In [1]:
```
import pandas as pd
#reading our data with pandas
movies = pd.read_csv("movies.csv")
```

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

Out[2]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [3]:
```
import re
#cleaning movie titles with RegEx
def clean_title(title):
    title = re.sub("[^a-zA-Z0-9 ]", "", title)
    return title
```

In [4]:
```
movies["clean_title"] = movies["title"].apply(clean_title)
```

In [5]:
```
movies
```

Out[5]:

	movieId	title	genres	clean_title
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	Toy Story 1995
1	2	Jumanji (1995)	Adventure|Children|Fantasy	Jumanji 1995
2	3	Grumpier Old Men (1995)	Comedy|Romance	Grumpier Old Men 1995
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	Waiting to Exhale 1995
4	5	Father of the Bride Part II (1995)	Comedy	Father of the Bride Part II 1995
...	...	...	...	...
62418	209157	We (2018)	Drama	We 2018
62419	209159	Window of the Soul (2001)	Documentary	Window of the Soul 2001
62420	209163	Bad Poems (2018)	Comedy|Drama	Bad Poems 2018
62421	209169	A Girl Thing (2001)	(no genres listed)	A Girl Thing 2001
62422	209171	Women of Devil's Island (1962)	Action|Adventure|Drama	Women of Devils Island 1962

62423 rows × 4 columns

In [6]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(1,2))

tfidf = vectorizer.fit_transform(movies["clean_title"])
```

In [7]:

```
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

def search(title):
    title = clean_title(title)
    query_vec = vectorizer.transform([title])
    similarity = cosine_similarity(query_vec, tfidf).flatten()
    indices = np.argpartition(similarity, -5)[-5:]
    results = movies.iloc[indices].iloc[::-1]

    return results
```

In [8]:
```
# pip install ipywidgets
#jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

In [9]:
```
#building an intergface search box with jupyter
import ipywidgets as widgets
from IPython.display import display

movie_input = widgets.Text(
    value='Toy Story',
    description='Movie Title:',
    disabled=False
)
movie_list = widgets.Output()

def on_type(data):
    with movie_list:
        movie_list.clear_output()
        title = data["new"]
        if len(title) > 5:
            display(search(title))

movie_input.observe(on_type, names='value')


display(movie_input, movie_list)
```

```
Text(value='Toy Story', description='Movie Title:')
Output()
```

In [10]:
```
movie_id = 89745

#def find_similar_movies(movie_id):
movie = movies[movies["movieId"] == movie_id]
```

In [11]:
```
ratings = pd.read_csv("ratings.csv")
```

In [12]:
```
ratings.dtypes
```

Out[12]:
```
userId         int64
movieId        int64
rating       float64
timestamp      int64
dtype: object
```

In [13]:
```
#finding users who liked the same movies
similar_users = ratings[(ratings["movieId"] == movie_id) & (ratings["rating"] > 4)]
similar_users
```

Out[13]:
```
array([    21,    187,    208, ..., 162469, 162485, 162532], dtype=int64)
```

In [14]:
```
similar_user_recs = ratings[(ratings["userId"].isin(similar_users)) & (ratings["rat
similar_user_recs
```

Out[14]:
```
3741          318
3742          527
3743          541
3744          589
3745          741
             ...
24998517    91542
24998518    92259
24998522    98809
24998523   102125
24998524   112852
Name: movieId, Length: 577796, dtype: int64
```

In [15]:
```
similar_user_recs = similar_user_recs.value_counts() / len(similar_users)

similar_user_recs = similar_user_recs[similar_user_recs > .10]
similar_user_recs
```

Out[15]:
```
movieId
89745    1.000000
58559    0.573393
59315    0.530649
79132    0.519715
2571     0.496687
           ...
47610    0.103545
780      0.103380
88744    0.103048
1258     0.101226
1193     0.100895
Name: count, Length: 193, dtype: float64
```

In [16]:
```
#finding how much all users like movies
all_users = ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratings["
```

In [17]:
```
all_user_recs = all_users["movieId"].value_counts() / len(all_users["userId"].uniqu
all_user_recs
```

Out[17]:
```
movieId
318       0.346395
296       0.288146
2571      0.247010
356       0.238136
593       0.228665
            ...
86332     0.010142
91630     0.009324
122900    0.008573
122926    0.008070
106072    0.005289
Name: count, Length: 193, dtype: float64
```

In [18]:
```
rec_percentages = pd.concat([similar_user_recs, all_user_recs], axis=1)
rec_percentages.columns = ["similar", "all"]
```

In [19]:
```
rec_percentages
```

Out[19]:

	similar	all
movieId		
89745	1.000000	0.040459
58559	0.573393	0.148256
59315	0.530649	0.054931
79132	0.519715	0.132987
2571	0.496687	0.247010
...	...	...
47610	0.103545	0.022770
780	0.103380	0.054723
88744	0.103048	0.010383
1258	0.101226	0.083887
1193	0.100895	0.120244

193 rows × 2 columns

In [20]:
```
rec_percentages["score"] = rec_percentages["similar"] / rec_percentages["all"]
```

In [21]:
```
rec_percentages = rec_percentages.sort_values("score", ascending=False)
```

In [22]:
```
rec_percentages.head(10).merge(movies, left_index=True, right_on="movieId")
```

Out[22]:

	similar	all	score	movieId	title	genr
17067	1.000000	0.040459	24.716368	89745	Avengers, The (2012)	Action|Adventure|Sci-Fi|IM/
20513	0.103711	0.005289	19.610199	106072	Thor: The Dark World (2013)	Action|Adventure|Fantasy|IM/
25058	0.241054	0.012367	19.491770	122892	Avengers: Age of Ultron (2015)	Action|Adventure|Sci-
19678	0.216534	0.012119	17.867419	102125	Iron Man 3 (2013)	Action|Sci-Fi|Thriller|IM/
16725	0.215043	0.012052	17.843074	88140	Captain America: The First Avenger (2011)	Action|Adventure|Sci-Fi|Thriller|W
16312	0.175447	0.010142	17.299824	86332	Thor (2011)	Action|Adventure|Drama|Fantasy|IM/
21348	0.287608	0.016737	17.183667	110102	Captain America: The Winter Soldier (2014)	Action|Adventure|Sci-Fi|IM/
25071	0.214049	0.012856	16.649399	122920	Captain America: Civil War (2016)	Action|Sci-Fi|Thrill
25061	0.136017	0.008573	15.865628	122900	Ant-Man (2015)	Action|Adventure|Sci-
14628	0.242876	0.015517	15.651921	77561	Iron Man 2 (2010)	Action|Adventure|Sci-Fi|Thriller|IM/

In [23]:

```
#building a recommendation function
def find_similar_movies(movie_id):
    similar_users = ratings[(ratings["movieId"] == movie_id) & (ratings["rating"] >
    similar_user_recs = ratings[(ratings["userId"].isin(similar_users)) & (ratings[
    similar_user_recs = similar_user_recs.value_counts() / len(similar_users)

    similar_user_recs = similar_user_recs[similar_user_recs > .10]
    all_users = ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratin
    all_user_recs = all_users["movieId"].value_counts() / len(all_users["userId"].u
    rec_percentages = pd.concat([similar_user_recs, all_user_recs], axis=1)
```

```
    rec_percentages.columns = ["similar", "all"]

    rec_percentages["score"] = rec_percentages["similar"] / rec_percentages["all"]
    rec_percentages = rec_percentages.sort_values("score", ascending=False)
    return rec_percentages.head(10).merge(movies, left_index=True, right_on="movieI
```

In [24]:
```
import ipywidgets as widgets
from IPython.display import display

movie_name_input = widgets.Text(
    value='Toy Story',
    description='Movie Title:',
    disabled=False
)
recommendation_list = widgets.Output()

def on_type(data):
    with recommendation_list:
        recommendation_list.clear_output()
        title = data["new"]
        if len(title) > 5:
            results = search(title)
            movie_id = results.iloc[0]["movieId"]
            display(find_similar_movies(movie_id))

movie_name_input.observe(on_type, names='value')

display(movie_name_input, recommendation_list)
```

```
Text(value='Toy Story', description='Movie Title:')
Output()
```