In [8]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
```

In [4]:
```
import kagglehub
path = kagglehub.dataset_download("rounakbanik/the-movies-dataset")
print("Path to dataset files:", path)
```

```
Warning: Looks like you're using an outdated `kagglehub` version, please consider
updating (latest version: 0.3.3)
Path to dataset files: C:\Users\dipto\.cache\kagglehub\datasets\rounakbanik\the-m
ovies-dataset\versions\7
```

In [6]:
```
md = pd. read_csv('C:/Users/dipto/.cache/kagglehub/datasets/rounakbanik/the-movi
md.head()
```

```
C:\Users\dipto\AppData\Local\Temp\ipykernel_5144\3706581673.py:1: DtypeWarning: C
olumns (10) have mixed types. Specify dtype option on import or set low_memory=Fa
lse.
  md = pd. read_csv('C:/Users/dipto/.cache/kagglehub/datasets/rounakbanik/the-mov
ies-dataset/versions/7/movies_metadata.csv')
```

Out[6]:

	adult	belongs_to_collection	budget	genres	homepage	
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	31
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}]	NaN	11

5 rows × 24 columns

In [9]:
```
md['genres'] = md['genres'].fillna('[]').apply(literal_eval).apply(lambda x: [i[
```

In [10]:
```
vote_counts = md[md['vote_count'].notnull()]['vote_count'].astype('int')
vote_averages = md[md['vote_average'].notnull()]['vote_average'].astype('int')
C = vote_averages.mean()
C
```

Out[10]:
```
5.244896612406511
```

In [11]:
```
m = vote_counts.quantile(0.95)
m
```

Out[11]:
```
434.0
```

In [12]:
```
md['year'] = pd.to_datetime(md['release_date'], errors='coerce').apply(lambda x:
```

In [13]:
```
qualified = md[(md['vote_count'] >= m) & (md['vote_count'].notnull()) & (md['vot
qualified['vote_count'] = qualified['vote_count'].astype('int')
```

```
qualified['vote_average'] = qualified['vote_average'].astype('int')
qualified.shape
```

Out[13]: (2274, 6)

In [14]:
```
def weightrate(x):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)
```

In [16]:
```
qualified['wr'] = qualified.apply(weightrate, axis=1)
qualified = qualified.sort_values('wr', ascending=False).head(250)
```

In [17]:
```
qualified.head(15)
```

Out[17]:

	title	year	vote_count	vote_average	popularity	genres	wr
15480	Inception	2010	14075	8	29.108149	[Action, Thriller, Science Fiction, Mystery, A...	7.917588
12481	The Dark Knight	2008	12269	8	123.167259	[Drama, Action, Crime, Thriller]	7.905871
22879	Interstellar	2014	11187	8	32.213481	[Adventure, Drama, Science Fiction]	7.897107
2843	Fight Club	1999	9678	8	63.869599	[Drama]	7.881753
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.070725	[Adventure, Fantasy, Action]	7.871787
292	Pulp Fiction	1994	8670	8	140.950236	[Thriller, Crime]	7.868660
314	The Shawshank Redemption	1994	8358	8	51.645403	[Drama, Crime]	7.864000
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.324358	[Adventure, Fantasy, Action]	7.861927
351	Forrest Gump	1994	8147	8	48.307194	[Comedy, Drama, Romance]	7.860656
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.423537	[Adventure, Fantasy, Action]	7.851924
256	Star Wars	1977	6778	8	42.149697	[Adventure, Action, Science Fiction]	7.834205
1225	Back to the Future	1985	6239	8	25.778509	[Adventure, Comedy, Science Fiction, Family]	7.820813
834	The Godfather	1972	6024	8	41.109264	[Drama, Crime]	7.814847

	title	year	vote_count	vote_average	popularity	genres	wr
1154	The Empire Strikes Back	1980	5998	8	19.470959	[Adventure, Action, Science Fiction]	7.814099
46	Se7en	1995	5915	8	18.45743	[Crime, Mystery, Thriller]	7.811669

In [18]:
```
s = md.apply(lambda x: pd.Series(x['genres']),axis=1).stack().reset_index(level=
s.name = 'genre'
gen_md = md.drop('genres', axis=1).join(s)
```

In [21]:
```
def chartify(genre, percentile=0.85):
    df = gen_md[gen_md['genre'] == genre]
    vote_counts = df[df['vote_count'].notnull()]['vote_count'].astype('int')
    vote_averages = df[df['vote_average'].notnull()]['vote_average'].astype('int
    C = vote_averages.mean()
    m = vote_counts.quantile(percentile)

    qualified = df[(df['vote_count'] >= m) & (df['vote_count'].notnull()) & (df[
    qualified['vote_count'] = qualified['vote_count'].astype('int')
    qualified['vote_average'] = qualified['vote_average'].astype('int')

    qualified['wr'] = qualified.apply(lambda x: (x['vote_count']/(x['vote_count'
    qualified = qualified.sort_values('wr', ascending=False).head(250)

    return qualified
```

In [22]:
```
chartify('Romance').head(15)
```

Out[22]:

	title	year	vote_count	vote_average	popularity	wr
10309	Dilwale Dulhania Le Jayenge	1995	661	9	34.457024	8.565285
351	Forrest Gump	1994	8147	8	48.307194	7.971357
876	Vertigo	1958	1162	8	18.20822	7.811667
40251	Your Name.	2016	1030	8	34.461252	7.789489
883	Some Like It Hot	1959	835	8	11.845107	7.745154
1132	Cinema Paradiso	1988	834	8	14.177005	7.744878
19901	Paperman	2012	734	8	7.198633	7.713951
37863	Sing Street	2016	669	8	10.672862	7.689483
882	The Apartment	1960	498	8	11.994281	7.599317
38718	The Handmaiden	2016	453	8	16.727405	7.566166
3189	City Lights	1931	444	8	10.891524	7.558867
24886	The Way He Looks	2014	262	8	5.711274	7.331363
45437	In a Heartbeat	2017	146	8	20.82178	7.003959
1639	Titanic	1997	7770	7	26.88907	6.981546
19731	Silver Linings Playbook	2012	4840	7	14.488111	6.970581

In [25]:
```
links_small = pd.read_csv('C:/Users/dipto/.cache/kagglehub/datasets/rounakbanik/
links_small = links_small[links_small['tmdbId'].notnull()]['tmdbId'].astype('int
md = md.drop([19730, 29503, 35587])
md['id'] = md['id'].astype('int')
smd = md[md['id'].isin(links_small)]
smd.shape
```

Out[25]:
```
(9099, 25)
```

In [28]:
```
smd['tagline'] = smd['tagline'].fillna('').astype(str)
smd['description'] = smd['overview'].fillna('') + ' ' + smd['tagline']
smd['description'] = smd['description'].fillna('').astype(str)
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=1, stop_words='
tfidf_matrix = tf.fit_transform(smd['description'])
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
smd = smd.reset_index()
titles = smd['title']
indices = pd.Series(smd.index, index=smd['title'])
def get_recommendations(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:31]  # Get top 30 recommendations
    movie_indices = [i[0] for i in sim_scores]
    return titles.iloc[movie_indices]
```

```
C:\Users\dipto\AppData\Local\Temp\ipykernel_5144\403399801.py:1: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  smd['tagline'] = smd['tagline'].fillna('').astype(str)
C:\Users\dipto\AppData\Local\Temp\ipykernel_5144\403399801.py:2: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  smd['description'] = smd['overview'].fillna('') + ' ' + smd['tagline']
C:\Users\dipto\AppData\Local\Temp\ipykernel_5144\403399801.py:3: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  smd['description'] = smd['description'].fillna('').astype(str)
```

In [29]:
```
get_recommendations('The Godfather').head(10)
```

Out[29]:
```
973      The Godfather: Part II
8387                The Family
3509                      Made
4196         Johnny Dangerously
29               Shanghai Triad
5667                      Fury
2412             American Movie
1582    The Godfather: Part III
4221                    8 Women
2159               Summer of Sam
Name: title, dtype: object
```

In [30]:
```
get_recommendations('The Dark Knight').head(10)
```

Out[30]:
```
7931                      The Dark Knight Rises
132                              Batman Forever
1113                             Batman Returns
8227    Batman: The Dark Knight Returns, Part 2
7565                 Batman: Under the Red Hood
524                                      Batman
7901                          Batman: Year One
2579              Batman: Mask of the Phantasm
2696                                       JFK
8165    Batman: The Dark Knight Returns, Part 1
Name: title, dtype: object
```

In [31]:
```
get_recommendations('The Intern').head(10)
```

Out[31]:
```
7937              Young Adult
2074            Hideous Kinky
691     The Ballad of Narayama
8812                    Ismael
1417           Mercury Rising
8823        While We're Young
336             Reality Bites
4586                 The Trip
7926        Bullet to the Head
6314              Late Spring
Name: title, dtype: object
```

In [ ]: