In [2]:

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

<IPython.core.display.Javascript object>

In [3]:

```
movies_df = pd.read_csv("tmdb_5000_movies.csv")
credits_df = pd.read_csv("tmdb_5000_credits.csv")
```

<IPython.core.display.Javascript object>

In [4]:

```
movies_df.head()
```

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...	
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	

<IPython.core.display.Javascript object>

In [5]:

```
credits_df.head()
```

Out[5]:

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

```
<IPython.core.display.Javascript object>
```

In [6]:

```
credits_df.columns = ["id", "title", "cast", "crew"]

movies_df = movies_df.merge(credits_df, on="id")
```

```
<IPython.core.display.Javascript object>
```

In [7]:

```
movies_df.head()
```

Out[7]:

	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	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	

5 rows × 23 columns

```
<IPython.core.display.Javascript object>
```

For Recommedation we used following features:

- Cast
- Crew
- Keywords
- Genre

In [8]:

```
from ast import literal_eval

features = ["cast", "crew", "keywords", "genres"]

for feature in features:
    movies_df[feature] = movies_df[feature].apply(literal_eval)

movies_df[features].head(10)
```

Out[8]:

	cast	crew	keywords	genres
0	[{'cast_id': 242, 'character': 'Jake Sully', '...	[{'credit_id': '52fe48009251416c750aca23', 'de...	[{'id': 1463, 'name': 'culture clash'}, {'id':...	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
1	[{'cast_id': 4, 'character': 'Captain Jack Spa...	[{'credit_id': '52fe4232c3a36847f800b579', 'de...	[{'id': 270, 'name': 'ocean'}, {'id': 726, 'na...	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...
2	[{'cast_id': 1, 'character': 'James Bond', 'cr...	[{'credit_id': '54805967c3a36829b5002c41', 'de...	[{'id': 470, 'name': 'spy'}, {'id': 818, 'name...	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
3	[{'cast_id': 2, 'character': 'Bruce Wayne / Ba...	[{'credit_id': '52fe4781c3a36847f81398c3', 'de...	[{'id': 849, 'name': 'dc comics'}, {'id': 853,...	[{'id': 28, 'name': 'Action'}, {'id': 80, 'nam...
4	[{'cast_id': 5, 'character': 'John Carter', 'c...	[{'credit_id': '52fe479ac3a36847f813eaa3', 'de...	[{'id': 818, 'name': 'based on novel'}, {'id':...	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
5	[{'cast_id': 30, 'character': 'Peter Parker / ...	[{'credit_id': '52fe4252c3a36847f80151a5', 'de...	[{'id': 851, 'name': 'dual identity'}, {'id': ...	[{'id': 14, 'name': 'Fantasy'}, {'id': 28, 'na...
6	[{'cast_id': 34, 'character': 'Flynn Rider (vo...	[{'credit_id': '52fe46db9251416c91062101', 'de...	[{'id': 1562, 'name': 'hostage'}, {'id': 2343,...	[{'id': 16, 'name': 'Animation'}, {'id': 10751...
7	[{'cast_id': 76, 'character': 'Tony Stark / Ir...	[{'credit_id': '55d5f7d4c3a3683e7e0016eb', 'de...	[{'id': 8828, 'name': 'marvel comic'}, {'id': ...	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
8	[{'cast_id': 3, 'character': 'Harry Potter', '...	[{'credit_id': '52fe4273c3a36847f801fab1', 'de...	[{'id': 616, 'name': 'witch'}, {'id': 2343, 'n...	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...
9	[{'cast_id': 18, 'character': 'Bruce Wayne / B...	[{'credit_id': '553bf23692514135c8002886', 'de...	[{'id': 849, 'name': 'dc comics'}, {'id': 7002...	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...

```
<IPython.core.display.Javascript object>
```

In [9]:

```
movies_df[features]["cast"][0][0]
```

Out[9]:

```
{'cast_id': 242,
 'character': 'Jake Sully',
 'credit_id': '5602a8a7c3a3685532001c9a',
 'gender': 2,
 'id': 65731,
 'name': 'Sam Worthington',
 'order': 0}


<IPython.core.display.Javascript object>
```

In [10]:

```
movies_df[features]["crew"][0][0]
```

Out[10]:

```
{'credit_id': '52fe48009251416c750aca23',
 'department': 'Editing',
 'gender': 0,
 'id': 1721,
 'job': 'Editor',
 'name': 'Stephen E. Rivkin'}

<IPython.core.display.Javascript object>
```

In [11]:

```
movies_df[features]["keywords"][0][0]
```

Out[11]:

```
{'id': 1463, 'name': 'culture clash'}

<IPython.core.display.Javascript object>
```

In [12]:

```
movies_df[features]["genres"][0][0]
```

Out[12]:

```
{'id': 28, 'name': 'Action'}

<IPython.core.display.Javascript object>
```

In [13]:

```
def get_director(x):
    for i in x:
        if i["job"] == "Director":
            return i["name"]
    return np.nan
```

```
<IPython.core.display.Javascript object>
```

In [14]:

```
def get_list(x):
    if isinstance(x, list):
        names = [i["name"] for i in x]

        if len(names) > 3:
            names = names[:3]

        return names

    return []
```

```
<IPython.core.display.Javascript object>
```

In [15]:

```
movies_df["director"] = movies_df["crew"].apply(get_director)

features = ["cast", "keywords", "genres"]
for feature in features:
    movies_df[feature] = movies_df[feature].apply(get_list)
```

```
<IPython.core.display.Javascript object>
```

- In the above code, we passed the “crew” information to the get_director() function, extracted the name, and created a new column “director”.
- For the features cast, keyword and genres we extracted the top information by applying the get_list() function

In [17]:

```
movies_df.columns
```

Out[17]:

```
Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'origina
l_language',
       'original_title', 'overview', 'popularity', 'production_co
mpanies',
       'production_countries', 'release_date', 'revenue', 'runtim
e',
       'spoken_languages', 'status', 'tagline', 'title_x', 'vote_
average',
       'vote_count', 'title_y', 'cast', 'crew', 'director'],
      dtype='object')
```

```
<IPython.core.display.Javascript object>
```

In [21]:

```
movies_df[["original_title", "cast", "director", "keywords", "genres"]].head()
```

Out[21]:

	original_title	cast	director	keywords	genres
0	Avatar	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	James Cameron	[culture clash, future, space war]	[Action, Adventure, Fantasy]
1	Pirates of the Caribbean: At World's End	[Johnny Depp, Orlando Bloom, Keira Knightley]	Gore Verbinski	[ocean, drug abuse, exotic island]	[Adventure, Fantasy, Action]
2	Spectre	[Daniel Craig, Christoph Waltz, Léa Seydoux]	Sam Mendes	[spy, based on novel, secret agent]	[Action, Adventure, Crime]
3	The Dark Knight Rises	[Christian Bale, Michael Caine, Gary Oldman]	Christopher Nolan	[dc comics, crime fighter, terrorist]	[Action, Crime, Drama]
4	John Carter	[Taylor Kitsch, Lynn Collins, Samantha Morton]	Andrew Stanton	[based on novel, mars, medallion]	[Action, Adventure, Science Fiction]

```
<IPython.core.display.Javascript object>
```

In [22]:

```
def clean_data(row):
    if isinstance(row, list):
        return [str.lower(i.replace(" ", "")) for i in row]
    else:
        if isinstance(row, str):
            return str.lower(row.replace(" ", ""))
        else:
            return ""


features = ["cast", "keywords", "director", "genres"]
for feature in features:
    movies_df[feature] = movies_df[feature].apply(clean_data)
```

```
<IPython.core.display.Javascript object>
```

In [23]:

```
def create_soup(features):
    return (
        " ".join(features["keywords"])
        + " "
        + " ".join(features["cast"])
        + " "
        + features["director"]
        + " "
        + " ".join(features["genres"])
    )


movies_df["soup"] = movies_df.apply(create_soup, axis=1)
print(movies_df["soup"].head())
```

```
0    cultureclash future spacewar samworthington zo...
1    ocean drugabuse exoticisland johnnydepp orland...
2    spy basedonnovel secretagent danielcraig chris...
3    dccomics crimefighter terrorist christianbale ...
4    basedonnovel mars medallion taylorkitsch lynnc...
Name: soup, dtype: object

<IPython.core.display.Javascript object>
```

In [24]:

```
movies_df["soup"][0]
```

Out[24]:

```
'cultureclash future spacewar samworthington zoesaldana sigourney
weaver jamescameron action adventure fantasy'
```

```
<IPython.core.display.Javascript object>
```

Working:

- **Movie Recommendation** engine works by suggesting movies to the user based on the metadata information.
- The similarity between the movies is calculated and then used to make recommendations.

In [25]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

```
<IPython.core.display.Javascript object>
```

In [26]:

```
count_vectorizer = CountVectorizer(stop_words="english")
count_matrix = count_vectorizer.fit_transform(movies_df["soup"])

print(count_matrix.shape)

cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
print(cosine_sim2.shape)

movies_df = movies_df.reset_index()
indices = pd.Series(movies_df.index, index=movies_df["original_title"])
```

```
(4803, 11520)
(4803, 4803)
```

```
<IPython.core.display.Javascript object>
```

Create a reverse mapping of movie titles to indices. By this, we can easily find the title of the movie based on the index.

In [28]:

```
indices = pd.Series(
    movies_df.index, index=movies_df["original_title"]
).drop_duplicates()

print(indices.head())
```

```
original_title
Avatar                                        0
Pirates of the Caribbean: At World's End      1
Spectre                                       2
The Dark Knight Rises                         3
John Carter                                   4
dtype: int64
```

```
<IPython.core.display.Javascript object>
```

Recommendations

In [42]:

```
def get_recommendations(title, cosine_sim=cosine_sim2):
    idx = indices[title]
    similarity_scores = list(enumerate(cosine_sim[idx]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)
    similarity_scores = similarity_scores[1:11]
    # (a, b) where a is id of movie, b is similarity_scores

    movies_indices = [ind[0] for ind in similarity_scores]
    movies = movies_df["original_title"].iloc[movies_indices]
    return movies
```

```
<IPython.core.display.Javascript object>
```

In [43]:

```
print("Recommendations for The Dark Knight Rises")
print(get_recommendations("The Dark Knight Rises", cosine_sim=cosine_sim2))
print("\n")
print("Recommendations for Avengers")
print(get_recommendations("The Avengers", cosine_sim=cosine_sim2))
```

```
Recommendations for The Dark Knight Rises
65               The Dark Knight
119                Batman Begins
4638    Amidst the Devil's Wings
1196               The Prestige
3073           Romeo Is Bleeding
3326              Black November
1503                      Takers
1986                      Faster
303                     Catwoman
747               Gangster Squad
Name: original_title, dtype: object


Recommendations for Avengers
7                   Avengers: Age of Ultron
26              Captain America: Civil War
79                              Iron Man 2
169     Captain America: The First Avenger
174                    The Incredible Hulk
85     Captain America: The Winter Soldier
31                              Iron Man 3
33                 X-Men: The Last Stand
68                               Iron Man
94                Guardians of the Galaxy
Name: original_title, dtype: object

<IPython.core.display.Javascript object>
```

In [37]:

```
print("Recommendations for The Matrix")
print(get_recommendations("The Matrix", cosine_sim=cosine_sim2))
```

```
Recommendations for The Matrix
123                 The Matrix Revolutions
125                    The Matrix Reloaded
93      Terminator 3: Rise of the Machines
266                                 I, Robot
43                    Terminator Salvation
108                     Terminator Genisys
3439                        The Terminator
487                             Red Planet
4401                   The Helix... Loaded
582                    Battle: Los Angeles
Name: original_title, dtype: object

<IPython.core.display.Javascript object>
```

In [41]:

```
print("Recommendations for Jurassic Park")
print(get_recommendations("Jurassic Park", cosine_sim=cosine_sim2))
```

```
Recommendations for Jurassic Park
508     The Lost World: Jurassic Park
334                  Jurassic Park III
91       Independence Day: Resurgence
28                      Jurassic World
185                  War of the Worlds
507                   Independence Day
2967       E.T. the Extra-Terrestrial
363      A.I. Artificial Intelligence
83                           The Lovers
193                        After Earth
Name: original_title, dtype: object


<IPython.core.display.Javascript object>
```

In [ ]:

```

```