In [ ]:

```
pip install numpy
pip install scikit-surprise
```

In [1]:

```
import pandas as pd
import numpy as np
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
```

In [2]:

```
books = pd.read_csv("D:/RISE - WPU/Internship/Hybrid/FinalData.csv")
books.head()
```

Out[2]:

	book_id	authors	title	Genres
0	1	Suzanne Collins	The Hunger Games (The Hunger Games, #1)	SciFi;Drama
1	2	J.K. Rowling, Mary GrandPré	Harry Potter and the Sorcerer's Stone (Harry P...	Fantasy;Young-Age
2	3	Stephenie Meyer	Twilight (Twilight, #1)	Fantasy
3	4	Harper Lee	To Kill a Mockingbird	Self-Help;Drama
4	5	F. Scott Fitzgerald	The Great Gatsby	Drama

In [22]:

```
books['Genres'] = books['Genres'].fillna('')
tf = TfidfVectorizer(ngram_range=(1, 2), min_df=0, stop_words='english')
```

In [23]:

```
tfidf_matrix = tf.fit_transform(books['Genres'])
tfidf_matrix.shape
```

Out[23]:

```
(999, 187)
```

In [24]:

```
cosine_sim = linear_kernel(tfidf_matrix)
cosine_sim
```

Out[24]:

```
array([[1.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 1.        , 0.27647653, ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.27647653, 1.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 1.        , 0.09738819,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.09738819, 1.        ,
        0.44573825],
       [0.        , 0.        , 0.        , ..., 0.        , 0.44573825,
        1.        ]])
```

In [29]:

```
def recommend(title):
    book_rec = books[books['title'] == title]
    if len(book_rec) > 1:
        print("There are duplications of same name. Choose index and use get_recommenda
tions(idx)")
        print(book_rec)
    else:
        indexes = get_recommendations(book_rec.index[0])
        recommend_books = books.iloc[indexes]
        return recommend_books[1:].set_index('book_id')


def get_recommendations(idx):
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    return [i[0] for i in sim_scores if i[1] > 0.01]
```

In [30]:

```
recommend('The Great Gatsby').head(10)
```

Out[30]:

book_id	authors	title	Genres
14	George Orwell	Animal Farm	Drama
115	Jeffrey Eugenides	Middlesex	Drama
123	John Grisham	The Firm (Penguin Readers, Level 5)	Drama
147	Jay Asher	Thirteen Reasons Why	Drama
250	R.J. Palacio	Wonder	Drama
268	Kazuo Ishiguro	Never Let Me Go	Drama
305	Stephen King	Pet Sematary	Drama
306	Wally Lamb	She's Come Undone	Drama
311	Colleen McCullough	The Thorn Birds	Drama
333	Lisa Genova	Still Alice	Drama

In [ ]:

```
from surprise import SVD

algo = SVD()
id_map = pd.read_csv("D:/RISE - WPU/Internship/Hybrid/AverageRatings.csv")

def hybrid(book_id, title):
    book = recommend(title)
    book['est'] = [algo.predict(userid, id_map.loc[x]['book_id']).est for x in book.ind
ex]
    book = book.sort_values('est', ascending=False)
    return book.head(10)
```

In [ ]:

```
hybrid(1,'The Great Gatsby')
```