```
import numpy as np
import pandas as pd
import os
```

```
movies = pd.read_csv('tmdb_5000_movies.csv')
credits = pd.read_csv('tmdb_5000_credits.csv')
```

```
movies.columns
```

```
Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'original_language',
       'original_title', 'overview', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'vote_average',
       'vote_count'],
      dtype='object')
```

```
credits.columns
```

```
Index(['movie_id', 'title', 'cast', 'crew'], dtype='object')
```

```
movies.head(1).columns
movies.shape
credits.shape
```

```
(4803, 4)
```

```
movies.head(1).columns
```

```
Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'original_language',
       'original_title', 'overview', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'vote_average',
       'vote_count'],
      dtype='object')
```

```
movies = movies.merge(credits,on='title',how='outer')
movies.shape
movies.head(1)
```

	budget	genres	homepage	id	keywords	original_language	original_title	ov
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	ce pa M

1 rows × 23 columns

```
movie_id      0
title         0
overview      3
genres        0
keywords      0
cast          0
crew          0
popularity    0
dtype: int64
```

```
movies = movies[['movie_id','title','overview','genres','keywords','cast','crew','popularity']]
```

```
movies.isnull().sum()
movies.dropna(inplace=True)
```

```
movies.duplicated().sum()
```

```
0
```

```
movies.iloc[0].genres
```

```
'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]'
```

```
import ast
def preprocess_genre(obj):
  L=[]
  for i in ast.literal_eval(obj):
    # print(i)
    L.append(i['name'].lower())
  return L
    # print(ast.literal_eval(x.genres[0].name))
# preprocess_genre(movies['genres'][0])
movies['genres']=movies['genres'].apply(preprocess_genre)
```

```
movies.head(1)
```

	movie_id	title	overview	genres	keywords	cast	crew	popularity
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[action, adventure, fantasy, science fiction]	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...	150.437577

```
def preprocess_kw(obj):
  L=[]
  for i in ast.literal_eval(obj):
    # print(i)
    L.append(i['name'].lower())
  return L
movies['keywords']=movies['keywords'].apply(preprocess_kw)
```

```
movies.head(1)
```

	movie_id	title	overview	genres	keywords	cast	crew	popularity
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[action, adventure, fantasy, science fiction]	[culture clash, future, space war, space colon...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...	150.437577

```
(ast.literal_eval(movies.head(1).cast[0]))
def preprocess_Cast(obj,k=3):
  L=[]
  for i in ast.literal_eval(obj):
    if(k>0):
      L.append(i['name'].lower().replace(' ',''))
      k-=1
  return L
movies['cast']=movies['cast'].apply(preprocess_Cast)
```

```
# (ast.literal_eval(movies.head(1).cast[0]))
def preprocess_Crew(obj):

  for i in ast.literal_eval(obj):
    if i['job']=='Director':
      return i['name'].lower().replace(" ","")
movies['crew']=movies['crew'].apply(preprocess_Crew)
```

```
def postProcess_Crew(dir):
  L=[]
  L.append(dir)
  return L
movies['crew']=movies['crew'].apply(postProcess_Crew)
```

```
movies['overview']=movies['overview'].apply(lambda x: (x.split()))
```

```
movies.head(1)
movies['genres']=movies['genres'].apply(lambda x:[i.lower().replace(" ","") for i in x])
movies['keywords']=movies['keywords'].apply(lambda x:[i.lower().replace(" ","") for i in x])
movies.head(1)
```

	movie_id	title	overview	genres	keywords	cast	crew	popularity
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[action, adventure, fantasy, sciencefiction]	[cultureclash, future, spacewar, spacecolony, ...	[samworthington, zoesaldana, sigourneyweaver]	[jamescameron]	150.437577

```
movies['tags']=movies['overview']+movies['genres']+movies['keywords'] + movies['cast']+movies['crew']
movies.head(1)
```

	movie_id	title	overview	genres	keywords	cast	crew	popularity	
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[action, adventure, fantasy, sciencefiction]	[cultureclash, future, spacewar, spacecolony, ...	[samworthington, zoesaldana, sigourneyweaver]	[jamescameron]	150.437577	cer par

```
new_df=movies[['title','movie_id','tags','popularity']]
new_df.head(1)
```

	title	movie_id	tags	popularity
0	Avatar	19995	[In, the, 22nd, century,, a, paraplegic, Marin...	150.437577

```
def applyIt(obj):
  try:

    return " ".join(list(filter(lambda item: item is not None, obj)))
  except:
    print(obj)
new_df['tags']=new_df['tags'].apply(applyIt)
```

```
<ipython-input-177-6a8ae226dbd5>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  new_df['tags']=new_df['tags'].apply(applyIt)
```

```
new_df['tags']=new_df['tags'].apply(lambda x: x.lower())
```

```
<ipython-input-178-150a467269a2>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  new_df['tags']=new_df['tags'].apply(lambda x: x.lower())
```

```
new_df.head(1)
```

```
from sklearn.feature_extraction.text import CountVectorizer
cv= CountVectorizer(max_features=5000,stop_words='english')
```

```
vectors = cv.fit_transform(new_df['tags']).toarray()
```

```
vectors
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
cv.get_feature_names_out()
```

```
array(['000', '007', '10', ..., 'zone', 'zoo', 'zooeydeschanel'],
      dtype=object)
```

```
import nltk
from nltk.stem.porter import PorterStemmer
ps=PorterStemmer()
```

```
def stem(txt):
  L=[]
  for i in txt.split():
    L.append(ps.stem(i))
  return " ".join(L)
new_df['tags']=new_df['tags'].apply(stem)
```

```
'in the 22nd century, a parapleg marin is dispatch to the moon pandora on a uniqu mission, but becom torn between follow order and pro
tect an alien civilization. action adventur fantasi sciencefict cultureclash futur spacewar spacecoloni societi spacetravel futurist r
omanc space alien tribe alienplanet cgi marin soldier battl loveaffair antiwar powerrel mindandsoul 3d samworthington zoesaldana sigou
rneyweav jamescameron'
```

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

```
similarity =cosine_similarity(vectors)
```

```
similarity.shape
```

```
(4806, 4806)
```

```
def recommend(movie):
  movie_index=new_df[new_df['title']==movie].index[0]
  distances=similarity[movie_index]
  movie_list= sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]
  for i in movie_list:
    print(new_df.iloc[i[0]].title)
```

```
recommend('Avatar')
```

```
Aliens vs Predator: Requiem
Aliens
Falcon Rising
Independence Day
Titan A.E.
```

✓ 0s completed at 5:57 PM