Preface:

I went to do the mechanics of Netflix movie recommendations, let's say if I want to go to Amazon to buy a phone, let's say I search for phone Iphone 14, then the next search will bring up related associations like iphone 14 pro, iphone 14 pro max, iphone 13 etc. So how does Amazon know that I am interested in the iphone banner.
So based on my behaviour, (my behavior), my interest, and even based on my purchase history on Amazon. There are two requirement systems here, the first becomes a content library and the other is called a collaborative filter, which contains user interests, user behaviour and user reads. The content I produce is based on the collaborative filter, let's say a user is watching an action movie on Netflix, it will push similar based on tags, for example if a user watches the movie Avengers, then Netflix will push Spider-Man, Iron Man, because these are action movies. The second point is that when user A gives a movie or a game a positive review, these will be recommended to user B because user B's type of movie preference is similar to user A's.

Jupter

You can see the total number of 10,000, the average value of 25%, the maximum value of 934,761 and the minimum value of 5. Check the total number of datasets and the classification. There are films in Hindi, English, etc. but language is not a major influence on viewing, so only id, title, genre, overview are needed.

In [20]:

```
movies.describe()
```

Out[20]:

	id	popularity	vote_average	vote_count
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	161243.505000	34.697267	6.621150	1547.309400
std	211422.046043	211.684175	0.766231	2648.295789
min	5.000000	0.600000	4.600000	200.000000
25%	10127.750000	9.154750	6.100000	315.000000
50%	30002.500000	13.637500	6.600000	583.500000
75%	310133.500000	25.651250	7.200000	1460.000000
max	934761.000000	10436.917000	8.700000	31917.000000

In [25]:

```
movies[['id','title','overview','genre']]
```

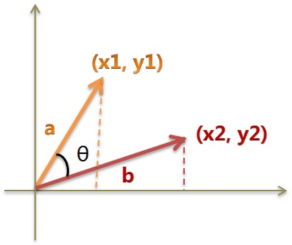
Out[25]:

	id	title	overview	genre
0	278	The Shawshank Redemption	Framed in the 1940s for the double murder of h...	Drama,Crime
1	19404	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...	Comedy,Drama,Romance
2	238	The Godfather	Spanning the years 1945 to 1955, a chronicle o...	Drama,Crime

Collection characteristics

Step 2, using the sklearn module, set up the stop word to detect english movies in 10000 data, then set up to use the cosine algorithm to set up the calculation of similarity, when we load the dataset here a red 10000, we fit it with the correct count vectorizer, this means that 10000 vectors in the dimension are created. Suppose there is a mobi movie that is an action movie and another movie that is also an action movie and most action movies are similar having the lowest angle, so this is also called cosine similarity.
So in a spatial axis of x, y, z, with 10,000 movies placed on the spatial axis, the smaller the distance between each movie, the more similar the movies are. The recommended similarity is similar to the cosine algorithm, so import the cosine similarity.

$$\cos\theta = \frac{\sum_{i=1}^{n}(A_i \times B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} = \frac{A \cdot B}{|A| \times |B|}$$



In [50]:

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

In [51]:

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

```
array([[1.        , 0.05634362, 0.12888482, ..., 0.07559289, 0.11065667,
        0.06388766],
       [0.05634362, 1.        , 0.07624929, ..., 0.        , 0.03636965,
        0.        ],
       [0.12888482, 0.07624929, 1.        , ..., 0.02273314, 0.06655583,
        0.08645856],
       ...,
       [0.07559289, 0.        , 0.02273314, ..., 1.        , 0.03253   ,
        0.02817181],
       [0.11065667, 0.03636965, 0.06655583, ..., 0.03253   , 1.        ,
        0.0412393 ],
       [0.06388766, 0.        , 0.08645856, ..., 0.02817181, 0.0412393 ,
        1.        ]])
```

Similarity data

```
sorted(list(enumerate(similarity[2])), reverse=True, key=lambda vector:vector[1])
```

```
[(2, 1.0000000000000004),
 (4, 0.48976229911514363),
 (7419, 0.3521803625302496),
```

Then we obtained 2,1.000, got 100 cosine similarities and got to 4.8,so set a search range of 0 to 5.

Out[18]:

	id	title	genre	original_l
0	278	The Shawshank Redemption	Drama,Crime	
1	19404	Dilwale Dulhania Le Jayenge	Comedy,Drama,Romance	
2	238	The Godfather	Drama,Crime	

In [72]:

```
def recommand(movies):
    index = new_data[new_data['title']==movies].index
    distance = sorted(list(enumerate(similarity[index
    for i in distance[0:5]:
        print(new_data.iloc[i[0]].title)
```

```
recommand("Iron Man")
```

```
Iron Man
Iron Man 3
Guardians of the Galaxy Vol. 2
Avengers: Age of Ultron
Star Wars: Episode III - Revenge of the Sith
```

Search for relevant recommended movies to get Iron Man. Then import pickle and call similarity rights To use this on a web page.

In [76]:
```
pickle.dump(new_data,open('movies_list.pkl','wb'))
```

In [79]:
```
pickle.dump(similarity,open('similarity.pkl','wb'))
```

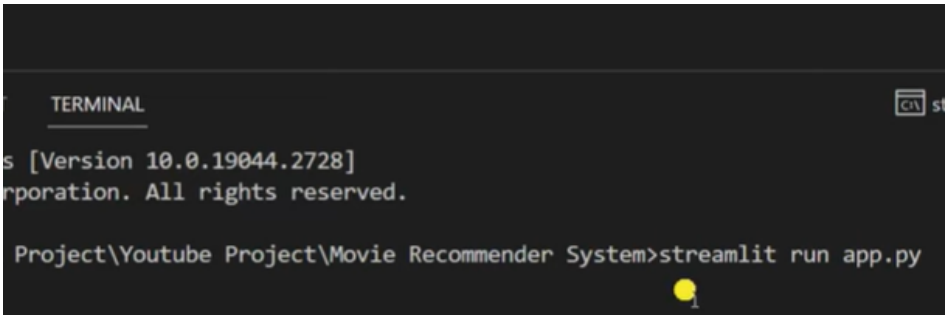
In [80]:
```
pickle.load(open('movies_list.pkl','rb'))
```

Out[80]:

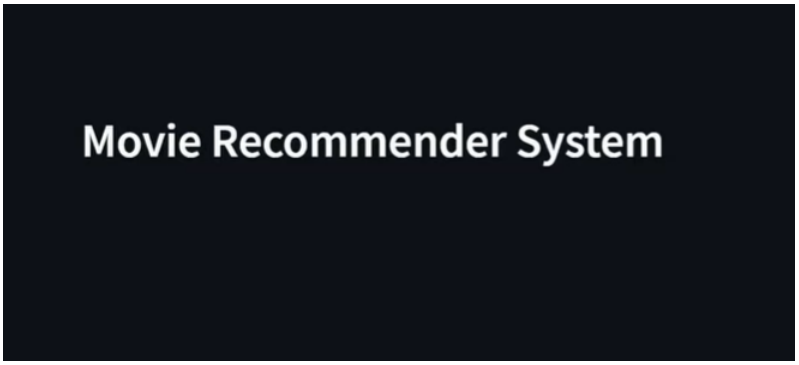
	id	title	original_language	popularity	release_date	vote_average	vote_count	
0	278	The Shawshank Redemption	en	94.075	1994-09-23	8.7	21862	th
1	19404	Dilwale Dulhania Le Jayenge	hi	25.408	1995-10-19	8.7	3731	R
2	238	The Godfather	en	90.585	1972-03-14	8.7	16280	cr

Web

writing web code.



Open the program on a local server or in any browser.

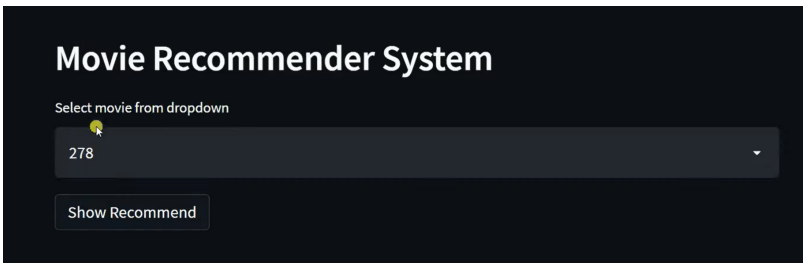


Open for web movie recommendation system.

```
st.header("Movie Recommender System")
```

```
if st.button("Show Recommend"):
    movie_name, movie_poster = recommend(selectvalue)
    col1,col2,col3,col4,col5=st.columns(5)
    with col1:
```

Set drop down menu, button.



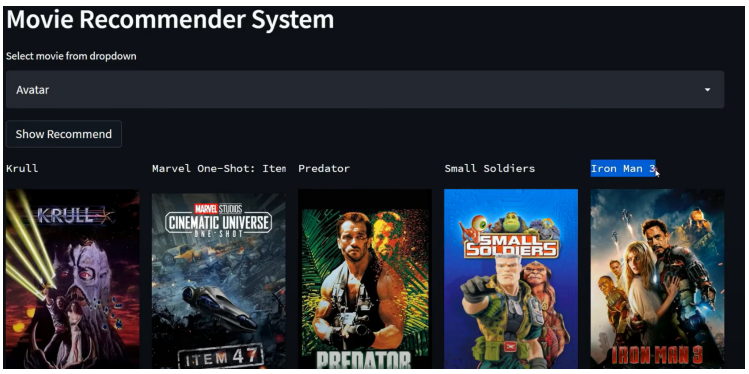

```
def recommend(movie):
    index=movies[movies['title']==movie].index[0]
    distance = sorted(list(enumerate(similarity[index])),
reverse=True, key=lambda vector:vector[1])
    recommend_movie=[]
    recommend_poster=[]
    for i in distance[1:6]:
        movies_id=movies.iloc[i[0]].id
        recommend_movie.append(movies.iloc[i[0]].title)
        recommend_poster.append(fetch_poster(movies_id))
    return recommend_movie, recommend_poster
```

Load the similarities correctly, search for 5 recommendations, then return when the search is complete.Set five index numbers there on the button.

```
if st.button("Show Recommend"):
    movie_name, movie_poster = recommend(selectvalue)
    col1,col2,col3,col4,col5=st.columns(5)
```

Next, associate the index with the relevant image, set the movie poster as a button and end.

```
with col1:
    st.text(movie_name[0])
    st.image(movie_poster[0])
```


Movie Recommender System
Select movie from dropdown
Avatar
Show Recommend
Krull
Marvel One-Shot: Item
Predator
Small Soldiers
Iron Man 3
KRULL
MARVEL STUDIOS
CINEMATIC UNIVERSE
ONE-SHOT
ITEM 47
PREDATOR
SMALL SOLDIERS
IRON MAN 3

出现的(2, 1.0)就是教父本身