1. Import Dependencies:

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

2. Data collection & preprocessing:

A- Dowlanding the data:

In [ ]:
```
data_movie = pd.read_csv("C:/Machine_learning Python/projets/movieRecomndation/m
```

B- Head of the data :

In [ ]:
```
data_movie.head()
```

Out[ ]:

	index	budget	genres	homepage	id	key
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	spa
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	eas
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy or
3	3	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	49026	dc te
4	4	260000000	Action Adventure Science Fiction	http://movies.disney.com/john-carter	49529	ba me

5 rows × 24 columns

C- Number of tow and columns:

In [ ]:
```
data_movie.shape
```

Out[ ]:
```
(4803, 24)
```

D- Type of the columns data :

In [ ]:
```
data_movie.dtypes
```

Out[ ]:
```
index                     int64
budget                    int64
genres                   object
homepage                 object
id                        int64
keywords                 object
original_language        object
original_title           object
overview                 object
popularity              float64
production_companies     object
production_countries     object
release_date             object
revenue                   int64
runtime                 float64
spoken_languages         object
status                   object
tagline                  object
title                    object
vote_average            float64
vote_count                int64
cast                     object
crew                     object
director                 object
dtype: object
```

E- Information about the data:

In [ ]:
```
data_movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   index                 4803 non-null   int64
 1   budget                4803 non-null   int64
 2   genres                4775 non-null   object
 3   homepage              1712 non-null   object
 4   id                    4803 non-null   int64
 5   keywords              4391 non-null   object
 6   original_language     4803 non-null   object
 7   original_title        4803 non-null   object
 8   overview              4800 non-null   object
 9   popularity            4803 non-null   float64
 10  production_companies  4803 non-null   object
 11  production_countries  4803 non-null   object
 12  release_date          4802 non-null   object
 13  revenue               4803 non-null   int64
 14  runtime               4801 non-null   float64
 15  spoken_languages      4803 non-null   object
 16  status                4803 non-null   object
 17  tagline               3959 non-null   object
 18  title                 4803 non-null   object
 19  vote_average          4803 non-null   float64
 20  vote_count            4803 non-null   int64
 21  cast                  4760 non-null   object
 22  crew                  4803 non-null   object
 23  director              4773 non-null   object
dtypes: float64(3), int64(5), object(16)
memory usage: 900.7+ KB
```

F- Selection the releveant features:

In [ ]:
```
selected_features = ["genres","keywords", "tagline", "cast","director"]
print(selected_features)
```

```
['genres', 'keywords', 'tagline', 'cast', 'director']
```

2. General statistical :

A- Replacing the null values (just for 5 columns):

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

B- Combining all the selectef_features:

In [ ]:
```
combined_feature = data_movie["genres"] + ' '+data_movie["keywords"] + ' '+data_
```

In [ ]:
```
print(combined_feature)
```

```
0       Action Adventure Fantasy Science Fiction cultu...
1       Adventure Fantasy Action ocean drug abuse exot...
2       Action Adventure Crime spy based on novel secr...
3       Action Crime Drama Thriller dc comics crime fi...
4       Action Adventure Science Fiction based on nove...
                               ...
4798    Action Crime Thriller united states\u2013mexic...
4799    Comedy Romance   A newlywed couple's honeymoon...
4800    Comedy Drama Romance TV Movie date love at fir...
4801        A New Yorker in Shanghai Daniel Henney Eli...
4802    Documentary obsession camcorder crush dream gi...
Length: 4803, dtype: object
```

C- Converting the text data to feature vector:

In [ ]:
```
vectorizer = TfidfVectorizer()
```

In [ ]:
```
feature_vector = vectorizer.fit_transform(combined_feature)
```

In [ ]:
```
print(feature_vector)
```

```
  (0, 2432)	0.17272411194153
  (0, 7755)	0.1128035714854756
  (0, 13024)	0.1942362060108871
  (0, 10229)	0.16058685400095302
  (0, 8756)	0.22709015857011816
  (0, 14608)	0.15150672398763912
  (0, 16668)	0.19843263965100372
  (0, 14064)	0.20596090415084142
  (0, 13319)	0.2177470539412484
  (0, 17290)	0.20197912553916567
  (0, 17007)	0.23643326319898797
  (0, 13349)	0.15021264094167086
  (0, 11503)	0.27211310056983656
  (0, 11192)	0.09049319826481456
  (0, 16998)	0.1282126322850579
  (0, 15261)	0.07095833561276566
  (0, 4945)	0.24025852494110758
  (0, 14271)	0.21392179219912877
  (0, 3225)	0.24960162956997736
  (0, 16587)	0.12549432354918996
  (0, 14378)	0.33962752210959823
  (0, 5836)	0.1646750903586285
  (0, 3065)	0.22208377802661425
  (0, 3678)	0.21392179219912877
  (0, 5437)	0.1036413987316636
  :	:
  (4801, 17266)	0.2886098184932947
  (4801, 4835)	0.24713765026963996
  (4801, 403)	0.17727585190343226
  (4801, 6935)	0.2886098184932947
  (4801, 11663)	0.21557500762727902
  (4801, 1672)	0.1564793427630879
  (4801, 10929)	0.13504166990041588
  (4801, 7474)	0.11307961713172225
  (4801, 3796)	0.3342808988877418
  (4802, 6996)	0.5700048226105303
  (4802, 5367)	0.22969114490410403
  (4802, 3654)	0.262512960498006
  (4802, 2425)	0.24002350969074696
  (4802, 4608)	0.24002350969074696
  (4802, 6417)	0.21753405888348784
  (4802, 4371)	0.1538239182675544
  (4802, 12989)	0.1696476532191718
  (4802, 1316)	0.1960747079005741
  (4802, 4528)	0.19504460807622875
  (4802, 3436)	0.21753405888348784
  (4802, 6155)	0.18056463596934083
  (4802, 4980)	0.16078053641367315
  (4802, 2129)	0.3099656128577656
  (4802, 4518)	0.16784466610624255
  (4802, 11161)	0.17867407682173203
```

3. Cosing Similarity:

A- Similarity scores:

In [ ]:
```
similarity_list = cosine_similarity(feature_vector)
```

In [ ]:
```
print(similarity_list)
```

```
[[1.         0.07219487 0.037733   ... 0.         0.         0.        ]
 [0.07219487 1.         0.03281499 ... 0.03575545 0.         0.        ]
 [0.037733   0.03281499 1.         ... 0.         0.05389661 0.        ]
 ...
 [0.         0.03575545 0.         ... 1.         0.         0.02651502]
 [0.         0.         0.05389661 ... 0.         1.         0.        ]
 [0.         0.         0.         ... 0.02651502 0.         1.        ]]
```

In [ ]:
```
print(similarity_list.shape)
```

```
(4803, 4803)
```

B- Getting the movie name:

In [ ]:
```
movie_name = input("Enter your favourite movie name: ")
```

C- Creating a list of all the movies names given the dataset:

In [ ]:
```
title_list = data_movie["title"].to_list()
```

D- Finding the close match for the movie name giving by the user:

In [ ]:
```
finding = difflib.get_close_matches(movie_name, title_list)
print(finding)
```

```
['Spectre', 'Sphere', 'Species']
```

In [ ]:
```
close_match = finding[0]
print(close_match) #Most similar movie
```

```
Spectre
```

E- Finding the index of the movie with title:

In [ ]:
```
index = data_movie[data_movie.title == close_match]['index'].values[0]
print(index)
```

```
2
```

F- Find the similar movie:

In [ ]:
```
similarity_score = list(enumerate(similarity_list[index]))
```

E- Sorting the movie based on hteir similarity score:

In [ ]:
```
sorted_list = sorted(similarity_score , key= lambda x:x[1] , reverse=True)
```

F- The most similar movie (suggestion):

In [ ]:
```
i = 1
print("The top 10 similar movies to '", movie_name, "' are: \n")
for index_score in sorted_list[1:11]:  # Assuming sorted_list contains tuples of
    movie_index = index_score[0]  # Extract the movie index from the tuple
    title = data_movie.iloc[movie_index]['title']  # Use iloc to access the Data
    print("TOP", i, ":", title)
    i += 1
```

```
The top 10 similar movies to ' avatar ' are:

TOP 1 : Skyfall
TOP 2 : Mission: Impossible - Ghost Protocol
TOP 3 : Johnny English Reborn
TOP 4 : Quantum of Solace
TOP 5 : Irreversible
TOP 6 : The Incredibles
TOP 7 : The Green Hornet
TOP 8 : Red Dragon
TOP 9 : The Sorcerer's Apprentice
TOP 10 : The Legend of Tarzan
```