Import Library

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

In [2]:
```
import numpy as np
```

Import Dataset

In [3]:
```
df = pd.read_csv('Movies Recommendation.csv')
```

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

Out[4]:

	Movie_ID	Movie_Title	Movie_Genre	Movie_Language	Movie_Budget	Movie_Popularity	Movie
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')
```

Get Feature Selection

In [8]:
```
df_features = df[['Movie_Genre','Movie_Keywords','Movie_Tagline','Movie_Cast',
```

```
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_featur
```

```
In [12]: X.shape
```

```
Out[12]: (4760,)
```

Get Feature Text Conversion To Tokens

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)	0.143259154038868
  (0, 15413)	0.1465525095337543
  (0, 9675)	0.14226057295252661
  (0, 9465)	0.165984136782097
  (0, 1390)	0.16898383612799558
  (0, 7825)	0.09799561597509843
  (0, 1214)	0.13865857545144072
  (0, 729)	0.13415063359531618
  (0, 13093)	0.143259154038868
  (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
```

Get Similarity Score Using Cosine Similarity

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)
```

Get Movie Name as Input from User and Validate for Closest Spelling

In [22]:
```
Favourite_Movie_Name = input('Enter your favourite movie name: ')
```

```
Enter your favourite movie name: avataar
```

In [23]:
```
All_Movies_Title_List = df['Movie_Title'].tolist()
```

In [24]:
```
import difflib
```

In [25]:
```
Movie_Reccomendation = difflib.get_close_matches(Favourite_Movie_Name, All_Mov
print(Movie_Reccomendation)
```

```
['Avatar']
```

In [26]:
```
Close_Match = Movie_Reccomendation[0]
print(Close_Match)
```

```
Avatar
```

In [27]:
```python
Index_of_Close_Match_Movie = df[df.Movie_Title == Close_Match]['Movie_ID'].val
print(Index_of_Close_Match_Movie)
```

```
2692
```

Getting a List Of Similar Movies

In [28]:
```python
Recommendation_Score = list(enumerate(Similarity_Score[Index_of_Close_Match_Mo
print(Recommendation_Score)
```

```
[(0, 0.009805093506053453), (1, 0.0), (2, 0.0), (3, 0.00800429043895183), (4,
0.0026759665928032302), (5, 0.009639835665946627), (6, 0.004963665756185002
6), (7, 0.012848827437220958), (8, 0.0027543335470164663), (9, 0.006078822904
16431), (10, 0.007539724639541887), (11, 0.0026263170118314906), (12, 0.00270
8340354961457), (13, 0.012904730427356216), (14, 0.0), (15, 0.022556564866386
044), (16, 0.005959078936688496), (17, 0.0), (18, 0.013639824714195078), (19,
0.008784739948684396), (20, 0.0026527570934446066), (21, 0.01521161402784047
1), (22, 0.006522322352622825), (23, 0.0026429172195160193), (24, 0.001656448
2636435309), (25, 0.025600660315408176), (26, 0.0024815199490618002), (27, 0.
0047922703978129), (28, 0.0), (29, 0.023288277583204436), (30, 0.004648836119
227042), (31, 0.006723965537835127), (32, 0.007984548069367697), (33, 0.01861
2326068635436), (34, 0.007439622267479848), (35, 0.0060612328203774185), (36,
0.0), (37, 0.0), (38, 0.008085428274959462), (39, 0.0046323263203813065), (4
0, 0.015305064222782005), (41, 0.0028220612513682524), (42, 0.007236825272071
698), (43, 0.014851289474516489), (44, 0.03961780430399104), (45, 0.089993246
43162435), (46, 0.01855499596172605), (47, 0.010374759033888029), (48, 0.0156
73410180680997), (49, 0.0), (50, 0.006986992676753986), (51, 0.01496597941178
2002), (52, 0.013600804094978335), (53, 0.0), (54, 0.0), (55, 0.0), (56, 0.00
6687995450791239), (57, 0.010835008478228547), (58, 0.0), (59, 0.0), (60, 0.0
```

In [29]:
```python
len(Recommendation_Score)
```

Out[29]:
```
4760
```

Get All Movies Sort Based on Recommendation Score with respect to Favourate Movie

sorting the movies based on their similarity score

In [30]:
```python
Sorted_Similar_Movies = sorted(Recommendation_Score, key = lambda x:x[1], reve
print(Sorted_Similar_Movies)
```

```
[(2692, 1.0000000000000002), (3276, 0.11904275527845871), (3779, 0.1018580579
7079382), (62, 0.10153560702418994), (2903, 0.10063787314386034), (1647, 0.09
397055536069451), (4614, 0.09362226751043302), (4375, 0.09117512301489193),
(45, 0.08999324643162435), (1383, 0.08425242441722802), (110, 0.0836178477502
9485), (628, 0.08334515876919323), (1994, 0.08287835345252216), (2558, 0.0826
7633224298852), (1070, 0.08104448918225104), (4378, 0.07894345402700793), (13
41, 0.07732693809361939), (1977, 0.07510309168081497), (3465, 0.0741108984125
5805), (3053, 0.0732438108456325), (4116, 0.07264153003988619), (1982, 0.0724
6569778553744), (2538, 0.06802035746289192), (3248, 0.06683400770968473), (39
46, 0.06577120166835922), (3480, 0.06560363079666712), (254, 0.06351452702158
421), (590, 0.06275727122098754), (3450, 0.06274272831079739), (1886, 0.06267
985852941994), (4594, 0.0624699521049894), (2112, 0.06218435141221765), (84,
0.0618237599684129), (675, 0.06176991517572303), (3854, 0.06161566270378365),
(1134, 0.06151448371353247), (3463, 0.060706045656025415), (4252, 0.060598155
08412411), (4137, 0.06047703709769184), (1118, 0.05998954734066491), (4389,
0.059627372790876695), (3385, 0.05898328865604495), (4062, 0.0589589942058893
6), (282, 0.05879285017883316), (4398, 0.05848106495843603), (424, 0.05839654
732699123), (2358, 0.05826769637272555), (3462, 0.057434079728437545), (2985,
0.05717355295839895), (2318, 0.05698746413620388), (1021, 0.0567199996476896
```

print the name of similar movies based on their index

In [31]:
```
print('Top 30 movies suggested : \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 : 

1 . Niagara
2 . Caravans
3 . My Week with Marilyn
4 . Brokeback Mountain
5 . Harry Brown
6 . Night of the Living Dead
7 . The Curse of Downers Grove
8 . The Boy Next Door
9 . Back to the Future
10 . The Juror
11 . Some Like It Hot
12 . Enough
13 . The Kentucky Fried Movie
14 . Eye for an Eye
15 . Welcome to the Sticks
16 . Alice Through the Looking Glass
17 . Superman III
18 . The Misfits
19 . Premium Rush
20 . Duel in the Sun
21 . Sabotage
22 . Small Soldiers
23 . All That Jazz
24 . Camping Sauvage
25 . The Raid
26 . Beyond the Black Rainbow
27 . To Kill a Mockingbird
28 . World Trade Center
29 . The Dark Knight Rises
30 . Tora! Tora! Tora!
```

Top 10- Movie Recommendation System

```
In [32]: Movie_Name = input('Enter your favourite 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], reve

print('Top 10 movies suggested : \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 favourite movie name: avataar
Top 10 movies suggested :

1 . ['Avatar']
2 . ['The Girl on the Train']
3 . ['Act of Valor']
4 . ['Donnie Darko']
5 . ['Precious']
6 . ['Freaky Friday']
7 . ['The Opposite Sex']
8 . ['Heaven is for Real']
9 . ['Run Lola Run']
10 . ['Elizabethtown']
```