Load the Libraries

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

Load the DataSet

In [2]:
```
df1 = pd.read_csv(r"C:\Users\LENOVO\Desktop\datasets\tmdb_5000_credits.csv")
df1.head()
```

Out[2]:

	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...

Get the basic infomation of the data

In [3]:
```
df1.shape
```

Out[3]:
```
(4803, 4)
```

In [4]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   movie_id  4803 non-null   int64 
 1   title     4803 non-null   object
 2   cast      4803 non-null   object
 3   crew      4803 non-null   object
dtypes: int64(1), object(3)
memory usage: 150.2+ KB
```

Load the second dataset: Movies dataset

In [5]:
```
df2 = pd.read_csv(r"C:\Users\LENOVO\Desktop\datasets\tmdb_5000_movies.csv")
df2.head()
```

Out[5]:

	budget	genres	homepage	id	keywords	original_la

	budget	genres	homepage	id	keywords	original_la
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	

The Basic information about the dataset

In [6]:
```
df2.shape
```

Out[6]:
```
(4803, 20)
```

In [7]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   budget                4803 non-null   int64  
 1   genres                4803 non-null   object 
 2   homepage              1712 non-null   object 
 3   id                    4803 non-null   int64  
 4   keywords              4803 non-null   object 
 5   original_language     4803 non-null   object 
 6   original_title        4803 non-null   object 
 7   overview              4800 non-null   object 
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object 
 10  production_countries  4803 non-null   object 
```

```
 11  release_date          4802 non-null   object
 12  revenue               4803 non-null   int64
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object
 15  status                4803 non-null   object
 16  tagline               3959 non-null   object
 17  title                 4803 non-null   object
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

Merge the two dataframes

In [8]:
```
df1.columns = ['id','title','cast','crew']
df2= df2.merge(df1,on='id')
```

In [9]:
```
df2.head()
```

Out[9]:

	budget	genres	homepage	id	keywords	original_la
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	

5 rows × 23 columns

In [10]:
```
df2.shape
```

Out[10]:
```
(4803, 23)
```

In [11]:
```
df2.columns
```

Out[11]:
```
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_x', 'vote_average',
       'vote_count', 'title_y', 'cast', 'crew'],
      dtype='object')
```

In [12]:
```
C= df2['vote_average'].mean()
C
```

Out[12]:
```
6.092171559442011
```

Minimum votes to be listed

In [13]:
```
m= df2['vote_count'].quantile(0.9) #movies having vote count greater than 90% from t
m
```

Out[13]:
```
1838.4000000000015
```

Getting the list of movies to be listed

In [14]:
```
lists_movies = df2.copy().loc[df2['vote_count'] >= m]
lists_movies.shape
```

Out[14]:
```
(481, 23)
```

Defining a function

In [15]:
```
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    # Calculation based on the IMDB formula (m=1838, c=6.09)
    return (v/(v+m) * R) + (m/(m+v) * C)
```

In [16]:
```
# Define a new feature 'score' and calculate its value with `weighted_rating()`
lists_movies['score'] = lists_movies.apply(weighted_rating, axis=1)
```

In [17]:
```
lists_movies.head(3)
```

Out[17]:

	budget	genres	homepage	id	keywords	original_la
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	

	budget	genres	homepage	id	keywords	original_la
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	

3 rows × 24 columns

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

Out[18]: (481, 24)

Sort the movies

In [19]:
```
#Sort movies based on score calculated above
lists_movies = lists_movies.sort_values('score', ascending=False)

#Print the top 10 movies
lists_movies[['title_x', 'vote_count', 'vote_average', 'score']].head(10)
```

Out[19]:

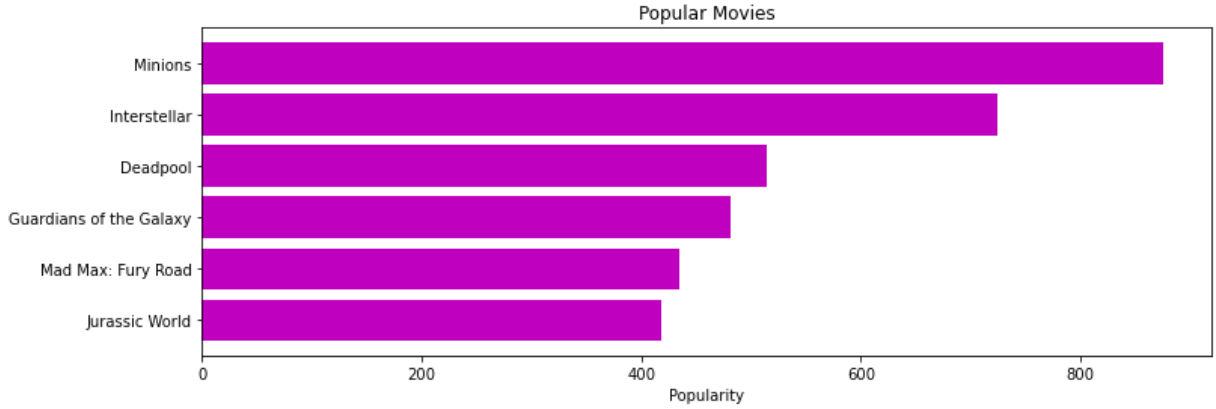
	title_x	vote_count	vote_average	score
1881	The Shawshank Redemption	8205	8.5	8.059258
662	Fight Club	9413	8.3	7.939256
65	The Dark Knight	12002	8.2	7.920020
3232	Pulp Fiction	8428	8.3	7.904645
96	Inception	13752	8.1	7.863239
3337	The Godfather	5893	8.4	7.851236
95	Interstellar	10867	8.1	7.809479
809	Forrest Gump	7927	8.2	7.803188
329	The Lord of the Rings: The Return of the King	8064	8.1	7.727243
1990	The Empire Strikes Back	5879	8.2	7.697884

Most Popular Movies

In [20]:
```
pop= df2.sort_values('popularity', ascending=False)
import matplotlib.pyplot as plt
plt.figure(figsize=(12,4))

plt.barh(pop['title_x'].head(6),pop['popularity'].head(6), align='center',
        color='m')
plt.gca().invert_yaxis()
plt.xlabel("Popularity")
plt.title("Popular Movies" )
```

Out[20]: Text(0.5, 1.0, 'Popular Movies')



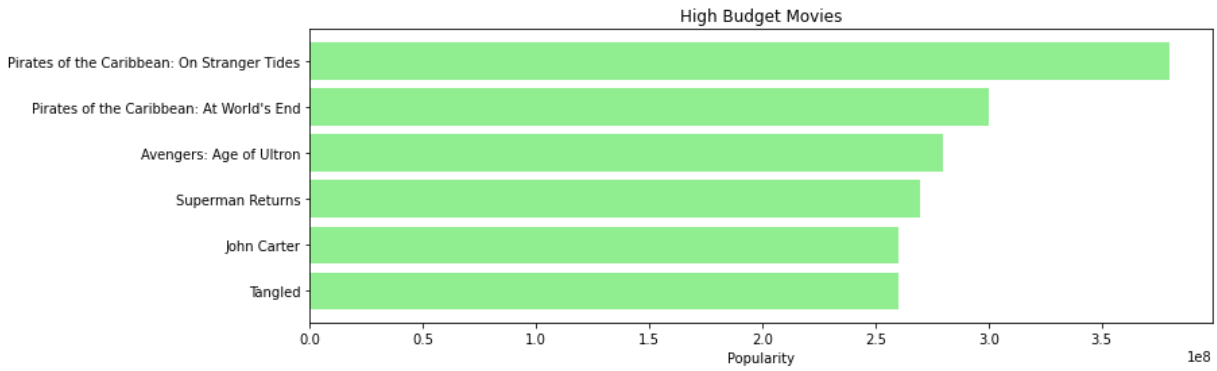
In [21]:
```
df2.columns
```

Out[21]:
```
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_x', 'vote_average',
       'vote_count', 'title_y', 'cast', 'crew'],
      dtype='object')
```

In [22]:
```
pop= df2.sort_values('budget', ascending=False)
import matplotlib.pyplot as plt
plt.figure(figsize=(12,4))

plt.barh(pop['title_x'].head(6),pop['budget'].head(6), align='center',
        color='lightgreen')
plt.gca().invert_yaxis()
plt.xlabel("Popularity")
plt.title("High Budget Movies" )
```

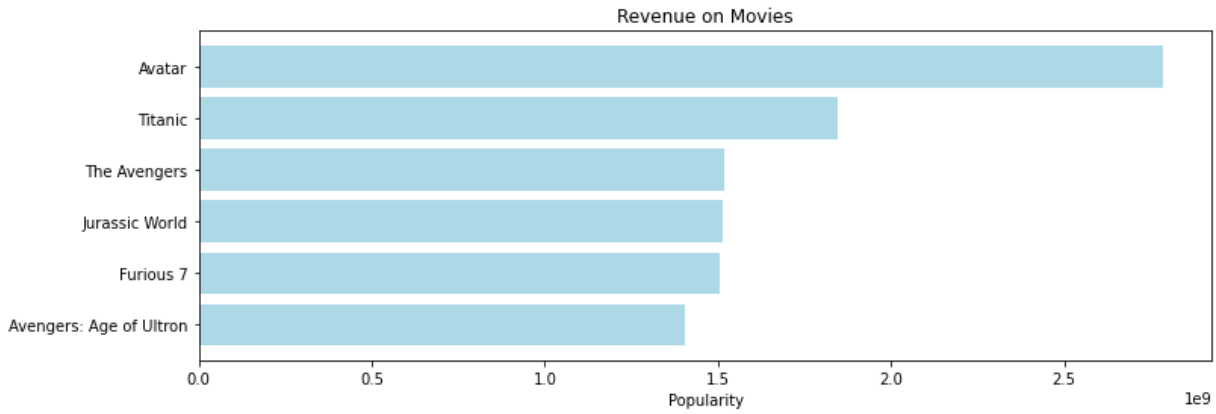
Out[22]:
```
Text(0.5, 1.0, 'High Budget Movies')
```



In [23]:
```
pop= df2.sort_values('revenue', ascending=False)
import matplotlib.pyplot as plt
plt.figure(figsize=(12,4))

plt.barh(pop['title_x'].head(6),pop['revenue'].head(6), align='center',
        color='lightblue')
plt.gca().invert_yaxis()
plt.xlabel("Popularity")
plt.title("Revenue on Movies" )
```

Out[23]:
```
Text(0.5, 1.0, 'Revenue on Movies')
```



Drop the title_y column from the DataFrame

In [24]:
```
lists_movies.drop(['title_y'], axis=1, inplace=True)
```

In [25]:
```
lists_movies.shape
```

Out[25]:
```
(481, 23)
```

In [26]:
```
lists_movies.head(2)
```

Out[26]:

	budget	genres	homepage	id	keywords	original_language
1881	25000000	[{"id": 18, "name": "Drama"}, {"id": 80, "name...	NaN	278	[{"id": 378, "name": "prison"}, {"id": 417, "n...	en
662	63000000	[{"id": 18, "name": "Drama"}]	http://www.foxmovies.com/movies/fight-club	550	[{"id": 825, "name": "support group"}, {"id": ...	en

2 rows × 23 columns

Overview column

In [29]:
```
df2['overview'].head(10)
```

Out[29]:
```
0    In the 22nd century, a paraplegic Marine is di...
1    Captain Barbossa, long believed to be dead, ha...
2    A cryptic message from Bond’s past sends him o...
3    Following the death of District Attorney Harve...
4    John Carter is a war-weary, former military ca...
5    The seemingly invincible Spider-Man goes up ag...
6    When the kingdom's most wanted-and most charmi...
7    When Tony Stark tries to jumpstart a dormant p...
8    As Harry begins his sixth year at Hogwarts, he...
9    Fearing the actions of a god-like Super Hero l...
Name: overview, dtype: object
```

based on the description we shall find the similarity among the movies.

In [30]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

#Define a TF-IDF Vectorizer Object. Remove all english stop words such as 'the', 'a'
tfidf = TfidfVectorizer(stop_words='english')

#Replace NaN with an empty string
df2['overview'] = df2['overview'].fillna('')

#Construct the required TF-IDF matrix by fitting and transforming the data
tfidf_matrix = tfidf.fit_transform(df2['overview'])

#Output the shape of tfidf_matrix
tfidf_matrix.shape
```

Out[30]:
```
(4803, 20978)
```

In [31]:
```
# Import linear_kernel
from sklearn.metrics.pairwise import linear_kernel

# Compute the cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

In [32]:
```
#Construct a reverse map of indices and movie titles
indices = pd.Series(df2.index, index=df2['title_x']).drop_duplicates()
```

In [33]:
```
# Function that takes in movie title as input and outputs most similar movies
def get_recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return df2['title_x'].iloc[movie_indices]
```

In [34]:
```
get_recommendations('The Dark Knight Rises')
```

Out[34]:
```
65                              The Dark Knight
299                              Batman Forever
428                              Batman Returns
1359                                     Batman
3854    Batman: The Dark Knight Returns, Part 2
119                               Batman Begins
2507                                  Slow Burn
9           Batman v Superman: Dawn of Justice
1181                                       JFK
210                              Batman & Robin
Name: title_x, dtype: object
```

In [35]:
```
get_recommendations('JFK')
```

Out[35]:
```
2507                Slow Burn
879       Law Abiding Citizen
2020                The Rookie
```

```
2193     Secret in Their Eyes
2697                   Bobby
753              The Sentinel
1202             Legal Eagles
817           American Wedding
65             The Dark Knight
3       The Dark Knight Rises
Name: title_x, dtype: object
```

In [36]:
```
get_recommendations('Avatar')
```

Out[36]:
```
3604                      Apollo 18
2130                    The American
634                       The Matrix
1341            The Inhabited Island
529                 Tears of the Sun
1610                           Hanna
311     The Adventures of Pluto Nash
847                          Semi-Pro
775                         Supernova
2628             Blood and Chocolate
Name: title_x, dtype: object
```

In [37]:
```
get_recommendations('The Matrix')
```

Out[37]:
```
1281                 Hackers
2996                Commando
2088                   Pulse
1341     The Inhabited Island
333             Transcendence
0                      Avatar
261     Live Free or Die Hard
775                 Supernova
125       The Matrix Reloaded
2614          The Love Letter
Name: title_x, dtype: object
```

Reeves movies: John Wick, Speed, Point Break.

PART 3: MOVIE RECOMMENDATION SYSTEM

Recommender System based on Genres, keywords, crew, and cast

In [39]:
```
# Parse the stringified features into their corresponding python objects
from ast import literal_eval

features = ['cast', 'crew', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(literal_eval)
```

In [40]:
```
#Get the director's name from the crew feature. If director is not listed, return Na
def get_director(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan
```

In [41]:
```
#Returns the list top 3 elements or entire list; whichever is more.
def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]
  #Check if more than 3 elements exist. If yes, return only first three. If no, retu
        if len(names) > 3:
```

```
            names = names[:3]
        return names

    #Return empty list in case of missing/malformed data
    return []
```

In [42]:
```
#Define new director, cast, genres and keywords features that are in a suitable form
df2['director'] = df2['crew'].apply(get_director)

features = ['cast', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(get_list)
```

In [43]:
```
# Print the new features of the first 5 films
df2[['title_x', 'cast', 'director', 'keywords', 'genres']].head(5)
```

Out[43]:

	title_x	cast	director	keywords	genres
0	Avatar	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	James Cameron	[culture clash, future, space war]	[Action, Adventure, Fantasy]
1	Pirates of the Caribbean: At World's End	[Johnny Depp, Orlando Bloom, Keira Knightley]	Gore Verbinski	[ocean, drug abuse, exotic island]	[Adventure, Fantasy, Action]
2	Spectre	[Daniel Craig, Christoph Waltz, Léa Seydoux]	Sam Mendes	[spy, based on novel, secret agent]	[Action, Adventure, Crime]
3	The Dark Knight Rises	[Christian Bale, Michael Caine, Gary Oldman]	Christopher Nolan	[dc comics, crime fighter, terrorist]	[Action, Crime, Drama]
4	John Carter	[Taylor Kitsch, Lynn Collins, Samantha Morton]	Andrew Stanton	[based on novel, mars, medallion]	[Action, Adventure, Science Fiction]

In [44]:
```
# Function to convert all strings to lower case and strip names of spaces
def clean_data(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ", "")) for i in x]
    else:
        #Check if director exists. If not, return empty string
        if isinstance(x, str):
            return str.lower(x.replace(" ", ""))
        else:
            return ''
```

In [45]:
```
# Apply clean_data function to your features.
features = ['cast', 'keywords', 'director', 'genres']

for feature in features:
    df2[feature] = df2[feature].apply(clean_data)
```

Creating MetaData

In [46]:
```
def create_soup(x):
    return ' '.join(x['keywords']) + ' ' + ' '.join(x['cast']) + ' ' + x['director']
df2['soup'] = df2.apply(create_soup, axis=1)
```

In [47]:
```
# Import CountVectorizer and create the count matrix
from sklearn.feature_extraction.text import CountVectorizer
```

```
count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(df2['soup'])
```

In [48]:
```
# Compute the Cosine Similarity matrix based on the count_matrix
from sklearn.metrics.pairwise import cosine_similarity

cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
```

In [49]:
```
# Reset index of our main DataFrame and construct reverse mapping as before
df2 = df2.reset_index()
indices = pd.Series(df2.index, index=df2['title_x'])
```

Make Recommendations

In [50]:
```
get_recommendations('JFK', cosine_sim2) # dir, cast, genres, keywords
```

Out[50]:
```
884              Zero Dark Thirty
1528                     Criminal
647            World Trade Center
737     Jack Ryan: Shadow Recruit
2008        In the Valley of Elah
3172                The Contender
940                       Syriana
991                     Fair Game
1091                        Nixon
1187              Bridge of Spies
Name: title_x, dtype: object
```

In [51]:
```
get_recommendations('The Godfather', cosine_sim2)
```

Out[51]:
```
867      The Godfather: Part III
2731      The Godfather: Part II
4638    Amidst the Devil's Wings
2649           The Son of No One
1525              Apocalypse Now
1018             The Cotton Club
1170     The Talented Mr. Ripley
1209               The Rainmaker
1394               Donnie Brasco
1850                    Scarface
Name: title_x, dtype: object
```

In [52]:
```
get_recommendations('Avatar',cosine_sim2)
```

Out[52]:
```
206                         Clash of the Titans
71        The Mummy: Tomb of the Dragon Emperor
786                           The Monkey King 2
103                   The Sorcerer's Apprentice
131                                     G-Force
215      Fantastic 4: Rise of the Silver Surfer
466                            The Time Machine
715                           The Scorpion King
1      Pirates of the Caribbean: At World's End
5                                   Spider-Man 3
Name: title_x, dtype: object
```

In [53]:
```
get_recommendations('Spectre', cosine_sim2)
```

Out[53]:
```
29                      Skyfall
11             Quantum of Solace
1084             The Glimmer Man
1234              The Art of War
2156                  Nancy Drew
4638    Amidst the Devil's Wings
```

```
62          The Legend of Tarzan
3373    The Other Side of Heaven
4                   John Carter
72                Suicide Squad
Name: title_x, dtype: object
```

In [ ]: