In [1]:

```
import pandas as pd
```

In [2]:

```
import numpy as np
```

In [3]:

```
df = pd.read_csv(r'https://github.com/YBI-Foundation/Dataset/raw/main/Movies%20Recommendati
```

In [4]:

```
df.head()
```

Out[4]:

	Movie_ID	Movie_Title	Movie_Genre	Movie_Language	Movie_Budget	Movie_Popularity	Mov
0	1	Four Rooms	Crime Comedy	en	4000000	22.876230	
1	2	Star Wars	Adventure Action Science Fiction	en	11000000	126.393695	
2	3	Finding Nemo	Animation Family	en	94000000	85.688789	
3	4	Forrest Gump	Comedy Drama Romance	en	55000000	138.133331	
4	5	American Beauty	Drama	en	15000000	80.878605	

5 rows × 21 columns

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4760 entries, 0 to 4759
Data columns (total 21 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Movie_ID                  4760 non-null   int64  
 1   Movie_Title               4760 non-null   object 
 2   Movie_Genre               4760 non-null   object 
 3   Movie_Language            4760 non-null   object 
 4   Movie_Budget              4760 non-null   int64  
 5   Movie_Popularity          4760 non-null   float64
 6   Movie_Release_Date        4760 non-null   object 
 7   Movie_Revenue             4760 non-null   int64  
 8   Movie_Runtime             4758 non-null   float64
 9   Movie_Vote                4760 non-null   float64
 10  Movie_Vote_Count          4760 non-null   int64  
 11  Movie_Homepage            1699 non-null   object 
 12  Movie_Keywords            4373 non-null   object 
 13  Movie_Overview            4757 non-null   object 
 14  Movie_Production_House    4760 non-null   object 
 15  Movie_Production_Country  4760 non-null   object 
 16  Movie_Spoken_Language     4760 non-null   object 
 17  Movie_Tagline             3942 non-null   object 
 18  Movie_Cast                4733 non-null   object 
 19  Movie_Crew                4760 non-null   object 
 20  Movie_Director            4738 non-null   object 
dtypes: float64(3), int64(4), object(14)
memory usage: 781.1+ KB
```

In [6]:

```
df.shape
```

Out[6]:

```
(4760, 21)
```

In [7]:

```
df.columns
```

Out[7]:

```
Index(['Movie_ID', 'Movie_Title', 'Movie_Genre', 'Movie_Language',
       'Movie_Budget', 'Movie_Popularity', 'Movie_Release_Date',
       'Movie_Revenue', 'Movie_Runtime', 'Movie_Vote', 'Movie_Vote_Count',
       'Movie_Homepage', 'Movie_Keywords', 'Movie_Overview',
       'Movie_Production_House', 'Movie_Production_Country',
       'Movie_Spoken_Language', 'Movie_Tagline', 'Movie_Cast', 'Movie_Crew',
       'Movie_Director'],
      dtype='object')
```

In [8]:

```
df_features = df[['Movie_Genre','Movie_Keywords','Movie_Tagline',
                  'Movie_Cast','Movie_Director']].fillna('')
```

In [9]:

```
df_features.shape
```

Out[9]:

```
(4760, 5)
```

In [10]:

```
df_features
```

Out[10]:

	Movie_Genre	Movie_Keywords	Movie_Tagline	Movie_Cast	Movie_Director
0	Crime Comedy	hotel new year's eve witch bet hotel room	Twelve outrageous guests. Four scandalous requ...	Tim Roth Antonio Banderas Jennifer Beals Madon...	Allison Anders
1	Adventure Action Science Fiction	android galaxy hermit death star lightsaber	A long time ago in a galaxy far, far away...	Mark Hamill Harrison Ford Carrie Fisher Peter ...	George Lucas
2	Animation Family	father son relationship harbor underwater fish...	There are 3.7 trillion fish in the ocean, they...	Albert Brooks Ellen DeGeneres Alexander Gould ...	Andrew Stanton
3	Comedy Drama Romance	vietnam veteran hippie mentally disabled runni...	The world will never be the same, once you've ...	Tom Hanks Robin Wright Gary Sinise Mykelti Wil...	Robert Zemeckis
4	Drama	male nudity female nudity adultery midlife cri...	Look closer.	Kevin Spacey Annette Bening Thora Birch Wes Be...	Sam Mendes
...	...	...	...	...	...
4755	Horror		The hot spot where Satan's waitin'.	Lisa Hart Carroll Michael Des Barres Paul Drak...	Pece Dingo
4756	Comedy Family Drama		It’s better to stand out than to fit in.	Roni Akurati Brighton Sharbino Jason Lee Anjul...	Frank Lotito
4757	Thriller Drama	christian film sex trafficking	She never knew it could happen to her...	Nicole Smolen Kim Baldwin Ariana Stephens Brys...	Jaco Booyens
4758	Family				
4759	Documentary	music actors legendary perfomer classic hollyw...		Tony Oppedisano	Simon Napier-Bell

4760 rows × 5 columns

In [11]:

```
X = df_features['Movie_Genre'] + ' ' + df_features['Movie_Keywords'] + ' ' + df_features['M
```

In [12]:

```
X.shape
```

Out[12]:

```
(4760,)
```

In [13]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [14]:

```
tfidf = TfidfVectorizer()
```

In [15]:

```
X = tfidf.fit_transform(X)
```

In [16]:

```
X.shape
```

Out[16]:

```
(4760, 17258)
```

In [17]:

```
print(X)
```

```
  (0, 617)      0.1633382144407513
  (0, 492)      0.1432591540388685
  (0, 15413)    0.1465525095337543
  (0, 9675)     0.14226057295252661
  (0, 9465)     0.1659841367820977
  (0, 1390)     0.16898383612799558
  (0, 7825)     0.09799561597509843
  (0, 1214)     0.13865857545144072
  (0, 729)      0.13415063359531618
  (0, 13093)    0.1432591540388685
  (0, 15355)    0.10477815972666779
  (0, 9048)     0.0866842116160778
  (0, 11161)    0.06250380151644369
  (0, 16773)    0.17654247479915475
  (0, 5612)     0.08603537588547631
  (0, 16735)    0.10690083751525419
  (0, 7904)     0.13348000542112332
  (0, 15219)    0.09800472886453934
  (0, 11242)    0.07277788238484746
  (0, 3878)     0.11998399582562203
  (0, 5499)     0.11454057510303811
  (0, 7071)     0.19822417598406614
  (0, 7454)     0.14745635785412262
  (0, 1495)     0.19712637387361423
  (0, 9206)     0.15186283580984414
  :     :
  (4757, 5455)  0.12491480594769522
  (4757, 2967)  0.16273475835631626
  (4757, 8464)  0.23522565554066333
  (4757, 6938)  0.17088173678136628
  (4757, 8379)  0.17480603856721913
  (4757, 15303) 0.07654356007668191
  (4757, 15384) 0.09754322497537371
  (4757, 7649)  0.11479421494340192
  (4757, 10896) 0.14546473055066447
  (4757, 4494)  0.05675298448720501
  (4758, 5238)  1.0
  (4759, 11264) 0.33947721804318337
  (4759, 11708) 0.33947721804318337
  (4759, 205)   0.3237911628497312
  (4759, 8902)  0.3040290704566037
  (4759, 14062) 0.3237911628497312
  (4759, 3058)  0.2812896191863103
  (4759, 7130)  0.26419662449963793
  (4759, 10761) 0.3126617295732147
  (4759, 4358)  0.18306542312175342
  (4759, 14051) 0.20084315377640435
  (4759, 5690)  0.19534291014627303
  (4759, 15431) 0.19628653185946862
  (4759, 1490)  0.21197258705292082
  (4759, 10666) 0.15888268987343043
```

In [18]:

```
from sklearn.metrics.pairwise import cosine_similarity
```

In [19]:

```
Similarity_Score = cosine_similarity(X)
```

In [20]:

```
Similarity_Score
```

Out[20]:

```
array([[1.        , 0.01351235, 0.03570468, ..., 0.        , 0.        ,
        0.        ],
       [0.01351235, 1.        , 0.00806674, ..., 0.        , 0.        ,
        0.        ],
       [0.03570468, 0.00806674, 1.        , ..., 0.        , 0.08014876,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 1.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.08014876, ..., 0.        , 1.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        1.        ]])
```

In [21]:

```
Similarity_Score.shape
```

Out[21]:

```
(4760, 4760)
```

In [22]:

```
Favourite_Movie_Name = input('Enter your favourite movie name: ')
```

```
Enter your favourite movie name: Finding Nemo
```

In [23]:

```
All_Movies_Title_List = df['Movie_Title'].tolist()
```

In [24]:

```
import difflib
```

In [25]:

```
Movie_Recommendation = difflib.get_close_matches(Favourite_Movie_Name, All_Movies_Title_Lis
print(Movie_Recommendation)
```

```
['Finding Nemo', 'Finding Neverland', 'Finding Forrester']
```

In [26]:

```
Close_Match = Movie_Recommendation[0]
print(Close_Match)
```

```
Finding Nemo
```

In [27]:

```
Index_of_Close_Match_Movie = df[df.Movie_Title == Close_Match]['Movie_ID'].values[0]
print(Index_of_Close_Match_Movie)
```

```
3
```

In [28]:

```
Recommendation_Score = list(enumerate(Similarity_Score[Index_of_Close_Match_Movie]))
print(Recommendation_Score)
```

```
0.012347817758613823), (380, 0.014086936499429128), (381, 0.01001525840385
19), (382, 0.01233226176852341), (383, 0.0031091833139591384), (384, 0.030
193485210942752), (385, 0.010625511059573618), (386, 0.02135950173653044
4), (387, 0.11477520957929307), (388, 0.11544228583648782), (389, 0.020916
018544659028), (390, 0.010110472495720288), (391, 0.06032128006813801), (3
92, 0.06305561111929797), (393, 0.0030893927598550116), (394, 0.0), (395,
0.02707697321692232), (396, 0.04845689624900107), (397, 0.030592172738379
7), (398, 0.009665162762812618), (399, 0.03158017602223937), (400, 0.01392
7199655767483), (401, 0.02712773314224609), (402, 0.028136244965485405),
(403, 0.09693738427823906), (404, 0.0033855308556482904), (405, 0.01028995
8645934943), (406, 0.01563808624088793), (407, 0.05790868288386336), (408,
0.028502059964978316), (409, 0.024172498521579863), (410, 0.03195504398330
047), (411, 0.02361223058208988), (412, 0.05871374808664351), (413, 0.0145
51613824223271), (414, 0.020280476108086583), (415, 0.02795426612126471),
(416, 0.019983394156780365), (417, 0.0230052171334187), (418, 0.0586447994
4341056), (419, 0.017992722270555483), (420, 0.013185636858915668), (421,
0.012181537796565504), (422, 0.02069406341634443), (423, 0.0), (424, 0.029
663060691212623), (425, 0.029637324686911222), (426, 0.02443561076389236),
(427, 0.0383700455685176), (428, 0.042568833525135276), (429, 0.0026908703
28435228), (430, 0.002763533643331422), (431, 0.014347634321983313), (432,
```

In [29]:

```
len(Recommendation_Score)
```

Out[29]:

```
4760
```

In [30]:

```
Sorted_Similar_Movies = sorted(Recommendation_Score, key = lambda x:x[1], reverse = True)
print(Sorted_Similar_Movies)
```

```
[(3, 1.0), (206, 0.179104799704356), (2650, 0.1477798342860161), (2027, 0.
14722613681312074), (164, 0.14589758456473872), (1087, 0.1446813560394558
7), (2158, 0.14253395697027288), (452, 0.13980809021421697), (3866, 0.1316
5771373972115), (2701, 0.12791112574372052), (524, 0.1274427335967744), (2
35, 0.12400604704949818), (3599, 0.12351841944961646), (307, 0.12287691296
12349), (12, 0.12113761972627259), (2595, 0.11955161342027029), (1329, 0.1
1954248931445911), (4630, 0.1184103619750066), (891, 0.11770483501438779),
(1798, 0.11569487202698034), (388, 0.11544228583648782), (387, 0.114775209
57929307), (2528, 0.11396862578533658), (751, 0.11383198385280849), (1048,
0.11304603430735162), (1936, 0.11300327831362768), (2403, 0.11285205452097
76), (3611, 0.10854547292694552), (4712, 0.10834119394067618), (1128, 0.10
817761505635788), (703, 0.10785985844708129), (1120, 0.10763674508378011),
(1367, 0.10484808320254088), (2043, 0.10393333953857428), (852, 0.10296479
032542376), (3965, 0.10294898963590221), (250, 0.10208695685084584), (714,
0.10113204323678006), (1340, 0.10096934364968312), (3018, 0.10093158118351
218), (197, 0.10020116492102216), (685, 0.09987862128515643), (2949, 0.099
80785526501688), (1707, 0.09901403963519267), (1278, 0.0989849410104913),
(1701, 0.09759434765027131), (403, 0.09693738427823906), (4267, 0.09640741
256017413), (919, 0.09634504757220615), (909, 0.09618020057477146), (2504,
```

In [31]:

```
print('Top 30 Movies Suggested for You:  \n')

i = 1
for movie in Sorted_Similar_Movies:
    index = movie[0]
    title_from_index = df[df.index==index]['Movie_Title'].values[0]
    if (i<31):
        print(i, '.',title_from_index)
        i+=1
```

```
Top 30 Movies Suggested for You:

1 . Forrest Gump
2 . The Green Mile
3 . A Home at the End of the World
4 . The Deer Hunter
5 . Rain Man
6 . Cast Away
7 . The Big Bounce
8 . Letters from Iwo Jima
9 . Silver Linings Playbook
10 . Say It Isn't So
11 . What's Eating Gilbert Grape
12 . Apollo 13
13 . Larry Crowne
14 . Contact
15 . Apocalypse Now
16 . A Christmas Carol
17 . You've Got Mail
18 . The Purge: Election Year
19 . Impostor
20 . Robin and Marian
21 . Dangerous Liaisons
22 . Saving Private Ryan
23 . Days of Heaven
24 . Memoirs of an Invisible Man
25 . Gigli
26 . The Musketeer
27 . Last Orders
28 . Moneyball
29 . Dancin' It's On
30 . Snake Eyes
```

In [*]:

```
Movie_Name = input ('Enter your favorite movie name: ')
list_of_all_titles = df['Movie_Title'].tolist()
Find_Close_Match = difflib.get_close_matches(Movie_Name, list_of_all_titles)
Close_Match = Find_Close_Match[0]
Index_of_Movie = df[df.Movie_Title == Close_Match]['Movie_ID'].values[0]
Recommendation_Score = list(enumerate(Similarity_Score[Index_of_Movie]))
sorted_similar_movies = sorted(Recommendation_Score, key = lambda x:x[1], reverse = True)
print('Top 10 Movies Suggestion for you: \n')
i = 1
for movie in sorted_similar_movies:
    index = movie[0]
    title_from_index = df[df.Movie_ID==index]['Movie_Title'].values
    if (i<11):
        print(i, '.',title_from_index)
        i+=1
```

```
Enter your favorite movie name: Finding Nemo
Top 10 Movies Suggestion for you:

1 . ['Finding Nemo']
2 . ['Borat: Cultural Learnings of America for Make Benefit Glorious Nation
of Kazakhstan']
3 . ['In Cold Blood']
4 . ['Intolerable Cruelty']
5 . ['A Nightmare on Elm Street']
6 . ['What the #$*! Do We (K)now!?']
7 . ['Hamlet 2']
8 . ['Ghost Rider']
9 . ['Chasing Mavericks']
10 . ['Chocolate: Deep Dark Secrets']
```

In [ ]: