In [16]:
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from nltk.corpus import stopwords
import string
```

In [17]:
```
data=pd.read_csv('IMDb names.csv')
data.head()
```

Out[17]:

	imdb_title_id	title	original_title	year	date_published	genre	duration	country	languag
0	tt0000574	The Story of the Kelly Gang	The Story of the Kelly Gang	1906	12/26/1906	Biography, Crime, Drama	70	Australia	Nal
1	tt0001892	Den sorte drøm	Den sorte drøm	1911	8/19/1911	Drama	53	Germany, Denmark	Nal
2	tt0002101	Cleopatra	Cleopatra	1912	11/13/1912	Drama, History	100	USA	Englis
3	tt0002130	L'Inferno	L'Inferno	1911	3/6/1911	Adventure, Drama, Fantasy	68	Italy	Italia
4	tt0002199	From the Manger to the Cross; or, Jesus of Naz...	From the Manger to the Cross; or, Jesus of Naz...	1912	1913	Biography, Drama	60	USA	Englis

5 rows × 22 columns

In [3]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\User\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[3]:
```
True
```

In [5]:
```
data.columns
```

Out[5]:
```
Index(['imdb_title_id', 'title', 'original_title', 'year', 'date_published',
       'genre', 'duration', 'country', 'language', 'director', 'writer',
       'production_company', 'actors', 'description', 'avg_vote', 'votes',
       'budget', 'usa_gross_income', 'worlwide_gross_income', 'metascore',
       'reviews_from_users', 'reviews_from_critics'],
      dtype='object')
```

In [6]:
```
def clean_data(x):
    return str.lower(x.replace(' ',''))
```

In [7]:
```
#Combining all important data
new_frame=pd.DataFrame()
data.fillna(value="",inplace=True)
data
new_frame['title']=data[data['country']=='USA']['original_title']
new_frame['combined_text']=data.apply(lambda x:x['title']+' '+x['original_title']+x['ge
                                                 +x['production_company']+x['actors']+x['descripti
new_frame.head(10)
```

Out[7]:

	title	combined_text
2	Cleopatra	Cleopatra CleopatraDrama, HistoryCharles L. Ga...
4	From the Manger to the Cross; or, Jesus of Naz...	From the Manger to the Cross; or, Jesus of Naz...
15	Home, Sweet Home	Home, Sweet Home Home, Sweet HomeDramaD.W. Gri...
17	Traffic in Souls	Traffic in Souls Traffic in SoulsCrime, DramaG...
20	The Avenging Conscience: or 'Thou Shalt Not Kill'	The Avenging Conscience: or 'Thou Shalt Not Ki...
21	The Bargain	The Bargain The BargainWesternReginald BarkerW...
23	Cinderella	Cinderella CinderellaFantasy, DramaJames Kirkw...
27	A Florida Enchantment	A Florida Enchantment A Florida EnchantmentCom...
30	His Majesty, the Scarecrow of Oz	His Majesty, the Scarecrow of Oz His Majesty, ...
31	Hypocrites	Hypocrites HypocritesDramaLois WeberLois Weber...

In [8]:
```
def combined_text(x):
    return x['title']+' '+x['original_title']+' '+x['genre']+' '+x['director']+' '+x['w

def textProcesing(combined_text):
    remove= [char for char in combined_text if char not in string.punctuation]
    remove =''.join(remove)
    return [word.lower() for word in remove.split() if word.lower() not in stopwords.wo
```

In [9]:
```
#Comvert all the documents into matrix

count =CountVectorizer(stop_words='english')
cMatrix=count.fit_transform(new_frame['combined_text'])
sim=cosine_similarity(cMatrix,cMatrix)
```

In [10]:
```
new_frame.head(10)
cMatrix.shape
```

Out[10]:
```
(27494, 189928)
```

In [11]:
```
new_frame.reset_index(inplace=True)
indices=pd.Series(new_frame.index,index=new_frame['title'])
indices
```

Out[11]:
```
title
Cleopatra                                                 0
From the Manger to the Cross; or, Jesus of Nazareth       1
Home, Sweet Home                                          2
Traffic in Souls                                          3
The Avenging Conscience: or 'Thou Shalt Not Kill'         4
                                                        ...
Love Struck Sick                                      27489
Nightmare Tenant                                      27490
Falling Inn Love                                      27491
Abduction 101                                         27492
The Pilgrim's Progress                                27493
Length: 27494, dtype: int64
```

In [12]:
```
def recommendation(title,sim=sim):
    index=indices[title]
    #Getting similarities score of the input movie with all movies
    sim_scores=list(enumerate(sim[index]))
    #Sort movies based on the score
    sort_movies=sorted(sim_scores, key=lambda x:x[1], reverse=True)
    #get top 10 sort movies based on scores
    sort_movies=sort_movies[1:10]
    #getting movies
    movies=[i[0] for i in sort_movies]
    #returning recomended movies
    return new_frame['title'].iloc[movies]
```

In [18]:
```
movie=input('Enter movie:')
recommendation(movie)
```

```
Enter movie:The Matrix
```

Out[18]:
```
15880    The Matrix Revolutions
15797       The Matrix Reloaded
13501                     Bound
25164                Jackrabbit
17191                   Replica
3473      The Mysterious Doctor
11040             Leonard Part 6
10501           American Flyers
16353            Terminal Error
Name: title, dtype: object
```