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

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

Import Data set

In [6]:
```
df = pd.read_csv(r'//home//apiiit-rkv//Desktop//Movies Recommendation.csv')
```

In [7]:
```
df.head()
```

Out[7]:

	Movie_ID	Movie_Title	Movie_Genre	Movie_Language	Movie_Budget	Movie_Popularity	Movie_Release_Date	Movie_Revenue	Movie_Runtime	Movie_Vote	...	Movie_Homepage	Movie_Keywords	Movie_Overview
0	1	Four Rooms	Crime Comedy	en	4000000	22.876230	09-12-1995	4300000	98.0	6.5	...	NaN	hotel new year's eve witch bet hotel room	It's Ted the Bellhop's first night on the job....
1	2	Star Wars	Adventure Action Science Fiction	en	11000000	126.393695	25-05-1977	775398007	121.0	8.1	...	http://www.starwars.com/films/star-wars-episod...	android galaxy hermit death star lightsaber	Princess Leia is captured and held hostage by ...
2	3	Finding Nemo	Animation Family	en	94000000	85.688789	30-05-2003	940335536	100.0	7.6	...	http://movies.disney.com/finding-nemo	father son relationship harbor underwater fish...	Nemo, an adventurous young clownfish, is unexp...
3	4	Forrest Gump	Comedy Drama Romance	en	55000000	138.133331	06-07-1994	677945399	142.0	8.2	...	NaN	vietnam veteran hippie mentally disabled runni...	A man with a low IQ has accomplished great thi...
4	5	American Beauty	Drama	en	15000000	80.878605	15-09-1999	356296601	122.0	7.9	...	http://www.dreamworks.com/ab/	male nudity female nudity adultery midlife cri...	Lester Burnham, a depressed suburban father in...

5 rows × 21 columns

In [8]:
```
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 [9]:
```
df.shape
```

Out[9]:
```
(4760, 21)
```

In [10]:
```
df.columns
```

Out[10]:
```
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 [11]:
```
df_features = df[['Movie_Genre','Movie_Keywords','Movie_Tagline','Movie_Cast','Movie_Director']].fillna('')
```

In [*]:
```
df_features.shape
```

In [13]:
```
df_features
```

Out[13]:

	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 [14]:
```
x = df_features['Movie_Genre']+ '' + df_features['Movie_Keywords']+ '' + df_features['Movie_Tagline']+ '' + df_features['Movie_Cast']+ '' + df_features['Movie_Directo
```

In [15]:
```
x
```

Out[15]:
```
0       Crime Comedyhotel new year's eve witch bet hot...
1       Adventure Action Science Fictionandroid galaxy...
2       Animation Familyfather son relationship harbor...
3       Comedy Drama Romancevietnam veteran hippie men...
4       Dramamale nudity female nudity adultery midlif...
                              ...
4755    HorrorThe hot spot where Satan's waitin'.Lisa ...
4756    Comedy Family DramaIt’s better to stand out th...
4757    Thriller Dramachristian film sex traffickingSh...
4758                                                Family
4759    Documentarymusic actors legendary perfomer cla...
Length: 4760, dtype: object
```

In [17]:
```
x.shape
```

Out[17]:
```
(4760,)
```


Get Features Text Conversion to Tokens

In [18]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [21]:
```
tfidf = TfidfVectorizer()
```

In [22]:
```
x = tfidf.fit_transform(x)
```

In [23]:
```
x.shape
```

Out[23]:
```
(4760, 27466)
```

In [25]:
```
print(x)
```

```
  (0, 1028)     0.16196019146631543
  (0, 24785)    0.1954632929283795
  (0, 15844)    0.14205053053187272
  (0, 15553)    0.17099186675469502
  (0, 2132)     0.18002354204307464
  (0, 13312)    0.09914387783149516
  (0, 1887)     0.14106037409792174
  (0, 1216)     0.13920306109638164
  (0, 21158)    0.14205053053187272
  (0, 24701)    0.11357423942624927
  (0, 14943)    0.091376722056839
  (0, 18098)    0.06200430666985742
  (0, 26738)    0.175053052455033
  (0, 9790)     0.08712552095655665
  (0, 26675)    0.1116831168780693
  (0, 13401)    0.13748876529263096
  (0, 24105)    0.10726395493180996
  (0, 18192)    0.07278761942152372
  (0, 6172)     0.11970212451073885
  (0, 9626)     0.11757910435818826
  (0, 11960)    0.20134029899961134
  (0, 12801)    0.1530338818199682
  (0, 2292)     0.1954632929283795
  (0, 15172)    0.1537691763994982
  (0, 18196)    0.08579029869987485
  :     :
  (4757, 1839)  0.19327629083107672
  (4757, 5410)  0.19734759150400596
  (4757, 11350) 0.21582294886514122
  (4757, 22017) 0.1646400247918531
  (4757, 17789) 0.18881341937258544
  (4757, 9484)  0.1411164779725638
  (4757, 14176) 0.2330831990045816
  (4757, 11762) 0.17321388936472645
  (4757, 14052) 0.1776312353410007
  (4757, 24232) 0.10947784435203887
  (4757, 24746) 0.09744940789814222
  (4757, 13079) 0.12400374714145113
  (4757, 17721) 0.1489085353667712
  (4758, 8651)  1.0
  (4759, 18229) 0.33527342183765224
  (4759, 22434) 0.33527342183765224
  (4759, 18841) 0.33527342183765224
  (4759, 6950)  0.33527342183765224
  (4759, 345)   0.31978160936741457
  (4759, 14742) 0.31978160936741457
  (4759, 12139) 0.2778063685558062
  (4759, 4446)  0.282306565154911
  (4759, 17552) 0.3087899934962816
  (4759, 9955)  0.21805075638656476
  (4759, 2285)  0.21465229435984196
```


Get Similarly Score using Cosine Similarly

In [26]:
```
from sklearn.metrics.pairwise import cosine_similarity
```

In [27]:
```
Similarity_score = cosine_similarity(x)
```

In [28]:
```
Similarity_score
```

Out[28]:
```
array([[1.        , 0.01438634, 0.03807033, ..., 0.        , 0.        ,
        0.        ],
       [0.01438634, 1.        , 0.00844858, ..., 0.        , 0.        ,
        0.        ],
       [0.03807033, 0.00844858, 1.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 1.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 1.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        1.        ]])
```

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

Out[29]:
```
(4760, 4760)
```

Get Movie Name as Input from User and Validate for CLOSEST SPELLING

In [*]:
```
Favourite_Movie_Name = input(' Enter Your favourite movie name : ')
```

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

In [*]:
```
import difflib
```

In [*]:
```
Movie_Recommendation = difflib.get_close_matches(Favourite_Movie_Name, All_Movie_Title_List)
```

In [*]:
```
print(Movie_Recommendation)
```

In [ ]:
```

```