In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import json
```

In [2]:

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

In [3]:

```
movies.shape, credits.shape
```

Out[3]:

```
((4803, 20), (4803, 4))
```

In [4]:

```
movies.head(2)
```

Out[4]:

	budget	genres	homepage	id	keywords	original_
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	

In [5]:

```
movies["genres"][0]
```

Out[5]:

```
'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 1
4, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]'
```

In [6]:

```
1 json.loads(movies["genres"][0])
```

Out[6]:

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

In [7]:

```
1 # Data Merging
```

In [8]:

```
1 movie_credits = pd.merge(movies,credits, left_on="id" ,right_on = "movie_id")
2 movie_credits.shape
```

Out[8]:

```
(4803, 24)
```

In [9]:

```
1 movie_credits.head(2)
```

Out[9]:

	budget	genres	homepage	id	keywords	original_
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	

2 rows × 24 columns

In [10]:

```
1 movie_credits = movie_credits[['movie_id','title_x','overview','genres','keywords','
```

In [11]:

```
movie_credits.dropna(inplace=True)
```

In [12]:

```
# Function to extract genre names from JSON data
def extract_values(str_lst):
    values = json.loads(str_lst)
    return [value['name'] for value in values]
```

In [13]:

```
# Apply the extract_genres function to each row in the DataFrame
movie_credits['genres'] = movie_credits['genres'].apply(extract_values)
movie_credits['genres'].head()
```

Out[13]:

```
0    [Action, Adventure, Fantasy, Science Fiction]
1                     [Adventure, Fantasy, Action]
2                       [Action, Adventure, Crime]
3                  [Action, Crime, Drama, Thriller]
4              [Action, Adventure, Science Fiction]
Name: genres, dtype: object
```

In [14]:

```
movie_credits['keywords'] = movie_credits['keywords'].apply(extract_values)
movie_credits['keywords'].head()
```

Out[14]:

```
0    [culture clash, future, space war, space colon...
1    [ocean, drug abuse, exotic island, east india ...
2    [spy, based on novel, secret agent, sequel, mi...
3    [dc comics, crime fighter, terrorist, secret i...
4    [based on novel, mars, medallion, space travel...
Name: keywords, dtype: object
```

In [15]:

```
movie_credits["cast"] = movie_credits["cast"].apply(extract_values).apply(lambda x:x[
movie_credits["cast"].head()
```

Out[15]:

```
0    [Sam Worthington, Zoe Saldana, Sigourney Weaver]
1       [Johnny Depp, Orlando Bloom, Keira Knightley]
2        [Daniel Craig, Christoph Waltz, Léa Seydoux]
3        [Christian Bale, Michael Caine, Gary Oldman]
4      [Taylor Kitsch, Lynn Collins, Samantha Morton]
Name: cast, dtype: object
```

In [16]:

```
# Function to extract genre names from JSON data
def fetch_director(str_lst):
    values = json.loads(str_lst)
    return [value['name'] for value in values if value['job'] == 'Director']
```

In [17]:

```
fetch_director(movie_credits['crew'][0])
```

Out[17]:

```
['James Cameron']
```

In [18]:

```
movie_credits['crew'] = movie_credits['crew'].apply(fetch_director)
movie_credits['crew'].head()
```

Out[18]:

```
0        [James Cameron]
1       [Gore Verbinski]
2           [Sam Mendes]
3    [Christopher Nolan]
4       [Andrew Stanton]
Name: crew, dtype: object
```

In [19]:

```
movie_credits['overview'] = movie_credits['overview'].str.split()
movie_credits['overview'].head()
```

Out[19]:

```
0    [In, the, 22nd, century,, a, paraplegic, Marin...
1    [Captain, Barbossa,, long, believed, to, be, d...
2    [A, cryptic, message, from, Bond’s, past, send...
3    [Following, the, death, of, District, Attorney...
4    [John, Carter, is, a, war-weary,, former, mili...
Name: overview, dtype: object
```

In [20]:

```
movie_credits.head()
```

Out[20]:

	movie_id	title_x	overview	genres	keywords	cast	crew
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[James Cameron]
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski]
2	206647	Spectre	[A, cryptic, message, from, Bond’s, past, send...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[Sam Mendes]
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[Christian Bale, Michael Caine, Gary Oldman]	[Christopher Nolan]
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[Andrew Stanton]

```
"Sam Worthington" --> "SamWorthington"
```

In [21]:

```
def collapse(lst):
    final_lst = []
    for i in lst:
        final_lst.append(i.replace(" ",""))
    return final_lst
```

In [22]:

```
#movie_credits["cast"].apply(collapse)

movie_credits["cast"] = movie_credits["cast"].apply(lambda x: [i.replace(" ","") for
movie_credits['crew'] = movie_credits['crew'].apply(collapse)
movie_credits['genres'] = movie_credits['genres'].apply(collapse)
movie_credits['keywords'] = movie_credits['keywords'].apply(collapse)
```

In [23]:

```
movie_credits["genres"].head()
```

Out[23]:

```
0    [Action, Adventure, Fantasy, ScienceFiction]
1                    [Adventure, Fantasy, Action]
2                      [Action, Adventure, Crime]
3                 [Action, Crime, Drama, Thriller]
4              [Action, Adventure, ScienceFiction]
Name: genres, dtype: object
```

In [24]:

```
movie_credits['tags'] = movie_credits['overview'] + movie_credits['genres'] + movie_c
```

In [25]:

```
final_df = movie_credits.drop(columns=['overview','genres','keywords','cast','crew'])
final_df.head()
```

Out[25]:

	movie_id	title_x	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...
2	206647	Spectre	[A, cryptic, message, from, Bond’s, past, send...
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...

In [26]:

```
final_df["tags"] = final_df["tags"].apply(lambda x: " ".join(x))
```

In [27]:

```
final_df["tags"] = final_df["tags"].str.lower()
```

In [28]:

```
final_df["tags"].head()
```

Out[28]:

```
0    in the 22nd century, a paraplegic marine is di...
1    captain barbossa, long believed to be dead, ha...
2    a cryptic message from bond’s past sends him o...
3    following the death of district attorney harve...
4    john carter is a war-weary, former military ca...
Name: tags, dtype: object
```

In [29]:

```
1 final_df.head()
```

Out[29]:

	movie_id	title_x	tags
0	19995	Avatar	in the 22nd century, a paraplegic marine is di...
1	285	Pirates of the Caribbean: At World's End	captain barbossa, long believed to be dead, ha...
2	206647	Spectre	a cryptic message from bond’s past sends him o...
3	49026	The Dark Knight Rises	following the death of district attorney harve...
4	49529	John Carter	john carter is a war-weary, former military ca...

In [30]:

```
1 # Text Vectorization
```

In [31]:

```
1 final_df["tags"][0]
```

Out[31]:

```
'in the 22nd century, a paraplegic marine is dispatched to the moon pandor
a on a unique mission, but becomes torn between following orders and prote
cting an alien civilization. action adventure fantasy sciencefiction cultu
reclash future spacewar spacecolony society spacetravel futuristic romance
space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar
powerrelations mindandsoul 3d samworthington zoesaldana sigourneyweaver ja
mescameron'
```

In [32]:

```
1 final_df.columns
```

Out[32]:

```
Index(['movie_id', 'title_x', 'tags'], dtype='object')
```

In [33]:

```
1 final_df.rename({"title_x":"title"}, axis=1, inplace = True)
2 final_df.columns
```

Out[33]:

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

In [34]:

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

In [35]:

```
vector = cv.fit_transform(final_df['tags'])
vector
```

Out[35]:

```
<4800x5000 sparse matrix of type '<class 'numpy.int64'>'
        with 133677 stored elements in Compressed Sparse Row format>
```

In [36]:

```
vector = vector.toarray()
vector
```

Out[36]:

```
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]], dtype=int64)
```

In [37]:

```
vector[0]
```

Out[37]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [38]:

```
1 print(cv.get_stop_words())
```

```
frozenset({'each', 'how', 'whence', 'moreover', 'else', 'even', 'been', 'm
ay', 'six', 'beside', 'but', 'either', 'con', 'your', 'behind', 'three',
'or', 'last', 'name', 'throughout', 'across', 'i', 'now', 'this', 'rathe
r', 'himself', 'front', 'whereas', 'can', 'itself', 'via', 'sixty', 'perha
ps', 'then', 'also', 'might', 'what', 'often', 'whoever', 'afterwards', 'f
ind', 'twelve', 'made', 'system', 'ltd', 'whenever', 'from', 'such', 'neit
her', 'sometimes', 'whereafter', 'eg', 'namely', 'are', 'forty', 'bill',
'toward', 'latterly', 'almost', 'side', 'someone', 'much', 'seemed', 'hasn
t', 'well', 'within', 'seeming', 'still', 'sincere', 'please', 'upon', 'am
ongst', 'over', 'become', 'meanwhile', 'cannot', 'fifty', 'nothing', 'mos
t', 'ours', 'take', 'nevertheless', 'although', 'to', 'found', 'therefor
e', 'be', 'cry', 'among', 'however', 'bottom', 'otherwise', 'give', 'ever
y', 'will', 'hers', 'if', 'always', 'except', 'since', 'towards', 'until',
'put', 'thereby', 'wherein', 'noone', 'wherever', 'along', 'while', 'anyon
e', 'she', 'third', 'anything', 'back', 'together', 'detail', 'they', 'fil
l', 'around', 'into', 'more', 'five', 'somehow', 'one', 'get', 'everywher
e', 'whether', 'down', 'after', 'out', 'everyone', 'there', 'being', 'tw
o', 'between', 'thereafter', 'others', 'anyhow', 'herein', 'couldnt', 'thr
ough', 'etc', 'becoming', 'whereby', 'why', 'describe', 'for', 'during',
'something', 'yourself', 'its', 'move', 'where', 'whereupon', 're', 'wit
h', 'so', 'about', 'thin', 'of', 'ourselves', 'without', 'thence', 'top',
'my', 'it', 'all', 'cant', 'has', 'keep', 'an', 'her', 'never', 'ie', 'who
se', 'under', 'is', 'already', 'myself', 'than', 'many', 'do', 'seem', 'el
sewhere', 'per', 'a', 'few', 'that', 'empty', 'him', 'twenty', 'in', 'mil
l', 'part', 'call', 'same', 'thereupon', 'onto', 'go', 'amount', 'next',
'serious', 'our', 'you', 'none', 'on', 'up', 'fire', 'interest', 'eleven',
'alone', 'them', 'was', 'at', 'show', 'he', 'nobody', 'off', 'those', 'les
s', 'somewhere', 'several', 'would', 'nine', 'some', 'me', 'hence', 'forme
r', 'four', 'too', 'amoungst', 'first', 'nowhere', 'before', 'further', 'i
nc', 'eight', 'everything', 'ever', 'who', 'not', 'whither', 'below', 'a
s', 'de', 'becomes', 'here', 'done', 'had', 'must', 'became', 'latter', 's
ometime', 'therein', 'anyway', 'very', 'both', 'due', 'see', 'their', 'bes
ides', 'nor', 'once', 'yet', 'indeed', 'his', 'beforehand', 'another', 'en
ough', 'thick', 'which', 'we', 'own', 'ten', 'herself', 'un', 'were', 'who
m', 'least', 'anywhere', 'thus', 'could', 'hereafter', 'am', 'formerly',
'yours', 'mine', 'full', 'whole', 'us', 'themselves', 'seems', 'have', 'fi
fteen', 'hereby', 'against', 'yourselves', 'these', 'by', 'above', 'shoul
d', 'and', 'because', 'only', 'when', 'hundred', 'whatever', 'co', 'beyon
d', 'though', 'any', 'thru', 'other', 'the', 'mostly', 'hereupon', 'agai
n', 'no'})
```

In [39]:

```
1 list(cv.get_feature_names_out())[50:60]
```

Out[39]:

```
['abandoned',
 'abducted',
 'abigailbreslin',
 'abilities',
 'ability',
 'able',
 'aboard',
 'abuse',
 'abusive',
 'academy']
```

In [40]:

```
1 import nltk
```

In [41]:

```
1 from nltk.stem.porter import PorterStemmer
2
3 ps = PorterStemmer()
```

In [42]:

```
1 ps.stem("love")
```

Out[42]:

```
'love'
```

In [43]:

```
1 ps.stem("loved")
```

Out[43]:

```
'love'
```

In [44]:

```
1 ps.stem("loving")
```

Out[44]:

```
'love'
```

In [45]:

```
def stem(txt):
    lst = []

    for i in txt.split():
        lst.append(ps.stem(i))

    return " ".join(lst)
```

In [46]:

```
stem(final_df["tags"][0])
```

Out[46]:

```
'in the 22nd century, a parapleg marin is dispatch to the moon pandora on
a uniqu mission, but becom torn between follow order and protect an alien
civilization. action adventur fantasi sciencefict cultureclash futur space
war spacecoloni societi spacetravel futurist romanc space alien tribe alie
nplanet cgi marin soldier battl loveaffair antiwar powerrel mindandsoul 3d
samworthington zoesaldana sigourneyweav jamescameron'
```

In [47]:

```
final_df["tags"] = final_df["tags"].apply(stem)
```

In [48]:

```
final_df.head()
```

Out[48]:

	movie_id	title	tags
0	19995	Avatar	in the 22nd century, a parapleg marin is dispa...
1	285	Pirates of the Caribbean: At World's End	captain barbossa, long believ to be dead, ha c...
2	206647	Spectre	a cryptic messag from bond’ past send him on a...
3	49026	The Dark Knight Rises	follow the death of district attorney harvey d...
4	49529	John Carter	john carter is a war-weary, former militari ca...

In [49]:

```
cv = CountVectorizer(max_features=5000,stop_words='english')
```

In [50]:

```
vector = cv.fit_transform(final_df['tags'])
vector
```

Out[50]:

```
<4800x5000 sparse matrix of type '<class 'numpy.int64'>'
        with 145203 stored elements in Compressed Sparse Row format>
```

In [51]:

```
vector = vector.toarray()
vector
```

Out[51]:

```
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]], dtype=int64)
```

In [52]:

```
list(cv.get_feature_names_out())[50:60]
```

Out[52]:

```
['500',
 '60',
 '70',
 '80',
 'aaron',
 'aaroneckhart',
 'abandon',
 'abduct',
 'abigailbreslin',
 'abil']
```

In [53]:

```
final_df.shape
```

Out[53]:

```
(4800, 3)
```

In [54]:

```
# Calculate vectors
```

In [55]:

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

In [56]:

```
similarity = cosine_similarity(vector)
```

In [57]:

```
similarity.shape
```

Out[57]:

```
(4800, 4800)
```

In [58]:

```
similarity
```

Out[58]:

```
array([[1.        , 0.08585457, 0.08718573, ..., 0.04559608, 0.        ,
        0.        ],
       [0.08585457, 1.        , 0.06154575, ..., 0.02414023, 0.        ,
        0.02654659],
       [0.08718573, 0.06154575, 1.        , ..., 0.02451452, 0.        ,
        0.        ],
       ...,
       [0.04559608, 0.02414023, 0.02451452, ..., 1.        , 0.03962144,
        0.04229549],
       [0.        , 0.        , 0.        , ..., 0.03962144, 1.        ,
        0.08714204],
       [0.        , 0.02654659, 0.        , ..., 0.04229549, 0.08714204,
        1.        ]])
```

In [59]:

```
similarity[0]
```

Out[59]:

```
array([1.        , 0.08585457, 0.08718573, ..., 0.04559608, 0.        ,
       0.        ])
```

In [60]:

```
# Recommendation
```

In [61]:

```
movie_index = final_df[final_df["title"] == "Avatar"].index[0]
movie_index
```

Out[61]:

```
0
```

In [62]:

```
distances = similarity[movie_index]
distances
```

Out[62]:

```
array([1.        , 0.08585457, 0.08718573, ..., 0.04559608, 0.        ,
       0.        ])
```

In [63]:

```
index_list = list(enumerate(distances))
index_list[:10]
```

Out[63]:

```
[(0, 0.9999999999999998),
 (1, 0.08585457105482137),
 (2, 0.08718572905786445),
 (3, 0.074458079104994),
 (4, 0.19184045508446734),
 (5, 0.1098436937909367),
 (6, 0.04078236951430929),
 (7, 0.1487044791289829),
 (8, 0.06003002251876642),
 (9, 0.09802861627917438)]
```

In [64]:

```
similar_movie = sorted(index_list, reverse = True, key = lambda x: x[1])[1:11]
similar_movie
```

Out[64]:

```
[(1213, 0.29061909685954823),
 (2403, 0.2726248784031353),
 (3723, 0.26401000024165),
 (507, 0.25903973506580724),
 (539, 0.2537477434955704),
 (582, 0.2484013136974297),
 (1201, 0.24784079854830487),
 (1191, 0.23490461932490855),
 (778, 0.23485569615051044),
 (4041, 0.23089735286521348)]
```

In [65]:

```
for i in similar_movie:
    print(final_df.iloc[i[0]].title)
```

```
Aliens vs Predator: Requiem
Aliens
Falcon Rising
Independence Day
Titan A.E.
Battle: Los Angeles
Predators
Small Soldiers
Meet Dave
U.F.O.
```

In [66]:

```
def recommender(movie_name):
    movie_index = final_df[final_df["title"] == movie_name].index[0]
    distances = similarity[movie_index]
    index_list = list(enumerate(distances))
    similar_movie = sorted(index_list, reverse = True, key = lambda x: x[1])[1:11]
    for i in similar_movie:
        print(final_df.iloc[i[0]].title)
```

In [67]:

```
recommender("Batman")
```

```
Batman & Robin
Batman Begins
Batman Returns
The R.M.
The Dark Knight Rises
Batman Forever
Code of Honor
Micmacs
Punisher: War Zone
Rockaway
```

In [68]:

```
recommender("Iron Man")
```

```
Iron Man 3
Iron Man 2
Avengers: Age of Ultron
The Avengers
Captain America: Civil War
Guardians of the Galaxy
X-Men
Thor: The Dark World
Ant-Man
X-Men Origins: Wolverine
```

In [69]:

```
recommender("Thor")
```

```
Thor: The Dark World
Clash of the Titans
After Earth
Ant-Man
Iron Man 2
Avengers: Age of Ultron
Rockaway
Little Nicky
Batman v Superman: Dawn of Justice
The Incredible Hulk
```

In [70]:

```
import pickle
```

In [71]:

```
pickle.dump(final_df,open('movie_list.pkl','wb'))
pickle.dump(similarity,open('similarity.pkl','wb'))
```

TMDB API

https://www.themoviedb.org (https://www.themoviedb.org), https://developer.themoviedb.org/docs (https://developer.themoviedb.org/docs)

In [72]:

```
import requests
```

In [73]:

```
url = f"https://api.themoviedb.org/3/movie/{11111}?api_key=75eb0685f1f9140663e33eb0ea
data = requests.get(url)
data = data.json()
```

In [74]:

```
poster_path = data["poster_path"]
poster_path
```

Out[74]:

```
'/5EB9LAzIePTQoMpg2M1GNJpNn9s.jpg'
```

In [75]:

```
full_path = "https://image.tmdb.org/t/p/w500/" + poster_path
print(full_path)
```

https://image.tmdb.org/t/p/w500//5EB9LAzIePTQoMpg2M1GNJpNn9s.jpg (https://image.tmdb.org/t/p/w500//5EB9LAzIePTQoMpg2M1GNJpNn9s.jpg)

In [76]:

```
def fetch_poster(movie_id):
    url = f"https://api.themoviedb.org/3/movie/{movie_id}?api_key=75eb0685f1f9140663e
    data = requests.get(url)
    data = data.json()
    poster_path = data['poster_path']
    full_path = "https://image.tmdb.org/t/p/w500/" + poster_path
    return full_path
```

In [77]:

```
print(fetch_poster(19995))
```

https://image.tmdb.org/t/p/w500//kyeqWdyUXW608qlYkRqosgbbJyK.jpg (https://image.tmdb.org/t/p/w500//kyeqWdyUXW608qlYkRqosgbbJyK.jpg)

In [78]:

```
def recommend(movie_name):
    movie_index = final_df[final_df["title"] == movie_name].index[0]
    distances = similarity[movie_index]
    index_list = list(enumerate(distances))
    similar_movie = sorted(index_list, reverse = True, key = lambda x: x[1])[1:11]

    recommended_movie_names = []
    recommended_movie_posters = []

    for i in similar_movie:
        # fetch the movie poster
        movie_id = final_df.iloc[i[0]].movie_id
        recommended_movie_posters.append(fetch_poster(movie_id))
        recommended_movie_names.append(final_df.iloc[i[0]].title)

    return recommended_movie_names,recommended_movie_posters
```

In [79]:

```
1 recommend("Iron Man 2")
```

Out[79]:

```
(['Krrish',
  'Ant-Man',
  'The Animal',
  'Iron Man 3',
  'The Adventures of Elmo in Grouchland',
  'Flying By',
  'All Is Lost',
  'The Truman Show',
  'Iron Man',
  '1982'],
 ['https://image.tmdb.org/t/p/w500//neJo0Xt9NH6aPBPNhKfHFQpwrcC.jpg',
  'https://image.tmdb.org/t/p/w500//8YxOIPrabqkQCOKKbuxaz9IcqhO.jpg',
  'https://image.tmdb.org/t/p/w500//oNxEXmKTZtECHs0bQbI6dQoXYMV.jpg',
  'https://image.tmdb.org/t/p/w500//qhPtAc1TKbMPqNvcdXSOn9Bn7hZ.jpg',
  'https://image.tmdb.org/t/p/w500//u9i4frT1XPaTqJxRYLJ8j2r8LYO.jpg',
  'https://image.tmdb.org/t/p/w500//xLMv1cpLK3qrvF0ehNEkowWXaFB.jpg',
  'https://image.tmdb.org/t/p/w500//9cVA4oX2xHgiglv6hemxwAaofsq.jpg',
  'https://image.tmdb.org/t/p/w500//vuza0WqY239yBXOadKlGwJsZJFE.jpg',
  'https://image.tmdb.org/t/p/w500//78lPtwv72eTNqFW9COBYI0dWDJa.jpg',
  'https://image.tmdb.org/t/p/w500//5vTgKqNjEVCrZIm4wclIz68O6xs.jpg'])
```