In [1]:
```
# 1: Import Required Libraries

import warnings
import kagglehub as kg
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

warnings.filterwarnings("ignore")
```

In [2]:
```
# 2: Importing the Dataset

# Download from kaggle
path = kg.dataset_download("harshitshankhdhar/imdb-dataset-of-top-1000-movies-an

#Making the dataframe
series_df = pd.read_csv(path + '/imdb_top_1000.csv')
series_df.head()
```

Out[2]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runt
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152
3	https://m.media-amazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	A	202
4	https://m.media-amazon.com/images/M/MV5BMWU4N2...	12 Angry Men	1957	U	96

In [3]:
```
# 3. Preprocess Text (Overview Field)

# Fill missing overviews with empty string
series_df['Overview'] = series_df['Overview'].fillna('')

# TF-IDF Vectorization on Overview
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(series_df['Overview'])
```

In [4]:
```
# 4. Compute Cosine Similarity

cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

In [5]:
```
# 5. Recommend Top-10 Similar Series for a Given Index

# Function to recommend top N similar TV Series
def recommend_series(title, top_n=10):
    if title not in series_df['Series_Title'].values:
        return "Series not found."

    idx = series_df[series_df['Series_Title'] == title].index[0]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Exclude the series itself and pick top_n
    top_similar = sim_scores[1:top_n+1]

    result = []
    for i, score in top_similar:
        result.append((series_df['Series_Title'][i], score))

    return pd.DataFrame(result, columns=['Recommended Series', 'Similarity Score
```

In [6]:
```
# 6: Choose a Sentence to Recommend From

# Choose the sentence index to find recommendations for
recommend_from =  series_df.iloc[0 , 7] # 0-based index
recommend_from
```

Out[6]:
```
'Two imprisoned men bond over a number of years, finding solace and eventual re
demption through acts of common decency.'
```

In [7]:
```
# 7. Get Recommendations

recommendations = recommend_series("The Invisible Man")
recommendations
```

Out[7]:

	Recommended Series	Similarity Score
0	Harvey	0.201524
1	Young Frankenstein	0.168055
2	The Butterfly Effect	0.146863
3	The Long Goodbye	0.123039
4	Rogue One	0.116654
5	Trois couleurs: Bleu	0.115169
6	Swades: We, the People	0.106805
7	Shutter Island	0.101418
8	Solaris	0.099157
9	Badhaai ho	0.090087