In [1]:
```
import numpy as np
import pandas as pd
import difflib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import os
```

In [2]:
```
file_path = r"D:\CV things\ML projects\movies.csv"  # Specify the file path
movies_data = pd.read_csv(file_path)  # Read the CSV file into a DataFrame

# Use 'movies_data' for further processing
print(movies_data.head())  # Displaying the first few rows as an example
```

```
   index     budget                                     genres  \
0      0  237000000  Action Adventure Fantasy Science Fiction
1      1  300000000                 Adventure Fantasy Action
2      2  245000000                   Action Adventure Crime
3      3  250000000              Action Crime Drama Thriller
4      4  260000000          Action Adventure Science Fiction

                                      homepage      id  \
0                  http://www.avatarmovie.com/   19995
1  http://disney.go.com/disneypictures/pirates/     285
2   http://www.sonypictures.com/movies/spectre/  206647
3           http://www.thedarkknightrises.com/   49026
4         http://movies.disney.com/john-carter   49529

                                        keywords original_language  \
0  culture clash future space war space colony so...                en
1  ocean drug abuse exotic island east india trad...                en
2        spy based on novel secret agent sequel mi6                 en
3  dc comics crime fighter terrorist secret ident...                en
4  based on novel mars medallion space travel pri...                en

                          original_title  \
0                                 Avatar
1  Pirates of the Caribbean: At World's End
2                                Spectre
3                  The Dark Knight Rises
4                           John Carter

                                        overview  popularity  ... runtime  \
0  In the 22nd century, a paraplegic Marine is di...  150.437577  ...   162.0
1  Captain Barbossa, long believed to be dead, ha...  139.082615  ...   169.0
2  A cryptic message from Bond’s past sends him o...  107.376788  ...   148.0
3  Following the death of District Attorney Harve...  112.312950  ...   165.0
4  John Carter is a war-weary, former military ca...   43.926995  ...   132.0

                                  spoken_languages    status  \
0  [{"iso_639_1": "en", "name": "English"}, {"iso...  Released
1          [{"iso_639_1": "en", "name": "English"}]  Released
2  [{"iso_639_1": "fr", "name": "Fran\u00e7ais"},...  Released
3          [{"iso_639_1": "en", "name": "English"}]  Released
4          [{"iso_639_1": "en", "name": "English"}]  Released

                                     tagline  \
0                  Enter the World of Pandora.
1  At the end of the world, the adventure begins.
2                         A Plan No One Escapes
3                               The Legend Ends
4           Lost in our world, found in another.

                                   title vote_average vote_count  \
0                                 Avatar          7.2      11800
1  Pirates of the Caribbean: At World's End          6.9       4500
2                                Spectre          6.3       4466
3                  The Dark Knight Rises          7.6       9106
4                           John Carter          6.1       2124

                                               cast  \
0  Sam Worthington Zoe Saldana Sigourney Weaver S...
1  Johnny Depp Orlando Bloom Keira Knightley Stel...
2  Daniel Craig Christoph Waltz L\u00e9a Seydoux ...
3  Christian Bale Michael Caine Gary Oldman Anne ...
4  Taylor Kitsch Lynn Collins Samantha Morton Wil...
```

```
                                                  crew           director
0  [{'name': 'Stephen E. Rivkin', 'gender': 0, 'd...      James Cameron
1  [{'name': 'Dariusz Wolski', 'gender': 2, 'depa...     Gore Verbinski
2  [{'name': 'Thomas Newman', 'gender': 2, 'depar...         Sam Mendes
3  [{'name': 'Hans Zimmer', 'gender': 2, 'departm...  Christopher Nolan
4  [{'name': 'Andrew Stanton', 'gender': 2, 'depa...     Andrew Stanton

[5 rows x 24 columns]
```

In [3]:
```
# Selecting the relevant features for recommendation
selected_features = ['genres', 'keywords', 'tagline', 'cast', 'director']
```

In [4]:
```
# Check for Null Values
print(movies_data.isna().sum())
```

```
index                      0
budget                     0
genres                    28
homepage                3091
id                         0
keywords                 412
original_language          0
original_title             0
overview                   3
popularity                 0
production_companies       0
production_countries       0
release_date               1
revenue                    0
runtime                    2
spoken_languages           0
status                     0
tagline                  844
title                      0
vote_average               0
vote_count                 0
cast                      43
crew                       0
director                  30
dtype: int64
```

In [5]:
```
# Replace the null values with empty strings
for feature in selected_features:
    movies_data[feature] = movies_data[feature].fillna('')
```

In [6]:
```
# Combine all the selected features
movies_data['combined_features'] = movies_data['genres'] + ' ' + movies_data['ke
    'tagline'] + ' ' + movies_data['cast'] + ' ' + movies_data['director']
```

In [7]:
```
# Convert the text data to feature vectors using TF-IDF
vectorizer = TfidfVectorizer()
feature_vectors = vectorizer.fit_transform(movies_data['combined_features'])
```

In [8]:
```
similarity = cosine_similarity(feature_vectors)
```

In [9]:
```
# Function to recommend movies
def recommend_movie(movie_name, movies_data, similarity):
    list_of_all_titles = movies_data['title'].tolist()

    # Finding the close match for the movie name given by the user
    find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
    close_match = find_close_match[0]

    # Finding the index of the movie with the title
    index_of_the_movie = movies_data[movies_data.title == close_match].index.val

    # Get the similarity row for the selected index
    similarity_score = list(enumerate(similarity[index_of_the_movie]))

    # Sorting the movies based on their similarity score
    sorted_similar_movies = sorted(similarity_score, key=lambda x: x[1], reverse

    # Print the name of similar movies based on the index - Top 30
    print('Movies suggested for you: \n')
    i = 1
    for movie in sorted_similar_movies:
        index = movie[0]
        title_from_index = movies_data[movies_data.index == index]['title'].valu
        if i < 30:
            print(i, '.', title_from_index)
            i += 1
```

In [10]:
```
# Get user input for a favorite movie
movie_name = input('Enter your favorite movie name: ')
```

```
Enter your favorite movie name: Spider man
```

In [11]:
```
# Make movie recommendations
recommend_movie(movie_name, movies_data, similarity)
```

Movies suggested for you:

1 . Spider-Man 3
2 . Spider-Man 2
3 . Spider-Man
4 . The Specials
5 . The One
6 . Oz: The Great and Powerful
7 . George of the Jungle
8 . The Good German
9 . Daddy's Home
10 . We Bought a Zoo
11 . Wimbledon
12 . John Carter
13 . The Calling
14 . Don McKay
15 . The Grace Card
16 . Fear and Loathing in Las Vegas
17 . Crazy/Beautiful
18 . Win a Date with Tad Hamilton!
19 . Eternal Sunshine of the Spotless Mind
20 . Imagine That
21 . Heaven is for Real
22 . Dick
23 . Max
24 . Drop Dead Gorgeous
25 . Predators
26 . Small Soldiers
27 . Batman
28 . Batman & Robin
29 . A History of Violence