In [ ]:
```
# Importing dependencies

import numpy as np
import pandas as pd
import difflib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

In [ ]:
```
# Loading the dataset as dataframe

df = pd.read_csv('dataset.csv')
df.head()
```

In [ ]:
```
# Viewing the number of rows and columns in the data frame

df.shape
```

In [ ]:
```
# Viewing the data-types for each feature

df.info()
```

In [ ]:
```
# Selecting relevant features from the dataset

selected_features = ['genres', 'keywords', 'tagline', 'cast', 'director', 's
print(selected_features)
```

In [ ]:
```
# Converting relevant features' Dtype to string type

df['genres'] = df['genres'].astype(str)
df['keywords'] = df['keywords'].astype(str)
df['tagline'] = df['tagline'].astype(str)
df['cast'] = df['cast'].astype(str)
df['director'] = df['director'].astype(str)
df['spoken_languages'] = df['spoken_languages'].astype(str)
df['vote_average'] = df['vote_average'].astype(str)
```

In [ ]:
```
# Viewing updated Dtypes

df[selected_features].info()
```

In [ ]:
```
# Replacing the null valuess with null string

for feature in selected_features:
  df[feature] = df[feature].fillna('')
```

In [ ]:
```
# Combining all the 7 selected features

combined_features = df['genres']+' '+df['keywords']+' '+df['tagline']+' '+df
print(combined_features)
```

In [ ]:
```
# Converting the text data to feature vectors
```

```
vectorizer = TfidfVectorizer()
feature_vectors = vectorizer.fit_transform(combined_features)
print(feature_vectors)
```

In [ ]:
```
# Similarity scores using cosine similarity

similarity = cosine_similarity(feature_vectors)
print(similarity)
```

In [ ]:
```
# Verifying if the data is whole or not

print(similarity.shape)
```

In [ ]:
```
# Input from the user

user_input = input(' Enter your favourite movie name : ')
print(user_input)
```

In [ ]:
```
# Creating a list of titles in the dataset

title_list = df['title'].tolist()
print(title_list)
```

In [ ]:
```
# Finding present matches in dataset

present_matches = difflib.get_close_matches(user_input, title_list)
print(present_matches)

if present_matches:
    close_match = present_matches[0]
    print(close_match)
else:
    print("No matches found")
```

In [ ]:
```
# Finding the index of the movie

index_of_the_movie = df[df.title == close_match]['index'].values[0]
print(index_of_the_movie)
```

In [ ]:
```
# Getting the list of similar movies

similarity_score = list(enumerate(similarity[index_of_the_movie]))
print(similarity_score)
```

In [ ]:
```
# Checking on the length of similarity score

len(similarity_score)
```

In [ ]:
```
# Sorting the movies based on their similarity score in descending order

sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], revers
print(sorted_similar_movies)
```

In [ ]:
```
# Printing the sorted list of movies

print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = df[df.index==index]['title'].values[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

In [ ]:
```
# Putting together the recommendation model in a single cell

user_input = input(' Enter your favourite movie name : ')

title_list = df['title'].tolist()

present_matches = difflib.get_close_matches(user_input, title_list)

close_match = present_matches[0]

index_of_the_movie = df[df.title == close_match]['index'].values[0]

similarity_score = list(enumerate(similarity[index_of_the_movie]))

sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], revers

print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = df[df.index==index]['title'].values[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```