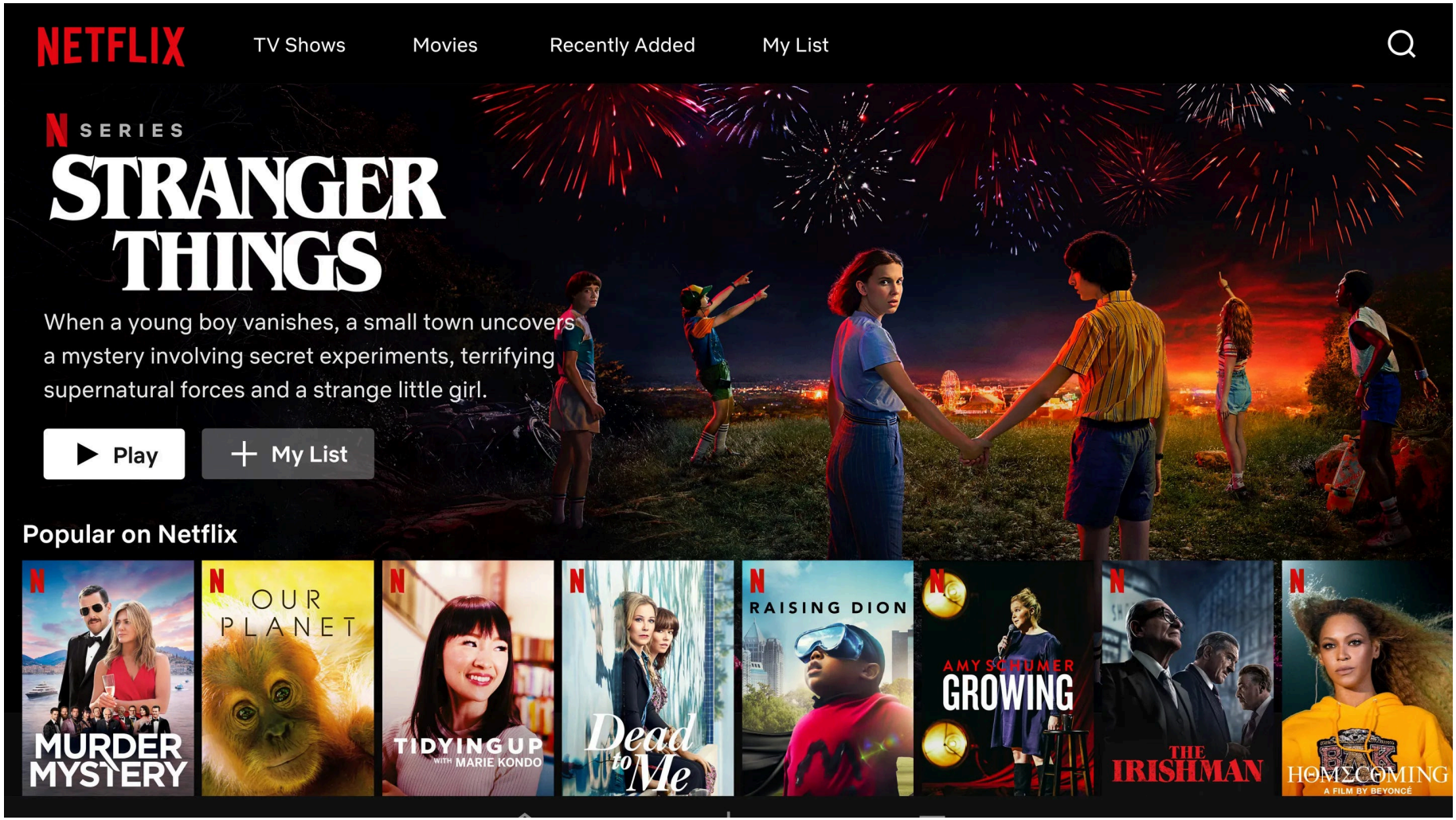
Movie Recommendation System (Content Based Recomm. System)

This is an **unsupervised model**, also known as the **Data Mining concept**. There is no data labelling like yes or no, and there is nothing to predict. This is **not a predictive modelling**.



Problem Statement:

- People often struggle to find movies they'll enjoy among thousands of options available online (like netflix,hotstar etc).
- The goal of this project is to build a movie recommendation system that can automatically suggest movies similar to a user's favorite one — based on content similarity such as genre, cast, crew, keywords, and overview.

Dataset Overview

This project uses two datasets from Kaggle:

- movies
- credits

---

movies

The movies file contains general information about movies such as title, genres, overview, keywords, popularity, release date, budget, and revenue.
Important columns for our model are **title**,**budget overview**, **genres**, and **keywords etc**. These help me understand what each movie is about and what themes or topics it covers.

---

credits

The credits file contains details about the cast and crew.
From this file, we mainly use the **cast** (actors) and **crew** (especially the director).

In [1]:
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
movies=pd.read_csv('/Users/abhisheksenapati/Desktop/Machine Learning & Stats/ML_final_project/Recommandat
movies.head(3)
```

Out[2]:

budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	produc
237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"n Fil
300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"nar Pictu
245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond’s past sends him o...	107.376788	[{"n

In [3]:
```
credits=pd.read_csv('/Users/abhisheksenapati/Desktop/Machine Learning & Stats/ML_final_project/Recommanda
credits.head()
```

Out[3]:

	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...

Pre Processing:

In [4]:
```
# shape of data
print(f"Shape of movies :{movies.shape}\nShape of credits :{credits.shape}")
```

```
Shape of movies :(4803, 20)
Shape of credits :(4803, 4)
```

In [5]:
```
# columns name
# shape of data
print(f"col of movies :{movies.columns}\ncol of credits :{credits.columns}")
```

```
col of movies :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')
col of credits :Index(['movie_id', 'title', 'cast', 'crew'], dtype='object')
```

In [6]:
```
# duplicate data by title, i am worry about on title only others are not significant
print('movies:', movies['title'].duplicated().sum())
print('credit:', credits['title'].duplicated().sum())
```

```
movies: 3
credit: 3
```

In [7]:
```
# missing data checking
print(movies.isnull().sum())
print('****************************************')
print(credits.isnull().sum())
```

```
budget                     0
genres                     0
homepage                3091
id                         0
keywords                   0
original_language          0
original_title             0
overview                   3
popularity                 0
production_companies       0
production_countries       0
release_date               1
revenue                    0
runtime                    2
spoken_languages           0
status                     0
tagline                  844
title                      0
vote_average               0
vote_count                 0
dtype: int64
****************************************
movie_id    0
title       0
cast        0
crew        0
dtype: int64
```

In [8]:
```
# will merge both the table--> movies+credit:
# both tables have common columns i.e, titles, will merge by titles:

movies=movies.merge(credits,on='title')
movies.shape
```

Out[8]:
```
(4809, 23)
```

- previously it was 4803, now 4809. That means some of the **movie's was not available in movies but available in credits**. That's why after merge it became 4809.
- previously movie data had 20 columns and credit data had 4 columns. The total should have been 24, **but after merging, it became 23 because both columns were common**, which was the title. So, the title is now considered as one column.

In [9]:
```
movies.columns
```

Out[9]:
```
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', 'movie_id', 'cast', 'crew'],
      dtype='object')
```

In [10]:
```
print('movies:', movies['title'].duplicated().sum())
```

```
movies: 9
```

In [11]:
```
# missing data checking
print(movies.isnull().sum())
```

```
budget                     0
genres                     0
homepage                3096
id                         0
keywords                   0
original_language          0
original_title             0
overview                   3
popularity                 0
production_companies       0
production_countries       0
release_date               1
revenue                    0
runtime                    2
spoken_languages           0
status                     0
tagline                  844
title                      0
vote_average               0
vote_count                 0
movie_id                   0
cast                       0
crew                       0
dtype: int64
```

For building a recommendation system that predicts movies based on similarities, some columns are not significant:

- Home page > isn't significant for the recommendation system- Drop it
- Id > also not significant - drop it
- budget > also not significant - drop it
- budget > also not significant - drop it
- original_title > also not significant - drop it
- popularity > also not significant - drop it
- production_companies > also not significant - drop it

- release_date > also not significant - drop it
- production_countries > also not significant - drop it
- revenue > also not significant - drop it
- runtime > also not significant - drop it
- ***original_language*** or ***spoken_languages*** > any one you have to take other need to - drop it (not both)
- tagline > also not significant - drop it
- ***movie_id*** or ***id*** > any one you have to take other need to - drop it (not both)
- vote_count > also not significant - drop it
- vote_average > also not significant - drop it
- & Rest I need to keep it for **recommendation system**.

In [12]:
```
1 movies.columns
```

Out[12]:
```
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', 'movie_id', 'cast', 'crew'],
      dtype='object')
```

In [13]:
```
1 print(movies['original_language'].value_counts(normalize=True))
```

```
original_language
en    0.937825
fr    0.014556
es    0.006654
zh    0.005614
de    0.005614
hi    0.003951
ja    0.003327
it    0.002911
ko    0.002495
cn    0.002495
ru    0.002287
pt    0.001871
da    0.001456
sv    0.001040
nl    0.000832
fa    0.000832
th    0.000624
he    0.000624
ta    0.000416
cs    0.000416
ro    0.000416
id    0.000416
ar    0.000416
vi    0.000208
sl    0.000208
ps    0.000208
no    0.000208
ky    0.000208
hu    0.000208
pl    0.000208
af    0.000208
nb    0.000208
tr    0.000208
is    0.000208
xx    0.000208
te    0.000208
el    0.000208
Name: proportion, dtype: float64
```

- In the ***original_language***, english itself is given ***93.78%***, while the remaining 6.22% is divided among different languages. since ***english is the default language, it influences my other data as well***. there's no need to use it in the recommender system. So, we can ***drop*** this too.

- mov_id is not required for now, but it becomes ***significant when I deploy*** this model at that time, it will be important because id is just a unique reference to each movie. ***for now, I can drop it*** but in deployment, it should be there.

In [14]:
```
movies=movies[['genres','keywords','overview','title', 'cast', 'crew']]
movies.columns
```

Out[14]:
```
Index(['genres', 'keywords', 'overview', 'title', 'cast', 'crew'], dtype='object')
```

In [15]:
```
movies.head(2)
```

Out[15]:

	genres	keywords	overview	title	cast	crew
0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 1463, "name": "culture clash"}, {"id":...	In the 22nd century, a paraplegic Marine is di...	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	Captain Barbossa, long believed to be dead, ha...	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...

In [16]:
```
movies['genres'][232] # randomly checking the data
```

Out[16]:
```
'[{"id": 28, "name": "Action"}, {"id": 878, "name": "Science Fiction"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}]'
```

Problems 1: - in the above data (different format) -or cleaning json data :

- The data columns such as genres, keywords, overview, cast, and crew are stored as strings instead of proper python lists or dictionaries.
- Before building a recommendation model, these columns must be converted into structured python objects ***(using ast.literal_eval-> its safely converting python object like dict,tuple, list,set)*** so that we can properly extract and process relevant information like genre names, actor names, or crew members.

Problems 2: Referencing id not required : need to drop

- When I checked movies['genres'] by random index, I noticed that a specific movie is associated with certain genres. However, these genres are referenced using numeric references, such as if a movie is 'action' then its id --> 28, and if it's 'Science Fiction', its ID --> 878. These id are not necessary because they are simply unique numbers for genres. I only need the types of genres, not the id as unique references. so, I will only select the genres, not those with id.
- Simillar problem in movies['keywords']

Handling these problem using (ast.literal_eval) :

In [17]:
```
import ast
```

For genres, keywords columns:

In [18]:
```
# define the 1st function using ast.literal_eval to extract
# only the name part from each dict for the genres columns:

def convert(text):
    extract=[]
    for i in ast.literal_eval(text):
        extract.append(i['name'])
    return extract
```

In [19]:
```
# apply the convert function in the genres columns :
movies['genres']=movies['genres'].apply(convert)
```

In [20]:
```
# after applying, check the results from the bottom.
movies['genres'].tail()
```

Out[20]:
```
4804             [Action, Crime, Thriller]
4805                     [Comedy, Romance]
4806    [Comedy, Drama, Romance, TV Movie]
4807                                    []
4808                           [Documentary]
Name: genres, dtype: object
```

In [21]:
```
#same for keywords col: 1st check the data:
movies['keywords'][0]
```

Out[21]:
```
'[{"id": 1463, "name": "culture clash"}, {"id": 2964, "name": "future"}, {"id": 3386, "name": "space war"},
{"id": 3388, "name": "space colony"}, {"id": 3679, "name": "society"}, {"id": 3801, "name": "space trave
l"}, {"id": 9685, "name": "futuristic"}, {"id": 9840, "name": "romance"}, {"id": 9882, "name": "space"},
{"id": 9951, "name": "alien"}, {"id": 10148, "name": "tribe"}, {"id": 10158, "name": "alien planet"}, {"i
d": 10987, "name": "cgi"}, {"id": 11399, "name": "marine"}, {"id": 13065, "name": "soldier"}, {"id": 14643,
"name": "battle"}, {"id": 14720, "name": "love affair"}, {"id": 165431, "name": "anti war"}, {"id": 193554,
"name": "power relations"}, {"id": 206690, "name": "mind and soul"}, {"id": 209714, "name": "3d"}]'
```

In [22]:
```
# apply the convert function in the keywords columns :
movies['keywords']=movies['keywords'].apply(convert)
```

In [23]:
```
# after applying, check the results from the bottom.
movies['keywords'].head()
```

Out[23]:
```
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
```

For cast columns:

In [24]:
```
movies['cast'][0]
```

Out[24]:

```
'[{"cast_id": 242, "character": "Jake Sully", "credit_id": "5602a8a7c3a3685532001c9a", "gender": 2, "i
d": 65731, "name": "Sam Worthington", "order": 0}, {"cast_id": 3, "character": "Neytiri", "credit_id":
"52fe48009251416c750ac9cb", "gender": 1, "id": 8691, "name": "Zoe Saldana", "order": 1}, {"cast_id": 25,
"character": "Dr. Grace Augustine", "credit_id": "52fe48009251416c750aca39", "gender": 1, "id": 10205,
"name": "Sigourney Weaver", "order": 2}, {"cast_id": 4, "character": "Col. Quaritch", "credit_id": "52fe
48009251416c750ac9cf", "gender": 2, "id": 32747, "name": "Stephen Lang", "order": 3}, {"cast_id": 5, "ch
aracter": "Trudy Chacon", "credit_id": "52fe48009251416c750ac9d3", "gender": 1, "id": 17647, "name": "Mi
chelle Rodriguez", "order": 4}, {"cast_id": 8, "character": "Selfridge", "credit_id": "52fe48009251416c7
50ac9e1", "gender": 2, "id": 1771, "name": "Giovanni Ribisi", "order": 5}, {"cast_id": 7, "character":
"Norm Spellman", "credit_id": "52fe48009251416c750ac9dd", "gender": 2, "id": 59231, "name": "Joel David
Moore", "order": 6}, {"cast_id": 9, "character": "Moat", "credit_id": "52fe48009251416c750ac9e5", "gende
r": 1, "id": 30485, "name": "CCH Pounder", "order": 7}, {"cast_id": 11, "character": "Eytukan", "credit_
id": "52fe48009251416c750ac9ed", "gender": 2, "id": 15853, "name": "Wes Studi", "order": 8}, {"cast_id":
10, "character": "Tsu\'Tey", "credit_id": "52fe48009251416c750ac9e9", "gender": 2, "id": 10964, "name":
"Laz Alonso", "order": 9}, {"cast_id": 12, "character": "Dr. Max Patel", "credit_id": "52fe48009251416c7
50ac9f1", "gender": 2, "id": 95697, "name": "Dileep Rao", "order": 10}, {"cast_id": 13, "character": "Ly
le Wainfleet", "credit_id": "52fe48009251416c750ac9f5", "gender": 2, "id": 98215, "name": "Matt Gerald",
"order": 11}, {"cast_id": 32, "character": "Private Fike", "credit_id": "52fe48009251416c750aca5b", "gen
der": 2, "id": 154153, "name": "Sean Anthony Moran", "order": 12}, {"cast_id": 33, "character": "Cryo Va
ult Med Tech", "credit_id": "52fe48009251416c750aca5f", "gender": 2, "id": 397312, "name": "Jason Wyat
```

- these are the frontend actors, all the actors in the movie were mentioned with their character names and original names, but not all the actor names are required.
- it's only required to extract the top 5 or top 10 actors, so I can do that part.
- From the top, I need the lead actor, followed by the second lead actor, and so on.
- **Extraction part:** I need the top 5 or 10 actors with their **original names Only**.

In [25]:
```
# define the 2nd function using ast.literal_eval to extract top 5 lead actors original names Only:

def get_top_actor(text):
    extract_2=[]
    counter=0
    for i in ast.literal_eval(text):
        if counter<5:
            extract_2.append(i['name'])
        counter=counter+1
    return extract_2
```

In [26]:
```
# apply the 2nd convert function in the cast columns :
movies['cast']=movies['cast'].apply(get_top_actor)
```

In [27]:
```
# after applying, check the results:
movies['cast'][0]
```

Out[27]:
```
['Sam Worthington',
 'Zoe Saldana',
 'Sigourney Weaver',
 'Stephen Lang',
 'Michelle Rodriguez']
```

For crew columns :

In [28]:
```
1 movies['crew'][0]
```


```
e10046fe", "department": "Editing", "gender": 0, "id": 1401815, "job": "Digital Intermediate", "name":
"Marvin Hall"}, {"credit_id": "5495a1f7c3a3686ae3004443", "department": "Production", "gender": 0, "id":
1401816, "job": "Publicist", "name": "Judy Alley"}, {"credit_id": "5592b29fc3a36869d100002f", "departmen
t": "Crew", "gender": 0, "id": 1418381, "job": "CG Supervisor", "name": "Mike Perry"}, {"credit_id": "55
92b23a9251415df8001081", "department": "Crew", "gender": 0, "id": 1426854, "job": "CG Supervisor", "nam
e": "Andrew Morley"}, {"credit_id": "55491e1192514104c40002d8", "department": "Art", "gender": 0, "id":
1438901, "job": "Conceptual Design", "name": "Seth Engstrom"}, {"credit_id": "5525d5809251417276002b06",
"department": "Crew", "gender": 0, "id": 1447362, "job": "Visual Effects Art Director", "name": "Eric Ol
iver"}, {"credit_id": "554427ca925141586500312a", "department": "Visual Effects", "gender": 0, "id": 144
7503, "job": "Modeling", "name": "Matsune Suzuki"}, {"credit_id": "551906889251415aab001c88", "departmen
t": "Art", "gender": 0, "id": 1447524, "job": "Art Department Manager", "name": "Paul Tobin"}, {"credit_
id": "5592af8492514152cc0010de", "department": "Costume & Make-Up", "gender": 0, "id": 1452643, "job":
"Hairstylist", "name": "Roxane Griffin"}, {"credit_id": "553d3c109251415852001318", "department": "Light
ing", "gender": 0, "id": 1453938, "job": "Lighting Artist", "name": "Arun Ram-Mohan"}, {"credit_id": "55
92af4692514152d5001355", "department": "Costume & Make-Up", "gender": 0, "id": 1457305, "job": "Makeup A
rtist", "name": "Georgia Lockhart-Adams"}, {"credit_id": "5592b2eac3a36877470012a5", "department": "Cre
w", "gender": 0, "id": 1466035, "job": "CG Supervisor", "name": "Thrain Shadbolt"}, {"credit_id": "5592b
032c3a36877450015f1", "department": "Crew", "gender": 0, "id": 1483220, "job": "CG Supervisor", "name":
"Brad Alexander"}, {"credit_id": "5592b05592514152d80012f6", "department": "Crew", "gender": 0, "id": 14
```


- This part is all about the backend char like movie dir, producer, editor,music director,sound designer, all are about the backend, but for a customer, all those people are not important except the movie director, who is very significant.
- **Extracttion**: So, I will extract the only and only **director's name Only**.

In [29]:
```
# define the 3rd function using ast.literal_eval to extract only director's name Only:

def get_director_only(text):
    director=[]
    for i in ast.literal_eval(text):
        if i['job']=='Director':
            director.append(i['name'])
    return director
```

In [30]:
```
# apply the 3rd function in the crew columns to get director name:
movies['crew']=movies['crew'].apply(get_director_only)
```

In [31]:
```
# after applying function, check the results:
movies['crew'][0]
```

Out[31]: ['James Cameron']

In [32]:
```
# final results

movies.head()
```

Out[32]:

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

In [33]:
```
movies.isnull().sum()
```

Out[33]:
```
genres      0
keywords    0
overview    3
title       0
cast        0
crew        0
dtype: int64
```

In [34]:
```
# this is text data so we can drop it

movies.dropna(inplace=True)
```

In [35]:
```
# check now

movies.isnull().sum()
```

Out[35]:
```
genres      0
keywords    0
overview    0
title       0
cast        0
crew        0
dtype: int64
```

EDA:

(Automated detailed EDA --> ydata_profiling)

In [36]:

```python
from ydata_profiling import ProfileReport

profile = ProfileReport(movies, title="Movie Dataset EDA Report", explorative=True)

profile.to_notebook_iframe()
profile.to_file("Movie_eda_report.html")
```

[Upgrade to ydata-sdk (https://ydata.ai/register)](https://ydata.ai/register)

Improve your data and profiling with ydata-sdk, featuring data quality scoring, redundancy detection, outlier identification, text validation, and synthetic data generation.

Summarize dataset: 100% 15/15 [00:02<00:00, 7.47it/s, Completed]

```
  0%|                                                 | 0/6 [00:00<?, ?it/s]
100%|██████████████████████████████████████████████████| 6/6 [00:00<00:00, 12.73it/s]
```

Generate report structure: 100% 1/1 [00:01<00:00, 1.48s/it]

Render HTML: 100% 1/1 [00:00<00:00, 5.30it/s]

Movie Dataset EDA Report

Overview

Brought to you by YData

Overview | Alerts 5 | Reproduction

Dataset statistics

Number of variables	6
Number of observations	4806
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	6
Duplicate rows (%)	0.1%
Total size in memory	4.5 MiB
Average record size in memory	975.4 B

Variable types

Unsupported	4
Text	2

Variables

Select Columns

genres

Unsupported

Rejected Unsupported

Missing	0
Missing (%)	0.0%
Memory size	494.3 KiB

Export report to file: 100% 1/1 [00:00<00:00, 148.66it/s]

Feature Engineering:

In [37]:
```
1 movies.head(10)
```

Out[37]:

	genres	keywords	overview	title	cast	crew
0	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	In the 22nd century, a paraplegic Marine is di...	Avatar	[Sam Worthington, Zoe Saldana, Sigourney Weave...	[James Cameron]
1	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	Captain Barbossa, long believed to be dead, ha...	Pirates of the Caribbean: At World's End	[Johnny Depp, Orlando Bloom, Keira Knightley, ...	[Gore Verbinski]
2	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	A cryptic message from Bond’s past sends him o...	Spectre	[Daniel Craig, Christoph Waltz, Léa Seydoux, R...	[Sam Mendes]
3	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	Following the death of District Attorney Harve...	The Dark Knight Rises	[Christian Bale, Michael Caine, Gary Oldman, A...	[Christopher Nolan]
4	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	John Carter is a war-weary, former military ca...	John Carter	[Taylor Kitsch, Lynn Collins, Samantha Morton,...	[Andrew Stanton]
5	[Fantasy, Action, Adventure]	[dual identity, amnesia, sandstorm, love of on...	The seemingly invincible Spider-Man goes up ag...	Spider-Man 3	[Tobey Maguire, Kirsten Dunst, James Franco, T...	[Sam Raimi]
6	[Animation, Family]	[hostage, magic, horse, fairy tale, musical, p...	When the kingdom's most wanted-and most charmi...	Tangled	[Zachary Levi, Mandy Moore, Donna Murphy, Ron ...	[Byron Howard, Nathan Greno]
7	[Action, Adventure, Science Fiction]	[marvel comic, sequel, superhero, based on com...	When Tony Stark tries to jumpstart a dormant p...	Avengers: Age of Ultron	[Robert Downey Jr., Chris Hemsworth, Mark Ruff...	[Joss Whedon]
8	[Adventure, Fantasy, Family]	[witch, magic, broom, school of witchcraft, wi...	As Harry begins his sixth year at Hogwarts, he...	Harry Potter and the Half-Blood Prince	[Daniel Radcliffe, Rupert Grint, Emma Watson, ...	[David Yates]
9	[Action, Adventure, Fantasy]	[dc comics, vigilante, superhero, based on com...	Fearing the actions of a god-like Super Hero l...	Batman v Superman: Dawn of Justice	[Ben Affleck, Henry Cavill, Gal Gadot, Amy Ada...	[Zack Snyder]

In [38]:
```
# exract some parts to know the problem:
movies.iloc[:3:2][['cast', 'crew']]
```

Out[38]:

	cast	crew
0	[Sam Worthington, Zoe Saldana, Sigourney Weave...	[James Cameron]
2	[Daniel Craig, Christoph Waltz, Léa Seydoux, R...	[Sam Mendes]

Now again some issue in the data (complexity of data)

- In the cast of row - **Sam Worthington** and in crew 1 row - **Sam Mendes**. The **first part is the same** when the system uses tokenization and splitting words by word. Then it will split cast - Sam and crew Sam. When I asked **director Sam, the answer may comes as actor Sam**. Similarly, if I asked actor Sam, it may come as **director**.

- So, it reduces similarity by removing spaces. Then it became **Sam_Worthington** which is actor and **Sam_Mendes** which is director. So, there is no confusion by machine. Python is case sensitive, so the machine will consider both as different.

- simillarly we need to handle this types of problem by removing space as python will tokenize the word by space.

In [39]:
```
def remove_space(space):
    space_extraction=[]
    for i in space:
        space_extraction.append(i.replace(" ",""))
    return space_extraction
```

In [40]:
```
1 movies.head(2)
```

Out[40]:

	genres	keywords	overview	title	cast	crew
0	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	In the 22nd century, a paraplegic Marine is di...	Avatar	[Sam Worthington, Zoe Saldana, Sigourney Weave...	[James Cameron]
1	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	Captain Barbossa, long believed to be dead, ha...	Pirates of the Caribbean: At World's End	[Johnny Depp, Orlando Bloom, Keira Knightley, ...	[Gore Verbinski]

In [41]:
```
1 movies['genres']=movies['genres'].apply(remove_space)
2 movies['keywords']=movies['keywords'].apply(remove_space)
3 movies['cast']=movies['cast'].apply(remove_space)
4 movies['crew']=movies['crew'].apply(remove_space)
```

In [42]:
```
1 movies.head(2)
```

Out[42]:

	genres	keywords	overview	title	cast	crew
0	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	In the 22nd century, a paraplegic Marine is di...	Avatar	[SamWorthington, ZoeSaldana, SigourneyWeaver, ...	[JamesCameron]
1	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticisland, eastindiatrad...	Captain Barbossa, long believed to be dead, ha...	Pirates of the Caribbean: At World's End	[JohnnyDepp, OrlandoBloom, KeiraKnightley, Ste...	[GoreVerbinski]

- We are not applying this function to overview , title columns.

- Because genres,keywords,cast,crew are stored as lists (inside square brackets []).

- Only those columns contain multiple items that need space removal or token cleaning.

- overview is a full paragraph of text or open text, so we should not modify or split it here.

- Therefore, we apply the function only to genres,keywords,cast,crew columns.

Now handle Overview

In [43]:
```
movies['overview'][0]
```

Out[43]:
```
'In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but become
s torn between following orders and protecting an alien civilization.'
```

In [44]:
```
# split word by word

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

In [45]:
```
1 movies['overview'][0]
```

Out[45]:
```
['In',
 'the',
 '22nd',
 'century,',
 'a',
 'paraplegic',
 'Marine',
 'is',
 'dispatched',
 'to',
 'the',
 'moon',
 'Pandora',
 'on',
 'a',
 'unique',
 'mission,',
 'but',
 'becomes',
 'torn',
 'between',
 'following',
 'orders',
 'and',
 'protecting',
 'an',
 'alien',
 'civilization.']
```

In [46]:
```
1 movies.head()
```

Out[46]:

	genres	keywords	overview	title	cast	crew
0	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[In, the, 22nd, century,, a, paraplegic, Marin...	Avatar	[SamWorthington, ZoeSaldana, SigourneyWeaver, ...	[JamesCameron]
1	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticisland, eastindiatrad...	[Captain, Barbossa,, long, believed, to, be, d...	Pirates of the Caribbean: At World's End	[JohnnyDepp, OrlandoBloom, KeiraKnightley, Ste...	[GoreVerbinski]
2	[Action, Adventure, Crime]	[spy, basedonnovel, secretagent, sequel, mi6, ...	[A, cryptic, message, from, Bond’s, past, send...	Spectre	[DanielCraig, ChristophWaltz, LéaSeydoux, Ralp...	[SamMendes]
3	[Action, Crime, Drama, Thriller]	[dccomics, crimefighter, terrorist, secretiden...	[Following, the, death, of, District, Attorney...	The Dark Knight Rises	[ChristianBale, MichaelCaine, GaryOldman, Anne...	[ChristopherNolan]
4	[Action, Adventure, ScienceFiction]	[basedonnovel, mars, medallion, spacetravel, p...	[John, Carter, is, a, war-weary,, former, mili...	John Carter	[TaylorKitsch, LynnCollins, SamanthaMorton, Wi...	[AndrewStanton]

Reason of split:

- overview contains movie descriptions (paragraphs).
- By splitting it, I can turn each overview into a list of words.
- Later, these tokens (by spliting) help the model find similarity between movie plots.
- it compare each word with others

Requirement for model building: (a new col tags)

- MY purpose is when I give the system the **movie title**, it should **recommend **the n-number movies**.
- So, for that, I can **combine all columns except the title**.
- Later, I can drop individual columns, so based on that, my system can recommend.
- So that it can find similarities.

In [47]:
```
movies.columns
```

Out[47]:
```
Index(['genres', 'keywords', 'overview', 'title', 'cast', 'crew'], dtype='object')
```

In [48]:
```
# concat all col data into tag-->

movies['tags']=movies['genres']+movies['keywords']+movies['overview']+movies['cast']+movies['crew']
```

In [49]:
```
movies.columns
```

Out[49]:
```
Index(['genres', 'keywords', 'overview', 'title', 'cast', 'crew', 'tags'], dtype='object')
```

In [50]:
```
movies.head(2)
```

Out[50]:

	genres	keywords	overview	title	cast	crew	tags
0	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[In, the, 22nd, century,, a, paraplegic, Marin...	Avatar	[SamWorthington, ZoeSaldana, SigourneyWeaver, ...	[JamesCameron]	[Action, Adventure, Fantasy, ScienceFiction, c...
1	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticisland, eastindiatrad...	[Captain, Barbossa,, long, believed, to, be, d...	Pirates of the Caribbean: At World's End	[JohnnyDepp, OrlandoBloom, KeiraKnightley, Ste...	[GoreVerbinski]	[Adventure, Fantasy, Action, ocean, drugabuse,...

In [51]:
```
# drop other col

new=movies.drop(columns=['genres', 'keywords', 'overview','cast', 'crew'],axis=1)
new.head()
```

Out[51]:

	title	tags
0	Avatar	[Action, Adventure, Fantasy, ScienceFiction, c...
1	Pirates of the Caribbean: At World's End	[Adventure, Fantasy, Action, ocean, drugabuse,...
2	Spectre	[Action, Adventure, Crime, spy, basedonnovel, ...
3	The Dark Knight Rises	[Action, Crime, Drama, Thriller, dccomics, cri...
4	John Carter	[Action, Adventure, ScienceFiction, basedonnov...

In [52]:
```
# tags coming as a list so need to make open paragraph:
# where ever comma there join everything, dont want comma separetor format

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

In [53]:
```
new.head()
```

Out[53]:

	title	tags
0	Avatar	Action Adventure Fantasy ScienceFiction cultur...
1	Pirates of the Caribbean: At World's End	Adventure Fantasy Action ocean drugabuse exoti...
2	Spectre	Action Adventure Crime spy basedonnovel secret...
3	The Dark Knight Rises	Action Crime Drama Thriller dccomics crimefigh...
4	John Carter	Action Adventure ScienceFiction basedonnovel m...

In [54]:
```
new['tags'][0]
```

Out[54]:
```
'Action Adventure Fantasy ScienceFiction cultureclash future spacewar spacecolony society spacetravel futur
istic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrelations min
dandsoul 3d In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission,
but becomes torn between following orders and protecting an alien civilization. SamWorthington ZoeSaldana S
igourneyWeaver StephenLang MichelleRodriguez JamesCameron'
```

Feature Engineering 2:

- transforming unstructured data **(text) into structured numerical format**.
- using **Vectorization Method**-> is the process of **converting text into numerical vectors**.
- and i will use **Bag of words** concept.

In [55]:
```
from sklearn.feature_extraction.text import CountVectorizer
count_Vect=CountVectorizer(max_features=5000,
                           stop_words='english',
                           binary=True)
vector=count_Vect.fit_transform(new['tags']).toarray()
```

In [56]:
```
vector.shape
```

Out[56]:
```
(4806, 5000)
```

In [57]:
```
# vector 5000 columns

pd.DataFrame(vector).head()
```

Out[57]:

	0	1	2	3	4	5	6	7	8	9	...	4990	4991	4992	4993	4994	4995	4996	4997	4998	4999
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 5000 columns

In [58]:
```
# all the unique words (features) extracted from the text

count_Vect.get_feature_names_out()
```

Out[58]:
```
array(['000', '007', '10', ..., 'zone', 'zoo', 'zooeydeschanel'],
      dtype=object)
```

Building Recommendation System: (Cosine Similarity)

Content Based Recomm. System:

In [60]:
```
from sklearn.metrics.pairwise import cosine_similarity
similarity=cosine_similarity(vector)
pd.DataFrame(similarity).head()
```

Out[60]:

	0	1	2	3	4	5	6	7	8	9	...	4796	4797	4798	4799	
0	1.000000	0.088273	0.064651	0.049237	0.171709	0.107443	0.025392	0.135932	0.067003	0.083624	...	0.000000	0.0000	0.027186	0.063564	0.0
1	0.088273	1.000000	0.062776	0.023905	0.083366	0.130410	0.024656	0.079195	0.065060	0.081200	...	0.000000	0.0000	0.026398	0.000000	0.0
2	0.064651	0.062776	1.000000	0.052523	0.091584	0.085960	0.027086	0.116003	0.071474	0.059470	...	0.049629	0.0000	0.000000	0.000000	0.0
3	0.049237	0.023905	0.052523	1.000000	0.046499	0.065465	0.061885	0.066259	0.027217	0.181164	...	0.037796	0.0343	0.022086	0.051640	0.0
4	0.171709	0.083366	0.091584	0.046499	1.000000	0.101469	0.071940	0.154049	0.031639	0.105300	...	0.043937	0.0000	0.000000	0.030015	0.0

5 rows × 4806 columns

In [63]:
```
1 new
```

Out[63]:

	title	tags
0	Avatar	Action Adventure Fantasy ScienceFiction cultur...
1	Pirates of the Caribbean: At World's End	Adventure Fantasy Action ocean drugabuse exoti...
2	Spectre	Action Adventure Crime spy basedonnovel secret...
3	The Dark Knight Rises	Action Crime Drama Thriller dccomics crimefigh...
4	John Carter	Action Adventure ScienceFiction basedonnovel m...
...	...	...
4804	El Mariachi	Action Crime Thriller unitedstates–mexicobarri...
4805	Newlyweds	Comedy Romance A newlywed couple's honeymoon i...
4806	Signed, Sealed, Delivered	Comedy Drama Romance TVMovie date loveatfirsts...
4807	Shanghai Calling	When ambitious New York attorney Sam is sent t...
4808	My Date with Drew	Documentary obsession camcorder crush dreamgir...

4806 rows × 2 columns

In [66]:
```
1 # this is giving index position by movie name
2 new[new['title']=='El Mariachi'].index[0]
```

Out[66]: 4804

In [93]:
```
def recommendationsystem(movies):
    index= new[new['title']==movies].index[0]  #find index/row_num the movie is in -->new df
    distance=sorted(list(enumerate(similarity[index])),reverse=True,key=lambda x: x[1])

    for i in distance[1:6]: # i wants top 5 simillar movies after sorting
        print(new.iloc[i[0]].title)

# distance=sorted(list(enumerate(similarity[index])),reverse=True,key=lambda x: x[1])-> **means**
# 1. enumerate(similarity[index]) → creates a list of tuples like (movie_index, similarity_score)
# 2. key=lambda x: x[1] → tells Python to sort based on the similarity_score (second value of each tuple)
# 3. reverse=True → sorts in descending order so most similar movies come first
# Final Output: 'distance' stores all movies ranked by their similarity to the selected movie
```

Final Chapter : By Movie Name 5 Recommendation

In [94]:
```
recommendationsystem("Veer-Zaara")
```

```
Sunshine State
All That Jazz
Good Intentions
Blood and Wine
Crossroads
```

In [ ]:
```

```