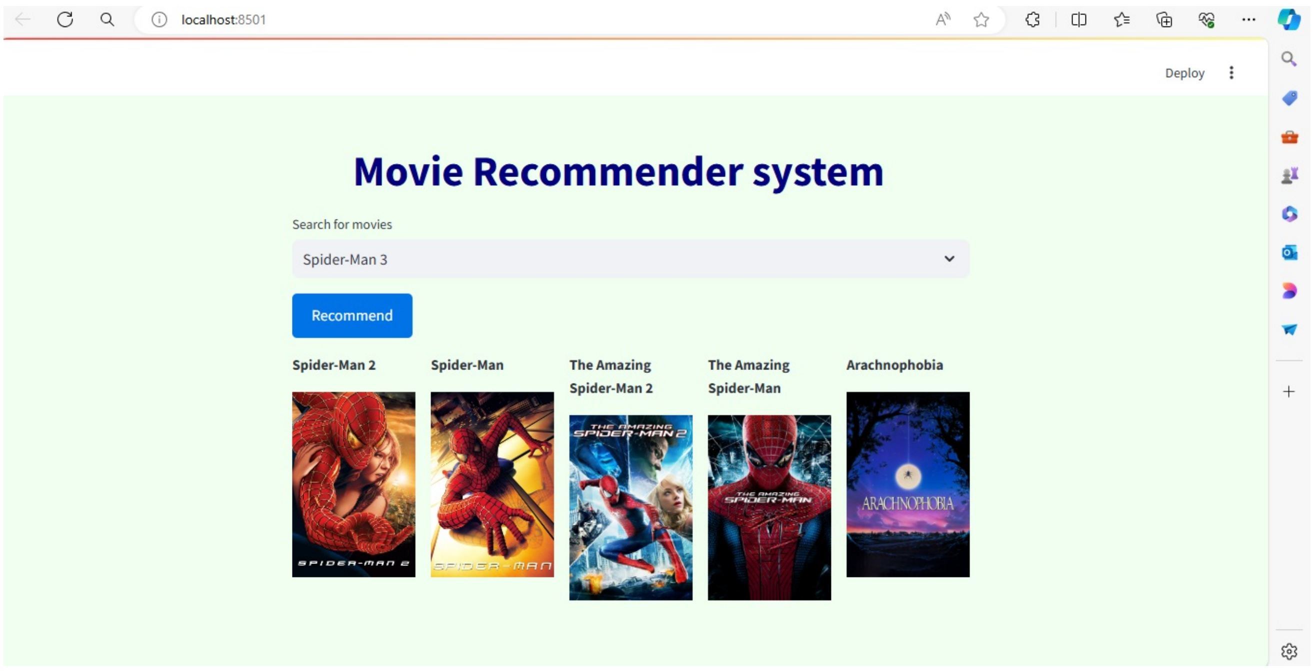
A Machine Learning Project

Content-Based Movie Recommender System



Movie Recommender system

Search for movies

Spider-Man 3

Recommend

Spider-Man 2 | Spider-Man | The Amazing Spider-Man 2 | The Amazing Spider-Man | Arachnophobia

Objective Statement:

- The objective of the Content-Based Movie Recommender System project is to develop an intelligent recommendation system that leverages movie attributes such as genre, cast, and plot to provide personalized movie suggestions to users. By analyzing these intrinsic characteristics of movies, the system aims to enhance user experience by offering relevant and engaging recommendations, thereby facilitating the discovery of new films aligned with individual preferences and tastes.

Dataset Overview:

- The dataset comprises information on 5000 movies sourced from Kaggle. Each entry contains various attributes and metadata associated with the movies, providing a comprehensive overview of the film industry. Key features may include movie titles, release dates, genres, directors, cast members, production companies, budget, revenue, ratings, and plot summaries. This dataset facilitates exploratory data analysis, machine learning model training, and the development of recommendation systems and predictive analytics applications within the domain of movie analysis and entertainment.

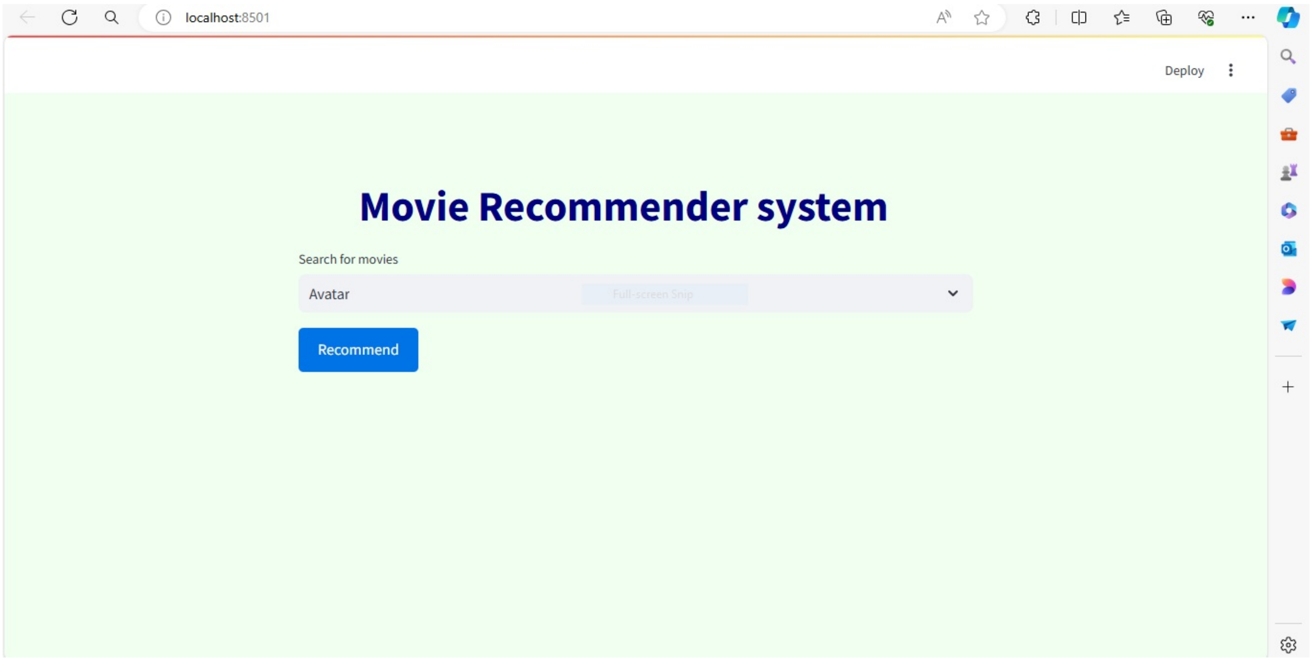
- Unlike collaborative filtering methods, which rely on user behavior and preferences, content-based systems focus on the intrinsic characteristics of movies, such as genres, actors, directors, and plot keywords.

Importance of Recommender System:

- A recommendation system, sometimes known as a recommendation engine, is a paradigm for information filtering that aims to anticipate user preferences and offer suggestions in accordance with these preferences. These technologies are now widely used in a variety of industries, including those that deal with utilities, books, music, movies, television, apparel, and restaurants. These systems gather data on a user's preferences and behavior, which they then employ to enhance their future suggestions.

- A movie recommendation engine is a system that suggests movies to users based on their preferences, ratings, or viewing history. There are different types of movie recommendation engines, such as content-based, collaborative filtering, or hybrid methods. Each of these methods uses different algorithms and data sources to provide personalized and accurate recommendations.

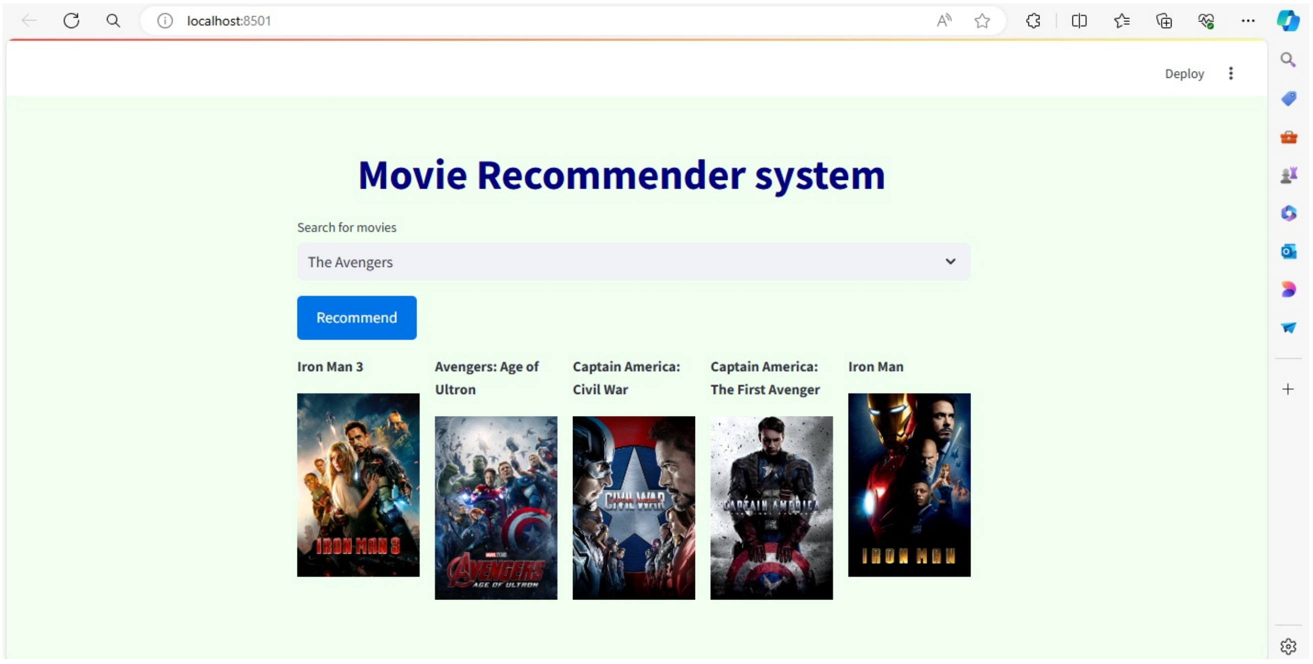
- Watching movies is fun, but figuring out what movie to watch next is a nerve-racking experience. Endlessly scrolling through Netflix, watching trailers on Youtube looking up IMDB Rating, wasting half an hour and still cannot decide what to watch?



For example:

- if a user enters "The Dark Knight" as their favorite movie, the recommender system may suggest movies like "The Dark Knight Rises," "Batman Begins," "Inception," "Interstellar," and "Fight Club" based on similarities in genre (action, thriller), cast (Christian Bale, Heath Ledger), director (Christopher Nolan), and themes (complex characters, moral ambiguity).

Key Attributes:



- These attributes provide valuable information about each movie and can be used for analysis, visualization, and modeling in various applications, such as recommendation systems, predictive analytics, and market research.

1. **Title**: The title of the movie.
2. **Release Date**: The date when the movie was released.
3. **Genres**: The categories or genres that the movie belongs to (e.g., action, comedy, drama).
4. **Director**: The director(s) of the movie.
5. **Cast**: The main actors or actresses starring in the movie.
6. **Production Company**: The company or companies responsible for producing the movie.
7. **Budget**: The estimated or actual budget of the movie.
8. **Revenue**: The total revenue generated by the movie, typically at the box office.
9. **Runtime**: The duration of the movie in minutes.
10. **Plot Summary**: A brief summary or description of the movie's plot or storyline.
11. **Rating**: The rating given to the movie by critics or audiences (e.g., IMDb rating, Rotten Tomatoes score).
12. **Popularity**: A measure of the movie's popularity or public interest, often based on factors like social media mentions or online searches.

13. **Awards**: Any awards or nominations received by the movie, such as Oscars or Golden Globes.
14. **Keywords**: Keywords or tags associated with the movie, describing its themes, topics, or motifs.
15. **Poster Image**: An image or poster associated with the movie, typically used for promotional purposes.

In [1]:
```
#Importing libraries
import numpy as np
import pandas as pd
```

In [2]:
```
#Importing datasets and creating dataframes
movies=pd.read_csv('tmdb_5000_movies.csv')
credits=pd.read_csv('tmdb_5000_credits.csv')
```

In [3]:
```
movies.head(2)
```

Out[3]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	p
0	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...	15
1	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...	13

In [4]:
```
credits.head(2)
```

Out[4]:

	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...

In [5]:
```
#merging movies & credits dataframes as movies
movies = movies.merge(credits,on='title')
```

In [6]:
```
#no.of rows & columns
movies.shape
```

Out[6]:
```
(4809, 23)
```

In [7]:
```
movies.head(2)
```

Out[7]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	p
0	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...	15
1	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...	13

2 rows × 23 columns

In [8]:
```
movies['original_language'].value_counts()
```

Out[8]:
```
original_language
en    4510
fr      70
es      32
zh      27
de      27
hi      19
ja      16
it      14
ko      12
cn      12
ru      11
pt       9
da       7
sv       5
nl       4
fa       4
th       3
he       3
ta       2
cs       2
ro       2
id       2
ar       2
vi       1
sl       1
ps       1
no       1
ky       1
hu       1
pl       1
af       1
nb       1
tr       1
is       1
xx       1
te       1
el       1
Name: count, dtype: int64
```

In [9]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4809 entries, 0 to 4808
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   budget                4809 non-null   int64  
 1   genres                4809 non-null   object 
 2   homepage              1713 non-null   object 
 3   id                    4809 non-null   int64  
 4   keywords              4809 non-null   object 
 5   original_language     4809 non-null   object 
 6   original_title        4809 non-null   object 
 7   overview              4806 non-null   object 
 8   popularity            4809 non-null   float64
 9   production_companies  4809 non-null   object 
 10  production_countries  4809 non-null   object 
 11  release_date          4808 non-null   object 
 12  revenue               4809 non-null   int64  
 13  runtime               4807 non-null   float64
 14  spoken_languages      4809 non-null   object 
 15  status                4809 non-null   object 
 16  tagline               3965 non-null   object 
 17  title                 4809 non-null   object 
 18  vote_average          4809 non-null   float64
 19  vote_count            4809 non-null   int64  
 20  movie_id              4809 non-null   int64  
 21  cast                  4809 non-null   object 
 22  crew                  4809 non-null   object 
dtypes: float64(3), int64(5), object(15)
memory usage: 864.2+ KB
```

Data Preprocessing:

- A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

- It involves below steps:

1. Getting the dataset

2. Importing libraries
3. Importing datasets
4. Finding Missing Data

Data Cleaning:

Data cleaning is the process of preparing raw data for analysis by removing or correcting inaccurate, incomplete, or irrelevant records. It involves identifying and handling missing values, normalizing data formats, filtering outliers, and ensuring consistency across the dataset. The goal is to enhance data quality and reliability, making it suitable for accurate analysis and decision-making.

In [10]:
```
#Removing unwanted columns & considering required
movies= movies[['movie_id','title','overview','genres', 'keywords','cast','crew']]
```

In [11]:
```
movies.head(2)
```

Out[11]:

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...

In [12]:
```
#checking for null values
movies.isnull().sum()
```

Out[12]:
```
movie_id    0
title       0
overview    3
genres      0
keywords    0
cast        0
crew        0
dtype: int64
```

In [13]:
```
#Removing NA
movies.dropna(inplace=True)
```

In [14]:
```
#checking for duplicates
movies.duplicated().sum()
```

Out[14]:
```
0
```

Pandas .iloc[] in Python

- In the Python Pandas library, .iloc[] is an indexer used for integer-location-based indexing of data in a DataFrame. It allows users to select specific rows and columns by providing integer indices, making it a valuable tool for data manipulation and extraction based on numerical positions within the DataFrame. This indexer is particularly useful when you want to access or manipulate data using integer-based positional indexing rather than labels.

In [15]:
```
#checking format of 0th row in genres which is list of dictionaries
movies.iloc[0].genres
```

Out[15]:
```
'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "na
me": "Science Fiction"}]'
```

ast module:

- The ast module provides a set of tools to generate, parse, and manipulate abstract syntax trees of the Python language.
- The ast.literal_eval() method is a useful tool for processing data that is stored in a string format, such as JSON data or configuration files. By using the ast.literal_eval() method, we can quickly and easily parse this data into a Python data structure that can be used in our code.The ast.literal_eval() method is a valuable tool for any Python programmer looking to process and manipulate data in their code.

In [16]:
```
#converting a list of strings into a list
import ast
ast.literal_eval('[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}
```

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

In [17]:
```
# converting string to list
# append - The append() method in Python adds a single item to the end of the existing list.
# After appending to the list, the size of the list increases by one.
def convert(obj):
    L=[]
    for i in ast.literal_eval(obj):
        L.append(i['name'])
    return L
```

In [18]:
```
#Applying convert function to genres
movies['genres'] = movies['genres'].apply(convert)
```

In [19]:
```
movies.head(2)
```

Out[19]:

	movie_id	title	overview	genres	keywords	cast	crew
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...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...

In [20]:
```
#Applying convert function to keywords
movies['keywords'].apply(convert)
```

Out[20]:
```
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...
                              ...
4804    [united states–mexico barrier, legs, arms, pap...
4805                                                   []
4806    [date, love at first sight, narration, investi...
4807                                                   []
4808            [obsession, camcorder, crush, dream girl]
Name: keywords, Length: 4806, dtype: object
```

In [21]:
```
movies['keywords']= movies['keywords'].apply(convert)
```

In [22]:
```
movies.head(1)
```

Out[22]:

	movie_id	title	overview	genres	keywords	cast	crew
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...

In [23]:
```
movies['cast'][0]
```

Out[23]:
```
'[{"cast_id": 242, "character": "Jake Sully", "credit_id": "5602a8a7c3a3685532001c9a", "gender": 2, "id": 65731
, "name": "Sam Worthington", "order": 0}, {"cast_id": 3, "character": "Neytiri", "credit_id": "52fe48009251416c
750ac9cb", "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", "ord
er": 2}, {"cast_id": 4, "character": "Col. Quaritch", "credit_id": "52fe48009251416c750ac9cf", "gender": 2, "id
": 32747, "name": "Stephen Lang", "order": 3}, {"cast_id": 5, "character": "Trudy Chacon", "credit_id": "52fe48
009251416c750ac9d3", "gender": 1, "id": 17647, "name": "Michelle Rodriguez", "order": 4}, {"cast_id": 8, "chara
cter": "Selfridge", "credit_id": "52fe48009251416c750ac9e1", "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": "52fe
48009251416c750ac9e5", "gender": 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": "52fe48009251416c
750ac9f1", "gender": 2, "id": 95697, "name": "Dileep Rao", "order": 10}, {"cast_id": 13, "character": "Lyle Wai
nfleet", "credit_id": "52fe48009251416c750ac9f5", "gender": 2, "id": 98215, "name": "Matt Gerald", "order": 11}
, {"cast_id": 32, "character": "Private Fike", "credit_id": "52fe48009251416c750aca5b", "gender": 2, "id": 1541
53, "name": "Sean Anthony Moran", "order": 12}, {"cast_id": 33, "character": "Cryo Vault Med Tech", "credit_id"
: "52fe48009251416c750aca5f", "gender": 2, "id": 397312, "name": "Jason Whyte", "order": 13}, {"cast_id": 34, "
character": "Venture Star Crew Chief", "credit_id": "52fe48009251416c750aca63", "gender": 2, "id": 42317, "name
": "Scott Lawrence", "order": 14}, {"cast_id": 35, "character": "Lock Up Trooper", "credit_id": "52fe4800925141
6c750aca67", "gender": 2, "id": 986734, "name": "Kelly Kilgour", "order": 15}, {"cast_id": 36, "character": "Sh
uttle Pilot", "credit_id": "52fe48009251416c750aca6b", "gender": 0, "id": 1207227, "name": "James Patrick Pitt"
```

, "order": 16}, {"cast_id": 37, "character": "Shuttle Co-Pilot", "credit_id": "52fe48009251416c750aca6f", "gend
er": 0, "id": 1180936, "name": "Sean Patrick Murphy", "order": 17}, {"cast_id": 38, "character": "Shuttle Crew
Chief", "credit_id": "52fe48009251416c750aca73", "gender": 2, "id": 1019578, "name": "Peter Dillon", "order": 1
8}, {"cast_id": 39, "character": "Tractor Operator / Troupe", "credit_id": "52fe48009251416c750aca77", "gender"
: 0, "id": 91443, "name": "Kevin Dorman", "order": 19}, {"cast_id": 40, "character": "Dragon Gunship Pilot", "c
redit_id": "52fe48009251416c750aca7b", "gender": 2, "id": 173391, "name": "Kelson Henderson", "order": 20}, {"c
ast_id": 41, "character": "Dragon Gunship Gunner", "credit_id": "52fe48009251416c750aca7f", "gender": 0, "id":
1207236, "name": "David Van Horn", "order": 21}, {"cast_id": 42, "character": "Dragon Gunship Navigator", "cred
it_id": "52fe48009251416c750aca83", "gender": 0, "id": 215913, "name": "Jacob Tomuri", "order": 22}, {"cast_id"
: 43, "character": "Suit #1", "credit_id": "52fe48009251416c750aca87", "gender": 0, "id": 143206, "name": "Mich
ael Blain-Rozgay", "order": 23}, {"cast_id": 44, "character": "Suit #2", "credit_id": "52fe48009251416c750aca8b
", "gender": 2, "id": 169676, "name": "Jon Curry", "order": 24}, {"cast_id": 46, "character": "Ambient Room Tec
h", "credit_id": "52fe48009251416c750aca8f", "gender": 0, "id": 1048610, "name": "Luke Hawker", "order": 25}, {
"cast_id": 47, "character": "Ambient Room Tech / Troupe", "credit_id": "52fe48009251416c750aca93", "gender": 0,
"id": 42288, "name": "Woody Schultz", "order": 26}, {"cast_id": 48, "character": "Horse Clan Leader", "credit_i
d": "52fe48009251416c750aca97", "gender": 2, "id": 68278, "name": "Peter Mensah", "order": 27}, {"cast_id": 49,
"character": "Link Room Tech", "credit_id": "52fe48009251416c750aca9b", "gender": 0, "id": 1207247, "name": "So
nia Yee", "order": 28}, {"cast_id": 50, "character": "Basketball Avatar / Troupe", "credit_id": "52fe4800925141
6c750aca9f", "gender": 1, "id": 1207248, "name": "Jahnel Curfman", "order": 29}, {"cast_id": 51, "character": "
Basketball Avatar", "credit_id": "52fe48009251416c750acaa3", "gender": 0, "id": 89714, "name": "Ilram Choi", "o
rder": 30}, {"cast_id": 52, "character": "Na\'vi Child", "credit_id": "52fe48009251416c750acaa7", "gender": 0,
"id": 1207249, "name": "Kyla Warren", "order": 31}, {"cast_id": 53, "character": "Troupe", "credit_id": "52fe48
009251416c750acaab", "gender": 0, "id": 1207250, "name": "Lisa Roumain", "order": 32}, {"cast_id": 54, "charact
er": "Troupe", "credit_id": "52fe48009251416c750acaaf", "gender": 1, "id": 83105, "name": "Debra Wilson", "orde
r": 33}, {"cast_id": 57, "character": "Troupe", "credit_id": "52fe48009251416c750acabb", "gender": 0, "id": 120
7253, "name": "Chris Mala", "order": 34}, {"cast_id": 55, "character": "Troupe", "credit_id": "52fe48009251416c
750acab3", "gender": 0, "id": 1207251, "name": "Taylor Kibby", "order": 35}, {"cast_id": 56, "character": "Trou
pe", "credit_id": "52fe48009251416c750acab7", "gender": 0, "id": 1207252, "name": "Jodie Landau", "order": 36},
{"cast_id": 58, "character": "Troupe", "credit_id": "52fe48009251416c750acabf", "gender": 0, "id": 1207254, "na
me": "Julie Lamm", "order": 37}, {"cast_id": 59, "character": "Troupe", "credit_id": "52fe48009251416c750acac3"
, "gender": 0, "id": 1207257, "name": "Cullen B. Madden", "order": 38}, {"cast_id": 60, "character": "Troupe",
"credit_id": "52fe48009251416c750acac7", "gender": 0, "id": 1207259, "name": "Joseph Brady Madden", "order": 39
}, {"cast_id": 61, "character": "Troupe", "credit_id": "52fe48009251416c750acacb", "gender": 0, "id": 1207262,
"name": "Frankie Torres", "order": 40}, {"cast_id": 62, "character": "Troupe", "credit_id": "52fe48009251416c75
0acacf", "gender": 1, "id": 1158600, "name": "Austin Wilson", "order": 41}, {"cast_id": 63, "character": "Troup
e", "credit_id": "52fe48019251416c750acad3", "gender": 1, "id": 983705, "name": "Sara Wilson", "order": 42}, {"
cast_id": 64, "character": "Troupe", "credit_id": "52fe48019251416c750acad7", "gender": 0, "id": 1207263, "name
": "Tamica Washington-Miller", "order": 43}, {"cast_id": 65, "character": "Op Center Staff", "credit_id": "52fe
48019251416c750acadb", "gender": 1, "id": 1145098, "name": "Lucy Briant", "order": 44}, {"cast_id": 66, "charac
ter": "Op Center Staff", "credit_id": "52fe48019251416c750acadf", "gender": 2, "id": 33305, "name": "Nathan Mei
ster", "order": 45}, {"cast_id": 67, "character": "Op Center Staff", "credit_id": "52fe48019251416c750acae3", "
gender": 0, "id": 1207264, "name": "Gerry Blair", "order": 46}, {"cast_id": 68, "character": "Op Center Staff",
"credit_id": "52fe48019251416c750acae7", "gender": 2, "id": 33311, "name": "Matthew Chamberlain", "order": 47},
{"cast_id": 69, "character": "Op Center Staff", "credit_id": "52fe48019251416c750acaeb", "gender": 0, "id": 120
7265, "name": "Paul Yates", "order": 48}, {"cast_id": 70, "character": "Op Center Duty Officer", "credit_id": "
52fe48019251416c750acaef", "gender": 0, "id": 1207266, "name": "Wray Wilson", "order": 49}, {"cast_id": 71, "ch
aracter": "Op Center Staff", "credit_id": "52fe48019251416c750acaf3", "gender": 2, "id": 54492, "name": "James
Gaylyn", "order": 50}, {"cast_id": 72, "character": "Dancer", "credit_id": "52fe48019251416c750acaf7", "gender"
: 0, "id": 1207267, "name": "Melvin Leno Clark III", "order": 51}, {"cast_id": 73, "character": "Dancer", "cred
it_id": "52fe48019251416c750acafb", "gender": 0, "id": 1207268, "name": "Carvon Futrell", "order": 52}, {"cast_
id": 74, "character": "Dancer", "credit_id": "52fe48019251416c750acaff", "gender": 0, "id": 1207269, "name": "B
randon Jelkes", "order": 53}, {"cast_id": 75, "character": "Dancer", "credit_id": "52fe48019251416c750acb03", "
gender": 0, "id": 1207270, "name": "Micah Moch", "order": 54}, {"cast_id": 76, "character": "Dancer", "credit_i
d": "52fe48019251416c750acb07", "gender": 0, "id": 1207271, "name": "Hanniyah Muhammad", "order": 55}, {"cast_i
d": 77, "character": "Dancer", "credit_id": "52fe48019251416c750acb0b", "gender": 0, "id": 1207272, "name": "Ch
ristopher Nolen", "order": 56}, {"cast_id": 78, "character": "Dancer", "credit_id": "52fe48019251416c750acb0f",
"gender": 0, "id": 1207273, "name": "Christa Oliver", "order": 57}, {"cast_id": 79, "character": "Dancer", "cre
dit_id": "52fe48019251416c750acb13", "gender": 0, "id": 1207274, "name": "April Marie Thomas", "order": 58}, {"
cast_id": 80, "character": "Dancer", "credit_id": "52fe48019251416c750acb17", "gender": 0, "id": 1207275, "name
": "Bravita A. Threatt", "order": 59}, {"cast_id": 81, "character": "Mining Chief (uncredited)", "credit_id": "
52fe48019251416c750acb1b", "gender": 0, "id": 1207276, "name": "Colin Bleasdale", "order": 60}, {"cast_id": 82,
"character": "Veteran Miner (uncredited)", "credit_id": "52fe48019251416c750acb1f", "gender": 0, "id": 107969,
"name": "Mike Bodnar", "order": 61}, {"cast_id": 83, "character": "Richard (uncredited)", "credit_id": "52fe480
19251416c750acb23", "gender": 0, "id": 1207278, "name": "Matt Clayton", "order": 62}, {"cast_id": 84, "characte
r": "Nav\'i (uncredited)", "credit_id": "52fe48019251416c750acb27", "gender": 1, "id": 147898, "name": "Nicole
Dionne", "order": 63}, {"cast_id": 85, "character": "Trooper (uncredited)", "credit_id": "52fe48019251416c750ac
b2b", "gender": 0, "id": 1207280, "name": "Jamie Harrison", "order": 64}, {"cast_id": 86, "character": "Trooper
(uncredited)", "credit_id": "52fe48019251416c750acb2f", "gender": 0, "id": 1207281, "name": "Allan Henry", "ord
er": 65}, {"cast_id": 87, "character": "Ground Technician (uncredited)", "credit_id": "52fe48019251416c750acb33
", "gender": 2, "id": 1207282, "name": "Anthony Ingruber", "order": 66}, {"cast_id": 88, "character": "Flight C
rew Mechanic (uncredited)", "credit_id": "52fe48019251416c750acb37", "gender": 0, "id": 1207283, "name": "Ashle
y Jeffery", "order": 67}, {"cast_id": 14, "character": "Samson Pilot", "credit_id": "52fe48009251416c750ac9f9",
"gender": 0, "id": 98216, "name": "Dean Knowsley", "order": 68}, {"cast_id": 89, "character": "Trooper (uncredi
ted)", "credit_id": "52fe48019251416c750acb3b", "gender": 0, "id": 1201399, "name": "Joseph Mika-Hunt", "order"
: 69}, {"cast_id": 90, "character": "Banshee (uncredited)", "credit_id": "52fe48019251416c750acb3f", "gender":
0, "id": 236696, "name": "Terry Notary", "order": 70}, {"cast_id": 91, "character": "Soldier (uncredited)", "cr
edit_id": "52fe48019251416c750acb43", "gender": 0, "id": 1207287, "name": "Kai Pantano", "order": 71}, {"cast_i
d": 92, "character": "Blast Technician (uncredited)", "credit_id": "52fe48019251416c750acb47", "gender": 0, "id
": 1207288, "name": "Logan Pithyou", "order": 72}, {"cast_id": 93, "character": "Vindum Raah (uncredited)", "cr
edit_id": "52fe48019251416c750acb4b", "gender": 0, "id": 1207289, "name": "Stuart Pollock", "order": 73}, {"cas
t_id": 94, "character": "Hero (uncredited)", "credit_id": "52fe48019251416c750acb4f", "gender": 0, "id": 584868
, "name": "Raja", "order": 74}, {"cast_id": 95, "character": "Ops Centreworker (uncredited)", "credit_id": "52f
e48019251416c750acb53", "gender": 0, "id": 1207290, "name": "Gareth Ruck", "order": 75}, {"cast_id": 96, "chara

```
cter": "Engineer (uncredited)", "credit_id": "52fe48019251416c750acb57", "gender": 0, "id": 1062463, "name": "R
hian Sheehan", "order": 76}, {"cast_id": 97, "character": "Col. Quaritch\'s Mech Suit (uncredited)", "credit_id
": "52fe48019251416c750acb5b", "gender": 0, "id": 60656, "name": "T. J. Storm", "order": 77}, {"cast_id": 98, "
character": "Female Marine (uncredited)", "credit_id": "52fe48019251416c750acb5f", "gender": 0, "id": 1207291,
"name": "Jodie Taylor", "order": 78}, {"cast_id": 99, "character": "Ikran Clan Leader (uncredited)", "credit_id
": "52fe48019251416c750acb63", "gender": 1, "id": 1186027, "name": "Alicia Vela-Bailey", "order": 79}, {"cast_i
d": 100, "character": "Geologist (uncredited)", "credit_id": "52fe48019251416c750acb67", "gender": 0, "id": 120
7292, "name": "Richard Whiteside", "order": 80}, {"cast_id": 101, "character": "Na\'vi (uncredited)", "credit_i
d": "52fe48019251416c750acb6b", "gender": 0, "id": 103259, "name": "Nikie Zambo", "order": 81}, {"cast_id": 102
, "character": "Ambient Room Tech / Troupe", "credit_id": "52fe48019251416c750acb6f", "gender": 1, "id": 42286,
"name": "Julene Renee", "order": 82}]'
```

In [24]:
```
#extracting top3 actors names from 1st three dictionaries of all movies
def convert3(obj):
    L=[]
    counter=0
    for i in ast.literal_eval(obj):
        if counter !=3:
           L.append(i['name'])
           counter+=1
        else:
            break
    return L
```

In [25]:
```
#Applying convert3 function to cast
movies['cast'].apply(convert3)
```

Out[25]:
```
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]
                              ...
4804    [Carlos Gallardo, Jaime de Hoyos, Peter Marqua...
4805          [Edward Burns, Kerry Bishé, Marsha Dietlein]
4806            [Eric Mabius, Kristin Booth, Crystal Lowe]
4807             [Daniel Henney, Eliza Coupe, Bill Paxton]
4808    [Drew Barrymore, Brian Herzlinger, Corey Feldman]
Name: cast, Length: 4806, dtype: object
```

In [26]:
```
#saving changes to movies_cast
movies['cast'] = movies['cast'].apply(convert3)
```

In [27]:
```
movies.head(2)
```

Out[27]:

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[{"credit_id": "52fe4232c3a36847f800b579", "de...

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

Out[28]:
```
'[{"credit_id": "52fe48009251416c750aca23", "department": "Editing", "gender": 0, "id": 1721, "job": "Editor",
"name": "Stephen E. Rivkin"}, {"credit_id": "539c47ecc3a36810e3001f87", "department": "Art", "gender": 2, "id":
496, "job": "Production Design", "name": "Rick Carter"}, {"credit_id": "54491c89c3a3680fb4001cf7", "department"
: "Sound", "gender": 0, "id": 900, "job": "Sound Designer", "name": "Christopher Boyes"}, {"credit_id": "54491c
b70e0a267480001bd0", "department": "Sound", "gender": 0, "id": 900, "job": "Supervising Sound Editor", "name":
"Christopher Boyes"}, {"credit_id": "539c4a4cc3a36810c9002101", "department": "Production", "gender": 1, "id":
1262, "job": "Casting", "name": "Mali Finn"}, {"credit_id": "5544ee3b925141499f0008fc", "department": "Sound",
"gender": 2, "id": 1729, "job": "Original Music Composer", "name": "James Horner"}, {"credit_id": "52fe48009251
416c750ac9c3", "department": "Directing", "gender": 2, "id": 2710, "job": "Director", "name": "James Cameron"},
{"credit_id": "52fe48009251416c750ac9d9", "department": "Writing", "gender": 2, "id": 2710, "job": "Writer", "n
ame": "James Cameron"}, {"credit_id": "52fe48009251416c750aca17", "department": "Editing", "gender": 2, "id": 2
710, "job": "Editor", "name": "James Cameron"}, {"credit_id": "52fe48009251416c750aca29", "department": "Produc
tion", "gender": 2, "id": 2710, "job": "Producer", "name": "James Cameron"}, {"credit_id": "52fe48009251416c750
aca3f", "department": "Writing", "gender": 2, "id": 2710, "job": "Screenplay", "name": "James Cameron"}, {"cred
it_id": "539c4987c3a36810ba0021a4", "department": "Art", "gender": 2, "id": 7236, "job": "Art Direction", "name
": "Andrew Menzies"}, {"credit_id": "549598c3c3a3686ae9004383", "department": "Visual Effects", "gender": 0, "i
d": 6690, "job": "Visual Effects Producer", "name": "Jill Brooks"}, {"credit_id": "52fe48009251416c750aca4b", "
department": "Production", "gender": 1, "id": 6347, "job": "Casting", "name": "Margery Simkin"}, {"credit_id":
"570b6f419251417da70032fe", "department": "Art", "gender": 2, "id": 6878, "job": "Supervising Art Director", "n
ame": "Kevin Ishioka"}, {"credit_id": "5495a0fac3a3686ae9004468", "department": "Sound", "gender": 0, "id": 688
3, "job": "Music Editor", "name": "Dick Bernstein"}, {"credit_id": "54959706c3a3686af3003e81", "department": "S
ound", "gender": 0, "id": 8159, "job": "Sound Effects Editor", "name": "Shannon Mills"}, {"credit_id": "54491d5
8c3a3680fb1001ccb", "department": "Sound", "gender": 0, "id": 8160, "job": "Foley", "name": "Dennie Thorpe"}, {
"credit_id": "54491d6cc3a3680fa5001b2c", "department": "Sound", "gender": 0, "id": 8163, "job": "Foley", "name"
```

```
: "Jana Vance"}, {"credit_id": "52fe48009251416c750aca57", "department": "Costume & Make-Up", "gender": 1, "id"
: 8527, "job": "Costume Design", "name": "Deborah Lynn Scott"}, {"credit_id": "52fe48009251416c750aca2f", "depa
rtment": "Production", "gender": 2, "id": 8529, "job": "Producer", "name": "Jon Landau"}, {"credit_id": "539c49
37c3a36810ba002194", "department": "Art", "gender": 0, "id": 9618, "job": "Art Direction", "name": "Sean Hawort
h"}, {"credit_id": "539c49b6c3a36810c10020e6", "department": "Art", "gender": 1, "id": 12653, "job": "Set Decor
ation", "name": "Kim Sinclair"}, {"credit_id": "570b6f2f9251413a0e00020d", "department": "Art", "gender": 1, "i
d": 12653, "job": "Supervising Art Director", "name": "Kim Sinclair"}, {"credit_id": "54491a6c0e0a26748c001b19"
, "department": "Art", "gender": 2, "id": 14350, "job": "Set Designer", "name": "Richard F. Mays"}, {"credit_id
": "56928cf4c3a3684cff0025c4", "department": "Production", "gender": 1, "id": 20294, "job": "Executive Producer
", "name": "Laeta Kalogridis"}, {"credit_id": "52fe48009251416c750aca51", "department": "Costume & Make-Up", "g
ender": 0, "id": 17675, "job": "Costume Design", "name": "Mayes C. Rubeo"}, {"credit_id": "52fe48009251416c750a
ca11", "department": "Camera", "gender": 2, "id": 18265, "job": "Director of Photography", "name": "Mauro Fiore
"}, {"credit_id": "5449194d0e0a26748f001b39", "department": "Art", "gender": 0, "id": 42281, "job": "Set Design
er", "name": "Scott Herbertson"}, {"credit_id": "52fe48009251416c750aca05", "department": "Crew", "gender": 0,
"id": 42288, "job": "Stunts", "name": "Woody Schultz"}, {"credit_id": "5592aefb92514152de0010f5", "department":
"Costume & Make-Up", "gender": 0, "id": 29067, "job": "Makeup Artist", "name": "Linda DeVetta"}, {"credit_id":
"5592afa492514152de00112c", "department": "Costume & Make-Up", "gender": 0, "id": 29067, "job": "Hairstylist",
"name": "Linda DeVetta"}, {"credit_id": "54959ed592514130fc002e5d", "department": "Camera", "gender": 2, "id":
33302, "job": "Camera Operator", "name": "Richard Bluck"}, {"credit_id": "539c4891c3a36810ba002147", "departmen
t": "Art", "gender": 2, "id": 33303, "job": "Art Direction", "name": "Simon Bright"}, {"credit_id": "54959c0692
51417a81001f3a", "department": "Visual Effects", "gender": 0, "id": 113145, "job": "Visual Effects Supervisor",
"name": "Richard Martin"}, {"credit_id": "54959a0dc3a3680ff5002c8d", "department": "Crew", "gender": 2, "id": 5
8188, "job": "Visual Effects Editor", "name": "Steve R. Moore"}, {"credit_id": "52fe48009251416c750aca1d", "dep
artment": "Editing", "gender": 2, "id": 58871, "job": "Editor", "name": "John Refoua"}, {"credit_id": "54491a4d
c3a3680fc30018ca", "department": "Art", "gender": 0, "id": 92359, "job": "Set Designer", "name": "Karl J. Marti
n"}, {"credit_id": "52fe48009251416c750aca35", "department": "Camera", "gender": 1, "id": 72201, "job": "Direct
or of Photography", "name": "Chiling Lin"}, {"credit_id": "52fe48009251416c750ac9ff", "department": "Crew", "ge
nder": 0, "id": 89714, "job": "Stunts", "name": "Ilram Choi"}, {"credit_id": "54959c529251416e2b004394", "depar
tment": "Visual Effects", "gender": 2, "id": 93214, "job": "Visual Effects Supervisor", "name": "Steven Quale"}
, {"credit_id": "54491edf0e0a267489001c37", "department": "Crew", "gender": 1, "id": 122607, "job": "Dialect Co
ach", "name": "Carla Meyer"}, {"credit_id": "539c485bc3a368653d001a3a", "department": "Art", "gender": 2, "id":
132585, "job": "Art Direction", "name": "Nick Bassett"}, {"credit_id": "539c4903c3a368653d001a74", "department"
: "Art", "gender": 0, "id": 132596, "job": "Art Direction", "name": "Jill Cormack"}, {"credit_id": "539c4967c3a
368653d001a94", "department": "Art", "gender": 0, "id": 132604, "job": "Art Direction", "name": "Andy McLaren"}
, {"credit_id": "52fe48009251416c750aca45", "department": "Crew", "gender": 0, "id": 236696, "job": "Motion Cap
ture Artist", "name": "Terry Notary"}, {"credit_id": "54959e02c3a3680fc60027d2", "department": "Crew", "gender"
: 2, "id": 956198, "job": "Stunt Coordinator", "name": "Garrett Warren"}, {"credit_id": "54959ca3c3a3686ae30043
8c", "department": "Visual Effects", "gender": 2, "id": 957874, "job": "Visual Effects Supervisor", "name": "Jo
nathan Rothbart"}, {"credit_id": "570b6f519251412c74001b2f", "department": "Art", "gender": 0, "id": 957889, "j
ob": "Supervising Art Director", "name": "Stefan Dechant"}, {"credit_id": "570b6f62c3a3680b77007460", "departme
nt": "Art", "gender": 2, "id": 959555, "job": "Supervising Art Director", "name": "Todd Cherniawsky"}, {"credit
_id": "539c4a3ac3a36810da0021cc", "department": "Production", "gender": 0, "id": 1016177, "job": "Casting", "na
me": "Miranda Rivers"}, {"credit_id": "539c482cc3a36810c1002062", "department": "Art", "gender": 0, "id": 10325
36, "job": "Production Design", "name": "Robert Stromberg"}, {"credit_id": "539c4b65c3a36810c9002125", "departm
ent": "Costume & Make-Up", "gender": 2, "id": 1071680, "job": "Costume Design", "name": "John Harding"}, {"cred
it_id": "54959e6692514130fc002e4e", "department": "Camera", "gender": 0, "id": 1177364, "job": "Steadicam Opera
tor", "name": "Roberto De Angelis"}, {"credit_id": "539c49f1c3a368653d001aac", "department": "Costume & Make-Up
", "gender": 2, "id": 1202850, "job": "Makeup Department Head", "name": "Mike Smithson"}, {"credit_id": "549599
9ec3a3686ae100460c", "department": "Visual Effects", "gender": 0, "id": 1204668, "job": "Visual Effects Produce
r", "name": "Alain Lalanne"}, {"credit_id": "54959cdfc3a3681153002729", "department": "Visual Effects", "gender
": 0, "id": 1206410, "job": "Visual Effects Supervisor", "name": "Lucas Salton"}, {"credit_id": "54959623925141
7a81001eae", "department": "Crew", "gender": 0, "id": 1234266, "job": "Post Production Supervisor", "name": "Ja
nace Tashjian"}, {"credit_id": "54959c859251416e1e003efe", "department": "Visual Effects", "gender": 0, "id": 1
271932, "job": "Visual Effects Supervisor", "name": "Stephen Rosenbaum"}, {"credit_id": "5592af28c3a368775a0010
5f", "department": "Costume & Make-Up", "gender": 0, "id": 1310064, "job": "Makeup Artist", "name": "Frankie Ka
rena"}, {"credit_id": "539c4adfc3a36810e300203b", "department": "Costume & Make-Up", "gender": 1, "id": 1319844
, "job": "Costume Supervisor", "name": "Lisa Lovaas"}, {"credit_id": "54959b579251416e2b004371", "department":
"Visual Effects", "gender": 0, "id": 1327028, "job": "Visual Effects Supervisor", "name": "Jonathan Fawkner"},
{"credit_id": "539c48a7c3a36810b5001fa7", "department": "Art", "gender": 0, "id": 1330561, "job": "Art Directio
n", "name": "Robert Bavin"}, {"credit_id": "539c4a71c3a36810da0021e0", "department": "Costume & Make-Up", "gend
er": 0, "id": 1330567, "job": "Costume Supervisor", "name": "Anthony Almaraz"}, {"credit_id": "539c4a8ac3a36810
ba0021e4", "department": "Costume & Make-Up", "gender": 0, "id": 1330570, "job": "Costume Supervisor", "name":
"Carolyn M. Fenton"}, {"credit_id": "539c4ab6c3a36810da0021f0", "department": "Costume & Make-Up", "gender": 0,
"id": 1330574, "job": "Costume Supervisor", "name": "Beth Koenigsberg"}, {"credit_id": "54491ab70e0a267480001ba
2", "department": "Art", "gender": 0, "id": 1336191, "job": "Set Designer", "name": "Sam Page"}, {"credit_id":
"544919d9c3a3680fc30018bd", "department": "Art", "gender": 0, "id": 1339441, "job": "Set Designer", "name": "Te
x Kadonaga"}, {"credit_id": "54491cf50e0a267483001b0c", "department": "Editing", "gender": 0, "id": 1352422, "j
ob": "Dialogue Editor", "name": "Kim Foscato"}, {"credit_id": "544919f40e0a26748c001b09", "department": "Art",
"gender": 0, "id": 1352962, "job": "Set Designer", "name": "Tammy S. Lee"}, {"credit_id": "5495a115c3a3680ff500
2d71", "department": "Crew", "gender": 0, "id": 1357070, "job": "Transportation Coordinator", "name": "Denny Ca
ira"}, {"credit_id": "5495a12f92514130fc002e94", "department": "Crew", "gender": 0, "id": 1357071, "job": "Tran
sportation Coordinator", "name": "James Waitkus"}, {"credit_id": "5495976fc3a36811530026b0", "department": "Sou
nd", "gender": 0, "id": 1360103, "job": "Supervising Sound Editor", "name": "Addison Teague"}, {"credit_id": "5
4491837c3a3680fb1001c5a", "department": "Art", "gender": 2, "id": 1376887, "job": "Set Designer", "name": "C. S
cott Baker"}, {"credit_id": "54491878c3a3680fb4001c9d", "department": "Art", "gender": 0, "id": 1376888, "job":
"Set Designer", "name": "Luke Caska"}, {"credit_id": "544918dac3a3680fa5001ae0", "department": "Art", "gender":
0, "id": 1376889, "job": "Set Designer", "name": "David Chow"}, {"credit_id": "544919110e0a267486001b68", "depa
rtment": "Art", "gender": 0, "id": 1376890, "job": "Set Designer", "name": "Jonathan Dyer"}, {"credit_id": "544
91967c3a3680faa001b5e", "department": "Art", "gender": 0, "id": 1376891, "job": "Set Designer", "name": "Joseph
Hiura"}, {"credit_id": "54491997c3a3680fb1001c8a", "department": "Art", "gender": 0, "id": 1376892, "job": "Art
Department Coordinator", "name": "Rebecca Jellie"}, {"credit_id": "544919ba0e0a26748f001b42", "department": "Ar
t", "gender": 0, "id": 1376893, "job": "Set Designer", "name": "Robert Andrew Johnson"}, {"credit_id": "54491b1
dc3a3680faa001b8c", "department": "Art", "gender": 0, "id": 1376895, "job": "Assistant Art Director", "name": "
```

```
Mike Stassi"}, {"credit_id": "54491b79c3a3680fbb001826", "department": "Art", "gender": 0, "id": 1376897, "job"
: "Construction Coordinator", "name": "John Villarino"}, {"credit_id": "54491baec3a3680fb4001ce6", "department"
: "Art", "gender": 2, "id": 1376898, "job": "Assistant Art Director", "name": "Jeffrey Wisniewski"}, {"credit_i
d": "54491d2fc3a3680fb4001d07", "department": "Editing", "gender": 0, "id": 1376899, "job": "Dialogue Editor",
 "name": "Cheryl Nardi"}, {"credit_id": "54491d86c3a3680fa5001b2f", "department": "Editing", "gender": 0, "id":
1376901, "job": "Dialogue Editor", "name": "Marshall Winn"}, {"credit_id": "54491d9dc3a3680faa001bb0", "departm
ent": "Sound", "gender": 0, "id": 1376902, "job": "Supervising Sound Editor", "name": "Gwendolyn Yates Whittle"
}, {"credit_id": "54491dc10e0a267486001bce", "department": "Sound", "gender": 0, "id": 1376903, "job": "Sound R
e-Recording Mixer", "name": "William Stein"}, {"credit_id": "54491f500e0a26747c001c07", "department": "Crew", "
gender": 0, "id": 1376909, "job": "Choreographer", "name": "Lula Washington"}, {"credit_id": "549599239251412c4
e002a2e", "department": "Visual Effects", "gender": 0, "id": 1391692, "job": "Visual Effects Producer", "name":
"Chris Del Conte"}, {"credit_id": "54959d54c3a36831b8001d9a", "department": "Visual Effects", "gender": 2, "id"
: 1391695, "job": "Visual Effects Supervisor", "name": "R. Christopher White"}, {"credit_id": "54959bdf9251412c
4e002a66", "department": "Visual Effects", "gender": 0, "id": 1394070, "job": "Visual Effects Supervisor", "nam
e": "Dan Lemmon"}, {"credit_id": "5495971d92514132ed002922", "department": "Sound", "gender": 0, "id": 1394129,
"job": "Sound Effects Editor", "name": "Tim Nielsen"}, {"credit_id": "5592b25792514152cc0011aa", "department":
"Crew", "gender": 0, "id": 1394286, "job": "CG Supervisor", "name": "Michael Mulholland"}, {"credit_id": "54959
a329251416e2b004355", "department": "Crew", "gender": 0, "id": 1394750, "job": "Visual Effects Editor", "name":
"Thomas Nittmann"}, {"credit_id": "54959d6dc3a3686ae9004401", "department": "Visual Effects", "gender": 0, "id"
: 1394755, "job": "Visual Effects Supervisor", "name": "Edson Williams"}, {"credit_id": "5495a08fc3a3686ae30044
1c", "department": "Editing", "gender": 0, "id": 1394953, "job": "Digital Intermediate", "name": "Christine Car
r"}, {"credit_id": "55402d659251413d6d000249", "department": "Visual Effects", "gender": 0, "id": 1395269, "job
": "Visual Effects Supervisor", "name": "John Bruno"}, {"credit_id": "54959e7b9251416e1e003f3e", "department":
"Camera", "gender": 0, "id": 1398970, "job": "Steadicam Operator", "name": "David Emmerichs"}, {"credit_id": "5
4959734c3a3686ae10045e0", "department": "Sound", "gender": 0, "id": 1400906, "job": "Sound Effects Editor", "na
me": "Christopher Scarabosio"}, {"credit_id": "549595dd92514130fc002d79", "department": "Production", "gender":
0, "id": 1401784, "job": "Production Supervisor", "name": "Jennifer Teves"}, {"credit_id": "549596009251413af70
028cc", "department": "Production", "gender": 0, "id": 1401785, "job": "Production Manager", "name": "Brigitte
Yorke"}, {"credit_id": "549596e892514130fc002d99", "department": "Sound", "gender": 0, "id": 1401786, "job": "S
ound Effects Editor", "name": "Ken Fischer"}, {"credit_id": "549598229251412c4e002a1c", "department": "Crew", "
gender": 0, "id": 1401787, "job": "Special Effects Coordinator", "name": "Iain Hutton"}, {"credit_id": "5495983
49251416e2b00432b", "department": "Crew", "gender": 0, "id": 1401788, "job": "Special Effects Coordinator", "na
me": "Steve Ingram"}, {"credit_id": "54959905c3a3686ae3004324", "department": "Visual Effects", "gender": 0, "i
d": 1401789, "job": "Visual Effects Producer", "name": "Joyce Cox"}, {"credit_id": "5495994b92514132ed002951",
"department": "Visual Effects", "gender": 0, "id": 1401790, "job": "Visual Effects Producer", "name": "Jenny Fo
ster"}, {"credit_id": "549599cbc3a3686ae1004613", "department": "Crew", "gender": 0, "id": 1401791, "job": "Vis
ual Effects Editor", "name": "Christopher Marino"}, {"credit_id": "549599f2c3a3686ae100461e", "department": "Cr
ew", "gender": 0, "id": 1401792, "job": "Visual Effects Editor", "name": "Jim Milton"}, {"credit_id": "54959a51
c3a3686af3003eb5", "department": "Visual Effects", "gender": 0, "id": 1401793, "job": "Visual Effects Producer"
, "name": "Cyndi Ochs"}, {"credit_id": "54959a7cc3a36811530026f4", "department": "Crew", "gender": 0, "id": 140
1794, "job": "Visual Effects Editor", "name": "Lucas Putnam"}, {"credit_id": "54959b91c3a3680ff5002cb4", "depar
tment": "Visual Effects", "gender": 0, "id": 1401795, "job": "Visual Effects Supervisor", "name": "Anthony \'Ma
x\' Ivins"}, {"credit_id": "54959bb69251412c4e002a5f", "department": "Visual Effects", "gender": 0, "id": 14017
96, "job": "Visual Effects Supervisor", "name": "John Knoll"}, {"credit_id": "54959cbbc3a3686ae3004391", "depar
tment": "Visual Effects", "gender": 2, "id": 1401799, "job": "Visual Effects Supervisor", "name": "Eric Saindon
"}, {"credit_id": "54959d06c3a3686ae90043f6", "department": "Visual Effects", "gender": 0, "id": 1401800, "job"
: "Visual Effects Supervisor", "name": "Wayne Stables"}, {"credit_id": "54959d259251416e1e003f11", "department"
: "Visual Effects", "gender": 0, "id": 1401801, "job": "Visual Effects Supervisor", "name": "David Stinnett"},
{"credit_id": "54959db49251413af7002975", "department": "Visual Effects", "gender": 0, "id": 1401803, "job": "V
isual Effects Supervisor", "name": "Guy Williams"}, {"credit_id": "54959de4c3a3681153002750", "department": "Cr
ew", "gender": 0, "id": 1401804, "job": "Stunt Coordinator", "name": "Stuart Thorp"}, {"credit_id": "54959ef2c3
a3680fc60027f2", "department": "Lighting", "gender": 0, "id": 1401805, "job": "Best Boy Electric", "name": "Gil
es Coburn"}, {"credit_id": "54959f07c3a3680fc60027f9", "department": "Camera", "gender": 2, "id": 1401806, "job
": "Still Photographer", "name": "Mark Fellman"}, {"credit_id": "54959f47c3a3681153002774", "department": "Ligh
ting", "gender": 0, "id": 1401807, "job": "Lighting Technician", "name": "Scott Sprague"}, {"credit_id": "54959
f8cc3a36831b8001df2", "department": "Visual Effects", "gender": 0, "id": 1401808, "job": "Animation Director",
 "name": "Jeremy Hollobon"}, {"credit_id": "54959fa0c3a36831b8001dfb", "department": "Visual Effects", "gender":
0, "id": 1401809, "job": "Animation Director", "name": "Orlando Meunier"}, {"credit_id": "54959fb6c3a3686af3003
f54", "department": "Visual Effects", "gender": 0, "id": 1401810, "job": "Animation Director", "name": "Taisuke
Tanimura"}, {"credit_id": "54959fd2c3a36831b8001e02", "department": "Costume & Make-Up", "gender": 0, "id": 140
1812, "job": "Set Costumer", "name": "Lilia Mishel Acevedo"}, {"credit_id": "54959ff9c3a3686ae300440c", "depart
ment": "Costume & Make-Up", "gender": 0, "id": 1401814, "job": "Set Costumer", "name": "Alejandro M. Hernandez"
}, {"credit_id": "5495a0ddc3a3686ae10046fe", "department": "Editing", "gender": 0, "id": 1401815, "job": "Digit
al Intermediate", "name": "Marvin Hall"}, {"credit_id": "5495a1f7c3a3686ae3004443", "department": "Production",
"gender": 0, "id": 1401816, "job": "Publicist", "name": "Judy Alley"}, {"credit_id": "5592b29fc3a36869d100002f"
, "department": "Crew", "gender": 0, "id": 1418381, "job": "CG Supervisor", "name": "Mike Perry"}, {"credit_id"
: "5592b23a9251415df8001081", "department": "Crew", "gender": 0, "id": 1426854, "job": "CG Supervisor", "name":
"Andrew Morley"}, {"credit_id": "55491e1192514104c40002d8", "department": "Art", "gender": 0, "id": 1438901, "j
ob": "Conceptual Design", "name": "Seth Engstrom"}, {"credit_id": "5525d5809251417276002b06", "department": "Cr
ew", "gender": 0, "id": 1447362, "job": "Visual Effects Art Director", "name": "Eric Oliver"}, {"credit_id": "5
54427ca925141586500312a", "department": "Visual Effects", "gender": 0, "id": 1447503, "job": "Modeling", "name"
: "Matsune Suzuki"}, {"credit_id": "551906889251415aab001c88", "department": "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": "Lighting", "gender": 0, "id": 1453938, "job": "Lighting Artist", "na
me": "Arun Ram-Mohan"}, {"credit_id": "5592af4692514152d5001355", "department": "Costume & Make-Up", "gender":
0, "id": 1457305, "job": "Makeup Artist", "name": "Georgia Lockhart-Adams"}, {"credit_id": "5592b2eac3a36877470
012a5", "department": "Crew", "gender": 0, "id": 1466035, "job": "CG Supervisor", "name": "Thrain Shadbolt"}, {
"credit_id": "5592b032c3a36877450015f1", "department": "Crew", "gender": 0, "id": 1483220, "job": "CG Superviso
r", "name": "Brad Alexander"}, {"credit_id": "5592b05592514152d80012f6", "department": "Crew", "gender": 0, "id
": 1483221, "job": "CG Supervisor", "name": "Shadi Almassizadeh"}, {"credit_id": "5592b090c3a36877570010b5", "d
epartment": "Crew", "gender": 0, "id": 1483222, "job": "CG Supervisor", "name": "Simon Clutterbuck"}, {"credit_
id": "5592b0dbc3a368774b00112c", "department": "Crew", "gender": 0, "id": 1483223, "job": "CG Supervisor", "nam
```

```
e": "Graeme Demmocks"}, {"credit_id": "5592b0fe92514152db0010c1", "department": "Crew", "gender": 0, "id": 1483
224, "job": "CG Supervisor", "name": "Adrian Fernandes"}, {"credit_id": "5592b11f9251415df8001059", "department
": "Crew", "gender": 0, "id": 1483225, "job": "CG Supervisor", "name": "Mitch Gates"}, {"credit_id": "5592b15dc
3a3687745001645", "department": "Crew", "gender": 0, "id": 1483226, "job": "CG Supervisor", "name": "Jerry Kung
"}, {"credit_id": "5592b18e925141645a0004ae", "department": "Crew", "gender": 0, "id": 1483227, "job": "CG Supe
rvisor", "name": "Andy Lomas"}, {"credit_id": "5592b1bfc3a368775d0010e7", "department": "Crew", "gender": 0, "i
d": 1483228, "job": "CG Supervisor", "name": "Sebastian Marino"}, {"credit_id": "5592b2049251415df8001078", "de
partment": "Crew", "gender": 0, "id": 1483229, "job": "CG Supervisor", "name": "Matthias Menz"}, {"credit_id":
"5592b27b92514152d800136a", "department": "Crew", "gender": 0, "id": 1483230, "job": "CG Supervisor", "name": "
Sergei Nevshupov"}, {"credit_id": "5592b2c3c3a36869e800003c", "department": "Crew", "gender": 0, "id": 1483231,
"job": "CG Supervisor", "name": "Philippe Rebours"}, {"credit_id": "5592b317c3a36877470012af", "department": "C
rew", "gender": 0, "id": 1483232, "job": "CG Supervisor", "name": "Michael Takarangi"}, {"credit_id": "5592b345
c3a36877470012bb", "department": "Crew", "gender": 0, "id": 1483233, "job": "CG Supervisor", "name": "David Wei
tzberg"}, {"credit_id": "5592b37cc3a368775100113b", "department": "Crew", "gender": 0, "id": 1483234, "job": "C
G Supervisor", "name": "Ben White"}, {"credit_id": "573c8e2f9251413f5d000094", "department": "Crew", "gender":
1, "id": 1621932, "job": "Stunts", "name": "Min Windle"}]'
```

In [29]:
```python
#fetching only director names of all movies
def fetch_director(obj):
    L=[]
    for i in ast.literal_eval(obj):
        if i ['job'] == 'Director':
            L.append(i['name'])
            break
    return L
```

In [30]:
```python
#Applying fetch_director function to crew
movies['crew'].apply(fetch_director)
```

Out[30]:
```
0           [James Cameron]
1          [Gore Verbinski]
2              [Sam Mendes]
3       [Christopher Nolan]
4          [Andrew Stanton]
               ...
4804     [Robert Rodriguez]
4805         [Edward Burns]
4806          [Scott Smith]
4807          [Daniel Hsia]
4808     [Brian Herzlinger]
Name: crew, Length: 4806, dtype: object
```

In [31]:
```python
#saving changes to movies_crew
movies['crew']= movies['crew'].apply(fetch_director)
```

In [32]:
```python
movies.head(2)
```

Out[32]:

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[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 dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski]

Lambda Function

- A lambda function is a small anonymous function.A lambda function can take any number of arguments, but can only have one expression.The power of lambda is better shown when you use them as an anonymous function inside another function.

Syntax

- lambda arguments : expression

Example

x = lambda a, b, c : a + b + c print(x(5, 6, 2))

Split Method:

- The split() method splits a string into a list.You can specify the separator, default separator is any whitespace.

In [33]:
```python
#converting overview column into list
movies['overview']=movies['overview'].apply(lambda x:x.split())
```

In [34]:
```python
movies.head(2)
```

Out[34]:

	movie_id	title	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]

In [35]:
```
#removing spaces between words to avoid confusion if names are similar
movies['genres'].apply(lambda x:[i.replace(" ","")for i in x])
```

Out[35]:
```
0        [Action, Adventure, Fantasy, ScienceFiction]
1                        [Adventure, Fantasy, Action]
2                          [Action, Adventure, Crime]
3                    [Action, Crime, Drama, Thriller]
4        [Action, Adventure, ScienceFiction]
                             ...
4804                        [Action, Crime, Thriller]
4805                                [Comedy, Romance]
4806              [Comedy, Drama, Romance, TVMovie]
4807                                               []
4808                                    [Documentary]
Name: genres, Length: 4806, dtype: object
```

In [36]:
```
#removing spaces between words to avoid confusion if names are similar
movies['genres']=movies['genres'].apply(lambda x:[i.replace(" ","")for i in x])
movies['keywords']=movies['keywords'].apply(lambda x:[i.replace(" ","")for i in x])
movies['cast']=movies['cast'].apply(lambda x:[i.replace(" ","")for i in x])
movies['crew']=movies['crew'].apply(lambda x:[i.replace(" ","")for i in x])
```

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

Out[37]:

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron]

In [38]:
```
#concatenating all columns as tags column
movies['tags']= movies['overview']+movies['genres']+movies['keywords']+movies['cast']+ movies['crew']
```

In [39]:
```
movies.head(1)
```

Out[39]:

	movie_id	title	overview	genres	keywords	cast	crew	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron]	[In, the, 22nd, century,, a, paraplegic, Marin...

In [40]:
```
#creating a new dataframe as new_df with necessary columns
new_df = movies[['movie_id','title','tags']]
```

In [41]:
```
#converting list to string
new_df['tags']= new_df['tags'].apply(lambda x:" ".join(x))
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11160\1369294267.py: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#retu
rning-a-view-versus-a-copy
  new_df['tags']= new_df['tags'].apply(lambda x:" ".join(x))
```

In [42]:
```
new_df.head(2)
```

Out[42]:

	movie_id	title	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...

In [43]:
```
# Viewing a particular row which is a concatenated large string
new_df['tags'][0]
```

```
Out[43]: 'In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becomes to
         rn between following orders and protecting an alien civilization. Action Adventure Fantasy ScienceFiction cultu
         reclash future spacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi ma
         rine soldier battle loveaffair antiwar powerrelations mindandsoul 3d SamWorthington ZoeSaldana SigourneyWeaver
         JamesCameron'
```

```
In [44]: #converting to lower case
         new_df['tags']=new_df['tags'].apply(lambda x:x.lower())
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11160\3413467545.py: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#retu
rning-a-view-versus-a-copy
  new_df['tags']=new_df['tags'].apply(lambda x:x.lower())
```

```
In [45]: new_df.head(2)
```

Out[45]:

	movie_id	title	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...

```
In [46]: #creating an object ps of class porterstemmer
         import nltk
         from nltk.stem.porter import PorterStemmer
         ps=PorterStemmer()
```

```
In [47]: #function stem - converts string into list and each word stemming
         #stemming-collects all inflected forms of a word in order to break them down to their root form
         def stem(text):
             y=[]

             for i in text.split():
                 y.append(ps.stem(i))

             return " ".join(y)
```

```
In [48]: #applying stem for tags & saved to new_df_tags
         new_df['tags']=new_df['tags'].apply(stem)
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11160\1642327719.py: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#retu
rning-a-view-versus-a-copy
  new_df['tags']=new_df['tags'].apply(stem)
```

Vectorization:

- Time complexity in the execution of any algorithm is very crucial deciding whether an application is reliable or not. To run a large algorithm in as much as optimal time possible is very important when it comes to real-time application of output. To do so, Python has some standard mathematical functions for fast operations on entire arrays of data without having to write loops. One of such library which contains such function is numpy.

- Vectorization is a technique that converts text or other types of data into numerical vectors that can be used for machine learning. You can use vectorization to represent movies or users as vectors based on their features, such as genre, cast, director, keywords, ratings, etc. You can also use vectorization to measure the similarity or dissimilarity between movies or users, such as cosine similarity, euclidean distance, or Jaccard index.

Bag of Words:

- The Bag of Words (BoW) model represents text by converting it into a collection of individual words, disregarding grammar and word order but keeping multiplicity. For a movie recommender system, each movie's description is converted into a vector of word frequencies.A feature extraction technique used in natural language processing (NLP). It converts text data into numerical representations that can be used by machine learning algorithms.

Key Points:

1. Text Representation: BoW represents text by counting the occurrence of each word in the text, creating a vector of word frequencies.
2. Vocabulary Creation: It builds a vocabulary of all unique words in the corpus.
3. Vectorization: Each document is then represented as a vector indicating the frequency of each word in the vocabulary.

Example:

1. Movie A description: "A thrilling adventure in space."
2. Movie B description: "A romantic adventure on Earth."
3. Corpus: ["A thrilling adventure in space.", "A romantic adventure on Earth."]
4. Combined vocabulary: {A, thrilling, adventure, in, space, romantic, on, Earth}

Vectors:

- Using these vectors, similarity between movies can be calculated to recommend similar movies to users.
- Movie A: [1, 1, 1, 1, 1, 0, 0, 0]
- Movie B: [1, 0, 1, 0, 0, 1, 1, 1]

Application:

- BoW vectors can be fed into machine learning models (e.g., Naive Bayes, SVM) for tasks like text classification, sentiment analysis, or recommendations, where the similarity between text documents is assessed based on their vector representations.

```
In [49]: #vectorization - stop words are excluded
         #converting concatenated large text into vector using BAG OF WORDS
         from sklearn.feature_extraction.text import CountVectorizer
         cv = CountVectorizer(max_features=5000,stop_words='english')
```

```
In [50]: #converting cv into a numpy array
         cv.fit_transform(new_df['tags']).toarray()
```

```
Out[50]: 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 [51]: #checking shape (movies v/s words)
         cv.fit_transform(new_df['tags']).toarray().shape
```

```
Out[51]: (4806, 5000)
```

```
In [52]: #saving as vectors
         vectors=cv.fit_transform(new_df['tags']).toarray()
```

```
In [53]: #1st movie vector
         vectors[0]
```

```
Out[53]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
In [54]: ['loved', 'loving', 'loved']
         ['love', 'love','love']
```

```
Out[54]: ['love', 'love', 'love']
```

```
In [55]: #stemming-root word
         ps.stem('loving')
```

```
Out[55]: 'love'
```

```
In [56]: #stemming
         stem('In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becom
```

```
Out[56]: 'in the 22nd century, a parapleg marin is dispatch to the moon pandora on a uniqu mission, but becom torn betwe
         en follow order and protect an alien civilization. action adventur fantasi sciencefict cultureclash futur space
         war spacecoloni societi spacetravel futurist romanc space alien tribe alienplanet cgi marin soldier battl lovea
         ffair antiwar powerrel mindandsoul 3d samworthington zoesaldana sigourneyweav jamescameron'
```

Similarity:

- Similarity is a measure of how alike two objects or vectors are. In the context of vector space, similarity quantifies the likeness between vectors representing items such as text documents, images, or user preferences. Higher similarity values indicate greater likeness.

Types of Similarity Measures:

1. **Cosine Similarity:** Measures the cosine of the angle between two vectors. Ranges from -1 (opposite) to 1 (identical).

2. **Euclidean Distance:** Measures the straight-line distance between two points in space.
3. **Jaccard Similarity:** Measures the size of the intersection divided by the size of the union of two sets.
4. **Manhattan Distance:** Measures the sum of absolute differences between the components of two vectors.
5. **Pearson Correlation:** Measures the linear correlation between two sets of data, ranging from -1 to 1.

Cosine Similarity:

- Cosine similarity and other similarity measures are used to determine the similarity between two vectors in a multi-dimensional space. These measures are crucial in applications like text analysis, information retrieval, and recommendation systems.

Definition:

- Measures the cosine of the angle between two non-zero vectors. It ranges from -1 (completely dissimilar) to 1 (completely similar).

  - Formula:

$$\text{Cosine Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

  where $\mathbf{A} \cdot \mathbf{B}$ is the dot product and $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ are the magnitudes of vectors $\mathbf{A}$ and $\mathbf{B}$.

Application in Movie Recommender System :

- In a movie recommender system, these similarity measures can compare the vector representations (e.g., BoW) of movie descriptions or user preferences to identify and recommend movies that are similar to those the user has liked or rated highly in the past.

In [57]:
```
# Importing cosine similarity it is between 0 to 1, 1-similar, 0-less similar
from sklearn.metrics.pairwise import cosine_similarity
```

In [58]:
```
#calculating vectors distance between each movie with other 4806 movies
cosine_similarity(vectors)
```

Out[58]:
```
array([[1.        , 0.08346223, 0.0860309 , ..., 0.04499213, 0.        ,
        0.        ],
       [0.08346223, 1.        , 0.06063391, ..., 0.02378257, 0.        ,
        0.02615329],
       [0.0860309 , 0.06063391, 1.        , ..., 0.02451452, 0.        ,
        0.        ],
       ...,
       [0.04499213, 0.02378257, 0.02451452, ..., 1.        , 0.03962144,
        0.04229549],
       [0.        , 0.        , 0.        , ..., 0.03962144, 1.        ,
        0.08714204],
       [0.        , 0.02615329, 0.        , ..., 0.04229549, 0.08714204,
        1.        ]])
```

In [59]:
```
cosine_similarity(vectors).shape
```

Out[59]:
```
(4806, 4806)
```

In [60]:
```
#storing
similarity=cosine_similarity(vectors)
```

In [61]:
```
#similarity metrics is distance of 1st movie with other 4806 movies
similarity[0]
```

Out[61]:
```
array([1.        , 0.08346223, 0.0860309 , ..., 0.04499213, 0.        ,
       0.        ])
```

In [62]:
```
#similarity metrics is array of arrays
similarity
```

```
Out[62]: array([[1.        , 0.08346223, 0.0860309 , ..., 0.04499213, 0.        ,
                 0.        ],
                [0.08346223, 1.        , 0.06063391, ..., 0.02378257, 0.        ,
                 0.02615329],
                [0.0860309 , 0.06063391, 1.        , ..., 0.02451452, 0.        ,
                 0.        ],
                ...,
                [0.04499213, 0.02378257, 0.02451452, ..., 1.        , 0.03962144,
                 0.04229549],
                [0.        , 0.        , 0.        , ..., 0.03962144, 1.        ,
                 0.08714204],
                [0.        , 0.02615329, 0.        , ..., 0.04229549, 0.08714204,
                 1.        ]])
```

```
In [63]: similarity.shape
```

```
Out[63]: (4806, 4806)
```

```
In [64]: #function for recommendation
def recommend(movie):
    movie_index=new_df[new_df['title'] == movie].index[0]
    distances=similarity[movie_index]
    movies_list=sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]

    for i in movies_list:
        print(new_df.iloc[i[0]].title)
```

```
In [65]: # Recommends top 5 similar movies
recommend('Batman Begins')
```

```
The Dark Knight
Batman
Batman
The Dark Knight Rises
10th & Wolf
```

```
In [66]: recommend('Avatar')
```

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

Pickle:

- Pickle is a standard library module in Python, simplifies the process of saving and loading complex data structures like lists, dictionaries, and custom objects, into a byte stream for storage or transmission and then reconstruct them back into their original form. This functionality is particularly useful for tasks like saving and loading machine learning models, caching data, or sharing objects between processes.

1. **Saving DataFrame as `movies.pkl`**:
   - I've chosen to store a DataFrame, labeled `new_df`, as `movies.pkl`.
   - This DataFrame typically holds comprehensive movie data like titles, genres, ratings, or descriptions.
   - By saving this DataFrame, I ensure swift access without the need to recompute or reload data from its original source.
2. **Saving DataFrame as Dictionary in `movie_dict.pkl`**:
   - I've transformed the DataFrame `new_df` into a dictionary format utilizing the `to_dict()` function.
   - Subsequently, I've retained this dictionary representation by saving it as `movie_dict.pkl`.
   - Storing the DataFrame as a dictionary proves advantageous for specific operations that benefit from dictionary-like access.
3. **Saving Similarity Matrix as `similarity.pkl`**:
   - I've archived a similarity matrix under the file name `similarity.pkl`.
   - This matrix typically encapsulates similarity scores between movies, typically calculated via a metric like cosine similarity.
   - The stored similarity matrix allows me to sidestep the necessity of recalculating movie similarities, particularly beneficial given their potentially intensive computational demands.

- Pickle facilitates the preservation of crucial data structures and outcomes within my movie recommender system, thereby streamlining future access and reuse across different project sessions.

```
In [67]: import pickle
```

```
In [69]: pickle.dump(new_df,open('movies.pkl','wb'))
```

```
In [72]: new_df['title'].values
```

Out[72]:
```
array(['Avatar', "Pirates of the Caribbean: At World's End", 'Spectre',
       ..., 'Signed, Sealed, Delivered', 'Shanghai Calling',
       'My Date with Drew'], dtype=object)
```

In [76]:
```
#converting the dataframe into dictionary
pickle.dump(new_df.to_dict(),open('movie_dict.pkl','wb'))
```

In [77]:
```
pickle.dump(similarity,open('similarity.pkl','wb'))
```

API:

- API (Application Programming Interface) is a set of rules facilitating communication between software components, enabling access to external data sources like movie databases. For example, the TMDB API provides movie-related information.

CSS:

- CSS (Cascading Style Sheets) is a language that dictates the presentation of HTML elements on web pages. It allows customization of appearance, including colors, fonts, and layout, in Streamlit apps. Additionally, CSS can be used to create animations and effects.

**TMDB:https://api.themoviedb.org/3/movie/%7Bmovie_id%7D?api_key=<<api_key>>&language=en-US

**Example: Give movie id=65 and API key = 8265bd1679663a7ea12ac168da84d2e8

https://api.themoviedb.org/3/movie/65?api_key=8265bd1679663a7ea12ac168da84d2e8&language=en-US

output is in JSON: A Specific **movie id=65** dictionary with all its key-value pairs.

{"adult":false,"backdrop_path":"/bfccQmQWNFQYRv4PHgCnjDu7PXn.jpg","belongs_to_collection":null,"budget":41000000,"genres":[{"id":18,"name":"Drama"},{"id":10402,"name":"Music"}],"homepage":"https://www.uphe.com/movies/8-mile%22,%22id%22:65,%22imdb_id%22:%22tt0298203%22,%22origin_country%22:%5B%22US%22%5D,%22original_language%22:%22e
Mile","overview":"For Jimmy Smith, Jr., life is a daily fight just to keep hope alive. Feeding his dreams in Detroit's vibrant music scene, Jimmy wages an extraordinary personal struggle to find his own voice - and earn a place in a world where rhymes rule, legends are born and every moment… is another chance.","popularity":66.294,"poster_path":"/7BmQj8qE1FLuLTf7Xjf9sdIHzoa.jpg","production_companies":[{"id":24,"logo_path":null,"name":"Mikona Productions","origin_country":"DE"},{"id":23,"logo_path":"/bJOFo2ufq7iFC1F4qZm8aLxF5aS.png","name":"Imagine Entertainment","origin_country":"US"}],"production_countries":[{"iso_3166_1":"DE","name":"Germany"},{"iso_3166_1":"US","name":"United States of America"}],"release_date":"2002-11-08","revenue":242875078,"runtime":111,"spoken_languages":[{"english_name":"English","iso_639_1":"en","name":"English"}],"status":"Released","tagline":"Every moment is another chance","title":"8 Mile","video":false,"vote_average":7.135,"vote_count":6993}

Pycharm code:

import streamlit as st import pickle import pandas as pd import requests

def fetch_poster(movie_id): response = requests.get('https://api.themoviedb.org/3/movie/%7B%7D?api_key=8265bd1679663a7ea12ac168da84d2e8&language=en-US%27.format(movie_id)) data = response.json()

```
    # tmdb image path + poster path = complete poster path
    return "https://image.tmdb.org/t/p/original" + data ['poster_path']
```

def recommend(movie): movie_index = movies[movies['title'] == movie].index[0] distances = similarity[movie_index] movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]

```
    recommended_movies=[]
    recommended_movies_posters=[]
    for i in movies_list:
        movie_id = movies.iloc[i[0]].movie_id

        recommended_movies.append(movies.iloc[i[0]].title)
        # fetch poster from API
        recommended_movies_posters.append(fetch_poster(movie_id))
    return recommended_movies, recommended_movies_posters
```

movies_dict = pickle.load(open('movie_dict.pkl','rb')) movies = pd.DataFrame(movies_dict)

similarity = pickle.load(open('similarity.pkl','rb'))

Custom CSS st.markdown ("""

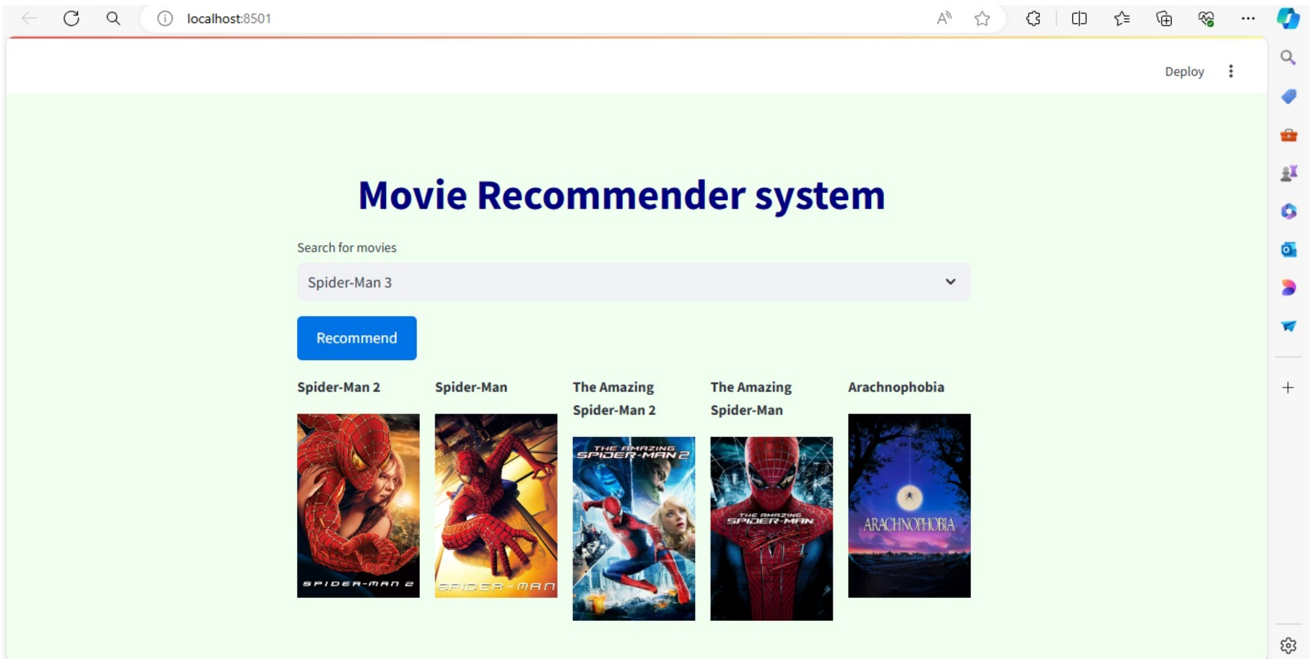
""", unsafe_allow_html=True)

st.title('Movie Recommender system')

selected_movie_name = st.selectbox("Search for movies", movies['title'].values)

if st.button("Recommend"): names,posters = recommend(selected_movie_name)

```
col1, col2, col3, col4, col5 = st.columns(5)

with col1:
    st.markdown(f'<p class="custom-text">{names[0]}</p>', unsafe_allow_html=True)
    st.image(posters[0])

with col2:
    st.markdown(f'<p class="custom-text">{names[1]}</p>', unsafe_allow_html=True)
    st.image(posters[1])

with col3:
    st.markdown(f'<p class="custom-text">{names[2]}</p>', unsafe_allow_html=True)
    st.image(posters[2])

with col4:
    st.markdown(f'<p class="custom-text">{names[3]}</p>', unsafe_allow_html=True)
    st.image(posters[3])

with col5:
    st.markdown(f'<p class="custom-text">{names[4]}</p>', unsafe_allow_html=True)
    st.image(posters[4])
```



Conclusion:

- The Content-Based Movie Recommender System leverages detailed movie attributes such as genre, cast, director, and plot to provide users with personalized movie recommendations. By focusing on the intrinsic characteristics of movies, the system effectively identifies and suggests films similar to a user's input, thereby enhancing the user's discovery experience.
- Throughout the development process, thorough data cleaning ensured the accuracy and consistency of the dataset, resulting in reliable recommendations. This recommender system demonstrates the potential of content-based filtering in delivering relevant and engaging movie suggestions, ultimately improving user satisfaction and engagement with the platform.
- By harnessing the rich information within the dataset, this system not only helps users find movies that match their preferences but also paves the way for future enhancements, such as integrating collaborative filtering techniques or incorporating user feedback to further refine recommendation accuracy.

Future Advancements:

1. **Exploring Advanced Algorithms**: Delve into deep learning or graph-based models to enhance recommendation accuracy.
2. **Enhancing Multi-criteria Recommendations**: Allow users to specify multiple preferences for a more tailored movie selection.
3. **Integrating Social Preferences**: Incorporate friends' movie preferences from social media platforms to enrich the recommendation

process.

4. **Expanding Content Diversity**: Broaden the movie database by including niche genres and international films, catering to diverse tastes.
5. **Implementing Dynamic Updates**: Reflect real-time changes in preferences by dynamically updating recommendations based on interactions.
6. **Adding User Engagement Features**: Enhance user engagement by incorporating gamification elements to make the movie discovery process more interactive.
7. **Cross-domain Recommendations**: Expand beyond movies and recommend other forms of entertainment like TV shows, music, and more for a comprehensive entertainment experience.

Thank you 😊