```
import pandas as pd
```

```
movies=pd.read_csv('/content/movie.csv')
```

```
movies
```

	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
...	...	...	...
27273	131254	Kein Bund für's Leben (2007)	Comedy
27274	131256	Feuer, Eis & Dosenbier (2002)	Comedy
27275	131258	The Pirates (2014)	Adventure
27276	131260	Rentun Ruusu (2001)	(no genres listed)
27277	131262	Innocence (2014)	Adventure|Fantasy|Horror

27278 rows × 3 columns

```
movies.isnull().sum()
```

	0
movieId	0
title	0
genres	0

dtype: int64

```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27278 entries, 0 to 27277
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   movieId  27278 non-null  int64 
 1   title    27278 non-null  object
 2   genres   27278 non-null  object
dtypes: int64(1), object(2)
```

```
memory usage: 639.5+ KB
```

```
import re
def clean_title(title):
  return re.sub("[^a-zA-Z0-9]"," ",title)
```

```
movies["clean_title"]=movies["title"].apply(clean_title)
movies.head(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

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer=TfidfVectorizer(ngram_range=(1,2))
tfidf=vectorizer.fit_transform(movies["clean_title"])
```

```
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][::-1]
  return results
```

```
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)
```

Movie Title : Toy Story

	movieId	title	genres	clean_title
15401	78499	Toy Story 3 (2010)	Adventure|Animation|Children|Comedy|Fantasy|IMAX	Toy Story 3 2010
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	Toy Story 1995
3027	3114	Toy Story 2 (1999)	Adventure|Animation|Children|Comedy|Fantasy	Toy Story 2 1999
21981	106022	Toy Story of Terror (2013)	Animation|Children|Comedy	Toy Story of Terror 2013
25463	120474	Toy Story That Time Forgot (2014)	Animation|Children	Toy Story That Time Forgot 2014

```
ratings=pd.read_csv("/content/rating.csv")
ratings
```

	userId	movieId	rating	timestamp
0	1	2	3.5	2005-04-02 23:53:47
1	1	29	3.5	2005-04-02 23:31:16
2	1	32	3.5	2005-04-02 23:33:39
3	1	47	3.5	2005-04-02 23:32:07
4	1	50	3.5	2005-04-02 23:29:40
...	...	...	...	...
1637252	11074	5995	4.0	2004-08-11 12:44:28
1637253	11074	6003	3.0	2004-12-04 20:20:57
1637254	11074	6016	4.0	2004-08-08 09:24:25
1637255	11074	6045	2.0	2004-09-11 17:15:05
1637256	11074	6078	2.0	NaN

1637257 rows × 4 columns

```
ratings.dtypes
```

	0
userId	int64
movieId	int64
rating	float64
timestamp	object

dtype: object

```
ratings.isnull().sum()
```

	0
userId	0
movieId	0
rating	0
timestamp	1

dtype: int64

```
movie_id=1
```

```
similar_users=ratings[(ratings["movieId"]==movie_id) & (ratings["rating"]>4)]["userId"].unique()
```

```
similar_users
```

```
array([    6,    11,    14, ..., 11034, 11050, 11063])
```

```
similar_user_recs=ratings[(ratings["userId"].isin(similar_users))&(ratings["rating"]>4)]["movieId"]
```

```
similar_user_recs
```

	movieId
517	1
519	7
520	17
521	52
522	62
...	...
1635304	1064
1635306	1566
1635309	1917
1635323	56174
1635324	78499

87482 rows × 1 columns

dtype: int64

```
similar_user_recs=similar_user_recs.value_counts()/len(similar_users)
similar_user_recs=similar_user_recs[similar_user_recs>.1]
```

```
similar_user_recs
```

movieId	count
1	1.000000
318	0.424127
260	0.400607
296	0.366464
356	0.345979
...	...
1732	0.102428
733	0.101669
1307	0.101669
1610	0.100152
1527	0.100152

106 rows × 1 columns

dtype: float64

```
all_users=ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratings["rating"]>4)]
```

```
all_users_recs=all_users["movieId"].value_counts()/len(all_users["userId"].unique())
```

```
all_users_recs
```

movieId	count
318	0.321211
296	0.277177
593	0.233045
356	0.229998
527	0.225378
...	...
2804	0.046983
1079	0.046589
2791	0.045213
745	0.044918
2355	0.026538

106 rows × 1 columns

dtype: float64

```
rec_percentages=pd.concat([similar_user_recs,all_users_recs],axis=1)
rec_percentages.columns=["similar","all"]
```

```
rec_percentages
```

movieId	similar	all
1	1.000000	0.129546
318	0.424127	0.321211
260	0.400607	0.223118
296	0.366464	0.277177
356	0.345979	0.229998
...	...	...
1732	0.102428	0.066051
733	0.101669	0.050423
1307	0.101669	0.050914
1610	0.100152	0.048653
1527	0.100152	0.058679

106 rows × 2 columns

```
rec_percentages["score"]=rec_percentages["similar"]/rec_percentages["all"]
rec_percentages=rec_percentages.sort_values("score",ascending=False)
rec_percentages
```

movieId	similar	all	score
1	1.000000	0.129546	7.719272
3114	0.263278	0.055730	4.724140
2355	0.118361	0.026538	4.460024
4886	0.175266	0.057500	3.048123
745	0.132018	0.044918	2.939066
...	...	...	...
2858	0.230653	0.176430	1.307331
1206	0.111533	0.085905	1.298321
593	0.301214	0.233045	1.292514
2959	0.213961	0.173973	1.229850
4973	0.113809	0.101238	1.124166

106 rows × 3 columns

```
rec_percentages.head(10).merge(movies,left_index=True,right_on="movieId")
```

	similar	all	score	movieId	title	genres	clean_title
0	1.000000	0.129546	7.719272	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	Toy Story 1995
3027	0.263278	0.055730	4.724140	3114	Toy Story 2 (1999)	Adventure|Animation|Children|Comedy|Fantasy	Toy Story 2 1999
2270	0.118361	0.026538	4.460024	2355	Bug's Life, A (1998)	Adventure|Animation|Children|Comedy	Bug s Life A 1998
4790	0.175266	0.057500	3.048123	4886	Monsters, Inc. (2001)	Adventure|Animation|Children|Comedy|Fantasy	Monsters Inc 2001
732	0.132018	0.044918	2.939066	745	Wallace & Gromit: A Close Shave (1995)	Animation|Children|Comedy	Wallace Gromit A Close Shave 1995
6271	0.174507	0.059760	2.920119	6377	Finding Nemo (2003)	Adventure|Animation|Children|Comedy	Finding Nemo 2003
8278	0.160850	0.056713	2.836197	8961	Incredibles, The (2004)	Action|Adventure|Animation|Children|Comedy	Incredibles The 2004
360	0.230653	0.081384	2.834129	364	Lion King, The (1994)	Adventure|Animation|Children|Drama|Musical|IMAX	Lion King The 1994
582	0.210167	0.074307	2.828357	588	Aladdin (1992)	Adventure|Animation|Children|Comedy|Musical	Aladdin 1992
2630	0.151745	0.056615	2.680303	2716	Ghostbusters (a.k.a. Ghost Busters) (1984)	Action|Comedy|Sci-Fi	Ghostbusters a k a Ghost Busters 1984

```
def find_similar_movies(movie_id):
  similar_users=ratings[(ratings["movieId"]==movie_id)&(ratings["rating"]>4)]["userId"].unique()
  similar_user_recs=ratings[(ratings["userId"].isin(similar_users)) & (ratings["rating"]>4)]["movieId"]

  similar_user_recs=similar_user_recs.value_counts()/len(similar_users)
```

```
  similar_user_recs=similar_user_recs[similar_user_recs>0.10]

  all_users=ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratings["rating"] >4)]
  all_users_recs=all_users["movieId"].value_counts() / len(all_users["userId"].unique())

  rec_percentages=pd.concat([similar_user_recs,all_users_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="movieId")[["score","title","genres"]]
```

```
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)
```

Movie Title : harry potter

	score	title	genres
4800	31.732308	Harry Potter and the Sorcerer's Stone (a.k.a. ...	Adventure|Children|Fantasy
5717	23.367724	Harry Potter and the Chamber of Secrets (2002)	Adventure|Fantasy
10600	21.154872	Harry Potter and the Goblet of Fire (2005)	Adventure|Fantasy|Thriller|IMAX
11974	20.280046	Harry Potter and the Order of the Phoenix (2007)	Adventure|Drama|Fantasy|IMAX
13935	18.054589	Harry Potter and the Half-Blood Prince (2009)	Adventure|Fantasy|Mystery|Romance|IMAX
16191	16.500800	Harry Potter and the Deathly Hallows: Part 1 (...	Action|Adventure|Fantasy|IMAX
7769	16.237602	Harry Potter and the Prisoner of Azkaban (2004)	Adventure|Fantasy|IMAX
17499	13.139159	Harry Potter and the Deathly Hallows: Part 2 (...	Action|Adventure|Drama|Fantasy|Mystery|IMAX
10643	12.509852	Chronicles of Narnia: The Lion, the Witch and ...	Adventure|Children|Fantasy
12002	8.855528	Stardust (2007)	Adventure|Comedy|Fantasy|Romance

```
Start coding or generate with AI.
```