```
import pandas as pd
```

```
import numpy as np
```

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

```
df.head()
```

	Movie_ID	Movie_Title	Movie_Genre	Movie_Language	Movie_Budget	Movie_Popularity
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	Comedy Drama omance	en	55000000	138.133331
4	5	American Beauty	Drama	en	15000000	80.878605

5 rows × 21 columns

Saved successfully!

```
df.info()
```

✓ 19s completed at 6:55 PM

```
<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
```

```
df.describe()
```

	Movie_ID	Movie_Budget	Movie_Popularity	Movie_Revenue	Movie_Runtime	Mov
count	4760.000000	4.760000e+03	4760.000000	4.760000e+03	4758.000000	476
			)7 21.599510	8.263743e+07	107.184111	
			)7 31.887919	1.630554e+08	21.960332	
min	1.000000	0.000000e+00	0.000372	0.000000e+00	0.000000	
25%	1190.750000	9.257500e+05	4.807074	0.000000e+00	94.000000	
50%	2380.500000	1.500000e+07	13.119058	1.944716e+07	104.000000	
75%	3572.250000	4.000000e+07	28.411929	9.341276e+07	118.000000	
max	4788.000000	3.800000e+08	875.581305	2.787965e+09	338.000000	1(

Saved successfully!

```
df.shape
```

```
(4760, 21)
```

```
df.columns
```

```
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 Feture Selection

```
df_features = df[['Movie_Genre','Movie_Keywords','Movie_Tagline','Movie_Cast','Movie_Direct
```

selected five existing feature to recommend movies .it may very form one project to another like one can add vote count budget language etc

```
df_features.shape
```

```
(4760, 5)
```

```
df_features
```

	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

Saved successfully!

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

```
x df_features['Movie_Genre']+''+df_features['Movie_Keywords']+''+df_features['Movie_Tagline
```

Saved successfully!

```
x
```

```
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
```

Get feature Text Conversion to Token

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

```
tfidf=TfidfVectorizer()
```

```
x=tfidf.fit_transform(x)
```

```
x.shape
```

```
(4760, 27466)
```

```
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
                0996
Saved successfully!  2372
                3885
  (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 Similarity Score using Cosine Similarity

cosine_similarity compute the l2-normalized dot product of vector Euclidean(L2)normalization prjects the vector onto the unit sphare and thair dot product is then the cosine of the angle between the point denoted by the vectoer

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

```
similarity_score=cosine_similarity(x)
```

```
similarity_score
```

Saved successfully!

```
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.        ]])
```

```
similarity_score.shape
```

```
(4760, 4760)
```

```
(4760, 4760)
```

Get movies name as input from user and validate for closest Spelling

```
Favourite_Movie_Name=input('Enter your favorate movies name')
```

```
Enter your favorate movies nameavtaar
```

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

```
import difflib
```

```
Movies_Recommendation=difflib.get_close_matches(Favourite_Movie_Name,All_Movies_Title_List
print(Movies_Recommendation)
```

```
['Avatar', 'Gattaca']
```

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

```
Avatar
```

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

```
2692
```

Saved successfully!

```
ovies
Recommendation_score=list(enumerate(similarity_score[Index_of_Close_Match_Movie]))
print(Recommendation_score)
```

```
[(0, 0.011030833064862307), (1, 0.0), (2, 0.0), (3, 0.008556266387176359), (4, 0.0),
```

```
len(Recommendation_score)
```

```
4760
```

get All Movies Sort Based on Recommendation Score wrt Favourite Movies

```
# sorting the movies based on the similarity score
Sorted_similar_movies = sorted(Recommendation_score, key = lambda x:x[1], reverse = True)
print(Sorted_similar_movies)
```

```
[(2692, 1.0000000000000007), (110, 0.08195740486074782), (1994, 0.08144186697343063),
```

```
# print the similer movies based on the index
print('top 30 movies suggested for you :\n')
i=1
for movies in Sorted_similar_movies:
  index = movies[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 . Niagara
2 . Some Like It Hot
3 . The Kentucky Fried Movie
4 . The Juror
5 . Enough
6 . Duel in the Sun
7 . Superman III
8 . Eye for an Eye
9 . The Misfits
10 . Beyond the Black Rainbow
11 . Brokeback Mountain
12 . All That Jazz
13 . Tora! Tora! Tora!
14 . Master and Commander: The Far Side of the World
15 . To Kill a Mockingbird
16 . Running with Scissors
19 . Edge of Darkness
20 . Man on Wire
21 . The Odd Life of Timothy Green
22 . Intolerable Cruelty
23 . The Curse of Downers Grove
24 . The Great Gatsby
25 . Mad Max 2: The Road Warrior
26 . Source Code
27 . Song One
28 . The Longest Yard
29 . The Boy Next Door
30 . The Lazarus Effect
```

Saved successfully!

Top 10 Movies Recommendation System

```
Movie_name = input('Enter your favourite movies name : ')
list_of_all_title = df['Movie_Title'].tolist()
Find_Close_Match = difflib.get_close_matches(Movie_name, list_of_all_title)
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_similear_movies= sorted(Recommendation_score,key = lambda x:x[1], reverse = True)
print('Top 10 movies suggested for you :\n ')

i=1
for movies in Sorted_similar_movies:
  index = movies[0]
  title_from_index=df[df.index == index]['Movie_Title'].values[0]
  if(i<11):
    print(i,'.',title_from_index)
    i+=1
```

```
 Enter your favourite movies name : avtaar
 Top 10 movies suggested for you :

 1 . Niagara
 2 . Some Like It Hot
 3 . The Kentucky Fried Movie
 4 . The Juror
 5 . Enough
 6 . Duel in the Sun
 7 . Superman III
 8 . Eye for an Eye
 9 . The Misfits
 10 . Beyond the Black Rainbow
```

Saved successfully!

Saved successfully!