Tittle of project:-

Movie Recommendation System

Objective:-

The objective of this project is to develop a basic movie recommendation system using Python and Pandas. The system will employ collaborative and content-based filtering techniques to predict and recommend movies based on user preferences and movie similarity. This project aims to explore how machine learning algorithms can be applied to personalized recommendations and enhance user experience by suggesting relevant movies.

Data Source:-

The dataset used for this project is "Movie Recommendation.csv", which contains movie titles, genres, ratings, and user interactions with movies. The dataset will be used to train and test the recommendation algorithms.

Import Library

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
```

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_Release_Date	Movie_Revenue	Movie_Runt
0	1	Four Rooms	Crime Comedy	en	4000000	22.876230	09-12-1995	4300000	9
1	2	Star Wars	Adventure Action Science Fiction	en	11000000	126.393695	25-05-1977	775398007	12
2	3	Finding Nemo	Animation Family	en	94000000	85.688789	30-05-2003	940335536	10
3	4	Forrest Gump	Comedy Drama Romance	en	55000000	138.133331	06-07-1994	677945399	14
4	5	American Beauty	Drama	en	15000000	80.878605	15-09-1999	356296601	12

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)

Get Feature Selection

In [9]:
```
df.columns
```

Out[9]:
```
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 [11]:
```
df_features = df[['Movie_Genre','Movie_Keywords','Movie_Tagline', 'Movie_Cast', 'Movie_Director']].fillna('')
```

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

Out[12]:
```
(4760, 5)
```

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 [51]:
```
X = df_features['Movie_Genre'] +''+ df_features['Movie_Keywords'] +' ' + df_features['Movie_Tagline'] +'' + df_fe
```

In [52]:
```
X
```

Out[52]:
```
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    Horror The hot spot where Satan's waitin'.Lisa...
4756    Comedy Family Drama It's better to stand out t...
4757    Thriller Dramachristian film sex trafficking S...
4758                                              Family
4759    Documentarymusic actors legendary perfomer cla...
Length: 4760, dtype: object
```

In [53]:
```
X.shape
```

Out[53]:
```
(4760,)
```

Get Feature Text Conversion to Token

In [54]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [55]:
```
tfidf = TfidfVectorizer()
```

In [56]:
```
X =tfidf.fit_transform(X)
```

In [46]:
```
X.shape
```

Out[46]:
```
(4760, 23793)
```

In [35]:
```
print(X)
```

```
  (0, 852)	0.16217249683336685
  (0, 21402)	0.1957195158049675
  (0, 13674)	0.14124528295872352
  (0, 13427)	0.1712160112849373
  (0, 1817)	0.18025952573650775
  (0, 11358)	0.0988350581246672
  (0, 1594)	0.14124528295872352
  (0, 1006)	0.13766899231333668
  (0, 18283)	0.1422367373383193
  (0, 21330)	0.11235434200282547
  (0, 12875)	0.08749958532574936
  (0, 15653)	0.062057722304633696
  (0, 23133)	0.17528252058679097
  (0, 8264)	0.08582179830177389
  (0, 23086)	0.10613790405866935
  (0, 11440)	0.13319322288674884
  (0, 20802)	0.09730528545238074
  (0, 15745)	0.07239084741433557
  (0, 5361)	0.11985903592984133
  (0, 8124)	0.11429504819312693
  (0, 10217)	0.19680948308476473
  (0, 10925)	0.1467726505080627
  (0, 1959)	0.1957195158049675
  (0, 13087)	0.153970744582032
  (0, 15749)	0.08224550765638705
  :	:
  (4757, 9644)	0.21830021697895316
  (4757, 19177)	0.16534759252826392
  (4757, 19050)	0.16310345137799656
  (4757, 15405)	0.18854241579183778
  (4757, 8074)	0.12577403371941828
  (4757, 12156)	0.2357585844526793
  (4757, 10034)	0.17173558080547904
  (4757, 12044)	0.17849495780797606
  (4757, 20915)	0.10698214459605472
  (4757, 21364)	0.09781388196944436
  (4757, 11134)	0.11539174587760775
  (4757, 15342)	0.14579429647497183
  (4758, 7373)	1.0
  (4759, 15777)	0.3301866073009095
  (4759, 19424)	0.3301866073009095
  (4759, 16330)	0.3301866073009095
  (4759, 5931)	0.3301866073009095
  (4759, 301)	0.3149298387433153
  (4759, 12690)	0.3149298387433153
  (4759, 3881)	0.27359145206697794
  (4759, 10379)	0.2569662483000291
  (4759, 15183)	0.30410498918216633
  (4759, 8403)	0.21220479182655572
  (4759, 21422)	0.20687479150919763
  (4759, 1952)	0.2106033662683427
```

Get Similarity Score Using Consine Similarity

In [57]:
```
from sklearn.metrics.pairwise import cosine_similarity
```

In [59]:
```
Similarity_Score = cosine_similarity(X)
```

In [60]:
```
Similarity_Score
```

Out[60]:
```
array([[1.        , 0.01336343, 0.03473927, ..., 0.        , 0.        ,
         0.        ],
       [0.01336343, 1.        , 0.00783257, ..., 0.        , 0.        ,
         0.        ],
       [0.03473927, 0.00783257, 1.        , ..., 0.        , 0.        ,
         0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 1.        , 0.        ,
         0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 1.        ,
         0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
         1.        ]])
```

In [61]:
```
Similarity_Score.shape
```

Out[61]:
```
(4760, 4760)
```

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

In [62]:
```
Favourite_Movie_Name = input(' Enter your favourite movie name : ')
```

```
 Enter your favourite movie name : avtaar
```

In [63]:
```
All_Movies_Title_List = df['Movie_Title'].tolist()
```

In [64]:
```
import difflib
```

In [65]:
```
Movie_Recommendation = difflib.get_close_matches(Favourite_Movie_Name, All_Movies_Title_List)
print(Movie_Recommendation)
```

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

In [66]:
```
Close_Match = Movie_Recommendation[0]
print(Close_Match)
```

```
Avatar
```

In [67]:
```
Index_of_Close_Match_Movie = df[df.Movie_Title == Close_Match]['Movie_ID'].values[0]
print(Index_of_Close_Match_Movie)
```

```
2692
```

In [69]:
```
# getting a list of similar movies
Recommendation_Score = list(enumerate(Similarity_Score[Index_of_Close_Match_Movie]))
print(Recommendation_Score)
```

```
06, 0.020608389956072188), (707, 0.003116334287315463), (708, 0.0), (709, 0.0), (710, 0.03603538208174131),
(711, 0.0), (712, 0.0031968792396109254), (713, 0.0), (714, 0.0), (715, 0.006818266898347443), (716, 0.003884
349033158063), (717, 0.0), (718, 0.0), (719, 0.008085834880600462), (720, 0.0), (721, 0.03501178133037393),
(722, 0.003301505992989483), (723, 0.0), (724, 0.0), (725, 0.0), (726, 0.0), (727, 0.0035546141238383786),
(728, 0.0038699331608586453), (729, 0.004003963965459868), (730, 0.0), (731, 0.0), (732, 0.00867424437007433
7), (733, 0.010735268384966124), (734, 0.009442714438358887), (735, 0.005182716254873506), (736, 0.0), (737,
0.007033337331426293), (738, 0.0), (739, 0.004081352650157756), (740, 0.0), (741, 0.0), (742, 0.0093729001814
60038), (743, 0.0037605963245735136), (744, 0.00334683235059458), (745, 0.0), (746, 0.0), (747, 0.0), (748,
0.00659324533607134), (749, 0.00356588168369434), (750, 0.00364297940750424), (751, 0.0), (752, 0.00668564454
26842605), (753, 0.009710664142885333), (754, 0.0), (755, 0.003565598257264353), (756, 0.0), (757, 0.01707938
143601082), (758, 0.006337000098831047), (759, 0.0), (760, 0.010408942337480437), (761, 0.02834450511325627),
(762, 0.011119102495657904), (763, 0.019994856274368557), (764, 0.0), (765, 0.0), (766, 0.00588546952125536
3), (767, 0.0), (768, 0.01797528839770504), (769, 0.0036190152835016156), (770, 0.0), (771, 0.0), (772, 0.0),
(773, 0.0), (774, 0.036335641581060826), (775, 0.00993246143456355), (776, 0.016670616643633302), (777, 0.0),
(778, 0.006602540601066455), (779, 0.01566623550901749), (780, 0.0), (781, 0.0), (782, 0.006680985414743898
5), (783, 0.003848127950300587), (784, 0.0), (785, 0.0), (786, 0.0), (787, 0.01564108226168861), (788, 0.0367
686510560816), (789, 0.0), (790, 0.008228147520133814), (791, 0.0), (792, 0.00379083198737289), (793, 0.0),
(794, 0.017119883575707293), (795, 0.01957097996915857), (796, 0.018356151708587364), (797, 0.0), (798, 0.003
4882844272049896), (799, 0.01195663429809342), (800, 0.008931321996412027), (801, 0.007869255443913854), (80
2, 0.0), (803, 0.0), (804, 0.003311187920159797), (805, 0.0), (806, 0.012450752543173607), (807, 0.0), (808,
```

In [70]:
```
len(Recommendation_Score)
```

Out[70]:
```
4760
```

Get All Movies Sort Recommendation Score Wrt Favourite Movie

In [71]:
```
# sorting the movies based on their similarity score

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

```
[(2692, 1.0), (3779, 0.09541442708476246), (4375, 0.08811373585254995), (110, 0.08209379514905785), (1383, 0.
08187851491634932), (1994, 0.08104978188369505), (628, 0.07908791546028547), (2558, 0.07886770559980592), (13
41, 0.07680860572145522), (3053, 0.07606439212124107), (1977, 0.0750851089148789), (3248, 0.0730901643337959
5), (1886, 0.06763079297286732), (2538, 0.06721483866343898), (3480, 0.06653196457770548), (62, 0.06478196756
609386), (1118, 0.062233830131116316), (254, 0.061943435050753726), (2903, 0.06132726547329475), (3450, 0.059
86105681769641), (1021, 0.05960286200687369), (2112, 0.05923474643280513), (2318, 0.058360022007445535), (406
2, 0.05821589115522707), (1134, 0.05611792693976014), (2026, 0.05596889427205487), (4614, 0.055678983753371325), (3385, 0.054667277036053265), (3655, 0.05453823673836775), (4398, 0.054489659904346344), (3728, 0.0542129
3743435012), (2358, 0.05415079031194871), (4389, 0.054045385053373224), (2985, 0.05363541019897365), (3946,
0.05331891512243263), (873, 0.05163869108306247), (3334, 0.051368610941424205), (4268, 0.05111087350005724),
(1243, 0.051092941006492226), (2214, 0.05013380204458546), (2294, 0.04930340177355187), (1809, 0.049058357261
160306), (227, 0.04868840592791839), (1878, 0.04859689516626076), (137, 0.04843632355909984), (213, 0.0481996
2256711177), (2550, 0.04813354802577181), (1023, 0.04790142824656811), (2579, 0.047798278357967246), (3203,
0.047706849167075085), (2944, 0.04755470181525129), (4682, 0.04747897666102893), (1821, 0.04746071084160482),
(3395, 0.047453604833711166), (45, 0.04716128465128405), (3552, 0.04695791194323787), (292, 0.046851371654091
45), (2191, 0.046543845730377065), (1070, 0.04652058148263319), (3294, 0.046353041953860113), (2133, 0.046300
445004110145), (2653, 0.04619208249696517), (3753, 0.04607820466378583), (408, 0.046034271807239575), (519,
0.046022597674199014), (3518, 0.04593342607421794), (877, 0.04587950998505488), (1030, 0.045878792655801255),
(3973, 0.0458593395252534), (3374, 0.04581581459898833), (2893, 0.04532051679577615), (3091, 0.045199104568
```

In [73]:
```
# print the name of similar movies based on the index

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 . Niagara
2 . My Week with Marilyn
3 . The Boy Next Door
4 . Some Like It Hot
5 . The Juror
6 . The Kentucky Fried Movie
7 . Enough
8 . Eye for an Eye
9 . Superman III
10 . Duel in the Sun
11 . The Misfits
12 . Camping Sauvage
13 . Tora! Tora! Tora!
14 . All That Jazz
15 . Beyond the Black Rainbow
16 . Brokeback Mountain
17 . Master and Commander: The Far Side of the World
18 . To Kill a Mockingbird
19 . Harry Brown
20 . The Dark Knight Rises
21 . Running with Scissors
22 . Edge of Darkness
23 . Man on Wire
24 . Broken Vessels
25 . Mad Max 2: The Road Warrior
26 . Intolerable Cruelty
27 . The Curse of Downers Grove
28 . Source Code
29 . The Great Gatsby
30 . Song One
```

Top 10 Movies Recommendation system

In [77]:
```
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], reverse = True)

print('Top 10 Movies suggested 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 favourite movie name : avtaar
Top 10 Movies suggested for you :

1 . ['Avatar']
2 . ['Act of Valor']
3 . ['Heaven is for Real']
4 . ['The Godfather']
5 . ['Elizabethtown']
6 . ['New Nightmare']
7 . ['Gerry']
8 . ['Bright Lights, Big City']
9 . ['A Prairie Home Companion']
10 . ['Cradle Will Rock']
```