


```
from google.colab import files
data=files.upload()
```



Choose Files

 No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving final rating[1].csv to final rating[1].csv

```
import pandas as pd
import numpy as np

movie=pd.read_csv('main_data (1).csv')
movie.head()
```

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title	comb
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action Adventure Fantasy Sci-Fi	avatar	CCH Pounder Joel David Moore Wes Studi James C...
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action Adventure Fantasy	pirates of the caribbean: at world's end	Johnny Depp Orlando Bloom Jack Davenport Gore ...
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action Adventure Thriller	spectre	Christoph Waltz Rory Kinnear Stephanie Sigman ...
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action Thriller	the dark knight rises	Tom Hardy Christian Bale Joseph Gordon-Levitt ...

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

cv = CountVectorizer()
count_matrix = cv.fit_transform(movie['comb'])
similarity = cosine_similarity(count_matrix)

def recommend(movie_name):
    index = movie[movie['movie_title'] == movie_name].index[0]
    lst = list(enumerate(similarity[index]))
    lst = sorted(lst, key=lambda x: x[1], reverse=True)
    lst = lst[1:11] # excluding the first item since it is the requested movie itself
    recommended_movies = []

    for i in range(len(lst)):
        a = lst[i][0]
        recommended_movies.append(movie['movie_title'][a])

    dff = movie[movie['movie_title'].isin(recommended_movies)]

    for index, row in dff.iterrows():
        print("Director:", row['director_name'])
        print("Genres:", row['genres'])
        print("Movie Title:", row['movie_title'])
        print("----")

recommend('avatar')
```

```
Director: Marc Webb
Genres: Action Adventure Fantasy Sci-Fi
Movie Title: the amazing spider-man 2
----
Director: Roland Emmerich
Genres: Action Adventure Sci-Fi
Movie Title: 2012
----
Director: James Cameron
Genres: Action Sci-Fi
Movie Title: terminator 2: judgment day
----
Director: Kinka Usher
Genres: Action Comedy Fantasy Sci-Fi
Movie Title: mystery men
----
Director: David S. Goyer
Genres: Action Adventure Fantasy Horror Sci-Fi Thriller
```

```

Movie Title: blade: trinity
----
Director: Stephen Sommers
Genres: Action Adventure Horror Sci-Fi
Movie Title: deep rising
----
Director: James Wong
Genres: Action Adventure Fantasy Sci-Fi Thriller
Movie Title: dragonball: evolution
----
Director: James Cameron
Genres: Action Adventure Sci-Fi
Movie Title: aliens
----
Director: unknown
Genres: Action Adventure Animation Drama Fantasy Sci-Fi
Movie Title: star wars: the clone wars
----
Director: James Cameron
Genres: Action Sci-Fi
Movie Title: the terminator
----

```

```
dff=movie[movie['movie_title'].isin(recommended_movies)]
```

```
dff['movie_title']
```

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
33	Marc Webb	Emma Stone	Andrew Garfield	B.J. Novak	Action Adventure Fantasy Sci-Fi	the amazing spider-man 2
52	Roland Emmerich	Oliver Platt	Liam James	Tom McCarthy	Action Adventure Sci-Fi	2012
260	James Cameron	Joe Morton	Jenette Goldstein	S. Epatha Merkerson	Action Sci-Fi	terminator 2: judgment day
595	Kinka Usher	Janeane Garofalo	Wes Studi	Eddie Izzard	Action Comedy Fantasy	mystery men

```

import pandas as pd
import streamlit as st

# Load the movie dataset
st.subheader('Welcome to the Movie Recommendation System')
try:
    movie = pd.read_csv('main_data (1).csv')
except FileNotFoundError:
    st.error("Dataset file not found. Please check the file path.")

# Define a function to get movies by a specific director
def get_movies_by_director(director_name):
    director_name = director_name.strip() # Remove leading/trailing whitespaces
    movies = movie[movie['director_name'].str.contains(director_name, case=False)]
    return movies['movie_title'].tolist()

# Get user input for director name
director_name_input = st.text_input('Enter Director Name:')
director_name_input = director_name_input.title() # Convert to title case for case-insensitive matching

# Display the button
if st.button('Click to Get Movies'):
    # Call the function only if a director name is provided and the button is clicked
    if director_name_input:
        movies_by_director = get_movies_by_director(director_name_input)

        if movies_by_director:
            st.write(movies_by_director)

```

```
        st.write(f"Movies directed by {director_name_input}:")
        for movie_title in movies_by_director:
            st.write(movie_title)
    else:
        st.warning(f"No movies found for director {director_name_input}.")
else:
    st.warning("Please enter a director name.")
```