In [33]:

```python
import pandas as pd
import numpy as np
from ast import literal_eval
import warnings; warnings.simplefilter('ignore')
```

In [34]:

```python
md = pd.read_csv("movies_metadata.csv")
print(md.head())
```

```
   adult                              belongs_to_collection    budget  \
0  False  {'id': 10194, 'name': 'Toy Story Collection', ...  30000000
1  False                                                NaN  65000000
2  False  {'id': 119050, 'name': 'Grumpy Old Men Collect...         0
3  False                                                NaN  16000000
4  False  {'id': 96871, 'name': 'Father of the Bride Col...         0

                                              genres  \
0  [{'id': 16, 'name': 'Animation'}, {'id': 35, '...
1  [{'id': 12, 'name': 'Adventure'}, {'id': 14, '...
2  [{'id': 10749, 'name': 'Romance'}, {'id': 35, ...
3  [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...
4                     [{'id': 35, 'name': 'Comedy'}]

                              homepage     id    imdb_id original_language  \
0  http://toystory.disney.com/toy-story    862  tt0114709                en
1                                  NaN   8844  tt0113497                en
2                                  NaN  15602  tt0113228                en
3                                  NaN  31357  tt0114885                en
4                                  NaN  11862  tt0113041                en

                original_title  \
0                    Toy Story
1                      Jumanji
2             Grumpier Old Men
3            Waiting to Exhale
4  Father of the Bride Part II

                                            overview  ... release_date  \
0  Led by Woody, Andy's toys live happily in his ...  ...   1995-10-30
1  When siblings Judy and Peter discover an encha...  ...   1995-12-15
2  A family wedding reignites the ancient feud be...  ...   1995-12-22
3  Cheated on, mistreated and stepped on, the wom...  ...   1995-12-22
4  Just when George Banks has recovered from his ...  ...   1995-02-10

       revenue runtime                                   spoken_languages  \
0  373554033.0    81.0           [{'iso_639_1': 'en', 'name': 'English'}]
1  262797249.0   104.0  [{'iso_639_1': 'en', 'name': 'English'}, {'iso...
2          0.0   101.0           [{'iso_639_1': 'en', 'name': 'English'}]
3   81452156.0   127.0           [{'iso_639_1': 'en', 'name': 'English'}]
4   76578911.0   106.0           [{'iso_639_1': 'en', 'name': 'English'}]

     status                                            tagline  \
0  Released                                                NaN
1  Released           Roll the dice and unleash the excitement!
2  Released  Still Yelling. Still Fighting. Still Ready for...
3  Released  Friends are the people who let you be yourself...
4  Released  Just When His World Is Back To Normal... He's ...

                         title  video vote_average vote_count
0                    Toy Story  False          7.7     5415.0
1                      Jumanji  False          6.9     2413.0
2             Grumpier Old Men  False          6.5       92.0
3            Waiting to Exhale  False          6.1       34.0
4  Father of the Bride Part II  False          5.7      173.0

[5 rows x 24 columns]
```

In [35]:

```python
md.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   adult                  45466 non-null  object
 1   belongs_to_collection  4494 non-null   object
 2   budget                 45466 non-null  object
 3   genres                 45466 non-null  object
 4   homepage               7782 non-null   object
 5   id                     45466 non-null  object
 6   imdb_id                45449 non-null  object
 7   original_language      45455 non-null  object
 8   original_title         45466 non-null  object
 9   overview               44512 non-null  object
 10  popularity             45461 non-null  object
 11  poster_path            45080 non-null  object
 12  production_companies   45463 non-null  object
 13  production_countries   45463 non-null  object
 14  release_date           45379 non-null  object
 15  revenue                45460 non-null  float64
 16  runtime                45203 non-null  float64
 17  spoken_languages       45460 non-null  object
 18  status                 45379 non-null  object
 19  tagline                20412 non-null  object
 20  title                  45460 non-null  object
 21  video                  45460 non-null  object
 22  vote_average           45460 non-null  float64
 23  vote_count             45460 non-null  float64
dtypes: float64(4), object(20)
memory usage: 8.3+ MB
```

In [36]:

```python
md.describe()
```

Out[36]:

	revenue	runtime	vote_average	vote_count
count	4.546000e+04	45203.000000	45460.000000	45460.000000
mean	1.120935e+07	94.128199	5.618207	109.897338
std	6.433225e+07	38.407810	1.924216	491.310374
min	0.000000e+00	0.000000	0.000000	0.000000
25%	0.000000e+00	85.000000	5.000000	3.000000
50%	0.000000e+00	95.000000	6.000000	10.000000
75%	0.000000e+00	107.000000	6.800000	34.000000
max	2.787965e+09	1256.000000	10.000000	14075.000000

In [37]:

```python
md.isnull().sum()
```

Out[37]:

```
adult                        0
belongs_to_collection    40972
budget                       0
genres                       0
homepage                 37684
id                           0
imdb_id                     17
original_language           11
original_title               0
overview                   954
popularity                   5
poster_path                386
production_companies         3
production_countries         3
release_date                87
revenue                      6
runtime                    263
spoken_languages             6
status                      87
tagline                  25054
title                        6
video                        6
vote_average                 6
vote_count                   6
dtype: int64
```

In [38]:

```python
mean_vote = md['vote_average'].mean()
print(mean_vote)
```

```
5.618207215133889
```

In [39]:

```python
min_vote = md['vote_count'].quantile(0.9)
print(min_vote)
```

```
160.0
```

In [40]:

```python
movies = md.copy().loc[md['vote_count'] >= min_vote]
movies.shape
```

Out[40]: (4555, 24)

In [41]:

```python
columns_with_missing_data = md.columns[md.isnull().any()]
for col in columns_with_missing_data:
    mode_value = md[col].mode()[0]
    md[col].fillna(mode_value, inplace = True)
```

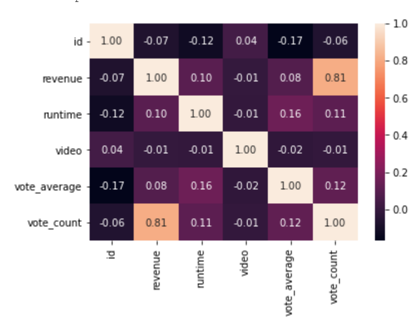
In [95]:

```python
import seaborn as sns
import matplotlib.pyplot as plt
```

In [96]:

```python
sns.heatmap(md.corr(), annot=True, fmt='.2f')
```

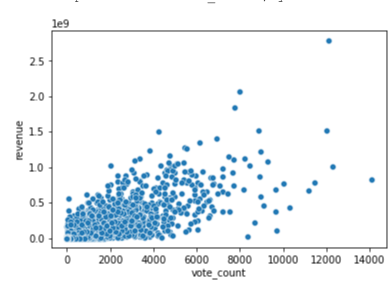
Out[96]: <AxesSubplot:>



In [97]:

```python
sns.scatterplot(x=md['vote_count'], y=md['revenue'])
```

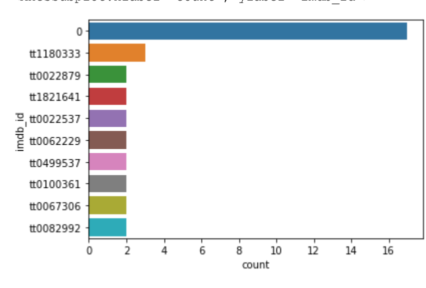
Out[97]: <AxesSubplot:xlabel='vote_count', ylabel='revenue'>



In [98]:

```python
sns.countplot(y=md['imdb_id'], order=md['imdb_id'].value_counts().index[0:10])
```

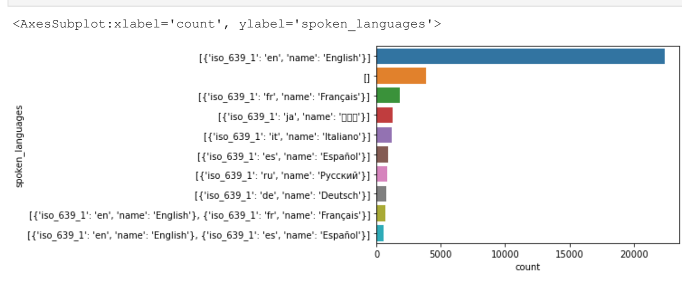
Out[98]: <AxesSubplot:xlabel='count', ylabel='imdb_id'>



In [99]:

```python
sns.countplot(y=md['spoken_languages'], order=md['spoken_languages'].value_counts().i
```

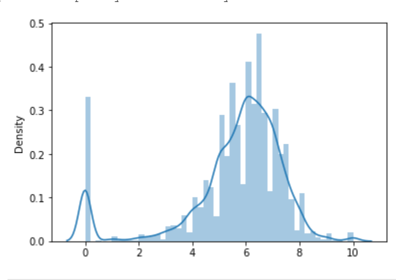
Out[99]: <AxesSubplot:xlabel='count', ylabel='spoken_languages'>



In [100...

```python
sns.distplot(x=md['vote_average'])
```

Out[100... <AxesSubplot:ylabel='Density'>



In [42]:

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

In [43]:

```python
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[43]: 5.24420446047596

In [44]:

```python
m = vote_counts.quantile(0.9)
m
```

Out[44]: 160.0

In [45]:

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

In [ ]:

In [48]:

```python
qualified = md[(md['vote_count'] >= m) & (md['vote_count'].notnull()) &
              (md['vote_average'].notnull())][['title','year', 'vote_count', 'vote_av
qualified['vote_count'] = qualified['vote_count'].astype('int')
qualified['vote_average'] = qualified['vote_average'].astype('int')
qualified.shape
```

Out[48]: (4555, 6)

In [49]:

```python
def weighted_rating(x):
    voters = x['vote_count']
    avg_vote = x['vote_average']
    return (voters/(voters + m) * avg_vote) + (m/(m + voters) * C)
```

In [50]:

```python
qualified['wr'] = qualified.apply(weighted_rating, axis=1)
```

In [51]:

```python
qualified = qualified.sort_values('wr', ascending=False).head(250)
```

In [52]:

```python
qualified.head(10)
```

Out[52]:

	title	year	vote_count	vote_average	popularity	genres	wr
10309	Dilwale Dulhania Le Jayenge	1995	661	9	34.457024	[Comedy, Drama, Romance]	8.268054
15480	Inception	2010	14075	8	29.108149	[Action, Thriller, Science Fiction, Mystery, A...	7.969025
12481	The Dark Knight	2008	12269	8	123.167259	[Drama, Action, Crime, Thriller]	7.964524
22879	Interstellar	2014	11187	8	32.213481	[Adventure, Drama, Science Fiction]	7.961142
2843	Fight Club	1999	9678	8	63.869599	[Drama]	7.955181
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.070725	[Adventure, Fantasy, Action]	7.951290
292	Pulp Fiction	1994	8670	8	140.950236	[Thriller, Crime]	7.950065
314	The Shawshank Redemption	1994	8358	8	51.645403	[Drama, Crime]	7.948236
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.324358	[Adventure, Fantasy, Action]	7.947421
351	Forrest Gump	1994	8147	8	48.307194	[Comedy, Drama, Romance]	7.946921

In [53]:

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

In [54]:

```python
def genre_wise(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['vote_
    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']+m) *
    qualified = qualified.sort_values('wr', ascending=False).head(250)

    return qualified
```

In [55]:

```python
genre_wise('Romance').head(15)
```

Out[55]:

	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 [56]:

```python
genre_wise('Action').head(10)
```

Out[56]:

	title	year	vote_count	vote_average	popularity	wr
15480	Inception	2010	14075	8	29.108149	7.955084
12481	The Dark Knight	2008	12269	8	123.167259	7.948593
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.070725	7.929555
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.324358	7.924005
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.423537	7.918355
256	Star Wars	1977	6778	8	42.149697	7.908296
1154	The Empire Strikes Back	1980	5998	8	19.470959	7.896805
4135	Scarface	1983	3017	8	11.299673	7.801977
9430	Oldboy	2003	2000	8	10.616859	7.711546
1910	Seven Samurai	1954	892	8	15.01777	7.425928

In [ ]:

In [71]:

```python
links_small = pd.read_csv('links_small.csv')
links_small.head()
```

Out[71]:

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0

In [72]:

```python
links_small.describe()
```

Out[72]:

	movieId	imdbId	tmdbId
count	9125.000000	9.125000e+03	9112.000000
mean	31123.291836	4.798244e+05	39104.545544
std	40782.633604	7.431774e+05	62814.519801
min	1.000000	4.170000e+02	2.000000
25%	2850.000000	8.884600e+04	9451.750000
50%	6290.000000	1.197780e+05	15852.000000
75%	56274.000000	4.284410e+05	39160.500000
max	164979.000000	5.794766e+06	416437.000000

In [73]:

```python
links_small.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9125 entries, 0 to 9124
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  9125 non-null   int64
 1   imdbId   9125 non-null   int64
 2   tmdbId   9112 non-null   float64
dtypes: float64(1), int64(2)
memory usage: 214.0 KB
```

In [77]:

```python
links_small.isnull().sum()
```

Out[77]:

```
movieId    0
imdbId     0
tmdbId     0
dtype: int64
```

In [76]:

```python
mean_value = links_small['tmdbId'].mean()  # Calculate the mean
links_small['tmdbId'].fillna(mean_value, inplace=True)
```

In [78]:

```python
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)]
```

In [79]:

```python
smd.shape
```

Out[79]: (9100, 25)

In [80]:

```python
smd['tagline'] = smd['tagline'].fillna('')
smd['description'] = smd['overview'] + smd['tagline']
smd['description'] = smd['description'].fillna('')
```

In [81]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
```

In [82]:

```python
tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0, stop_words='english
tfidf_matrix = tf.fit_transform(smd['description'])
```

In [83]:

```python
tfidf_matrix.shape
```

Out[83]: (9100, 269623)

In [84]:

```python
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

In [85]:

```python
cosine_sim[1]
```

Out[85]:

```
array([0.00672473, 1.        , 0.01531072, ..., 0.00357074, 0.00759553,
       0.        ])
```

In [86]:

```python
smd = smd.reset_index()
titles = smd['title']
indices = pd.Series(smd.index, index=smd['title'])
```

In [87]:

```python
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]
    movie_indices = [i[0] for i in sim_scores]
    return titles.iloc[movie_indices]
```

In [88]:

```python
get_recommendations('The Godfather').head(10)
```

Out[88]:

```
973      The Godfather: Part II
8388                 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 [89]:

```python
get_recommendations('The Dark Knight').head(10)
```

Out[89]:

```
7931                      The Dark Knight Rises
132                              Batman Forever
1113                             Batman Returns
8228    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
8166    Batman: The Dark Knight Returns, Part 1
Name: title, dtype: object
```

In [94]:

```python
get_recommendations('Seven Samurai').head(10)
```

Out[94]:

```
3425                The Magnificent Seven
4016                      Shogun Assassin
7162           The Good, The Bad, The Weird
7391      Did You Hear About the Morgans?
2371                      The Story of Us
5705           Samurai I: Musashi Miyamoto
4619                   Destry Rides Again
4136                             Mr. Deeds
3605                       Erik the Viking
2741    Teenage Mutant Ninja Turtles III
Name: title, dtype: object
```