is 0.5 while the average rating received by any movie is 3.52.
2. Highest no. of movie releases (i.e. 12902) was witnessed during the period 1990-2010.

2) EXPLORATORY ANALYSIS

- The density distribution plot for the average rating received by all movies shows that maximum movies received an average rating between 3 and 4. The bell shaped curve represents a normal distribution.

- We create a categorical column 'Remarks' based on average rating of movies:

  1. For ratings between (0,1], the movie is given the remark 'Super-flop'
  2. For ratings between (1,2], the movie is given the remark 'Flop'
  3. For ratings between (2,3], the movie is given the remark 'Hit'
  4. For ratings between (3,4], the movie is given the remark 'Superhit'
  5. For ratings between (4,5], the movie is given the remark 'Blockbuster'

- We find that our dataset has maximum no. of Super-Hit movies (movies which have received rating between 3 and 4)

- Top 10 most commonly watched genres are:

Genre	No. of movies released
Drama	4416
Comedy	2251
Documentary	1879
Comedy|Drama	1241
Drama|Romance	1043
Comedy|Romance	741
Comedy|Drama|Romance	594
Horror	556
Crime|Drama	435
Drama|Thriller	421

- Maximum movies released are of the genre 'Drama'.

- Amongst the top 5 most commonly watched genres, maximum no. of superhit and blockbuster movies are of the genre 'Drama'.

- Temporal trends of the top 5 most commonly watched genre:-

1. *Drama* has maintained its position as the most watched genre from beginning till the end.
2. *Documentary* genre started gaining popularity from the year 1970 (notice the sharp rise) reached a peak from 1990-2010 going head to head with *Comedy*. However, after year 2010, it started losing popularity once again.
3. It is also interesting to note that the *Romance* genre did not make an appearance until the year 1910.

- We find that year 1890-1910 has the highest average rating for movies.
- Majority of movies released during 1890-1910 are from genre Documentary
- We also find that 2009 is the most popular year. The reason is twofold:

  1- 2009 is the year with highest no. of movie releases- 1114.

  2- We also witness the maximum number of blockbuster movie releases (43 blockbuster movies) in 2009.

- Amongst the top 5 most common genres (Drama, Comedy, Documentary, Comedy|Drama, Drama|Romance) we find that movies of genres *Drama* and *Documentary* performed well in the year 2009 too with 120 superhit movies releases of the Drama genre and 100 superhit releases of the Documentary genre.
- Pulp Fiction (1994) has the most viewership/ maximum no. of ratings. Note that the movie falls into the Drama, Comedy genre, proving yet again that Drama and Comedy genres are indeed popular. The movie has also been categorized as a blockbuster movie. More than 50% of the ratings received by ‘Pulp Fiction’ fall between values 4 and 5.

PREDICTIVE ANALYSIS

We make a basic recommendation system using item-based filtering. The approach is quite simple. We find the correlation between a movie say "movieA" (which has been watched and rated by people) and other movies. The movie having the highest correlation with "movieA" is then recommended.

Bibliography

- https://www.kaggle.com/grouplens/movielens-20m-dataset
- https://stackoverflow.com/
- https://www.geeksforgeeks.org/
- https://www.google.com/