TMDB 5000 Movie Dataset (Movie Recommeddation System)

Types of Recommendation Systems :

1. Content Based Recommendation System :

This system promotes or recommends movies to user based on the movies that they have watched before. For example , if person watched action movies before, then it will recommend action movies for him.

2. Popularity Based Recommendation System :

This system will recommend top movies in film platforms such as Netflix or cinemas.

3. Collaborative Recommendation System :

This system groups people based on their watching pattern. Then if a user watch a film of this group's films then the system will recommend the films watched by this group to the user. (Recommend based on other previous data).

> 🚨 **My note:** We will take an input from the user : So, we can use the second and third systems (Techniques).

Workflow :

1. Data Collection :

We need to have a data of this movies. (Movie description, Type of the movie . . . etc).

2. Data PreProcessing :

Clean data for any missing or incomplete values.

3. Feature Extraction :

There are textual features in data frame , we can not ues it directly. So, We need to convet into meaningful numerical values)

4. Find the Similarity :

We have 5000 movie and we want to find which movies are similar to each other by giving them a similarity score (Similarity Confidence Score).

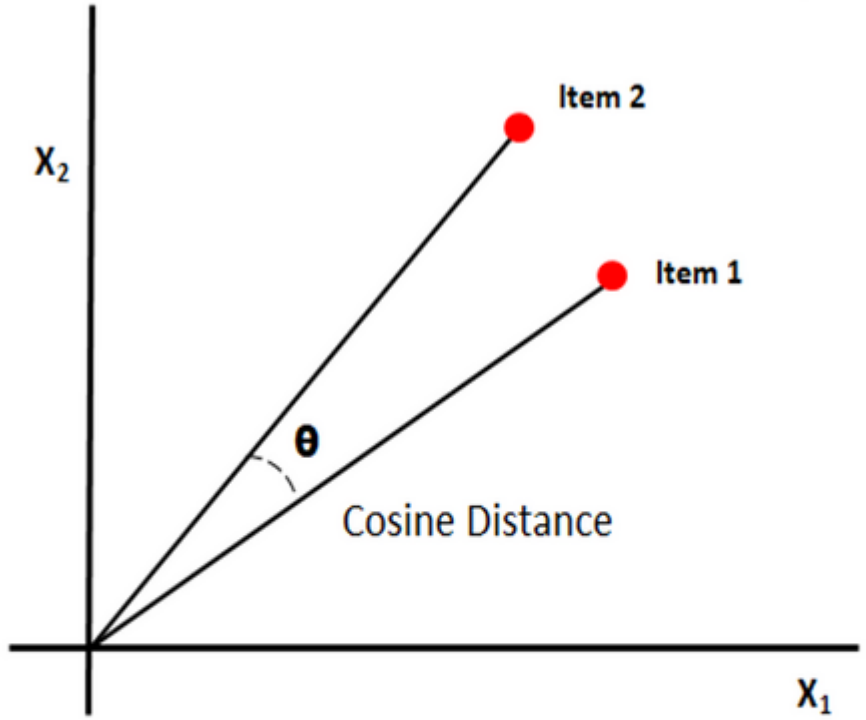
5. User Input :

Ask user for his input , so based on user input we should suggest which movie user can watch.

6. Use Cosine Similarity :

This percent similarity algorithm is used in order to find the similarity between the vectors so here we will just converting each movies into a kind of a vector and we will try to find the similarity between them using Cosine-similarity. So when a user gives a movie name, we will try to compare that movie and we will just try to find which movies are similar to the one given by the user. now we will get a list of movies and we can

Cosine Distance/Similarity



$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Import Libraries :

In [1]:

```
import ast
import numpy as np              # NumPy is a Python library used for working with arrays.
import pandas as pd             # Pandas is mainly used for data analysis. Pandas allows importing data from various file
```

```
import seaborn as sns           # Seaborn is a library in Python predominantly used for making statistical graphics. Seab
import matplotlib.pyplot as plt # Matplotlib is a cross-platform, data visualization and graphical plotting library for P
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
```

In [2]:

```
# Loading the dataframe
df = pd.read_csv('tmdb_5000_movies.csv')
df_credits = pd.read_csv('tmdb_5000_credits.csv')
df = df.merge(df_credits,on='title')
```

In [3]:

```
df.head() # Show the first 5 rows in the data
```

Out[3]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond’s past sends him o...	107.376788
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	John Carter	John Carter is a war-weary, former military ca...	43.926995

5 rows × 23 columns

In [4]:
```
df_credits.head()
```

Out[4]:

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

Describe the data :

In [5]:
```
df.shape # Show the number of rows and columns as a tuple (number of rows, number of columns).
# There is 4803 rows and 20 column
```

Out[5]:
```
(4809, 23)
```

In [6]:
```
df.columns # Show name of columns
```

Out[6]:
```
Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'original_language',
       'original_title', 'overview', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
```

```
       'spoken_languages', 'status', 'tagline', 'title', 'vote_average',
       'vote_count', 'movie_id', 'cast', 'crew'],
      dtype='object')
```

In [7]:

```
#==================================== Column Names and its meaning ====================================
# Budget : The budget in which the movie was made.
# Genre : Type of the movie action , drama , horror ...
# Homepage : Offical page link (Where you can watch the movie).
# Id : ID of the film.
# Keyword : The keywords or tags related to the movie : Words tell the kind of the idea of about what the movie is.
# Original language : The language in which the movie was made.
# Original title : The title of the movie before translation or adaptation.
# Overview : A brief description of the movie.
# Popularity : A numeric quantity specifying the movie popularity.
# Production companies : The production house of the movie.
# Production countries : The country in which it was produced.
# Release date : The date on which it was released.
# Revenue : The worldwide revenue generated by the movie.
# Runtime : The running time of the movie in minutes.
# Status : "Released" or "Rumored".
# Tagline : Movie's tagline.
# Title : Title of the movie.
# Vote average : average ratings the movie recieved.
# Vote count : the count of votes recieved.
```

In [8]:

```
df.describe() # It calculate some statistical data like percentile, mean and std of the numerical values of DataFrame.
```

Out[8]:

	budget	id	popularity	revenue	runtime	vote_average	vote_count	movie_id
count	4.809000e+03	4809.000000	4809.000000	4.809000e+03	4807.000000	4809.000000	4809.000000	4809.000000
mean	2.902780e+07	57120.571429	21.491664	8.227511e+07	106.882255	6.092514	690.331670	57120.571429
std	4.070473e+07	88653.369849	31.803366	1.628379e+08	22.602535	1.193989	1234.187111	88653.369849
min	0.000000e+00	5.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000	5.000000
25%	7.800000e+05	9012.000000	4.667230	0.000000e+00	94.000000	5.600000	54.000000	9012.000000
50%	1.500000e+07	14624.000000	12.921594	1.917000e+07	103.000000	6.200000	235.000000	14624.000000
75%	4.000000e+07	58595.000000	28.350529	9.291317e+07	118.000000	6.800000	737.000000	58595.000000
max	3.800000e+08	459488.000000	875.581305	2.787965e+09	338.000000	10.000000	13752.000000	459488.000000

How to deal with outliers data ?

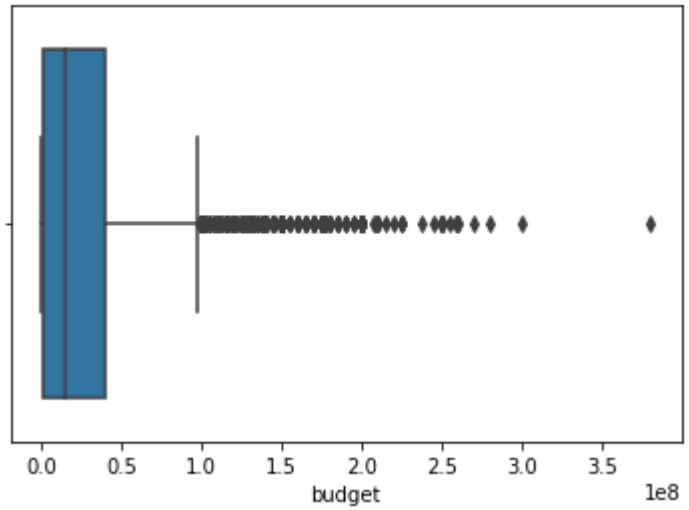
In [9]:

```
print("Number of films that have a budget less that 100 : ",len(df[df['budget'] < 100]))
sns.boxplot(df['budget'])
```

```
Number of films that have a budget less that 100 :  1062
C:\Users\com\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keywo
rd arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[9]:

```
<AxesSubplot:xlabel='budget'>
```



In [10]:
```
# Budget
Q1 = df.budget.quantile(0.25)
Q3 = df.budget.quantile(0.75)
IQR = Q3 - Q1
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit # Upper limit allowed , Lower limit allowed
```

Out[10]:
```
(-58050000.0, 98830000.0)
```

In [11]:
```
df[df["budget"] >upper_limit]
```

Out[11]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	p
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	1
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"},	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to	1

	budget	genres	homepage	id	keywords	original_language	original_title	overview	p
					{"id": 726, "na...			be dead, ha...	
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	1
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	1
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	John Carter	John Carter is a war-weary, former military ca...	
...	...	...	...	...	...	...	...	...	
565	150000000	[{"id": 12, "name": "Adventure"}, {"id": 16, "...	http://www.shrek2.com/	809	[{"id": 378, "name": "prison"}, {"id": 2343, "...	en	Shrek 2	Shrek, Fiona and Donkey set off to Far, Far Aw...	
566	120000000	[{"id": 16, "name": "Animation"}, {"id": 12, "...	http://disney.go.com/disneyvideos/animatedfilm...	920	[{"id": 830, "name": "car race"}, {"id": 1926,...	en	Cars	Lightning McQueen, a hotshot rookie race car d...	
692	150000000	[{"id": 16, "name": "Animation"}, {"id": 10751...	http://movies.disney.com/chicken-little	9982	[{"id": 1357, "name": "fish"},	en	Chicken Little	When the sky really is falling and sanity has ...	

	budget	genres	homepage	id	keywords	original_language	original_title	overview	p
1065	120000000	[{"id": 12, "name": "Adventure"}, {"id": 16, "...	http://movies.disney.com/a-bugs-life	9487	[{"id": 1442, "name": "winter"}, {"id": 1721, ...	en	A Bug's Life	On behalf of "oppressed bugs everywhere," an i...	
1658	100000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	NaN	14164	[{"id": 3436, "name": "karate"}, {"id": 9715, ...	en	Dragonball Evolution	The young warrior Son Goku sets out on a quest...	

321 rows × 23 columns

In [12]:
```
median = df['budget'].median()
df["budget"] = np.where(df["budget"] >upper_limit, median,df['budget'])
```

In [13]:
```
df.shape
```

Out[13]:
```
(4809, 23)
```

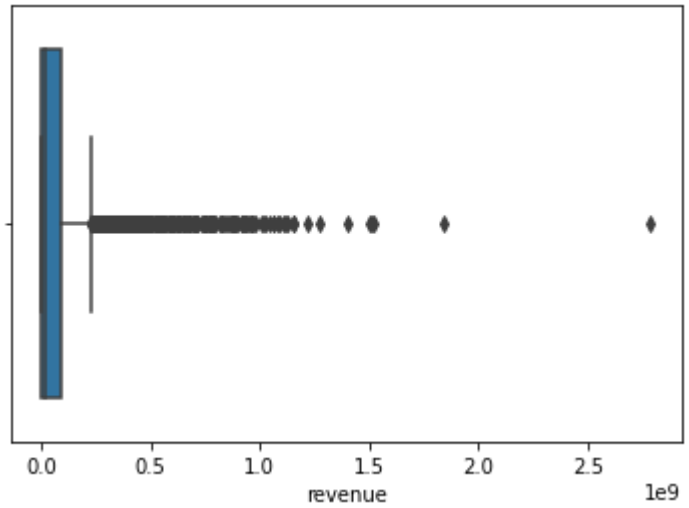
🚨 **Budget column:** 1. There are films that have budget less than 100 which is so small. 2. There are high percentage of outliers in this column (It will be not good to drop them) ,So I will replace it with median. 3. Fortunately, This column is not important in recommendation process so i will exclude it later.

In [14]:
```
print("Number of films that have a revenue less that 100 : ",len(df[df['revenue'] < 100]))
len(df[df['revenue'] < 100])
sns.boxplot(df['revenue'])
```

```
Number of films that have a revenue less that 100 :  1448
C:\Users\com\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keywo
rd arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an e
```

```
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[14]:
```
<AxesSubplot:xlabel='revenue'>
```



In [15]:
```
# Revenue
Q1 = df.revenue.quantile(0.25)
Q3 = df.revenue.quantile(0.75)
IQR = Q3 - Q1
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit
```

Out[15]:
```
(-139369756.5, 232282927.5)
```

In [16]:
```
df[df['revenue']> upper_limit]
```

Out[16]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	pop
0	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.

	budget	genres	homepage	id	keywords	original_language	original_title	overview	pop
1	15000000.0	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.(
2	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	107.3
3	15000000.0	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112.3
4	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	John Carter	John Carter is a war-weary, former military ca...	43.9
...	...	...	...	...	...	...	...	...	
3703	6000000.0	[{"id": 35, "name": "Comedy"}, {"id": 18, "nam...	http://workingtitlefilms.com/film.php?filmID=59	712	[{"id": 213, "name": "upper class"}, {"id": 69...	en	Four Weddings and a Funeral	Four Weddings And A Funeral is a British comed...	29.8
3820	4000000.0	[{"id": 18, "name": "Drama"}, {"id": 10749, "n...	NaN	770	[{"id": 314, "name": "life and death"}, {"id":...	en	Gone with the Wind	An American classic in which a manipulative wo...	48.9

	budget	genres	homepage	id	keywords	original_language	original_title	overview	pop
3831	3500000.0	[{"id": 35, "name": "Comedy"}]	NaN	9427	[{"id": 1252, "name": "suicide attempt"}, {"id...	en	The Full Monty	Sheffield, England. Gaz, a jobless steelworker...	17.(
4447	858000.0	[{"id": 16, "name": "Animation"}, {"id": 18, "...	http://movies.disney.com/bambi	3170	[{"id": 5774, "name": "forest"}, {"id": 10683,...	en	Bambi	Bambi's tale unfolds from season to season as ...	47.(
4502	60000.0	[{"id": 27, "name": "Horror"}, {"id": 9648, "n...	http://www.blairwitch.com/	2667	[{"id": 616, "name": "witch"}, {"id": 3392, "n...	en	The Blair Witch Project	In October of 1994 three student filmmakers di...	41.(

473 rows × 23 columns

In [17]:
```
median = df['revenue'].median()
df["revenue"] = np.where(df["revenue"] >upper_limit, median,df['revenue'])
```

In [18]:
```
df.shape
```

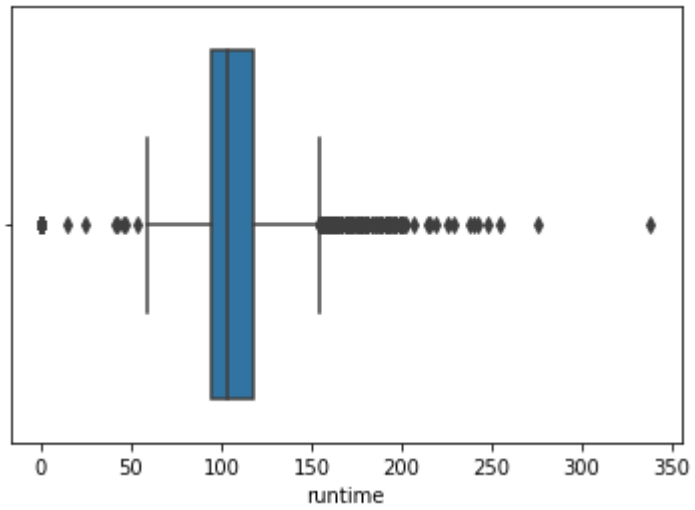
Out[18]:
```
(4809, 23)
```

🚨 **Revenue column:** 1. There are films that have revenue less than 100 which is so small. 2. There are some outliers in this column (It will not be good to drop them) ,So I will replace it with median. 3. Fortunately, This column is not important in recommendation process so i will exclude it later.

In [19]:
```
len(df[df['runtime'] == 0])
sns.boxplot(df['runtime'])
```

```
C:\Users\com\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keywo
rd arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[19]:
```
<AxesSubplot:xlabel='runtime'>
```



In [20]:
```
# Runtime
Q1 = df.runtime.quantile(0.25)
Q3 = df.runtime.quantile(0.75)
IQR = Q3 - Q1
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit
```

Out[20]:
```
(58.0, 154.0)
```

In [21]:
```
df[df['runtime']> upper_limit]
```

Out[21]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	po
0	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150

	budget	genres	homepage	id	keywords	original_language	original_title	overview	po
					clash"}, {"id":...				
1	15000000.0	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139
3	15000000.0	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112
22	15000000.0	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://www.thehobbit.com/	57158	[{"id": 603, "name": "elves"}, {"id": 604, "na...	en	The Hobbit: The Desolation of Smaug	The Dwarves, Bilbo and Gandalf have successful...	94
24	15000000.0	[{"id": 12, "name": "Adventure"}, {"id": 18, "...	NaN	254	[{"id": 774, "name": "film business"}, {"id": ...	en	King Kong	In 1933 New York, an overly ambitious movie pr...	61
...	...	...	...	...	...	...	...	...	
4395	0.0	[{"id": 53, "name": "Thriller"}]	NaN	20296	[]	en	Chocolate: Deep Dark Secrets	Christmas Eve, London. While the snow-clad cit...	0
4486	700000.0	[{"id": 99, "name": "Documentary"}]	NaN	14275	[{"id": 520, "name": "chicago"}, {"id": 1483, ...	en	Hoop Dreams	This documentary follows two inner-city Chicag...	9
4503	600000.0	[{"id": 36, "name": "History"}, {"id": 99, "na...	NaN	9459	[{"id": 458, "name": "hippie"},	en	Woodstock	An intimate look at the Woodstock	3

	budget	genres	homepage	id	keywords	original_language	original_title	overview	po
					{"id": 460, "n...			Music & Art ...	
4541	2000000.0	[{"id": 28, "name": "Action"}, {"id": 18, "nam...	NaN	346	[{"id": 233, "name": "japan"}, {"id": 1462, "n...	ja	七人の侍	A samurai answers a village's request for prot...	39
4598	385907.0	[{"id": 18, "name": "Drama"}]	NaN	3059	[{"id": 279, "name": "usa"}, {"id": 2487, "nam...	en	Intolerance	The story of a poor young woman, separated by ...	3

140 rows × 23 columns

In [22]:
```
print(len(df[df['runtime']< lower_limit]))
df[df['runtime']< lower_limit]
```

42

Out[22]:

	budget	genres	homepage	id	keywords	original_language	original_title	overv
1014	0.0	[{"id": 27, "name": "Horror"}]	NaN	53953	[{"id": 10292, "name": "gore"}, {"id": 12339, ...	de	The Tooth Fairy	A wo and daug (Ni Mu enco
3117	0.0	[{"id": 18, "name": "Drama"}, {"id": 80, "name...	NaN	41894	[]	en	Blood Done Sign My Name	A dr based on true sto which a l
3359	0.0	[{"id": 99, "name": "Documentary"}]	NaN	24977	[{"id": 6075, "name": "sport"}]	en	Michael Jordan to the Max	documen showc baske player

	budget	genres	homepage	id	keywords	original_language	original_title	overv
3408	0.0	[{"id": 10751, "name": "Family"}, {"id": 16, "...	NaN	294512	[]	en	Alpha and Omega: The Legend of the Saw Tooth Cave	The Alp and Ome sha thri adve
3476	6000000.0	[{"id": 99, "name": "Documentary"}]	NaN	57612	[{"id": 630, "name": "dolphin"}, {"id": 4676, ...	en	Dolphins and Whales: Tribes of the Ocean	documen goes to c reefs of
3631	5000000.0	[{"id": 99, "name": "Documentary"}]	NaN	78394	[{"id": 10506, "name": "prehistoric"}, {"id": ...	en	Sea Rex 3D: Journey to a Prehistoric World	Through powe IMAX experien w
3677	0.0	[{"id": 35, "name": "Comedy"}, {"id": 18, "nam...	http://www.romeothemovie.com/	113406	[]	en	Should've Been Romeo	A cente middle-a pitchmar a
3816	4000000.0	[{"id": 35, "name": "Comedy"}, {"id": 10749, "...	NaN	158150	[]	en	How to Fall in Love	account who n quite g out of
3960	0.0	[{"id": 10752, "name": "War"}, {"id": 18, "nam...	NaN	281230	[{"id": 187056, "name": "woman director"}]	en	Fort McCoy	Unabl serv World W because
3999	0.0	[]	NaN	346081	[]	en	Sardaarji	A g hunter bottle cap troul
4075	0.0	[]	NaN	371085	[]	en	Sharkskin	The Post ll sto Manha

	budget	genres	homepage	id	keywords	original_language	original_title	overv
								born M
4125	0.0	[]	NaN	325140	[]	en	Hum To Mohabbat Karega	Ra waiter, love with famous T
4212	0.0	[{"id": 18, "name": "Drama"}, {"id": 80, "name...	http://www.imdb.com/title/tt1289419/	66468	[]	en	N-Secure	N-Secure no hc ba thri dran
4217	0.0	[{"id": 10749, "name": "Romance"}]	NaN	74084	[]	hi	दिल जो भी कहे	During British ru India, sev Ir
4248	1500000.0	[{"id": 35, "name": "Comedy"}]	NaN	51820	[{"id": 10183, "name": "independent film"}]	en	The Salon	A Be shop ov f romanc she str
4319	0.0	[{"id": 53, "name": "Thriller"}, {"id": 27, "n...	NaN	107315	[{"id": 888, "name": "screenwriter"}]	en	Below Zero	When (Edv Furlong) dange r
4324	0.0	[{"id": 27, "name": "Horror"}]	NaN	310933	[]	en	Bleeding Hearts	Captu Hearts insane s killer/ho
4328	0.0	[{"id": 99, "name": "Documentary"}]	NaN	102840	[]	en	Sex With Strangers	For s mar couples, i obsessi
4334	0.0	[{"id": 27, "name": "Horror"}, {"id": 99, "nam...	NaN	202604	[{"id": 2626, "name": "exorcism"}]	en	The Vatican Exorcisms	Documen following film-m Joe Mari

	budget	genres	homepage	id	keywords	original_language	original_title	overv
4411	0.0	[{"id": 10751, "name": "Family"}, {"id": 35, "...	https://www.epicbuzz.net/movies/karachi-se-lahore	357441	[]	en	Karachi se Lahore	A road from Kar to Lal where 5
4441	0.0	[{"id": 27, "name": "Horror"}]	NaN	323270	[{"id": 9706, "name": "anthology"}]	en	The Horror Network Vol. 1	Serial kil gho phone c inner de
4464	0.0	[]	NaN	279759	[]	en	Harrison Montgomery	Film f Daniel Da
4472	0.0	[{"id": 27, "name": "Horror"}, {"id": 878, "na...	NaN	211557	[]	en	Vessel	Vessel is story of passen of Fli
4508	0.0	[{"id": 80, "name": "Crime"}, {"id": 18, "name...	NaN	263503	[]	en	Water & Power	brot nickna "Water" "Power'
4510	0.0	[]	NaN	331493	[]	en	Light from the Darkroom	Light in Darkroo the stor two b
4559	0.0	[]	NaN	380097	[]	en	America Is Still the Place	1971 civil ri Franc seeme
4564	0.0	[{"id": 10402, "name": "Music"}, {"id": 27, "n...	http://www.thedevilscarnival.com/	285743	[{"id": 3473, "name": "carnival"}, {"id": 4344...	en	Alleluia! The Devil's Carnival	The De Carn Allelui the secor
4570	0.0	[{"id": 18, "name": "Drama"}]	NaN	94072	[]	en	Straight Out of Brooklyn	A Spe Jury Av winner at Sunda

	budget	genres	homepage	id	keywords	original_language	original_title	overv
4572	0.0	[]	NaN	325579	[]	en	Diamond Ruff	Acti Orphan, artist, c boss
4575	0.0	[]	http://mutualfriendsmovie.com/	198370	[]	en	Mutual Friends	Surp parties ra go well. one i
4577	0.0	[]	NaN	328307	[]	en	Rise of the Entrepreneur: The Search for a Bet...	The wor chan faster t e Tech
4587	0.0	[]	NaN	281189	[{"id": 187056, "name": "woman director"}]	en	Gory Gory Hallelujah	Four ac compete the ro Jesus -
4590	0.0	[{"id": 27, "name": "Horror"}, {"id": 35, "nam...	NaN	189711	[]	en	Love in the Time of Monsters	Two sis travel ch tourist
4617	0.0	[]	NaN	162396	[]	en	The Big Swap	In this Br drama, E (So Brooks
4626	0.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	NaN	47534	[{"id": 2792, "name": "boxer"}, {"id": 4076, "...	en	Fighting Tommy Riley	An a trainer a yo fighter, b
4639	0.0	[]	NaN	300327	[]	en	Death Calls	An act packed story on Mex b

	budget	genres	homepage	id	keywords	original_language	original_title	overv
4663	0.0	[]	NaN	320435	[]	en	UnDivided	UnDivi docum the true s of how a
4664	0.0	[{"id": 27, "name": "Horror"}, {"id": 53, "nam...	NaN	150211	[{"id": 177972, "name": "bickering"}, {"id": 2...	en	The Frozen	Aft harrov snowmc accide you
4668	0.0	[{"id": 35, "name": "Comedy"}]	NaN	40963	[{"id": 10183, "name": "independent film"}]	en	Little Big Top	An aging of v cl returns to sn
4715	0.0	[{"id": 16, "name": "Animation"}, {"id": 10751...	NaN	13187	[{"id": 65, "name": "holiday"}, {"id": 207317,...	en	A Charlie Brown Christmas	When Ch Br compl about overw
4735	0.0	[{"id": 10751, "name": "Family"}]	NaN	272726	[]	en	Dude Where's My Dog?	Left h alone his Harry, yo R
4762	50000.0	[{"id": 27, "name": "Horror"}, {"id": 53, "nam...	http://www.cthulhulives.org/cocmovie/index.html	20981	[{"id": 1523, "name": "obsession"}, {"id": 303...	en	The Call of Cthulhu	A d profe leave gr nephe

42 rows × 23 columns

In [23]:

```
df = df[df['runtime']> lower_limit]
df = df[df['runtime']< upper_limit]
```

In [24]:
```
df.shape
```

Out[24]:
```
(4613, 23)
```

🚨 **Runtime:** 1. There are some rows that have runtime (film duration equal to 0) Zero , which is wrong values. 2. This column have outliers (Not much). 3. This column will be important in the process of recommendation , so I drop outlier columns.

Null values:

In [25]:
```
df.isna().sum() # Returns the number of missing values in each column.
```

Out[25]:
```
budget                     0
genres                     0
homepage                2950
id                         0
keywords                   0
original_language          0
original_title             0
overview                   1
popularity                 0
production_companies       0
production_countries       0
release_date               0
revenue                    0
runtime                    0
spoken_languages           0
status                     0
tagline                  782
title                      0
vote_average               0
vote_count                 0
movie_id                   0
cast                       0
crew                       0
dtype: int64
```

In [26]:
```
df_credits.isna().sum() # There is no null values in this data frame
```

Out[26]:
```
movie_id    0
title       0
cast        0
crew        0
dtype: int64
```

🚨 **My note:** Homepage : - It has 3091 null values. - How I deal with it : I delete this column , It is not useful for me in the recommendation. Overview : - It has 3 null values. - How I deal with it : I drop these 3 rows. Release date : - It has 1 null values. - How I deal with it : I drop these 1 row. Runtime : - It has 2 null values. - How I deal with it : I drop these 2 rows. Tagline : - It has 844 null values. - How I deal with it : I replace these rows with empty string as num of rows are very big. Also, this feature will be helpfull for me in the in the recommendation process.

Drop some columns and rows with null value :

In [27]:
```
df.drop('homepage', inplace=True, axis=1) # Drop Homepage column
df.drop(df[(df['runtime'] == 0)].index, inplace = True) #4. Drop Columns with runtime == 0 , as there is no film have 0 d
df["tagline"].fillna("", inplace = True) #5. Drop Tagline column
df = df.dropna()  #6. Drop null rows (Overview, Release date, Runtime)
```

In [28]:
```
df.isna().sum() # Now, there are not any null values
```

Out[28]:
```
budget                0
genres                0
id                    0
keywords              0
original_language     0
original_title        0
overview              0
popularity            0
production_companies  0
production_countries  0
release_date          0
revenue               0
runtime               0
spoken_languages      0
status                0
tagline               0
title                 0
```

```
vote_average            0
vote_count              0
movie_id                0
cast                    0
crew                    0
dtype: int64
```

In [29]:
```
df.shape
```

Out[29]:
```
(4612, 22)
```

In [30]:
```
df.head()
```

Out[30]:

	budget	genres	id	keywords	original_language	original_title	overview	popularity	production_companies	production_count
2	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"iso_3166_1": " "name": "Un Kingdo
4	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	John Carter	John Carter is a war-weary, former military ca...	43.926995	[{"name": "Walt Disney Pictures", "id": 2}]	[{"iso_3166_1": " "name": "United St
5	15000000.0	[{"id": 14, "name": "Fantasy"}, {"id": 28, "na...	559	[{"id": 851, "name": "dual identity"}, {"id": ...	en	Spider-Man 3	The seemingly invincible Spider-Man goes up ag...	115.699814	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"iso_3166_1": " "name": "United St
6	15000000.0	[{"id": 16, "name": "Animation"}, {"id": 10751...	38757	[{"id": 1562, "name": "hostage"}, {"id": 2343,...	en	Tangled	When the kingdom's most wanted-and most charmi...	48.681969	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"iso_3166_1": " "name": "United St

	budget	genres	id	keywords	original_language	original_title	overview	popularity	production_companies	production_count
7	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	99861	[{"id": 8828, "name": "marvel comic"}, {"id": ...	en	Avengers: Age of Ultron	When Tony Stark tries to jumpstart a dormant p...	134.279229	[{"name": "Marvel Studios", "id": 420}, {"name...	[{"iso_3166_1": " "name": "United St

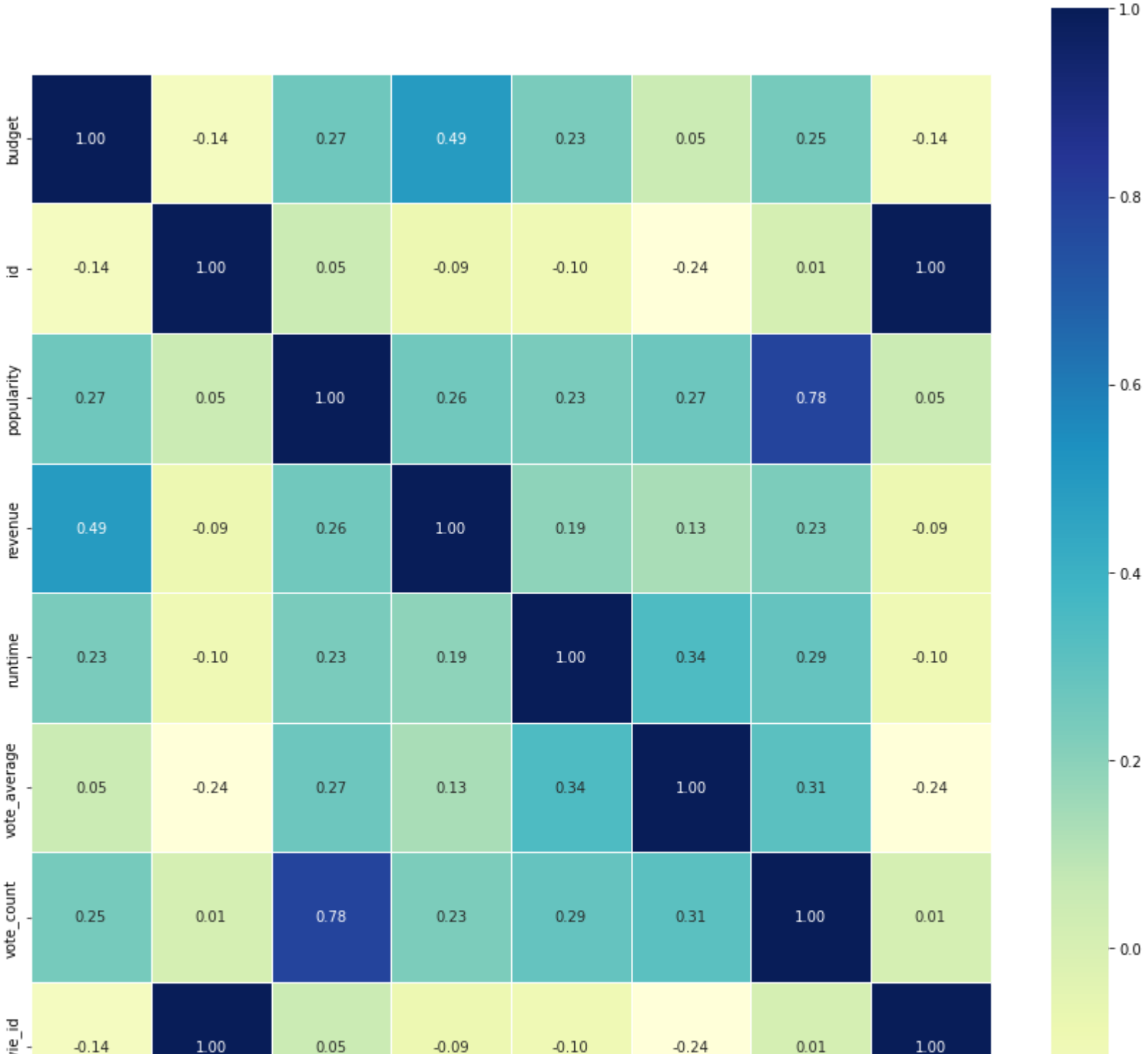
5 rows × 22 columns

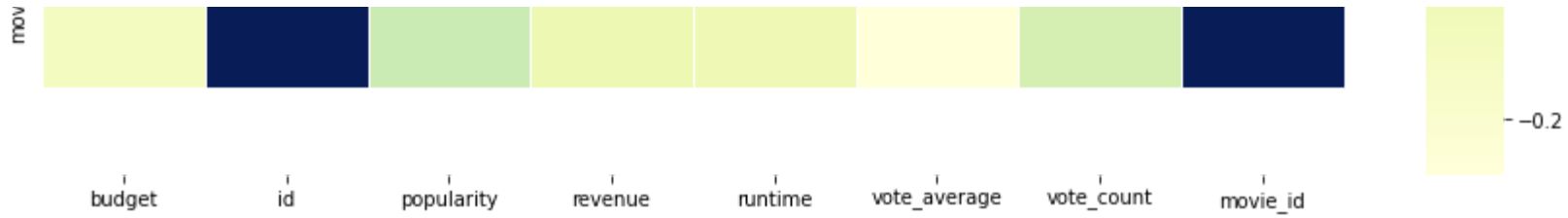
> **My note:** Number of column and rows not affect (38 rows deleted , 3 column deleted)

Correlation Matrix :

In [31]:
```
# Let's make our correlation matrix a little prettier
corr_matrix =df.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.9,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Out[31]:
```
(8.5, -0.5)
```


budget
id
popularity
revenue
runtime
vote_average
vote_count
ie_id
1.00 -0.14 0.27 0.49 0.23 0.05 0.25 -0.14
-0.14 1.00 0.05 -0.09 -0.10 -0.24 0.01 1.00
0.27 0.05 1.00 0.26 0.23 0.27 0.78 0.05
0.49 -0.09 0.26 1.00 0.19 0.13 0.23 -0.09
0.23 -0.10 0.23 0.19 1.00 0.34 0.29 -0.10
0.05 -0.24 0.27 0.13 0.34 1.00 0.31 -0.24
0.25 0.01 0.78 0.23 0.29 0.31 1.00 0.01
-0.14 1.00 0.05 -0.09 -0.10 -0.24 0.01 1.00
1.0
0.8
0.6
0.4
0.2
0.0



My note: 1. Vote Count has a strong correlation with popularity. This is logical. It means the more vote count the more popularity the film is 2. Vote count & vote average & Popularit have a low correlation with runtime. Thats mean that the time of the film not affect in the vote of people and its popularity. 3. Revenue has very low correlation with Runtime. So that means that the time of the film have no relation with its revenue.

Change release_date to year to easily deal with column :

In [32]:
```
df['release_year'] = pd.to_datetime(df['release_date']).dt.year # Add a new column with the year of the film
```

In [33]:
```
df.drop('release_date', inplace=True, axis=1) # Drop the column of release_date because we replace it with the year colum
```

In [34]:
```
df.head()
```

Out[34]:

	budget	genres	id	keywords	original_language	original_title	overview	popularity	production_companies	production_count
2	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"iso_3166_1": " "name": "Un Kingdo
4	15000000.0	[{"id": 28, "name": "Action"},	49529	[{"id": 818, "name": "based on	en	John Carter	John Carter is a war-weary,	43.926995	[{"name": "Walt Disney Pictures", "id": 2}]	[{"iso_3166_1": " "name": "United St

	budget	genres	id	keywords	original_language	original_title	overview	popularity	production_companies	production_count
		{"id": 12, "nam...		novel"}, {"id":...			former military ca...			
5	15000000.0	[{"id": 14, "name": "Fantasy"}, {"id": 28, "na...	559	[{"id": 851, "name": "dual identity"}, {"id": ...	en	Spider-Man 3	The seemingly invincible Spider-Man goes up ag...	115.699814	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"iso_3166_1": " "name": "United St
6	15000000.0	[{"id": 16, "name": "Animation"}, {"id": 10751...	38757	[{"id": 1562, "name": "hostage"}, {"id": 2343,...	en	Tangled	When the kingdom's most wanted-and most charmi...	48.681969	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"iso_3166_1": " "name": "United St
7	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	99861	[{"id": 8828, "name": "marvel comic"}, {"id": ...	en	Avengers: Age of Ultron	When Tony Stark tries to jumpstart a dormant p...	134.279229	[{"name": "Marvel Studios", "id": 420}, {"name...	[{"iso_3166_1": " "name": "United St

5 rows × 22 columns

Make a new column for runtime types :

```
0 =>
duration <= 40        => Short Movies
1 => 40 < duration <= 70  => Mediam Duration Movies
2 => duration > 70        => Long Duration Movies
```

In [35]:

```
duration_genres = np.array([])
for i in df['runtime']:
    if (i<=40):
        duration_genres = np.append(duration_genres, 0)
    elif (i>40 and i<=75):
```

```
        duration_genres = np.append(duration_genres, 1)
    if (i>75):
        duration_genres = np.append(duration_genres, 2)
```

In [36]:
```
duration_genres
```

Out[36]:
```
array([2., 2., 2., ..., 2., 2., 2.])
```

In [37]:
```
df['duration_genres'] = duration_genres
```

In [38]:
```
df.head()
```

Out[38]:

	budget	genres	id	keywords	original_language	original_title	overview	popularity	production_companies	production_count
2	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"iso_3166_1": " "name": "Un Kingdo
4	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	John Carter	John Carter is a war-weary, former military ca...	43.926995	[{"name": "Walt Disney Pictures", "id": 2}]	[{"iso_3166_1": " "name": "United St
5	15000000.0	[{"id": 14, "name": "Fantasy"}, {"id": 28, "na...	559	[{"id": 851, "name": "dual identity"}, {"id": ...	en	Spider-Man 3	The seemingly invincible Spider-Man goes up ag...	115.699814	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"iso_3166_1": " "name": "United St
6	15000000.0	[{"id": 16, "name": "Animation"}, {"id": 10751...	38757	[{"id": 1562, "name": "hostage"},	en	Tangled	When the kingdom's most wanted-	48.681969	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"iso_3166_1": " "name": "United St

	budget	genres	id	keywords	original_language	original_title	overview	popularity	production_companies	production_count
				{"id": 2343,...			and most charmi...			
7	15000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	99861	[{"id": 8828, "name": "marvel comic"}, {"id": ...	en	Avengers: Age of Ultron	When Tony Stark tries to jumpstart a dormant p...	134.279229	[{"name": "Marvel Studios", "id": 420}, {"name...	[{"iso_3166_1": " "name": "United St

5 rows × 23 columns

Genre Extraction function : from raw data for the creation of tags :

In [39]:
```
def convert(obj):
    L = []
    for i in ast.literal_eval(obj):
        L.append(i['name'])
    return L
```

In [40]:
```
#FirsT Data Frame :

df['genres'] = df['genres'].apply(convert)
df['keywords'] = df['keywords'].apply(convert)
df['spoken_languages'] = df['spoken_languages'].apply(convert)
df['production_countries'] = df['production_countries'].apply(convert)
df['production_companies'] = df['production_companies'].apply(convert)
#-----------------------------------------------------------------------------------------
```

In [41]:
```
df.head()
```

Out[41]:

	budget	genres	id	keywords	original_language	original_title	overview	popularity	production_companies	production_countri
2	15000000.0	[Action, Adventure, Crime]	206647	[spy, based on novel, secret agent, sequel, mi...	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[Columbia Pictures, Danjaq, B24]	[United Kingdo United States Americ
4	15000000.0	[Action, Adventure, Science Fiction]	49529	[based on novel, mars, medallion, space travel...	en	John Carter	John Carter is a war-weary, former military ca...	43.926995	[Walt Disney Pictures]	[United States Americ
5	15000000.0	[Fantasy, Action, Adventure]	559	[dual identity, amnesia, sandstorm, love of on...	en	Spider-Man 3	The seemingly invincible Spider-Man goes up ag...	115.699814	[Columbia Pictures, Laura Ziskin Productions, ...	[United States Americ
6	15000000.0	[Animation, Family]	38757	[hostage, magic, horse, fairy tale, musical, p...	en	Tangled	When the kingdom's most wanted-and most charmi...	48.681969	[Walt Disney Pictures, Walt Disney Animation S...	[United States Americ
7	15000000.0	[Action, Adventure, Science Fiction]	99861	[marvel comic, sequel, superhero, based on com...	en	Avengers: Age of Ultron	When Tony Stark tries to jumpstart a dormant p...	134.279229	[Marvel Studios, Prime Focus, Revolution Sun S...	[United States Americ

5 rows × 23 columns

Function for extracting top(first) 8 actors from the movie :

In [42]:
```
def convert_actors(obj):
    L = []
    counter = 0
    for i in ast.literal_eval(obj):
        if counter !=8:
            L.append(i['name'])
            counter+=1
        else:
            break
    return L
```

In [43]:
```
df['cast'] = df['cast'].apply(convert_actors)
```

Function to fetch the director of movie from the crew column :

In [44]:
```
def fetch_director(obj):
    L = []
    for i in ast.literal_eval(obj):
        if i['job'] == 'Director':
            L.append(i['name'])
            break
    return L
```

In [45]:
```
df['crew'] = df['crew'].apply(fetch_director)
df['overview'] = df['overview'].apply(lambda x:x.split())
```

Remove spaces between words :

In [46]:
```
df['genres'] = df['genres'].apply(lambda x:[i.replace(" ","") for i in x])
df['keywords'] = df['keywords'].apply(lambda x:[i.replace(" ","") for i in x])
df['cast'] = df['cast'].apply(lambda x:[i.replace(" ","") for i in x])
df['crew'] = df['crew'].apply(lambda x:[i.replace(" ","") for i in x])
```

In [47]:
```
df['features'] = df['overview'] + df['genres'] + df['keywords'] + df['cast'] + df['crew']
df['features'] = df['features'].apply(lambda x:" ".join(x))
```

Lower casing all the alphabets in the tags column :

In [48]:
```
df['features'] = df['features'].apply(lambda x:x.lower())
```

Apply Steming to remove similarities/duplications in words list :

In [49]:
```
import nltk
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

def xStem(txt):
    y = []
    for x in txt.split():
        y.append(ps.stem(x))
    return " ".join(y)
```

In [50]:
```
df['features'] = df['features'].apply(xStem)
```

Convert text to matrix :

In [51]:
```
from sklearn.feature_extraction.text import HashingVectorizer

hv=HashingVectorizer(stop_words="english",n_features=7000)
hv_vector= hv.fit_transform(df['features']).toarray()
```

In [52]:
```
similarity = cosine_similarity(hv_vector)
```

In [53]:
```
pd.DataFrame(similarity,index=df['title'],columns=df['title'])
```

Out[53]:

title	Spectre	John Carter	Spider-Man 3	Tangled	Avengers: Age of Ultron	Harry Potter and the Half-Blood Prince	Batman v Superman: Dawn of Justice	Quantum of Solace	Pirates of the Caribbean: Dead Man's Chest	The Lone Ranger	...	On The Downlow	Sanctuary Quite a Conundrum
title													
Spectre	1.000000	0.055670	0.054447	0.015386	0.095730	0.046524	0.057438	0.265908	0.045980	0.017293	...	0.049568	0.000000
John Carter	0.055670	1.000000	0.074092	0.037689	0.140694	0.056980	0.062531	0.108556	0.112628	0.028239	...	0.020236	0.000000
Spider-Man 3	0.054447	0.074092	1.000000	0.024574	0.122312	0.055728	0.122312	0.091003	0.073435	0.013809	...	0.000000	0.000000
Tangled	0.015386	0.037689	0.024574	1.000000	0.038886	0.031497	0.051848	-0.012859	0.000000	0.058537	...	0.000000	0.000000
Avengers: Age of Ultron	0.095730	0.140694	0.122312	0.038886	1.000000	0.039193	0.145161	0.080003	0.096837	0.043704	...	0.000000	0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
El Mariachi	0.032141	0.118094	0.064166	0.021760	0.108306	0.098693	0.027077	0.161165	0.097538	0.036684	...	0.000000	0.038292
Newlyweds	-0.036564	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.04356
Signed, Sealed, Delivered	0.018282	0.014927	0.014599	0.012377	0.000000	0.037424	0.015401	0.000000	-0.018493	0.041731	...	0.039873	0.02178
Shanghai Calling	0.019627	0.048075	0.047019	0.039862	0.049602	0.080354	0.016534	0.065609	0.039707	0.000000	...	0.000000	0.02338
My Date with Drew	0.000000	0.000000	0.031083	0.000000	0.065583	0.079682	0.016396	0.048795	0.000000	0.000000	...	0.000000	0.000000

4612 rows × 4612 columns

In [54]:
```
distances = similarity[32] ## Similarites for the movie
sorted(distances,reverse=True)[0:10]
```

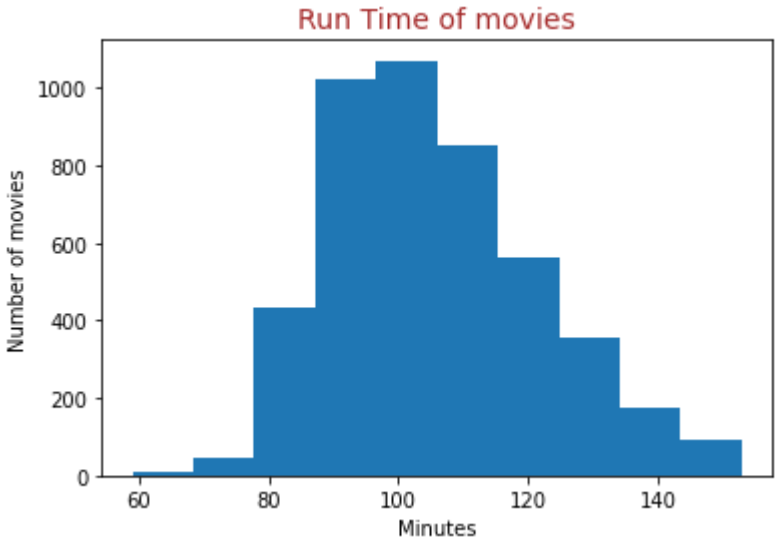
Out[54]:
```
[1.0000000000000002,
 0.3360537729058417,
 0.2832856927186045,
 0.2783349703706405,
 0.26122949691608693,
 0.2545454545454546,
 0.24003840921845832,
 0.23994948963429277,
 0.2397457108377597,
 0.23472626340651012]
```

In [55]:
```
def recommend(movie):
    movie_index = df[df['title'] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)),reverse=True, key=lambda x:x[1])[1:16]
    mov=[]
    id=[]
    scores=[]
    for i in movies_list:
        mov.append(df.iloc[i[0]].title)
        id.append(df.iloc[i[0]].movie_id)
        scores.append(i[1])
    dic={'movie_id':id,'title':mov,'Similarity Score':scores}
    return pd.DataFrame(dic)
```

Visulization :

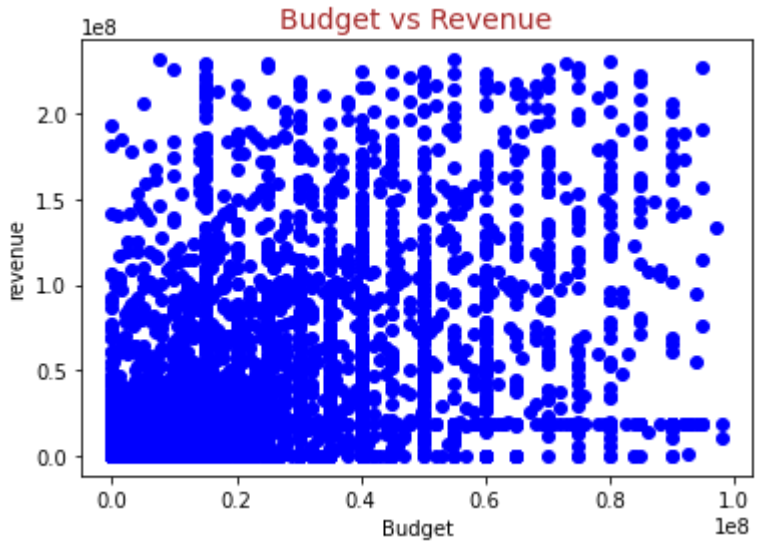
In [56]:
```
#Histogram to represent runtime of movies
#plt.hist(df['runtime'])
plt.hist(df['runtime'], bins=10)
plt.title('Run Time of movies', fontdict={'fontsize':14, 'color':'brown'})
```

```
plt.xlabel("Minutes")
plt.ylabel('Number of movies')
plt.show()
```



In [57]:

```
#Scatter plot to compare between Budget and Revenue
plt.scatter(df['budget'], df['revenue'], marker='o', c='blue')

plt.title('Budget vs Revenue', fontdict={'fontsize':14, 'color':'brown'})
plt.xlabel("Budget")
plt.ylabel('revenue')
plt.show()
```

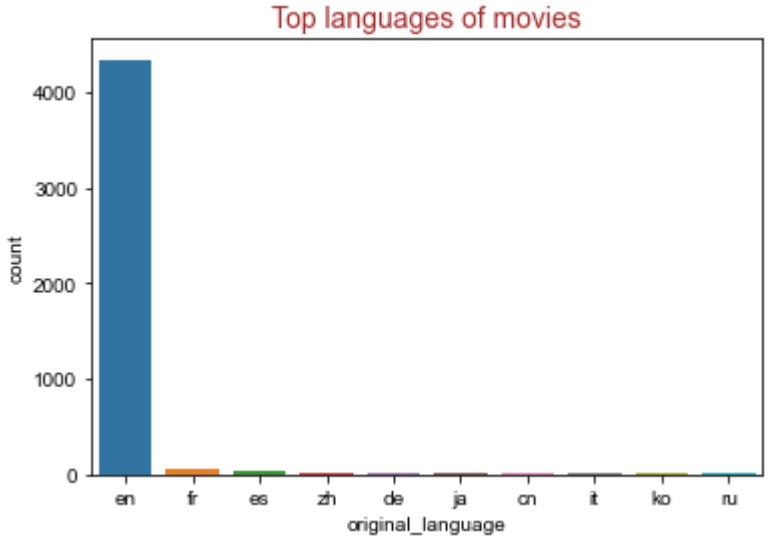


In [58]:

```
#countplot to show the top languages in the movies
sns.countplot(x='original_language',data=df,order=pd.value_counts(df['original_language']).iloc[:10].index)
sns.set(rc = {'figure.figsize':(12,8)})
plt.title('Top languages of movies', fontdict={'fontsize':14, 'color':'brown'})
```

Out[58]:

```
Text(0.5, 1.0, 'Top languages of movies')
```



In [59]:
```
language=df.original_language.value_counts()
```

In [60]:
```
#Pie chart to represent the top language
from IPython.display import HTML
import plotly.express as px
plt.figure(figsize=(20,14))
fig= px.pie(df,
            values=language.iloc[:10].values,
            names=language.iloc[:10].index,
            title='Top 10 Languages',
            height=1050,
            width=700)
HTML(fig.to_html())
```

Out[60]:

<Figure size 1440x1008 with 0 Axes>

In [61]:

```
#This bar used to represent the number of movies that has a specific ratings
fig=px.bar(df,
           x=df['vote_average'].value_counts().index,
           y=df['vote_average'].value_counts(),
           title='Overall Ratings',
           text=df['vote_average'].value_counts(),
```

```
            height=700
            )
HTML(fig.to_html())
```

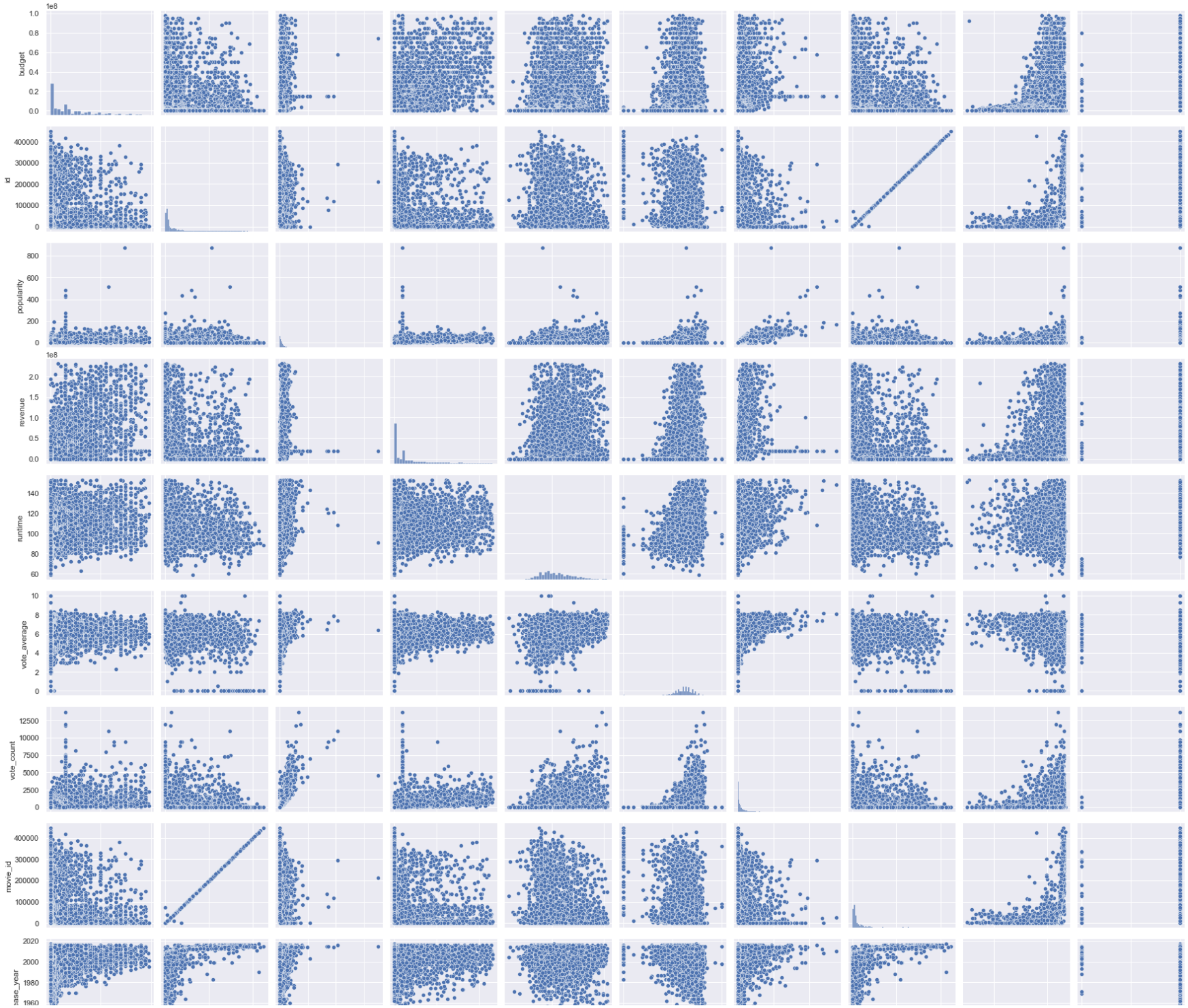
Out[61]:

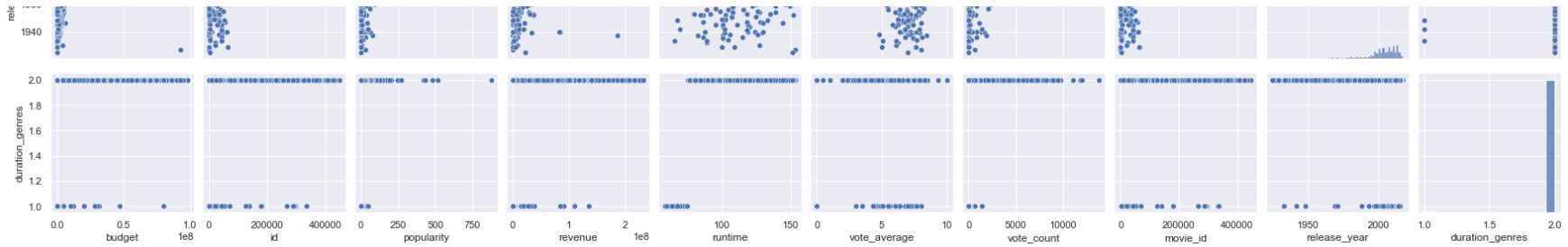
In [62]:

```
#The pairwise scatterplot used to show the relation between every columns
import seaborn as sns
# Create the default pairplot
sns.pairplot(df)
```

Out[62]:

```
<seaborn.axisgrid.PairGrid at 0x133242a1490>
```


budget
id
popularity
revenue
runtime
vote_average
vote_count
movie_id
release_year
1e8



In [63]:

```
all_countries=[]
for i in df['production_countries']:
    for j in i:
        all_countries.append(j)
#all_countries
```

In [64]:

```
from collections import Counter
cont = Counter()

for text in all_countries:
    cont[text] += 1
# See most common ten countries
cont.most_common()
```

Out[64]:

```
[('United States of America', 3830),
 ('United Kingdom', 605),
 ('Germany', 314),
 ('France', 298),
 ('Canada', 253),
 ('Australia', 109),
 ('Spain', 69),
 ('Italy', 59),
 ('China', 58),
 ('Japan', 56),
 ('Hong Kong', 46),
 ('India', 38),
 ('Ireland', 37),
 ('Mexico', 29),
 ('Belgium', 25),
 ('Czech Republic', 24),
 ('New Zealand', 22),
 ('South Africa', 20),
 ('Denmark', 20),
```

```
('Switzerland', 18),
('South Korea', 18),
('Sweden', 18),
('Russia', 17),
('Netherlands', 17),
('United Arab Emirates', 14),
('Hungary', 13),
('Brazil', 13),
('Norway', 13),
('Romania', 11),
('Luxembourg', 11),
('Argentina', 9),
('Poland', 6),
('Iceland', 6),
('Israel', 6),
('Finland', 5),
('Austria', 5),
('Thailand', 5),
('Taiwan', 4),
('Morocco', 4),
('Bulgaria', 4),
('Iran', 4),
('Bahamas', 3),
('Malta', 3),
('Greece', 3),
('Jamaica', 2),
('Slovenia', 2),
('Pakistan', 2),
('Malaysia', 2),
('Peru', 2),
('Kazakhstan', 2),
('Chile', 2),
('Slovakia', 2),
('Colombia', 2),
('Dominica', 1),
('Monaco', 1),
('Tunisia', 1),
('Philippines', 1),
('Bosnia and Herzegovina', 1),
('Portugal', 1),
('Singapore', 1),
('Aruba', 1),
('Serbia', 1),
('Ukraine', 1),
('Panama', 1),
```

```
('Lithuania', 1),
('Cambodia', 1),
('Fiji', 1),
('Serbia and Montenegro', 1),
('Turkey', 1),
('Nigeria', 1),
('Cyprus', 1),
('Jordan', 1),
('Bolivia', 1),
('Ecuador', 1),
('Egypt', 1),
('Bhutan', 1),
('Lebanon', 1),
('Kyrgyz Republic', 1),
('Algeria', 1),
('Indonesia', 1),
('Guyana', 1),
('Guadaloupe', 1),
('Afghanistan', 1),
('Angola', 1),
('Dominican Republic', 1),
('Cameroon', 1),
('Kenya', 1)]
```

In [65]:

```
count_freq = pd.DataFrame(cont.most_common(10), columns=['countries', 'count'])
count_freq
```

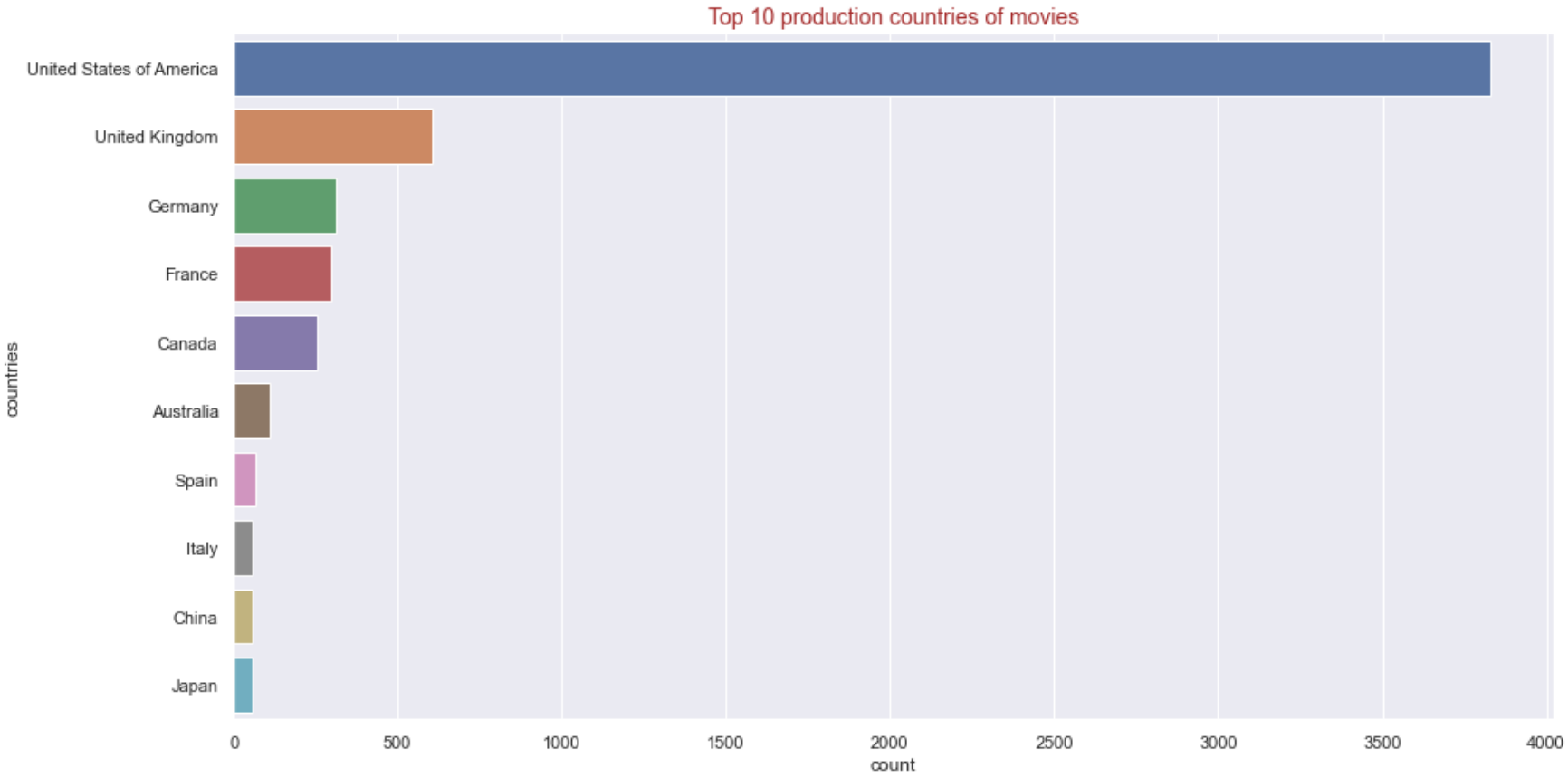
Out[65]:

	countries	count
0	United States of America	3830
1	United Kingdom	605
2	Germany	314
3	France	298
4	Canada	253
5	Australia	109
6	Spain	69
7	Italy	59
8	China	58

	countries	count
9	Japan	56

In [66]:

```
#The barplot used to show the Top 10 production countries
plt.figure(figsize=(15,8))
plt.title('Top 10 production countries of movies', fontdict={'fontsize':14, 'color':'brown'})
sns.barplot(x='count', y='countries', data=count_freq);
```



Recommendation

How the recommender function works ?

In [67]:
```
# we first get the index of the movie for example Alice in Wonderland index is 32
movie_index = df[df['title'] == 'Alice in Wonderland'].index[0]
movie_index
```

Out[67]:
```
32
```

In [68]:
```
# we get the similarities for the movie
distances = similarity[32]
# we sort the similarities in descending order
sorted(distances,reverse=True)[0:10]
```

Out[68]:
```
[1.0000000000000002,
 0.3360537729058417,
 0.2832856927186045,
 0.2783349703706405,
 0.26122949691608693,
 0.2545454545454546,
 0.24003840921845832,
 0.23994948963429277,
 0.2397457108377597,
 0.23472626340651012]
```

In [69]:
```
#then we get the top 3 movies and their Index and similarity score
num=3
movies_list = sorted(list(enumerate(distances)),reverse=True, key=lambda x:x[1])[1:num+1]
movies_list
```

Out[69]:
```
[(525, 0.3360537729058417),
 (885, 0.2832856927186045),
 (135, 0.2783349703706405)]
```

In [70]:
```
# finally we access them using Index
for i in movies_list:
 print(df.iloc[i[0]].title)
 print(df.iloc[i[0]].movie_id)
```

```
Cars
920
The Book of Life
228326
```

```
Kung Fu Panda 3
140300
```

In [71]:

```python
#generate random title from the list of movies
import random
def random_title():
    return random.choice(df['title'])

random_title()
```

Out[71]:

```
'The Cookout'
```

In [72]:

```python
movie_name='Alice in Wonderland'
popular_movies = recommend(movie_name)
popular_movies
```

Out[72]:

	movie_id	title	Similarity Score
0	920	Cars	0.336054
1	228326	The Book of Life	0.283286
2	140300	Kung Fu Panda 3	0.278335
3	332	Inspector Gadget	0.261229
4	13053	Bolt	0.254545
5	49519	The Croods	0.240038
6	10477	Driven	0.239949
7	17711	The Adventures of Rocky & Bullwinkle	0.239746
8	145220	Muppets Most Wanted	0.234726
9	20542	Delgo	0.234726
10	10996	Stuart Little 2	0.230170
11	12703	The Brown Bunny	0.227260
12	62206	30 Minutes or Less	0.224733
13	2270	Stardust	0.222475

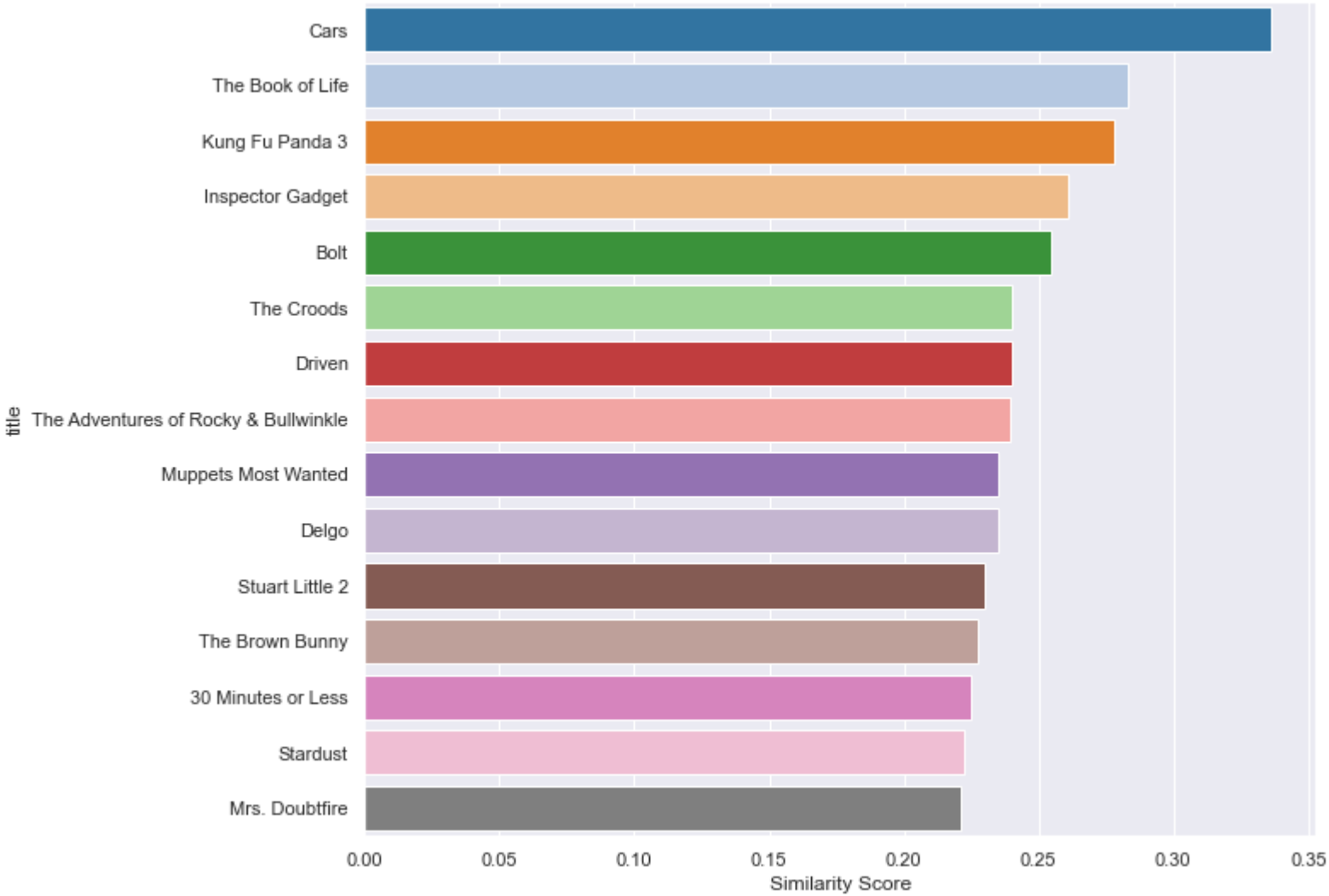
	movie_id	title	Similarity Score
14	788	Mrs. Doubtfire	0.220946

Bar plot titles and similarity scores :

In [73]:

```
import pandas as pd
plt.rcParams['figure.figsize'] = (10, 9)
sns.barplot(popular_movies['Similarity Score'],popular_movies['title'],palette='tab20')
plt.show();
```

```
C:\Users\com\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```



In [74]:

```
import requests
from IPython.display import Image, HTML, display
def movie_display(popular_movies):
 getList_name = {}
 for x, xRows in popular_movies.iterrows():
    # we get the movie id from the dataframe and we use it to get the movie poster
    getResponse = requests.get('https://api.themoviedb.org/3/movie/{}?api_key=c0bda0be71f7815fd6ba2eb5f5c86fd8'.format(xR
    getData = getResponse.json() # we request the data from the API and convert it to json

    # a bug fixed because sometimes there are is no poster so it returrns error
```

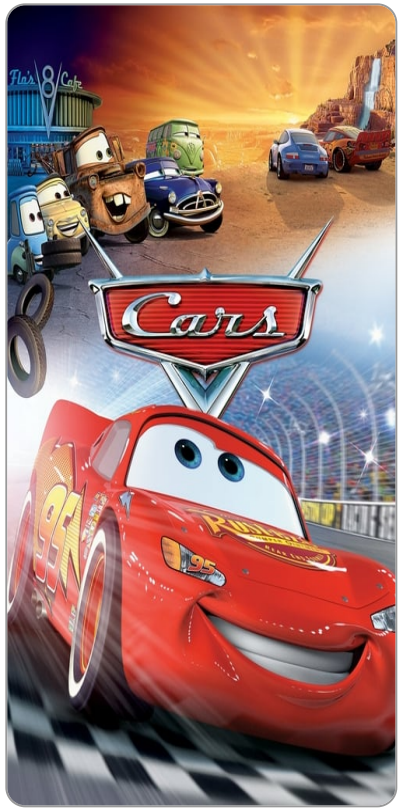
```
    if getData['poster_path']==None:
     continue
    else:
     getPath = "http://image.tmdb.org/t/p/w500" + getData['poster_path']    # get the path of the poster
     getList_name[xRows['title']] = getPath


display(HTML(f"""  < <div style="font-size:24px; font-weight:Bold; color:#fff; text-align:center; padding-top:8px; heigh
# in here he loops on the number of movies to be recommended which is in num_recommend variable in recommend() function
for i in range(0,popular_movies.shape[0],5):


 display( HTML(f"""
                   <table>
                   <tr>
                       <td><img src={list(getList_name.values())[i]} style='border-radius:10px; height:400px; width:575p
                       <td><img src={list(getList_name.values())[i+2]} style='border-radius:10px; height:400px; width:57
                       <td><img src={list(getList_name.values())[i+3]} style='border-radius:10px; height:400px; width:57
                       <td><img src={list(getList_name.values())[i+4]} style='border-radius:10px; height:400px; width:57

                   </tr>

                       <td><div style="height:60px; padding-top:15px; text-align:center; font-size:14px; font-weight:bol
                       <td><div style="height:60px; padding-top:15px; text-align:center; font-size:14px; font-weight:bol
                       <td><div style="height:60px; padding-top:15px; text-align:center; font-size:14px; font-weight:bol
                       <td><div style="height:60px; padding-top:15px; text-align:center; font-size:14px; font-weight:bol
                   </tr>

               </table>"""))
```

In [75]:

```
movie_display(recommend(movie_name,))
```

<

Alice in Wonderland



Cars

Kung Fu Panda 3

Inspector Gadget

Bolt

The Croods

The Adventures of Rocky & Bullwinkle

Muppets Most Wanted

Delgo

Stuart Little 2

30 Minutes or Less

Stardust

Mrs. Doubtfire

Machine Learning Model

In this section we will further clean the data and then we will create a model that predicts the rating of a movie

Cleaning the data

Importing needed packages

In [76]:
```
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')
from IPython.display import HTML
```

Importing the csv files

In [77]:
```
#we are reopening the files in new dataframes, this was just made because it was easier to collaborate this way
df_score=pd.read_csv('tmdb_5000_movies.csv')
df_score2=pd.read_csv('tmdb_5000_credits.csv')
```

Merging the two files together

In [78]:
```
df_score = df_score.merge(df_score2,on='title')
#the title will be like our key that merges the columns to the correct position, just like the database forgein key
```

Droping things with low correlation with vote_average

We referred to the correlation matrix at the start of the section. It must be clarified that we have droped the budget and revenue, even though they seem to be important and a good predictor for a movies success. However, the correlation matrix displayed very poor relationship of between budget or revenue with vote_average

In [79]:
```
df_score.drop(['vote_count', 'movie_id', 'runtime','homepage','budget','revenue','overview','release_date','spoken_langua
```

In [80]:
```
df_score.head(2)
```

Out[80]:

	genres	id	keywords	original_title	popularity	production_companies	production_countries	vote_average	cast	
0	[{"id": 28, "name": "Action"},	19995	[{"id": 1463, "name": "culture	Avatar	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166_1": "US", "name": "United States o...	7.2	[{"cast_id": 242, "character":	"52fe480092

	genres	id	keywords	original_title	popularity	production_companies	production_countries	vote_average	cast	
	{"id": 12, "nam...		clash"}, {"id":...						"Jake Sully", "...	
1	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	Pirates of the Caribbean: At World's End	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"iso_3166_1": "US", "name": "United States o...	6.9	[{"cast_id": 4, "character": "Captain Jack Spa...	"52fe4232c3a

Change the json files to lists

1. Change the list of dictionaries to the needed extractions in a list
2. Pinpoint the rows that we will need to drop because they are empty
3. Update the rows of our dataframe, to maintain the integrity of the rows' numbering

In [81]:

```
#a set that takes all the empty rows we need to remove
empty_rows=set()
def json_to_list(col):
    # if crew we will parse from a specific key
    if col == 'crew':
        # will move through the entire dataframe
        for i in range(0, len(df_score)):
            film_director =""
            # change the row content into a list
            df_list_ = eval(df_score.loc[i,col])
            for j in range(0,len(df_list_)):
                # parse the parts with directors only
                if df_list_[j]['job'] == 'Director':
                    film_director = df_list_[j]['name']
                    df_score.loc[i,col] = str(film_director)
                    # break because we will not need to search more
                    break
            if film_director == "":
                # if we are empty then add to the empty set
                empty_rows.add(i)
                df_score.loc[i,col] = ''
    elif col == 'cast':
        # loop through all the dataframe
```

```
        for i in range(0, len(df_score)):
            # this list will hold the parsed data
            list_ = []
            # we will convert the string list of dictionaries, into a list of dictionary that we can access
            df_list_ = eval(df_score.loc[i,col])
            # we will loop through this list of dictionary to extract the needed data
            for j in range(0, len(df_list_)):
                temp_ = df_list_[j]["name"]
                list_.append(temp_.lower())
            # if there is nothing then we will add this to rows that need to be deleted
            if len(df_list_) == 0:
                empty_rows.add(i)
            # we are only interested in the first 4 actors
            selected = list_[0:4]
            #we add this to the row
            df_score.loc[i,col] =  str(selected)
    else:
        for i in range(0, len(df_score)):
            #loop through the entire dataframe
            list_ = []
            #we will take each content in the row and change it to a list of dictionaries to be able to parse
            df_list_ = eval(df_score.loc[i,col])
            for j in range(0, len(df_list_)):
                # we will select the part of data we are interested in
                temp_ = df_list_[j]["name"]
                list_.append(temp_.lower())
            # this will happen when our list is empty, thus we will need to add it to the rows to be deleted
            if len(df_list_) == 0:
                empty_rows.add(i)
            df_score.loc[i,col] = str(list_)
```

In [82]:

```
# we change the jason text to the list
json_to_list('genres')
json_to_list('keywords')
json_to_list('production_companies')
json_to_list('production_countries')
json_to_list('cast')
json_to_list('crew')
# we delete the rows that have any empty rows or columns
empty_rows = list(empty_rows)
df_score.drop(empty_rows, axis=0, inplace=True)
```

In [83]:
```
#we need to reset the index of the dataframe
df_score.reset_index(level=None, drop=True, inplace=True, col_level=0)
```

In [84]:
```
df_score
```

Out[84]:

	genres	id	keywords	original_title	popularity	production_companies	production_countries	vote_average	cast	
0	['action', 'adventure', 'fantasy', 'science fi...	19995	['culture clash', 'future', 'space war', 'spac...	Avatar	150.437577	['ingenious film partners', 'twentieth century...	['united states of america', 'united kingdom']	7.2	['sam worthington', 'zoe saldana', 'sigourney ...	
1	['adventure', 'fantasy', 'action']	285	['ocean', 'drug abuse', 'exotic island', 'east...	Pirates of the Caribbean: At World's End	139.082615	['walt disney pictures', 'jerry bruckheimer fi...	['united states of america']	6.9	['johnny depp', 'orlando bloom', 'keira knight...	
2	['action', 'adventure', 'crime']	206647	['spy', 'based on novel', 'secret agent', 'seq...	Spectre	107.376788	['columbia pictures', 'danjaq', 'b24']	['united kingdom', 'united states of america']	6.3	['daniel craig', 'christoph waltz', 'léa seydo...	
3	['action', 'crime', 'drama', 'thriller']	49026	['dc comics', 'crime fighter', 'terrorist', 's...	The Dark Knight Rises	112.312950	['legendary pictures', 'warner bros.', 'dc ent...	['united states of america']	7.6	['christian bale', 'michael caine', 'gary oldm...	Cl
4	['action', 'adventure', 'science fiction']	49529	['based on novel', 'mars', 'medallion', 'space...	John Carter	43.926995	['walt disney pictures']	['united states of america']	6.1	['taylor kitsch', 'lynn collins', 'samantha mo...	
...	...	...	...	...	...	...	...	...	...	
4171	['drama']	124606	['gang', 'audition', 'police	Bang	0.918116	['asylum films', 'fm entertainment', 'eagle ey...	['united states of america']	6.0	['darling narita', 'peter	A

	genres	id	keywords	original_title	popularity	production_companies	production_countries	vote_average	cast	
			fake', 'homeless'...						greene', 'michael ne...	
4172	['science fiction', 'drama', 'thriller']	14337	['distrust', 'garage', 'identity crisis', 'tim...	Primer	23.307949	['thinkfilm']	['united states of america']	6.9	['shane carruth', 'david sullivan', 'casey goo...	
4173	['action', 'crime', 'thriller']	9367	['united states–mexico barrier', 'legs', 'arms...	El Mariachi	14.269792	['columbia pictures']	['mexico', 'united states of america']	6.6	['carlos gallardo', 'jaime de hoyos', 'peter m...	I
4174	['comedy', 'drama', 'romance', 'tv movie']	231617	['date', 'love at first sight', 'narration', '...	Signed, Sealed, Delivered	1.444476	['front street pictures', 'muse entertainment ...	['united states of america']	7.0	['eric mabius', 'kristin booth', 'crystal lowe...	Sc
4175	['documentary']	25975	['obsession', 'camcorder', 'crush', 'dream girl']	My Date with Drew	1.929883	['rusty bear entertainment', 'lucky crow films']	['united states of america']	6.3	['drew barrymore', 'brian herzlinger', 'corey ...	H

4176 rows × 10 columns

Generate binary lists 🔢

In [85]:

```
#function change to binary representations
def binary(genre_list, uniqueList):
    #this is the list that will contain the zeros and ones representing the presence or no presence of an entity
    binaryList = []
    for genre in uniqueList:
        if genre in genre_list:
            binaryList.append(1)
        else:
```

```
            binaryList.append(0)

    return binaryList
```

In [86]:

```
#function that gets the unique value
def unique_list(col):
    # we used a set to not count any repeats
    unique_set = set()
    for i in range(0, len(df_score)):
        #loop on the entire dataframe
        l = df_score.iloc[i,col]
        for j in range(len(l)):
            #add the entities in the list to the set
            unique_set.add(l[j])
    unique_list = list(unique_set)
    return unique_list
```

In [87]:

```
#converting all the cells that contain a  list of strings to a list of binary to be able to use it in the score predictor

genreL = unique_list(0)
df_score['Genres_bin'] = df_score.iloc[:,0].apply(lambda x: binary(x,genreL))

castL = unique_list(8)
df_score['Actors_bin'] = df_score.iloc[:,8].apply(lambda x: binary(x,castL))

crewL = unique_list(9)
df_score['Directors_bin'] = df_score.iloc[:,9].apply(lambda x: binary(x,crewL))

keywordsL = unique_list(2)
df_score['Keywords_bin'] = df_score.iloc[:,2].apply(lambda x: binary(x,keywordsL))
```

In [88]:

```
df_score.head(3)
```

Out[88]:

	genres	id	keywords	original_title	popularity	production_companies	production_countries	vote_average	cast	crew
0	['action', 'adventure', 'fantasy', 'science fi...	19995	['culture clash', 'future', 'space	Avatar	150.437577	['ingenious film partners', 'twentieth century...	['united states of america', 'united kingdom']	7.2	['sam worthington', 'zoe saldana', 'sigourney ...	James Cameron

	genres	id	keywords	original_title	popularity	production_companies	production_countries	vote_average	cast	crew
			war', 'spac...							
1	['adventure', 'fantasy', 'action']	285	['ocean', 'drug abuse', 'exotic island', 'east...	Pirates of the Caribbean: At World's End	139.082615	['walt disney pictures', 'jerry bruckheimer fi...	['united states of america']	6.9	['johnny depp', 'orlando bloom', 'keira knight...	Gore Verbinski
2	['action', 'adventure', 'crime']	206647	['spy', 'based on novel', 'secret agent', 'seq...	Spectre	107.376788	['columbia pictures', 'danjaq', 'b24']	['united kingdom', 'united states of america']	6.3	['daniel craig', 'christoph waltz', 'léa seydo...	Sam Mendes

Prediction Model 🔢

Similarity Function 👀

In [89]:

```
from scipy import spatial
#this is a function that uses cosine similarity to find how similar two movies are
# the similarity is upon
#1) genre
#2) Actors
#3) Directors
#4) Keywords

def Similarity(movieId1, movieId2):
    a = df_score.iloc[movieId1]
    b = df_score.iloc[movieId2]

    genresA = a['Genres_bin']
    genresB = b['Genres_bin']

    genreDistance = spatial.distance.cosine(genresA, genresB)

    scoreA = a['Actors_bin']
    scoreB = b['Actors_bin']
```

```
    scoreDistance = spatial.distance.cosine(scoreA, scoreB)

    directA = a['Directors_bin']
    directB = b['Directors_bin']
    directDistance = spatial.distance.cosine(directA, directB)

    wordsA = a['Keywords_bin']
    wordsB = b['Keywords_bin']
    wordsDistance = spatial.distance.cosine(wordsA, wordsB)
    # because we are adding 4 cosine similarity values the values will be between  0  and 4 inclusive
    return genreDistance + directDistance + scoreDistance + wordsDistance
```

In [90]:
```
Similarity(3,234) #lets check similarity between any 2 random movies
```

Out[90]:
```
0.9062687954957778
```

Score Predictor 🔮

1. Input the movie you want to predict its score
2. Find the similarity between all other movies
3. Find the top 10 similar movies
4. Find the average rating

In [91]:
```
#This is a function that calculates the similarity of a choosen movie and compares it with the rest, and finds the top
# 10 most similar movies and gets the average rating
def score_prdictor(movieID):
    #create a disctionary that key: the index of the row,  value: the similarity score
    similarity_dic = dict()
    #loop through the dataframe to get the similarity scores
    for i in range(0, len(df_score)):
        if i == movieID:
            #we don't want to consider the movieID that we chose itself, because it will already get a score of 0
            continue
        temp_similarity =  Similarity(movieID,i)
        similarity_dic[i]= temp_similarity
    #change the dictionary to a list of tuples
    similarity_list = list(similarity_dic.items())
    #sort the list according to the similarity score
    similarity_list = sorted(similarity_list, key = lambda x: x[1])[:10]
    #total_score is used so we can get the averahe vote prediction
```

```
    total_score = 0
    for i in similarity_list:
        temp = df_score.loc[int(i[0]), 'vote_average']
        total_score+= temp
    return round(total_score/10,2)
```

In [92]:
```
score_prdictor(4)
```

Out[92]: 6.64

Our Accuracy 📉

In [93]:
```
# real will store the actual vote_average
#predicted will store the predicted vote_average using cosine similarity function explained above
real =[]
predicted =[]

#we will test on the first 100 rows
for i in range(0,100):
    real.append(df_score.loc[i,'vote_average'])
    predicted.append(score_prdictor(i))
```

In [94]:
```
# created a new dataframe to have the data of real and predicted
df_accuracy = pd.DataFrame()
df_accuracy['Real']=real
df_accuracy['Predicted']=predicted
```

In [95]:
```
# we will calculate the percentage error
percentage_error =[]
for i in range(0, len(df_accuracy)):
    approx = df_accuracy.loc[i,'Predicted']
    exact = df_accuracy.loc[i,'Real']
    error = (abs(approx-exact)/exact)*100
    percentage_error.append(round(error,2))

df_accuracy["Percentage Error%"] = percentage_error
```

In [96]:
```
df_accuracy.head(3)
```

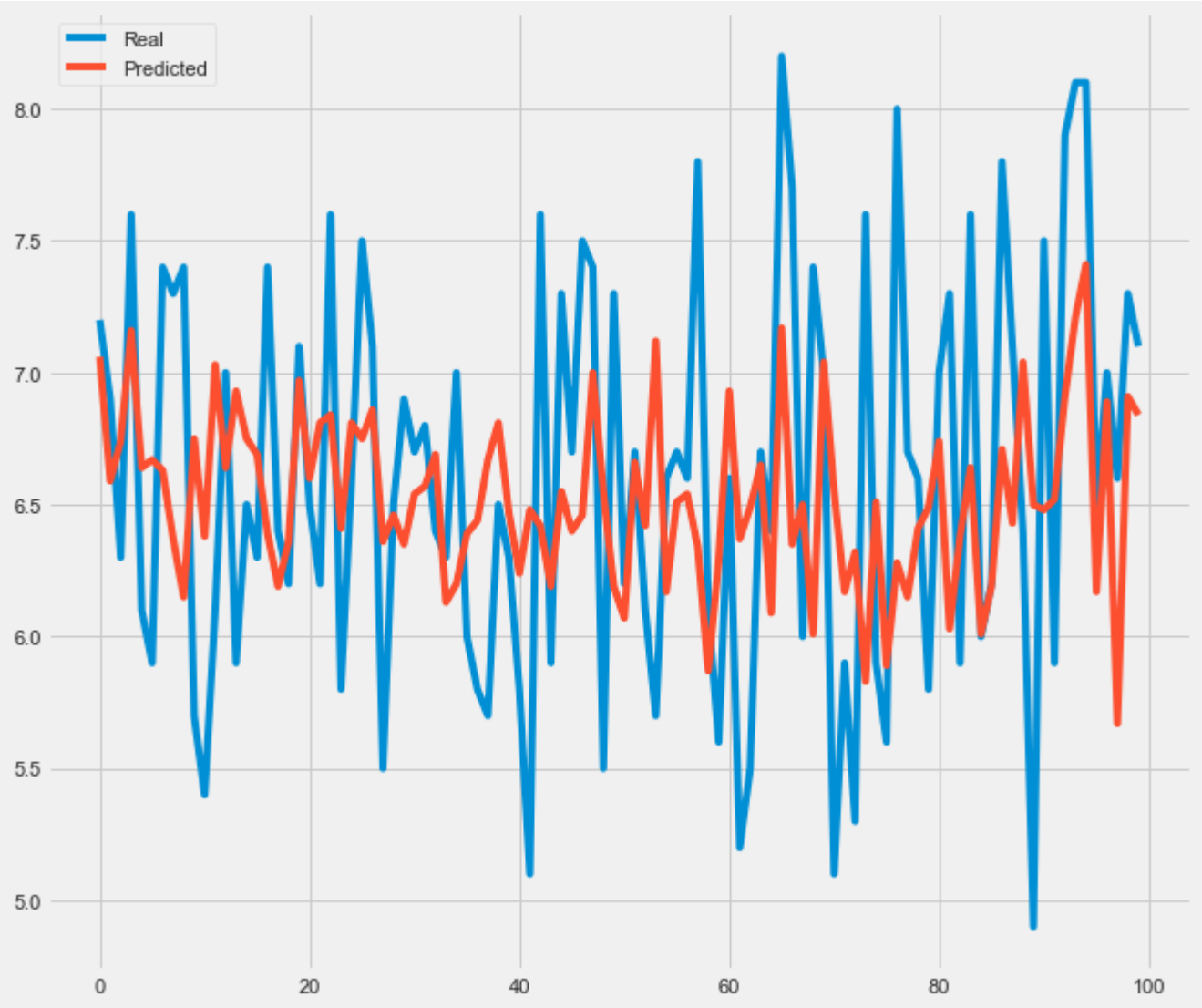
Out[96]:

	Real	Predicted	Percentage Error%
0	7.2	7.06	1.94
1	6.9	6.59	4.49
2	6.3	6.74	6.98

In [97]:
```
Average_error = sum(percentage_error)/len(percentage_error)
print("Average Percentage Error: "+str(round(Average_error,2))+" %")
```

```
Average Percentage Error: 9.66 %
```

In [98]:
```
df_accuracy.plot( y=['Real','Predicted'], kind = 'line')
plt.show()
```



Answering Questions

What are the movie genres available?

In [99]:

```
genres = pd.Series([categ for row in df['genres'] for categ in row])
#geners
```

In [100…

```
#list of all genres in the column "with repetition"
all_genres=[]
for i in genres:
    all_genres.append(i)
#all_genres
```

In [101…

```
#get all different geners in list
diff_genres = list( dict.fromkeys(all_genres) )
#print(diff_genres)
```

In [102…

```
#cloud of words of all genres
import matplotlib.pyplot as plt
from wordcloud import WordCloud
my_list=diff_genres
#convert list to string and generate
unique_string=(" ").join(my_list)
wordcloud = WordCloud(width = 1000, height = 500).generate(unique_string)
plt.figure(figsize=(15,8))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
plt.close()
```

Create a counter for the frequency table

In [103…

```
from collections import Counter
cnt = Counter()

for text in all_genres:
    cnt[text] += 1
# See most common ten words
cnt.most_common()
```

Out[103…

```
[('Drama', 2172),
 ('Comedy', 1705),
 ('Thriller', 1248),
 ('Action', 1122),
 ('Romance', 865),
 ('Adventure', 763),
```

```
('Crime', 674),
('ScienceFiction', 525),
('Horror', 507),
('Family', 504),
('Fantasy', 410),
('Mystery', 335),
('Animation', 232),
('Music', 175),
('History', 152),
('War', 116),
('Documentary', 101),
('Western', 72),
('Foreign', 32),
('TVMovie', 5)]
```

Create a frequency table

In [104…

```
#make a data frame of all words and there frequency
import pandas as pd
word_freq = pd.DataFrame(cnt.most_common(), columns=['words', 'count'])
word_freq
```

Out[104…

	words	count
0	Drama	2172
1	Comedy	1705
2	Thriller	1248
3	Action	1122
4	Romance	865
5	Adventure	763
6	Crime	674
7	ScienceFiction	525
8	Horror	507
9	Family	504
10	Fantasy	410
11	Mystery	335

	words	count
12	Animation	232
13	Music	175
14	History	152
15	War	116
16	Documentary	101
17	Western	72
18	Foreign	32
19	TVMovie	5

Create the plot

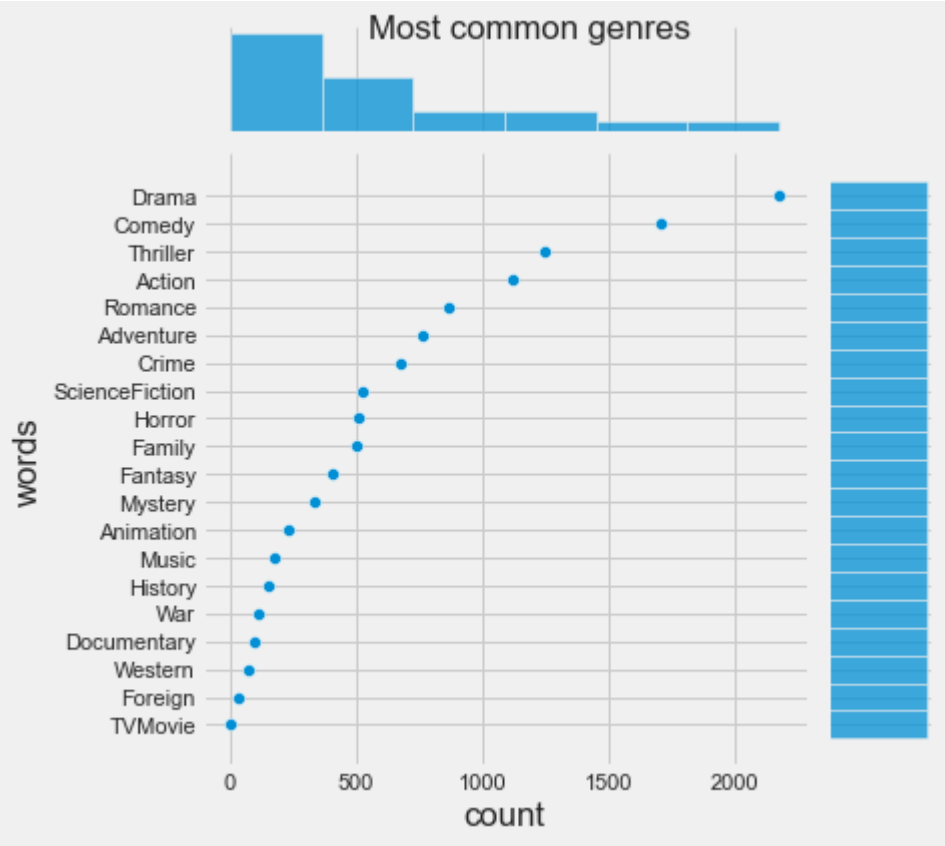
In [105…

```
plt.figure(figsize=(10,10))
p=sns.jointplot(x='count', y='words', data=word_freq);
p.fig.suptitle('Most common genres')
```

Out[105…

```
Text(0.5, 0.98, 'Most common genres')
```

```
<Figure size 720x720 with 0 Axes>
```

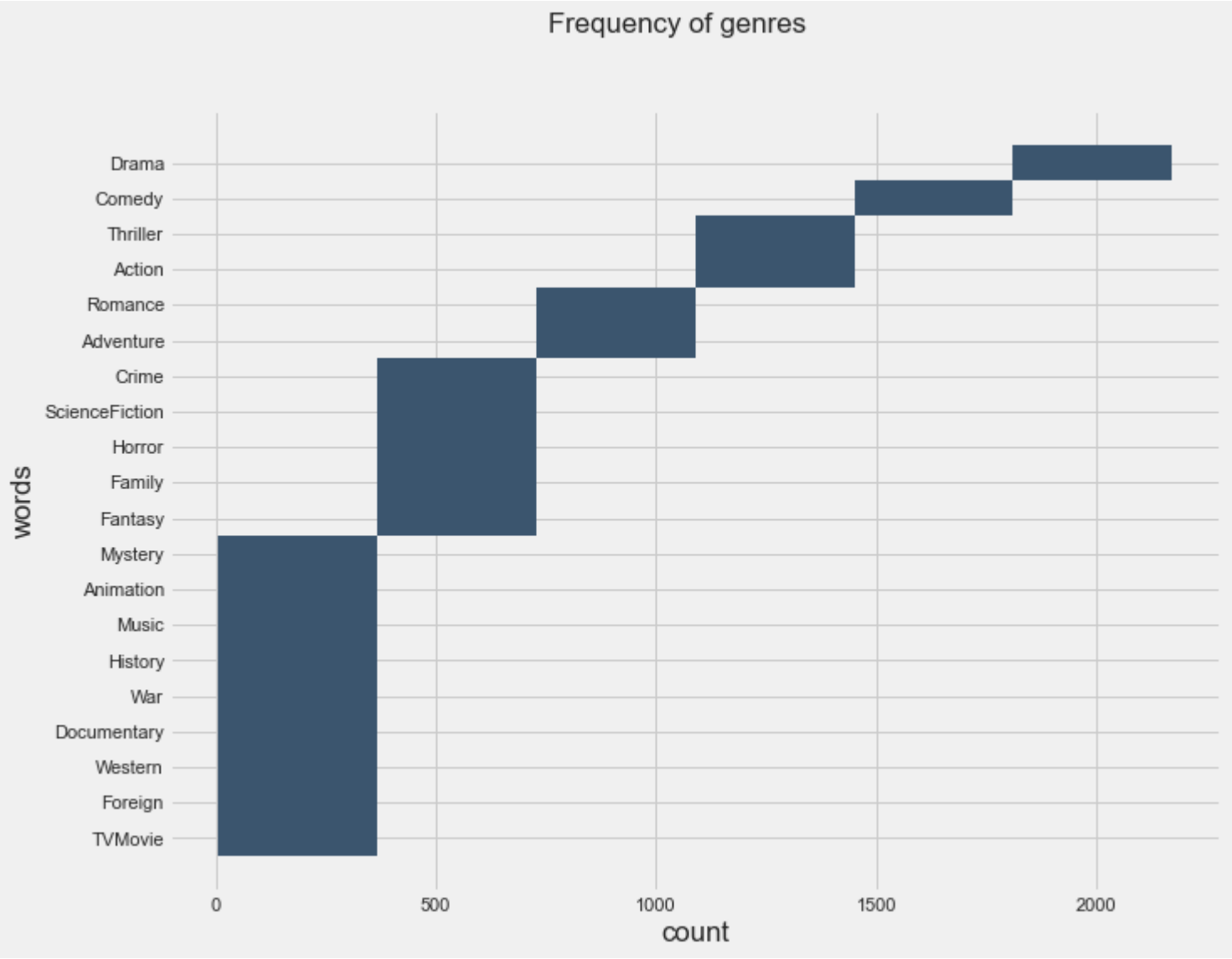


In [106…

```
plt.figure(figsize=(10,8))
sns.histplot(x='count', y='words', data=word_freq);

import pylab as pl
pl.suptitle("Frequency of genres")
```

Out[106…

```
Text(0.5, 0.98, 'Frequency of genres')
```



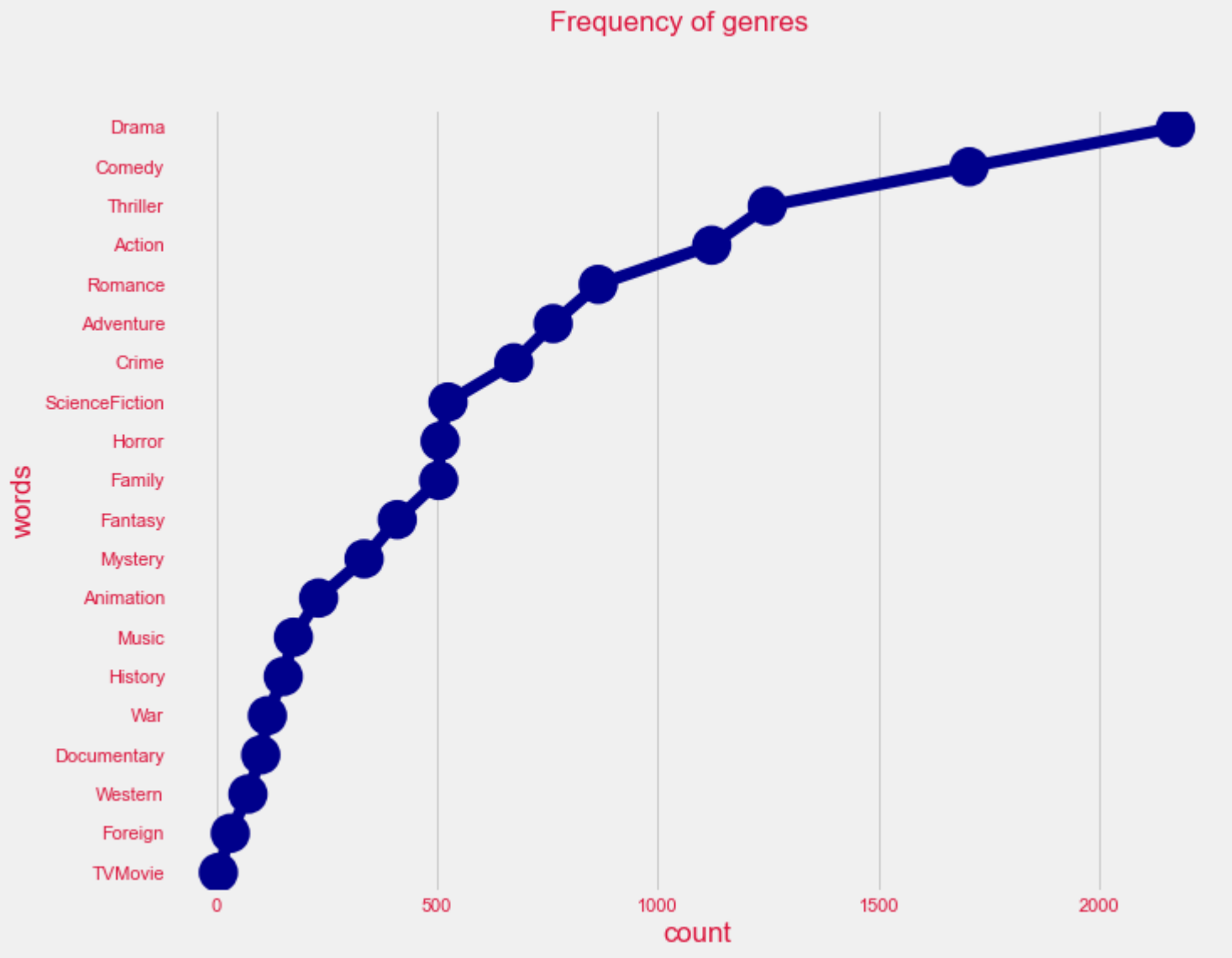
In [107…

```
from cycler import cycler
import matplotlib as mpl
COLOR = 'crimson'
mpl.rcParams['text.color'] = COLOR
mpl.rcParams['axes.labelcolor'] = COLOR
mpl.rcParams['xtick.color'] = COLOR
mpl.rcParams['ytick.color'] = COLOR
plt.figure(figsize=(10,8))
```

```
sns.pointplot(x='count', y='words', data=word_freq,color="darkblue");
import pylab as pl
pl.suptitle("Frequency of genres")
```

Out[107…

```
Text(0.5, 0.98, 'Frequency of genres')
```



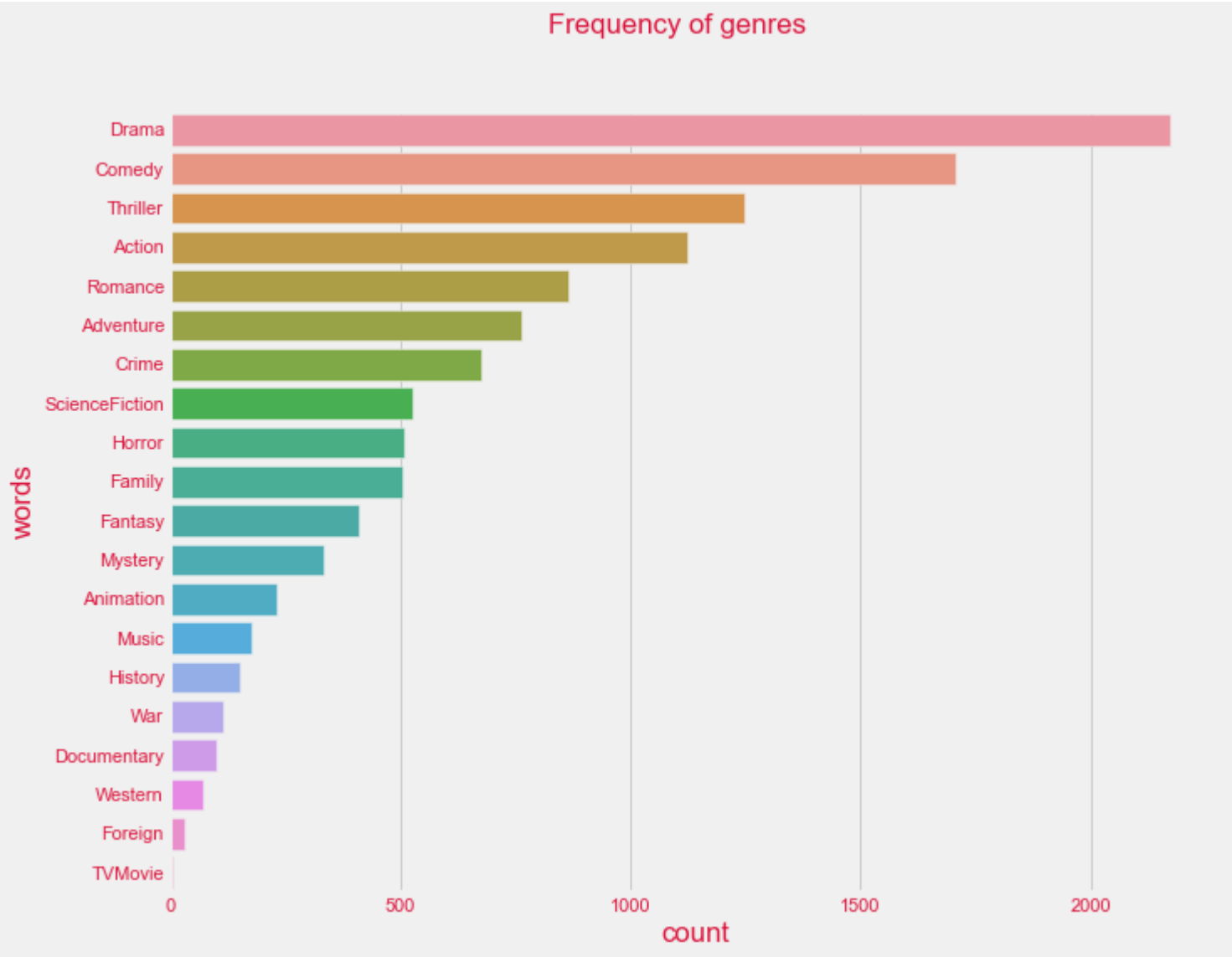
In [108…

```
plt.figure(figsize=(10,8))
sns.barplot(x='count', y='words', data=word_freq);
```

```
import pylab as pl
pl.suptitle("Frequency of genres")
```

Out[108…

```
Text(0.5, 0.98, 'Frequency of genres')
```



As we can see from the visualization the top three repeated genres are (Drama, comedy, thriller)

What are the top 10 rated movies and what do they have in common?

In [109…

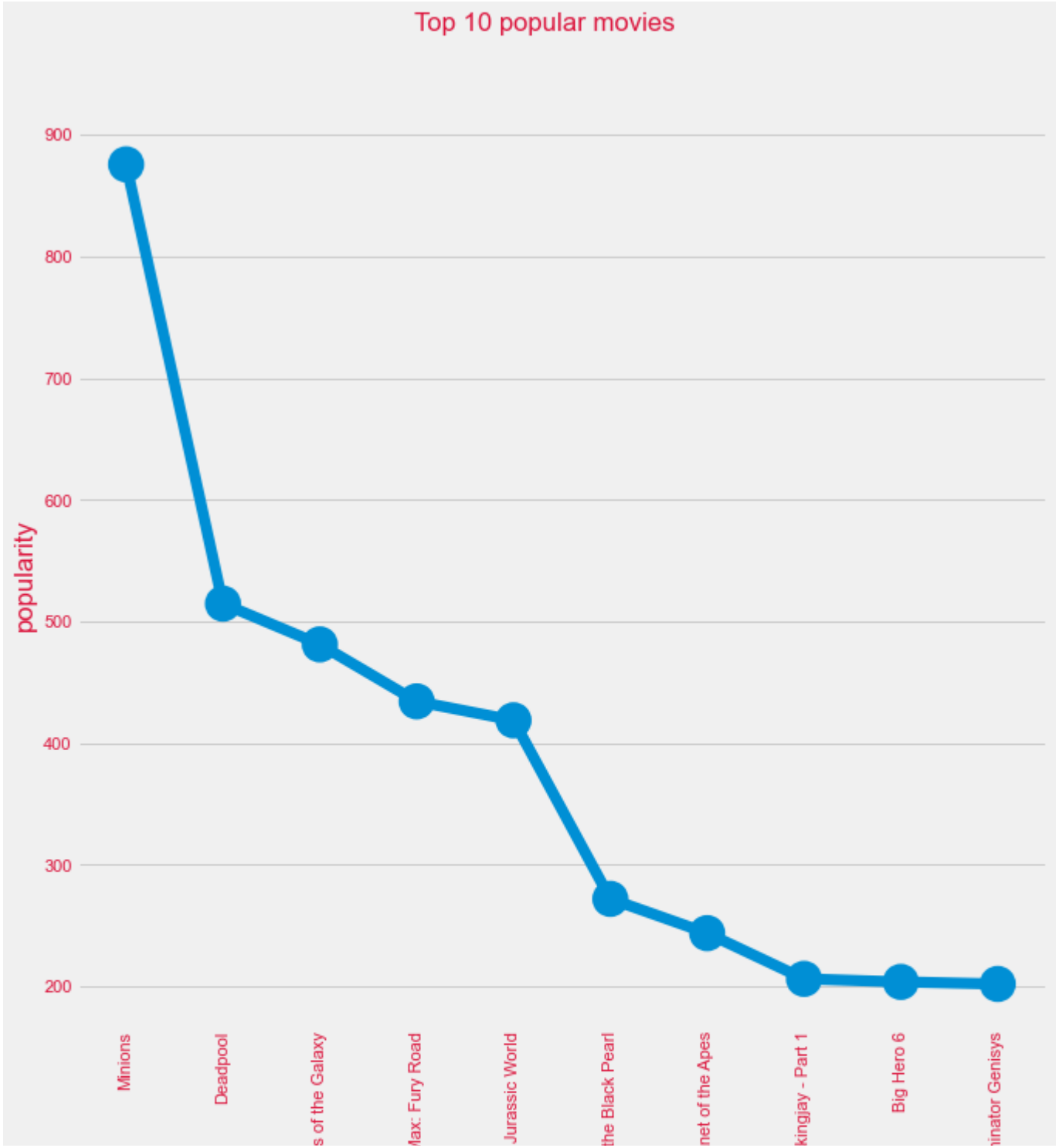
```
#arranging the rows according to the popularity in descending order
import pandas as pd
p= pd.DataFrame(df.sort_values(by=['popularity'], ascending=False).head(10))
```

In [110…

```
plt.figure(figsize=(10,10))
sns.pointplot(x='original_title', y='popularity', data=p);
plt.xticks(rotation=90)
import pylab as pl
pl.suptitle("Top 10 popular movies")
```

Out[110…

```
Text(0.5, 0.98, 'Top 10 popular movies')
```

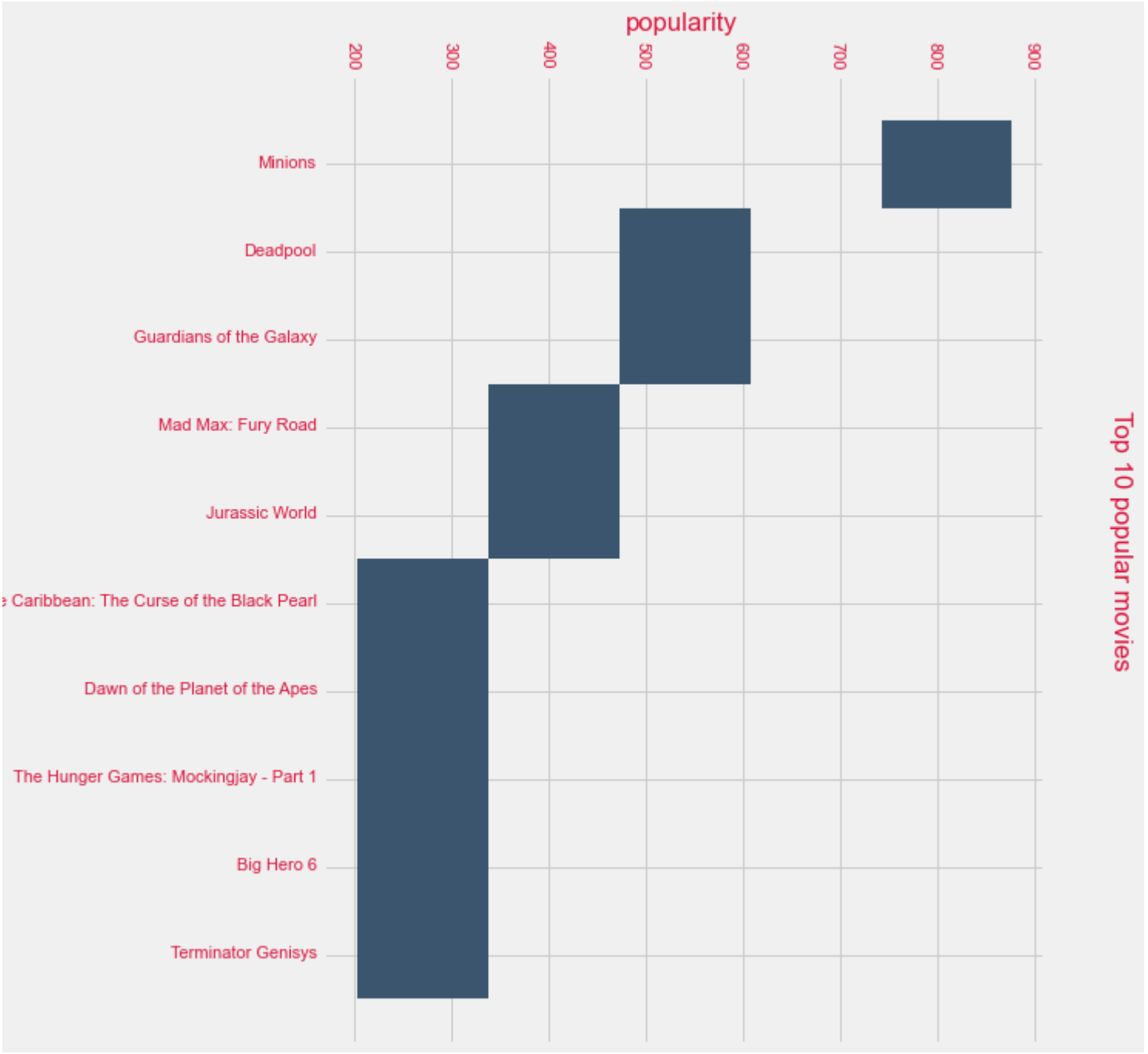

Top 10 popular movies
popularity
900
800
700
600
500
400
300
200
Minions
Deadpool
s of the Galaxy
Max: Fury Road
Jurassic World
the Black Pearl
net of the Apes
kingjay - Part 1
Big Hero 6
inator Genisys

In [111…

```
plt.figure(figsize=(10,8))
sns.histplot(x='original_title', y='popularity', data=p);
plt.xticks(rotation=90)
import pylab as pl
pl.suptitle("Top 10 popular movies")
```

Out[111…

```
Text(0.5, 0.98, 'Top 10 popular movies')
```

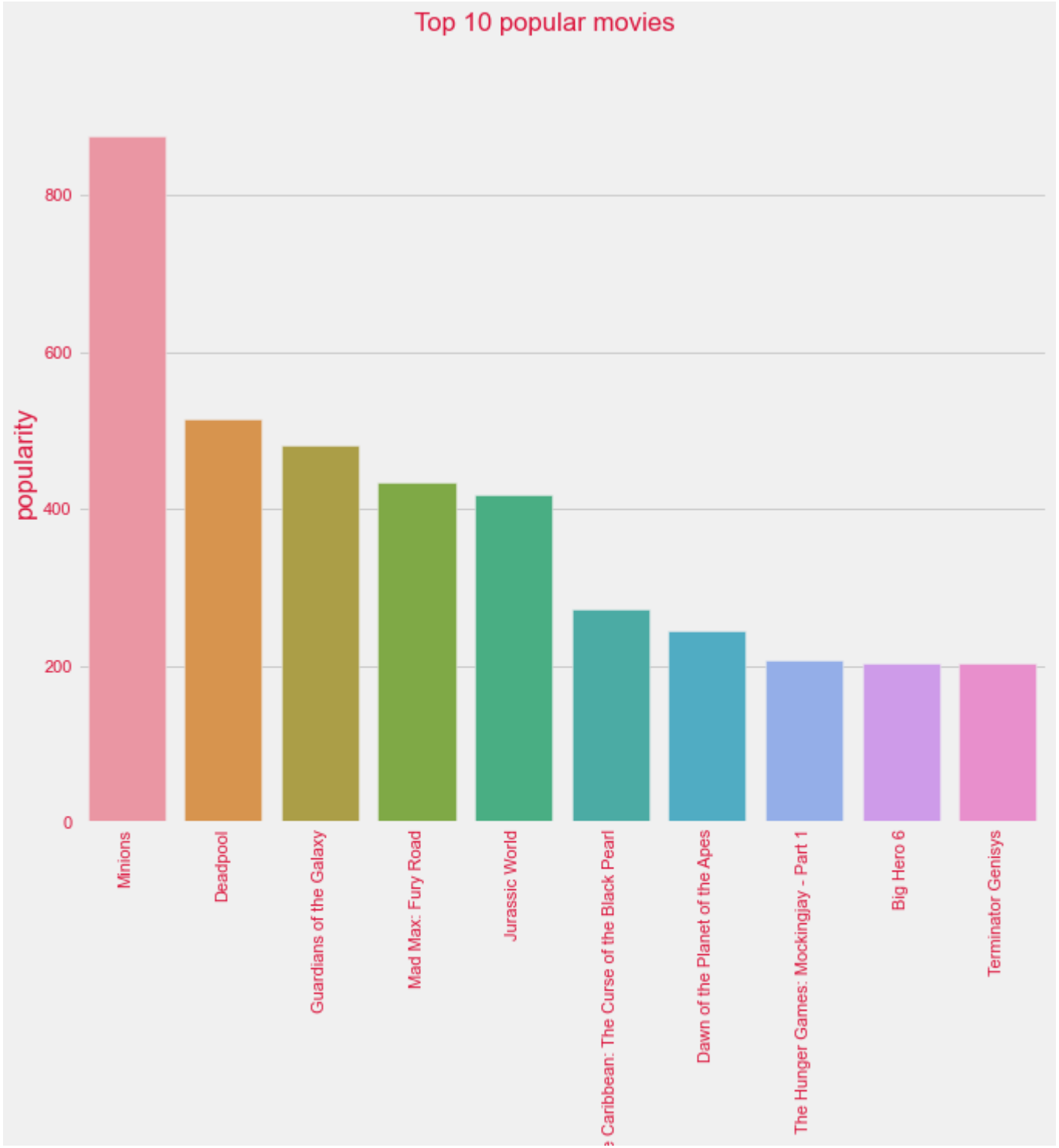

popularity
200
300
400
500
600
700
800
900
Minions
Deadpool
Guardians of the Galaxy
Mad Max: Fury Road
Jurassic World
e Caribbean: The Curse of the Black Pearl
Dawn of the Planet of the Apes
The Hunger Games: Mockingjay - Part 1
Big Hero 6
Terminator Genisys
Top 10 popular movies

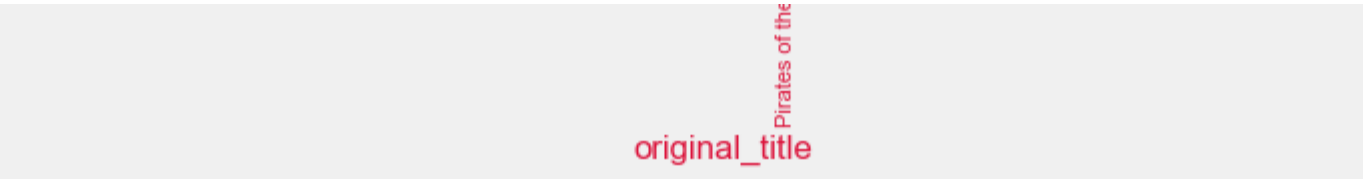
In [112…

```
plt.figure(figsize=(10,8))
sns.barplot(x='original_title', y='popularity', data=p);
plt.xticks(rotation=90)
import pylab as pl
pl.suptitle("Top 10 popular movies")
```

Out[112…

```
Text(0.5, 0.98, 'Top 10 popular movies')
```


Top 10 popular movies
popularity
800
600
400
200
0
Minions
Deadpool
Guardians of the Galaxy
Mad Max: Fury Road
Jurassic World
Caribbean: The Curse of the Black Pearl
Dawn of the Planet of the Apes
The Hunger Games: Mockingjay - Part 1
Big Hero 6
Terminator Genisys



As we can see from the visualizations that the Minions is the most popular movie.

What are the number of movies per year?

In [113…

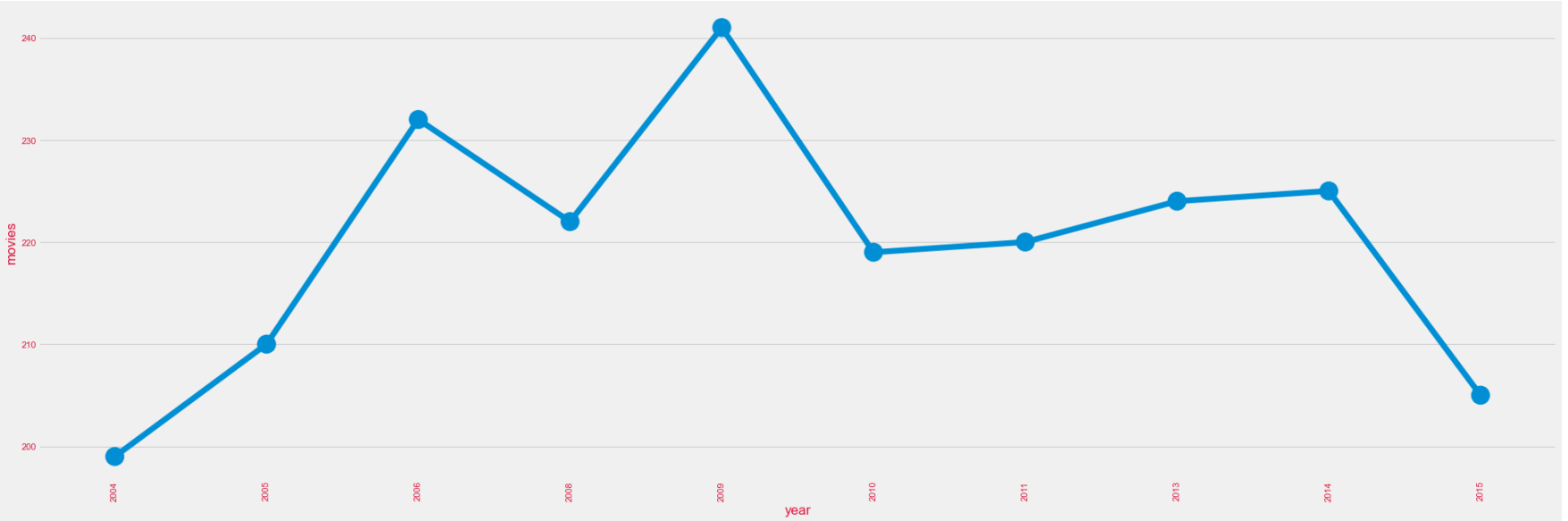
```
#Dataframe of years and movies
year_df = pd.DataFrame(df['release_year'].value_counts().reset_index())
year_df.columns = ['year', 'movies']
year_df=pd.DataFrame(year_df.sort_values(by=['movies'], ascending=False).head(10))
```

In [114…

```
plt.figure(figsize=(30,10))
sns.pointplot(x='year', y='movies', data=year_df);
plt.xticks(rotation=90)
```

Out[114…

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, '2004'),
  Text(1, 0, '2005'),
  Text(2, 0, '2006'),
  Text(3, 0, '2008'),
  Text(4, 0, '2009'),
  Text(5, 0, '2010'),
  Text(6, 0, '2011'),
  Text(7, 0, '2013'),
  Text(8, 0, '2014'),
  Text(9, 0, '2015')])
```



Answer: 2009 was the top year that had a 241 movie

The top 10 actors who contributed in most successful movies

In [115...

```
#make actor dataframe that had four columns first_actor, second_actor, third_actor, and popularity
actor= pd.DataFrame(df_score['cast'])
#use the cast column and split it by commas
actor=actor['cast'].str.split(',', expand=True)
#there were extra empty columns
actor=actor.drop(columns=[3, 4])
#remove "["
actor[0]=actor[0].str.replace("[","")
actor.columns = ['first_actor', 'second_actor', 'third_actor']
popularity = df_score["popularity"]
actor = actor.join(popularity)
actor
```

Out[115...

	first_actor	second_actor	third_actor	popularity
0	'sam worthington'	'zoe saldana'	'sigourney weaver'	150.437577
1	'johnny depp'	'orlando bloom'	'keira knightley'	139.082615
2	'daniel craig'	'christoph waltz'	'léa seydoux'	107.376788

	first_actor	second_actor	third_actor	popularity
3	'christian bale'	'michael caine'	'gary oldman'	112.312950
4	'taylor kitsch'	'lynn collins'	'samantha morton'	43.926995
...	...	...	...	...
4171	'darling narita'	'peter greene'	'michael newland'	0.918116
4172	'shane carruth'	'david sullivan'	'casey gooden'	23.307949
4173	'carlos gallardo'	'jaime de hoyos'	'peter marquardt'	14.269792
4174	'eric mabius'	'kristin booth'	'crystal lowe'	1.444476
4175	'drew barrymore'	'brian herzlinger'	'corey feldman'	1.929883

4176 rows × 4 columns

In [116...
```
#group each actor column with the popularity
one=actor.groupby('first_actor', as_index=False).sum()
two=actor.groupby('second_actor', as_index=False).sum()
three=actor.groupby('third_actor', as_index=False).sum()
```

In [117...
```
#name in each dataframe the two columns the same
one.columns = ['actors', 'popularity']
two.columns = ['actors', 'popularity']
three.columns = ['actors', 'popularity']
```

In [118...
```
#combine all data frames together to have all actors in same column and each one has the popularity of his movies
frames = [one, two,three]
result = pd.concat(frames)
result=result.groupby('actors', as_index=False).sum()
```

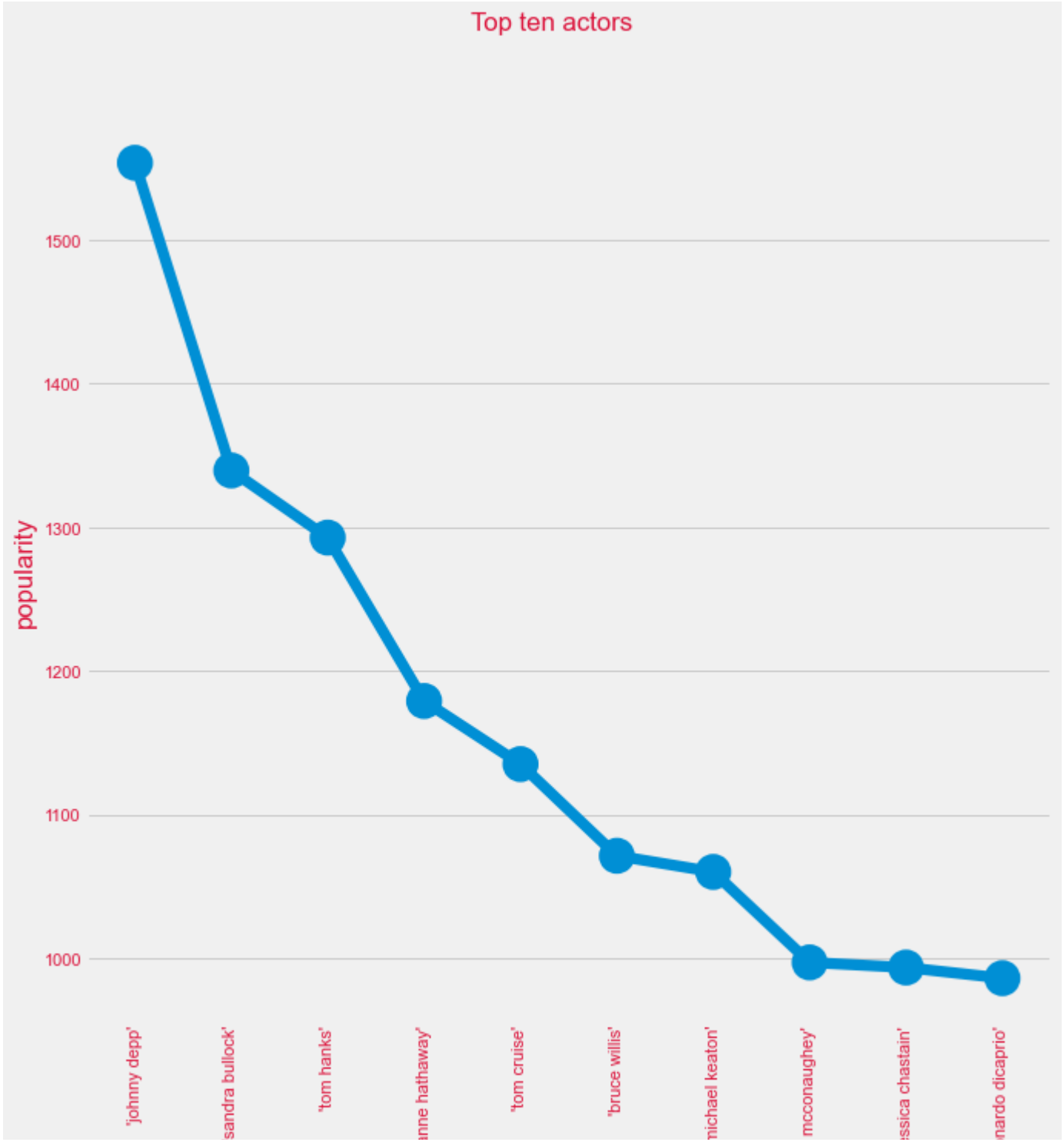
In [119...
```
#get the top actors
top_actors= pd.DataFrame(result.sort_values(by=['popularity'], ascending=False).head(10))
```

In [120...
```
plt.figure(figsize=(10,10))
sns.pointplot(x='actors', y='popularity', data=top_actors);
```

```
plt.xticks(rotation=90)
import pylab as pl
pl.suptitle("Top ten actors")
```

Out[120…

```
Text(0.5, 0.98, 'Top ten actors')
```

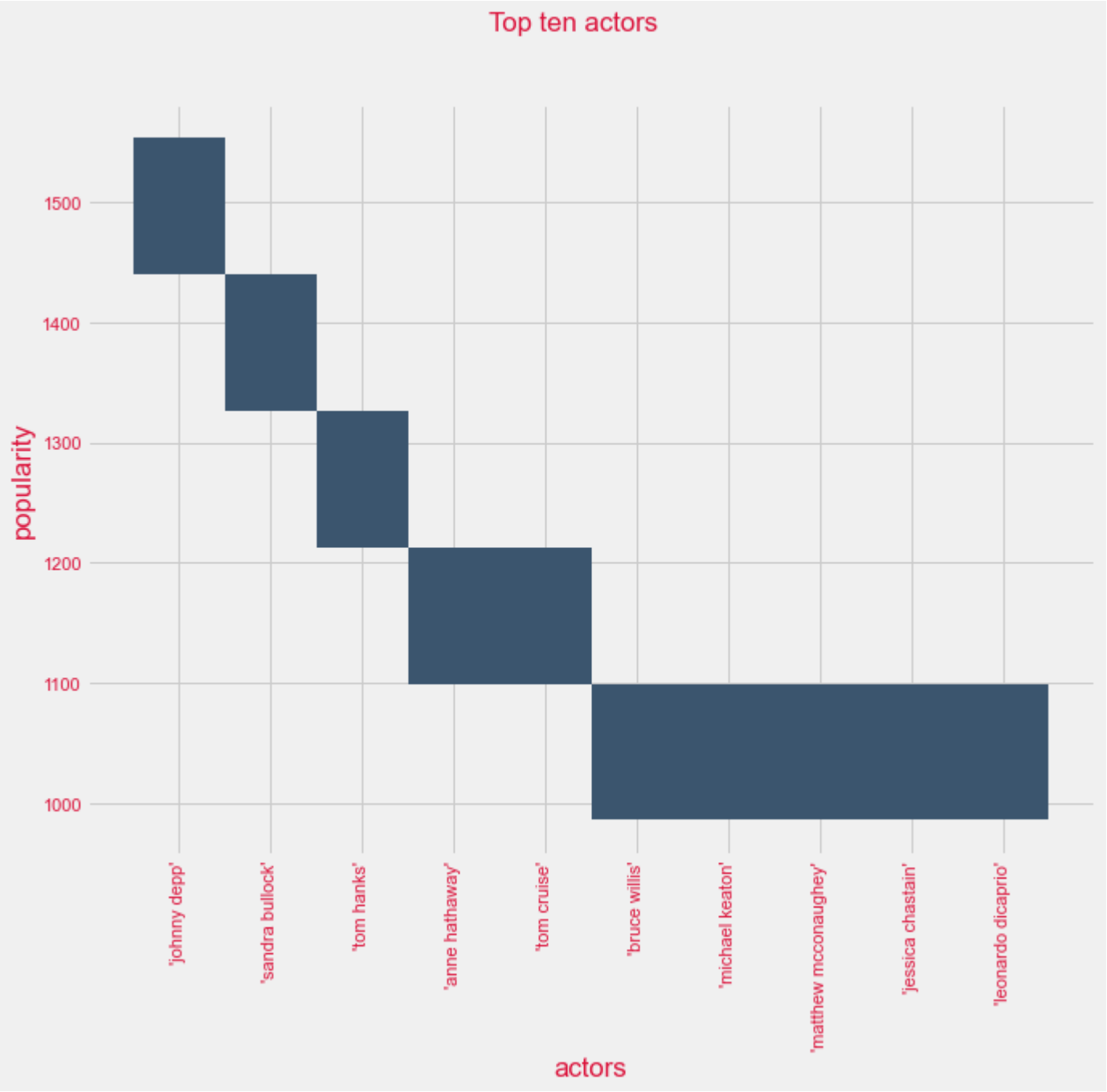

Top ten actors
popularity
1500
1400
1300
1200
1100
1000
'johnny depp'
sandra bullock'
'tom hanks'
nne hathaway'
'tom cruise'
'bruce willis'
michael keaton'
mcconaughey'
ssica chastain'
nardo dicaprio'

In [121…

```
plt.figure(figsize=(10,8))
sns.histplot(x='actors', y='popularity', data=top_actors);
plt.xticks(rotation=90)
import pylab as pl
pl.suptitle("Top ten actors")
```

Out[121…

```
Text(0.5, 0.98, 'Top ten actors')
```



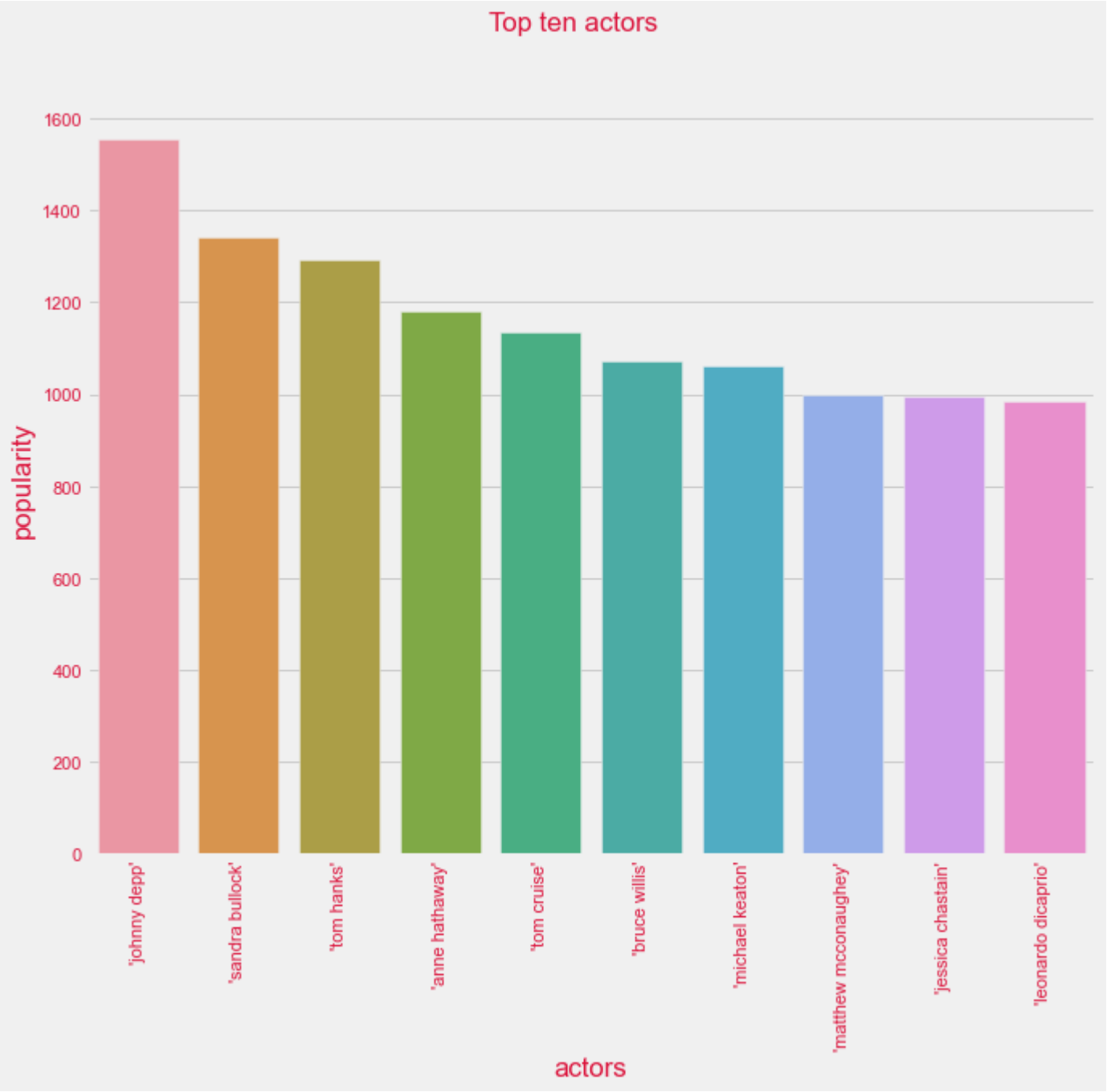
In [122…

```
plt.figure(figsize=(10,8))
sns.barplot(x='actors', y='popularity', data=top_actors);
```

```
plt.xticks(rotation=90)
import pylab as pl
pl.suptitle("Top ten actors")
```

Out[122…

```
Text(0.5, 0.98, 'Top ten actors')
```



The actor that had the most successful movies is "Johnny Depp"

Which movie has the highest budget?

In [123...

```
df_score_no=pd.read_csv('tmdb_5000_movies.csv')
df_score_no[df_score_no['budget'] == df_score_no['budget'].max()]
```

Out[123...

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	productio
17	380000000	[{"id": 12, "name": "Adventure"}, {"id": 28, "...	http://disney.go.com/pirates/index-on-stranger...	1865	[{"id": 658, "name": "sea"}, {"id": 1316, "nam...	en	Pirates of the Caribbean: On Stranger Tides	Captain Jack Sparrow crosses paths with a woma...	135.413856	[{"name" Picture

In [124...

```
df_score_no[['id','title','budget', 'revenue']].sort_values(['budget'], ascending=False).head(10).style.background_gradie
```

Out[124...

	id	title	budget	revenue
17	1865	Pirates of the Caribbean: On Stranger Tides	380000000	1045713802
1	285	Pirates of the Caribbean: At World's End	300000000	961000000
7	99861	Avengers: Age of Ultron	280000000	1405403694
10	1452	Superman Returns	270000000	391081192
4	49529	John Carter	260000000	284139100
6	38757	Tangled	260000000	591794936
5	559	Spider-Man 3	258000000	890871626
13	57201	The Lone Ranger	255000000	89289910
46	127585	X-Men: Days of Future Past	250000000	747862775
22	57158	The Hobbit: The Desolation of Smaug	250000000	958400000

"Pirates of the caribbean:On Stranger Tides" had the highest budget

Which movie is longest movie?

In [125...

```
df_score_no[df_score_no['runtime'] == df_score_no['runtime'].max()]
```

Out[125...

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	produ
2384	18000000	[{"id": 80, "name": "Crime"}, {"id": 18, "name...	NaN	43434	[{"id": 1419, "name": "gun"}, {"id": 7336, "na...	en	Carlos	The story of Venezuelan revolutionary, Ilich R...	1.138383	[{"name": "Egoli Tossell Film AG", "id": 2254}...	[{"is "r

"The story of Venezuelan revolutionary" is the longest movie

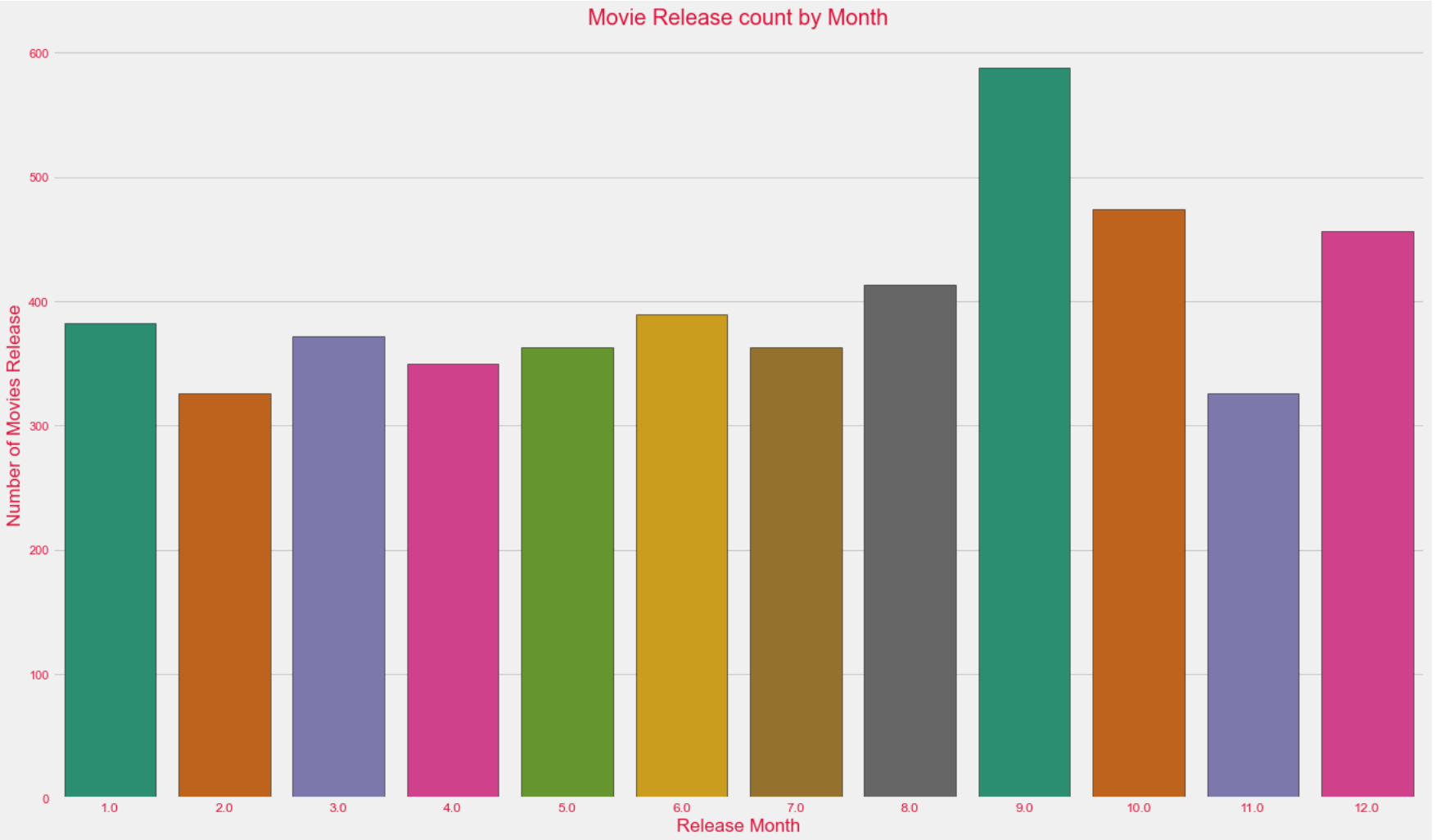
In which month most movies are released from 1921 to 2017?

In [126...

```
df_score_no['release_date'] = pd.to_datetime(df_score_no['release_date'], infer_datetime_format=True)
df_score_no['release_day'] = df_score_no['release_date'].apply(lambda t: t.day)
df_score_no['release_weekday'] = df_score_no['release_date'].apply(lambda t: t.weekday())
df_score_no['release_month'] = df_score_no['release_date'].apply(lambda t: t.month)

df_score_no['release_year'] = df_score_no['release_date'].apply(lambda t: t.year if t.year < 2018 else t.year -100)
```

In [127...

```
plt.figure(figsize=(20,12))
edgecolor=(0,0,0),
sns.countplot(df_score_no['release_month'].sort_values(), palette = "Dark2", edgecolor=(0,0,0))
plt.title("Movie Release count by Month",fontsize=20)
plt.xlabel('Release Month')
plt.ylabel('Number of Movies Release')
plt.xticks(fontsize=12)
plt.show()
```


Movie Release count by Month
Number of Movies Release
0
100
200
300
400
500
600
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
Release Month

In [ ]: