DM_week_3 数据分析

assignment

数据摘要

标称属性，给出每个可能取值的频数

数值属性，给出5数概括及缺失值的个数

数据可视化

使用直方图、盒图等检查数据分布及离群点

数据缺失的处理

观察数据集中缺失数据，分析其缺失的原因。分别使用下列四种策略对缺失值进行处理:

- 将缺失部分剔除
- 用最高频率值来填补缺失值
- 通过属性的相关关系来填补缺失值
- 通过数据对象之间的相似性来填补缺失值

注意：在处理后完成，要对比新旧数据集的差异。

0.加载数据集

Columns

`id` : movie's unique id

`title` : movie's name

`storyline` : a short description of the movie

`views` : no. of clicks per movie

`downloads` : no. of downloads per movie

`IMDb-rating` : rating

`appropriate_for` : R-rated, PG-13, etc

`language` : this can be multiple languages also

`industry` : Hollywood, Bollywood, etc.

`posted_date` : when the movie is posted on the platform

`release_date` : when the movie is released worldwide

`run_time` : in minutes

`director` : director's name

`writer` : list of all the writers

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Load data set.
mov_data = pd.read_csv('datasets/movies_dataset.csv')
```

```
# Check the columns.
mov_data.columns
```

Out[ ]:
```
Index(['Unnamed: 0', 'IMDb-rating', 'appropriate_for', 'director', 'downloads',
       'id', 'industry', 'language', 'posted_date', 'release_date', 'run_time',
       'storyline', 'title', 'views', 'writer'],
      dtype='object')
```

In [ ]:
```
# 去除不相关列
mov_data = mov_data.drop(columns='Unnamed: 0')
```

In [ ]:
```
mov_data.columns
```

Out[ ]:
```
Index(['IMDb-rating', 'appropriate_for', 'director', 'downloads', 'id',
       'industry', 'language', 'posted_date', 'release_date', 'run_time',
       'storyline', 'title', 'views', 'writer'],
      dtype='object')
```

1.数据摘要

属性预览

分析不同属性的类型，样例，唯一值个数，空值比例

In [ ]:
```
def dtype_uni_miss(mov_data):
    cols, dtype, specimens, num_unique, null_share = [], [], [], [], []

    for column in mov_data:
        cols.append(column)
        col_dtype = mov_data[column].dtype
        dtype.append(col_dtype)

        specimen = mov_data.loc[mov_data[column].first_valid_index(), column]
        specimens.append(specimen)

        num_unique.append(mov_data[column].nunique())

        null_sum = mov_data[column].isna().sum()
        null_to_len = null_sum / len(mov_data[column])
        null_share.append(f'{null_to_len:.2%}')

    df = pd.DataFrame(list(zip(dtype, specimens, num_unique, null_share)), index=cols)
    df.columns=['dtype', 'eg.', 'num_unique', 'null_share']

    return df
```

In [ ]:
```
dtype_uni_miss(mov_data)
```

Out[ ]:

	dtype	eg.	num_unique	null_share
IMDb-rating	float64	4.8	85	4.09%
appropriate_for	object	R	21	46.12%
director	object	John Swab	9672	9.43%
downloads	object	304	10625	0.00%
id	int64	372092	17086	0.00%
industry	object	Hollywood / English	10	0.00%
language	object	English	1167	2.66%
posted_date	object	20 Feb, 2023	4123	0.00%
release_date	object	Jan 28 2023	4886	0.00%
run_time	object	105	415	8.60%
storyline	object	Doc\r\n facilitates a fragile truce between th...	15748	8.28%
title	object	Little Dixie	16572	0.00%
views	object	2,794	16821	0.00%
writer	object	John Swab	13603	10.67%

对属性进行预览：

In [ ]:
```
mov_data.head()
```

Out[ ]:

	IMDb-rating	appropriate_for	director	downloads	id	industry	language	posted_date	release_date	run_time	storyline
0	4.8	R	John Swab	304	372092	Hollywood / English	English	20 Feb, 2023	Jan 28 2023	105	Doc\r\n facilitates a fragile truce between th...
1	6.4	TV-PG	Paul Ziller	73	372091	Hollywood / English	English	20 Feb, 2023	Feb 05 2023	84	Caterer\r\n Goldy Berry reunites with detectiv...
2	5.2	R	Ben Wheatley	1,427	343381	Hollywood / English	English,Hindi	20 Apr, 2021	Jun 18 2021	1h 47min	As the world searches for a cure to a disastro...
3	8.1	NaN	Venky Atluri	1,549	372090	Tollywood	Hindi	20 Feb, 2023	Feb 17 2023	139	The life of a young man and his struggles agai...
4	4.6	NaN	Shaji Kailas	657	372089	Tollywood	Hindi	20 Feb, 2023	Jan 26 2023	122	A man named Kalidas gets stranded due to the p...

downloads和views本应是数值属性，于是将downloads和views从字符串转换成数值属性

In [ ]:
```
for col in 'downloads', 'views':
    mov_data[col] = mov_data[col].str.replace(',','')
    mov_data[col] = mov_data[col].astype('float')

mov_data['id'] = mov_data['id'].astype('str')
```

In [ ]:
```
# for col in 'downloads', 'views':
#     mov_data[col] = mov_data[col].astype('int')
dtype_uni_miss(mov_data)
```

Out[ ]:

	dtype	eg.	num_unique	null_share
IMDb-rating	float64	4.8	85	4.09%
appropriate_for	object	R	21	46.12%
director	object	John Swab	9672	9.43%
downloads	float64	304.0	10625	0.00%
id	object	372092	17086	0.00%
industry	object	Hollywood / English	10	0.00%
language	object	English	1167	2.66%
posted_date	object	20 Feb, 2023	4123	0.00%
release_date	object	Jan 28 2023	4886	0.00%
run_time	object	105	415	8.60%
storyline	object	Doc\r\n facilitates a fragile truce between th...	15748	8.28%
title	object	Little Dixie	16572	0.00%
views	float64	2794.0	16821	0.00%
writer	object	John Swab	13603	10.67%

标称属性 nominal attributes

`id` : movie's unique id

`title` : movie's name

`storyline` : a short description of the movie

`appropriate_for` : R-rated, PG-13, etc

`language` : this can be multiple languages also

`industry` : Hollywood, Bollywood, etc.

`posted_date` : when the movie is posted on the platform

`release_date` : when the movie is released worldwide

`runtime` : in minutes

`director` : director's name

`writer` : list of all the writers

列出每个属性的每个值的频数

In [ ]:
```
def list_frequencies(mov_data):
    result_list = []
    for col in mov_data:
        if mov_data[col].dtype == 'object':
            nominal_frequencies = mov_data[col].value_counts()
            for value, count in nominal_frequencies.items():
                result_list.append({'':col,'value':value,'count':count})

    df = pd.DataFrame(result_list)
    return df

# 假设你的数据集存储在一个名为mov_data的DataFrame中
# 假设标称属性的列名为'category'
# nominal_column_name = 'director'  # 请替换为你的标称属性列名

# nominal_frequencies = calculate_nominal_frequencies(mov_data, nominal_column_name)
# print("标称属性的频数：")
# print(nominal_frequencies)
```

In [ ]:
```
list_frequencies(mov_data)
```

Out[ ]:

		value	count
0	appropriate_for	R	4384
1	appropriate_for	Not Rated	2142
2	appropriate_for	PG-13	1968
3	appropriate_for	PG	886
4	appropriate_for	TV-14	694
...	...	...	...
83298	writer	Barbara Samuels, Joseph Boyden	1
83299	writer	Maria Allred	1
83300	writer	Pia Mechler	1
83301	writer	Paul Flannery, David Ryan Keith	1
83302	writer	Khwaja Ahmad Abbas, Khwaja Ahmad Abbas	1

83303 rows × 3 columns

对每个标称属性进行分析

`id`

In [ ]:
```
mov_data['id'].value_counts()
```

Out[ ]:
```
id
372090    402
371744    402
371877    402
372092    202
371991    202
          ...
303381      1
303380      1
303379      1
303377      1
30459       1
Name: count, Length: 17086, dtype: int64
```

可以看出，存在重复的id，说明一部电影可能存在多条数据记录

In [ ]:
```
# 查看其中一部电影的记录
mov_data[mov_data['id'] == '372090']
```

Out[ ]:

	IMDb-rating	appropriate_for	director	downloads	id	industry	language	posted_date	release_date	run_time	storyline	
3	8.1	NaN	Venky Atluri	1549.0	372090	Tollywood	Hindi	20 Feb, 2023	Feb 17 2023	139	The life of a young man and his struggles agai...	V
38	8.1	NaN	Venky Atluri	1550.0	372090	Tollywood	Hindi	20 Feb, 2023	Feb 17 2023	139	The life of a young man and his struggles agai...	V
82	8.1	NaN	Venky Atluri	1552.0	372090	Tollywood	Hindi	20 Feb, 2023	Feb 17 2023	139	The life of a young man and his struggles agai...	V
126	8.1	NaN	Venky Atluri	1557.0	372090	Tollywood	Hindi	20 Feb, 2023	Feb 17 2023	139	The life of a young man and his struggles agai...	V
170	8.1	NaN	Venky Atluri	1558.0	372090	Tollywood	Hindi	20 Feb, 2023	Feb 17 2023	139	The life of a young man and his struggles agai...	V
...	...	...	...	...	...	...	...	...	...	...	...	
17461	8.1	NaN	Venky Atluri	2369.0	372090	Tollywood	Hindi	20 Feb, 2023	Feb 17 2023	139	The life of a young man and his struggles agai...	V
17505	8.1	NaN	Venky Atluri	2374.0	372090	Tollywood	Hindi	20 Feb, 2023	Feb 17 2023	139	The life of a young man and his struggles agai...	V
17549	8.1	NaN	Venky Atluri	2375.0	372090	Tollywood	Hindi	20 Feb, 2023	Feb 17 2023	139	The life of a young man and his struggles agai...	V
17593	8.1	NaN	Venky Atluri	2378.0	372090	Tollywood	Hindi	20 Feb, 2023	Feb 17 2023	139	The life of a young man and his struggles agai...	V
17637	8.1	NaN	Venky Atluri	2379.0	372090	Tollywood	Hindi	20 Feb, 2023	Feb 17 2023	139	The life of a young man and his struggles agai...	V

402 rows × 14 columns

只有downloads与views变化，说明可能是随着时间推移，对同一部电影添加了多次记录

`title`

In [ ]:
```
mov_data['title'].value_counts()
```

Out[ ]:
```
title
The Girl Who Escaped: The Kara Robinson Story    402
Vaathi                                           402
Who Invited Charlie?                             402
Little Dixie                                     202
The Inspection                                   202
                                                ...
Kesari                                             1
Old Boys                                           1
American Exit                                      1
Adventures of Aladdin                              1
Madhumati                                          1
Name: count, Length: 16572, dtype: int64
```

In [ ]:
```
mov_data['title'].value_counts().head(30)
```

Out[ ]:
```
title
The Girl Who Escaped: The Kara Robinson Story    402
Vaathi                                           402
Who Invited Charlie?                             402
Little Dixie                                     202
The Inspection                                   202
WWE Smackdown 2023-02-10                         202
Consent                                          202
Vacation Home Nightmare                          202
Your Place or Mine                               201
Baby Ruby                                        201
Carnifex                                         201
Marlowe                                          201
Shehzada                                         201
TNA.Impact 2023-02-09                             78
WWE Raw 2023-02-13                                78
Pinocchio                                          6
Alone                                              5
Sacrifice                                          5
True Justice                                       5
Gold                                               4
The Jungle Book                                    4
Devdas                                             4
Bodyguard                                          4
Don                                                4
Remember Me                                        4
Blackbird                                          4
Saathi                                             4
Ghost                                              4
Blood Money                                        4
Lucky                                              4
Name: count, dtype: int64
```

`storyline`

In [ ]:
```
mov_data['storyline'].value_counts()
```

Out[ ]:
```
storyline
The life of a young man and his struggles against the privatization of education.
402
Follows\r\n a New York City family hiding out in the Hamptons whose bubble is \r\npopped when a Bloody Mary-swi
lling, pot-smoking 'Charlie' comes to bring\r\n a lifetime of hurt that might heal them all.
402
It follows Kara Robinson as she survives an abduction and ultimately brings down a serial killer.
402
Doc\r\n facilitates a fragile truce between the Governor and Cartel, trading \r\nprosecutorial leniency for fin
ance. With no more truce, Doc is left to \r\nfend for himself and protect the one untainted thing in his life:
his \r\ndaughter, Little Dixie.
202
A\r\n young, gay Black man, rejected by his mother and with few options for \r\nhis future, decides to join the
Marines, doing whatever it takes to \r\nsucceed in a system that would cast him aside.
202

...
Four waves of increasingly deadly attacks have left most of Earth in ruin. Against a backdrop of fear and distr
ust, Cassie is on the run, desperately trying to save her younger brother. As she prepares for the inevitable a
nd lethal fifth wave, Cassie teams up with a young man who may become her final hope - if she can only trust hi
m.
1
Yamuna along with her son Laxman locates to Mumbai leaving behind her abusive husband.She takes shelter in the
house of her aunt Chandra whom she calls Akka.Yamuna's only aim is to give a better education to her son.Chandr
a finds her a job as sweeper in a art school.Yamuna finds that Chandra poses as a nude model to the students of
the school.Chandra confines Yamuna to take up the job being nude out there the students don't look at you in lu
st but as a project.      1
A young violinist struggles to assert her individuality amidst the intense pressure of her pianist father, and
the weight of her own musical ability.
1
A right wing talk show host's life takes a sudden turn when his 16 year old niece comes crashing into his life.
1
While driving his car on a rainy night, Anand's car breaks down, and he goes to seek shelter in a nearby house.
He is let into the house by the servant, and he is permitted to stay until the rains stop be able to get his ca
r fixed. It is here that he will find out about his previous birth, his true love, Madhumati, their ill-fated,
star-crossed and tragic romance, and how events in his previous birth are going to effect him in this life-time
.                          1
Name: count, Length: 15748, dtype: int64
```

`appropriate_for`

In [ ]:
```
mov_data['appropriate_for'].value_counts()
```

Out[ ]:
```
appropriate_for
R                4384
Not Rated        2142
PG-13            1968
PG                886
TV-14             694
TV-MA             406
G                 152
Unrated           132
TV-PG             115
TV-G               99
TV-Y7              45
TV-Y               25
Approved            9
NC-17               4
TV-Y7-FV            3
Passed              3
MA-17               1
TV-13               1
Drama               1
Drama, Romance      1
18+                 1
Name: count, dtype: int64
```

`language`

In [ ]:
```
mov_data['language'].value_counts()
```

Out[ ]:
```
language
English                                   12657
Hindi                                      2558
English,Spanish                             391
Punjabi                                     310
English,Hindi                               304
                                           ...
English,Korean,Spanish                        1
Norwegian,Swedish                             1
Spanish,Chinese,English,Maori,French          1
Urdu,Punjabi,English                          1
Spanish,German,English                        1
Name: count, Length: 1167, dtype: int64
```

In [ ]:
```
mov_data['language'].value_counts().head(30)
```

Out[ ]:
```
language
English                 12657
Hindi                    2558
English,Spanish           391
Punjabi                   310
English,Hindi             304
Telugu                    298
Tamil                     198
Hindi,English             191
English,French            174
English,Russian            71
English,German             65
English,Italian            54
Urdu                       52
English,Japanese           50
English,Mandarin           48
Malayalam                  48
Kannada                    43
English,Arabic             38
French                     37
Spanish,English            34
Japanese                   34
Russian                    30
English,Chinese            30
Hindi,Urdu                 29
English,Ukrainian          29
English Hindi              29
Spanish                    28
English,Latin              26
Hindi,Marathi,English      25
German                     24
Name: count, dtype: int64
```

可以看出，English为使用最广泛的语言，而许多电影都支持双语甚至多种不同语言，一共存在1167种不同的语言组合

In [ ]:
```
mov_language = mov_data['language'].astype('str')
mov_language
```

Out[ ]:
```
0             English
1             English
2        English,Hindi
3               Hindi
4               Hindi
             ...
20543           Hindi
20544           Hindi
20545           Hindi
20546         English
20547         English
Name: language, Length: 20548, dtype: object
```

In [ ]:
```
# 拆分不同语言
for i in range(len(mov_language)):
    mov_language.iloc[i] = mov_language.iloc[i].split(',')
    for j in range(len(mov_language.iloc[i])):
        mov_language.iloc[i][j] = mov_language.iloc[i][j].lstrip(' ')

mov_language
```

Out[ ]:
```
0               [English]
1               [English]
2       [English, Hindi]
3                 [Hindi]
4                 [Hindi]
               ...
20543             [Hindi]
20544             [Hindi]
20545             [Hindi]
20546           [English]
20547           [English]
Name: language, Length: 20548, dtype: object
```

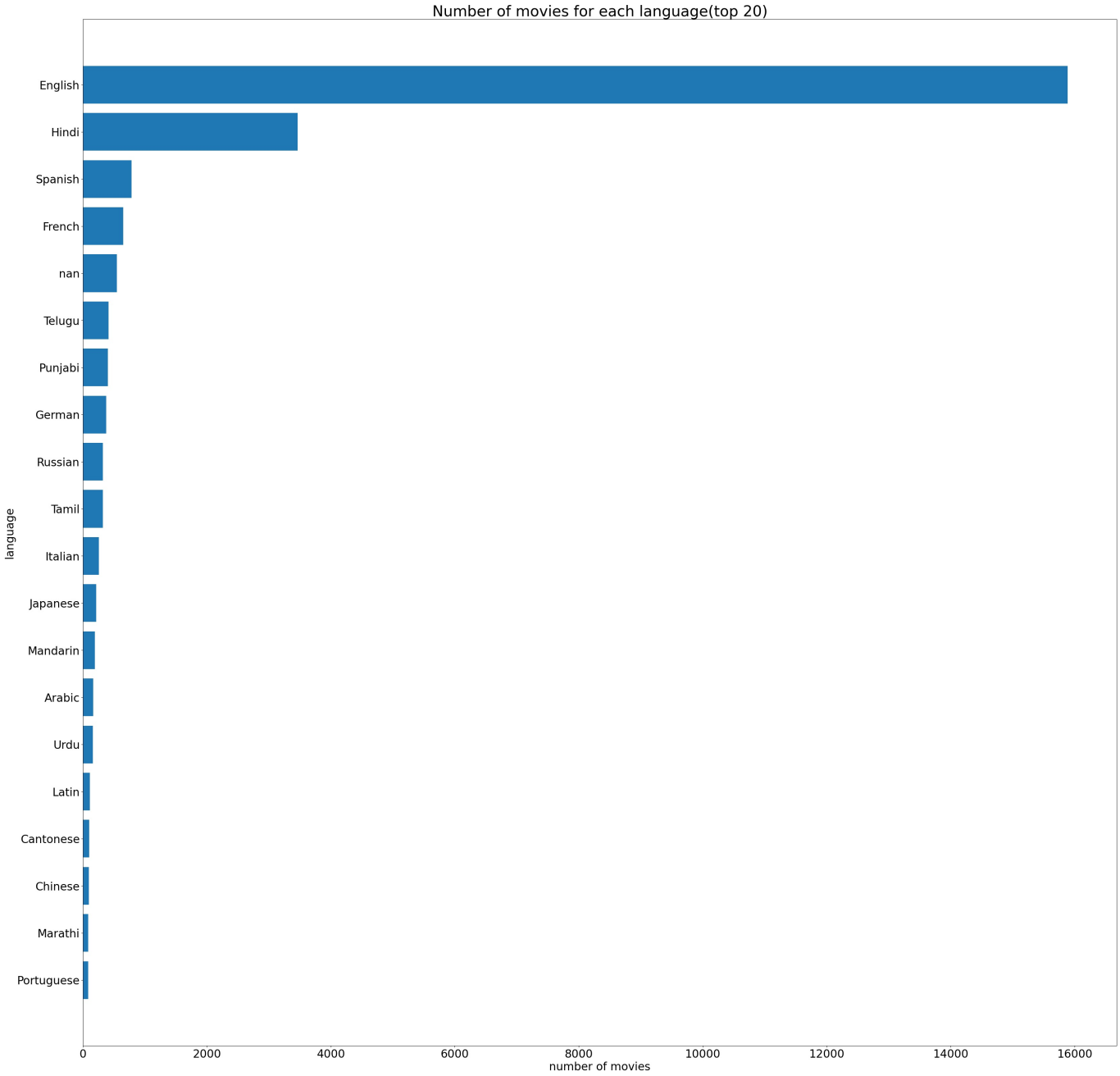
In [ ]:
```
# 创造一个字典用于记录不同语言
mov_language_dict = {}

for i in range(len(mov_language)):
    for j in range(len(mov_language[i])):
        if mov_language[i][j] not in mov_language_dict :
            mov_language_dict['{}'.format(mov_language[i][j])] = 0
        mov_language_dict['{}'.format(mov_language[i][j])] += 1
```

In [ ]:
```
print("一共有{}种不同语言 ".format(len(mov_language_dict)))
```

一共有187种不同语言

In [ ]:
```
mov_language_dict = dict(sorted(mov_language_dict.items(), key=lambda x: x[1]))
```

In [ ]:
```
plt.figure(figsize=(40, 40))
plt.yticks(fontsize=24)
plt.xticks(fontsize=24)
plt.barh(list(mov_language_dict.keys())[-20:], width=list(mov_language_dict.values())[-20:])
plt.ylabel('language', fontsize=24)
plt.xlabel('number of movies', fontsize=24)
plt.title('Number of movies for each language(top 20)', fontsize=32, loc='center')
plt.show()
```



In [ ]:
```
mov_language_dict.keys()
```

Out[ ]:
```
dict_keys(['Bambara', 'Sumerian', 'Ojibwa', 'Chechen', 'Shoshoni', 'Nahuatl', 'Bable', 'Tonga (Tonga Islands)',
'Spanish Sign Language', 'Wolof', 'Dinka', 'Chaozhou', 'Russian Sign Language', 'Parsee', 'Visayan', 'Rhaetian'
, 'Turkmen', 'Middle English', 'p', 'British Sign Language', 'Igbo', 'Hokkien', 'Nama', 'Polynesian', 'Cheyenne
', 'Uzbek', 'Mohawk', 'Kikuyu', 'Azerbaijani', 'Assamese', 'Abkhazian', 'Pawnee', 'Australian Sign Language', '
Konkani', 'Syriac', 'Greenlandic', 'Indian Sign Language', 'English.Hindi', 'Quenya', 'Sinhala', 'Tok Pisin', '
Sicilian', 'Slovak', 'Acholi', 'Apache languages', 'Low German', 'Kru', 'Mende', 'Algonquin', 'Akan', 'Maltese'
, 'Samoan', 'Cree', 'Tupi', 'Macedonian', 'Georgian', 'Amharic', 'Micmac', 'Tswana', 'Basque', 'Quechua', 'Nava
jo', 'Ibo', 'Nyanja', 'Sotho', 'Sioux', 'Papiamento', 'Slovenian', 'Oriya', 'Lao', 'Southern Sotho', 'Aramaic',
'Brazilian Sign Language', 'Tulu', 'Maya', 'Khmer', 'Romany', 'Awadhi', 'Kazakh', 'Kashmiri', 'Pushto', 'Burmes
e', 'Kurdish', 'Saami', 'Lingala', 'Catalan', 'Klingon', 'Shanghainese', 'Sindarin', 'Yoruba', 'Norse', 'Old',
'Ancient (to 1453)', 'Dari', 'Mongolian', 'Lithuanian', 'Scottish Gaelic', 'Old English', 'Flemish', 'Egyptian
(Ancient)', 'Scots', 'Sindhi', 'Min Nan', 'Nepali', 'Estonian', 'Tibetan', 'None', 'Swiss German', 'Zulu', 'Cro
atian', 'Albanian', 'Inuktitut', 'Sign Languages', 'Bosnian', 'Esperanto', 'Maori', 'Haryanvi', 'Malay', 'Serbo
-Croatian', 'Central Khmer', 'Hawaiian', 'Somali', 'Tagalog', 'Armenian', 'Irish', 'Filipino', 'North American
Indian', 'Bhojpuri', 'Swahili', 'Irish Gaelic', 'Welsh', 'Xhosa', 'Sanskrit', 'Yiddish', 'Pashtu', 'Rajasthani'
, 'Finnish', 'Gaelic', 'Bulgarian', 'Aboriginal', 'Gujarati', 'Icelandic', 'Afrikaans', 'Panjabi', 'Czech', 'Se
rbian', 'Indonesian', 'Hungarian', 'Vietnamese', 'English Hindi', 'Romanian', 'Persian', 'Bengali', 'Danish', '
Turkish', 'American Sign Language', 'Swedish', 'Norwegian', 'Greek', 'Polish', 'Dutch', 'Thai', 'Hebrew', 'Ukra
inian', 'Kannada', 'Malayalam', 'Korean', 'Portuguese', 'Marathi', 'Chinese', 'Cantonese', 'Latin', 'Urdu', 'Ar
abic', 'Mandarin', 'Japanese', 'Italian', 'Tamil', 'Russian', 'German', 'Punjabi', 'Telugu', 'nan', 'French', '
Spanish', 'Hindi', 'English'])
```

`industry`

In [ ]:
```
mov_data['industry'].value_counts()
```

Out[ ]:
```
industry
Hollywood / English    14649
Bollywood / Indian      2645
Tollywood               1172
Anime / Kids            1049
Wrestling                433
Punjabi                  332
Stage shows              129
Pakistani                 92
Dub / Dual Audio          45
3D Movies                  1
Name: count, dtype: int64
```

`posted_date`

In [ ]:
```
mov_data['posted_date'].value_counts()
```

Out[ ]:
```
posted_date
13 Feb, 2023    812
20 Feb, 2023    607
15 Feb, 2023    607
10 Feb, 2023    485
16 Feb, 2023    406
                ...
12 Sep, 2009      1
08 Sep, 2009      1
01 Sep, 2009      1
18 Aug, 2009      1
30 Nov, 2011      1
Name: count, Length: 4123, dtype: int64
```

In [ ]:
```
mov_data['posted_date'] = pd.to_datetime(mov_data['posted_date'])
mov_data['posted_date']
```

Out[ ]:
```
0       2023-02-20
1       2023-02-20
2       2021-04-20
3       2023-02-20
4       2023-02-20
           ...
20543   1970-01-01
20544   1970-01-01
20545   1970-01-01
20546   2023-02-10
20547   2023-02-14
Name: posted_date, Length: 20548, dtype: datetime64[ns]
```

`release_date`

In [ ]:
```
mov_data['release_date'] = pd.to_datetime(mov_data['release_date'])
mov_data['release_date']
```

Out[ ]:
```
0       2023-01-28
1       2023-02-05
2       2021-06-18
3       2023-02-17
4       2023-01-26
           ...
20543   1959-03-13
20544   1955-05-13
20545   1958-03-28
20546   2023-02-09
20547   2023-02-13
Name: release_date, Length: 20548, dtype: datetime64[ns]
```

In [ ]:
```
mov_data['release_date'].value_counts()
```

Out[ ]:
```
release_date
1970-01-01    962
2023-02-03    616
2023-02-17    607
2023-02-10    410
2023-02-11    402
             ...
2003-09-05      1
2022-12-29      1
2013-08-24      1
2014-01-12      1
1958-03-28      1
Name: count, Length: 4886, dtype: int64
```

`run_time`

In [ ]:
```
mov_data['run_time'].value_counts()
```

Out[ ]:
```
run_time
93          652
88          622
101         568
139         454
95          454
           ...
74 min        1
288           1
220           1
49min         1
3h 13min      1
Name: count, Length: 415, dtype: int64
```

In [ ]:
```
# pd.set_option('mode.chained_assignment', None)
# for i in range(len(mov_data['run_time'])):
#     # print(mov_data['run_time'].iloc[i])
#     if mov_data['run_time'].isna().iloc[i] == False:
#         time_list = mov_data['run_time'].astype('str').iloc[i].rstrip('min').rstrip('h').split('h ')
#         if len(time_list) == 2:
#             run_time = int(time_list[0]) * 60 + int(time_list[1])
#         elif len(time_list) == 1:
#             run_time = int(time_list[0])

#         mov_data['run_time'].iloc[i] = run_time
```

In [ ]:
```
mov_data['run_time'].value_counts().head(30)
```

Out[ ]:
```
run_time
93          652
88          622
101         568
139         454
95          454
90          431
105         368
1h 30min    365
110         353
109         326
92          265
142         254
98          240
85          237
96          234
100         223
91          218
94          216
87          199
97          199
86          198
1h 35min    191
89          181
99          178
84          176
104         175
1h 32min    165
107         164
1h 25min    163
102         159
Name: count, dtype: int64
```

director

In [ ]:
```
mov_data['director'].value_counts()
```

Out[ ]:
```
director
Venky Atluri                                  405
Simone Stock                                  403
Xavier Manrique                               403
John Swab                                     205
Neil Jordan                                   205
                                             ...
Agnieszka Smoczynska                            1
Dylan Thomas Ellis                              1
Sunil Thakur, Sunil Dhawan, Shivani Thakur      1
Suman Mukhopadhyay                              1
Shea Sizemore                                   1
Name: count, Length: 9672, dtype: int64
```

writer

In [ ]:
```
mov_data['writer'].value_counts()
```

Out[ ]:
```
writer
Nicholas Schutt                            403
Venky Atluri                               402
Haley Harris                               402
John Swab                                  205
Elegance Bratton                           202
                                          ...
Barbara Samuels, Joseph Boyden               1
Maria Allred                                 1
Pia Mechler                                  1
Paul Flannery, David Ryan Keith              1
Khwaja Ahmad Abbas, Khwaja Ahmad Abbas       1
Name: count, Length: 13603, dtype: int64
```

数值属性 Numeric Attributes

给出5数概括及缺失值的个数

`views` : no. of clicks per movie

`downloads` : no. of downloads per movie

`IMDb-rating` : rating

In [ ]:
```
numeric_mov_data = pd.DataFrame(mov_data, columns=['views', 'downloads', 'IMDb-rating'])
numeric_mov_data.head()
```

Out[ ]:

	views	downloads	IMDb-rating
0	2794.0	304.0	4.8
1	1002.0	73.0	6.4
2	14419.0	1427.0	5.2
3	4878.0	1549.0	8.1
4	2438.0	657.0	4.6

In [ ]:
```
mov_data['views'].describe()
```

Out[ ]:
```
count    2.054700e+04
mean     3.559551e+04
std      6.247242e+04
min      6.670000e+02
25%      7.571500e+03
50%      1.522200e+04
75%      3.657100e+04
max      1.638533e+06
Name: views, dtype: float64
```

In [ ]:
```
mov_data['downloads'].describe()
```

Out[ ]:
```
count     20547.000000
mean      10795.238916
std       23716.181987
min           0.000000
25%         855.500000
50%        2716.000000
75%       10070.000000
max      391272.000000
Name: downloads, dtype: float64
```

In [ ]:
```
mov_data['IMDb-rating'].describe()
```

Out[ ]:
```
count    19707.000000
mean         5.762151
std          1.374041
min          1.100000
25%          4.800000
50%          5.700000
75%          6.600000
max          9.900000
Name: IMDb-rating, dtype: float64
```

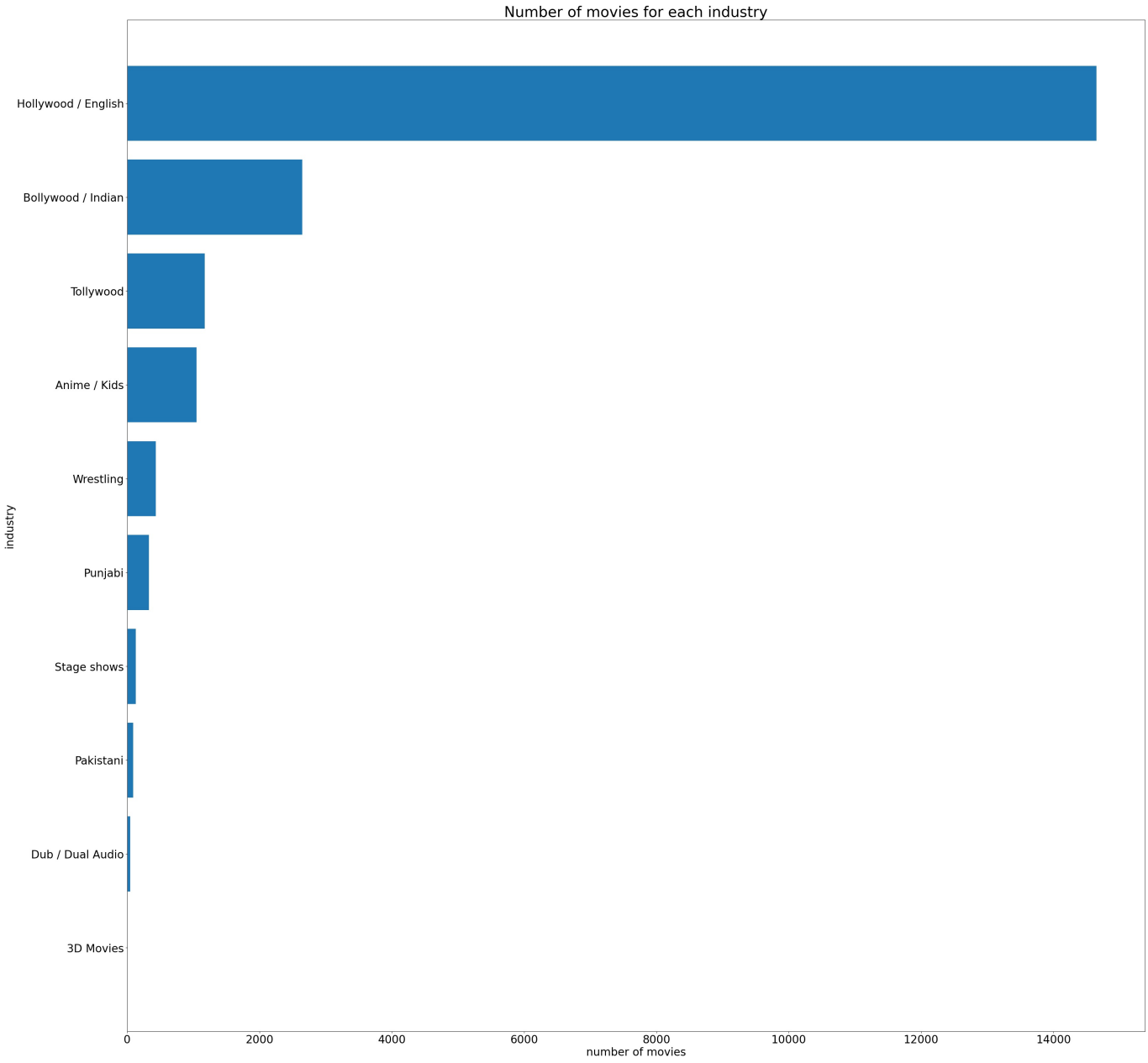
2.数据可视化

直方图

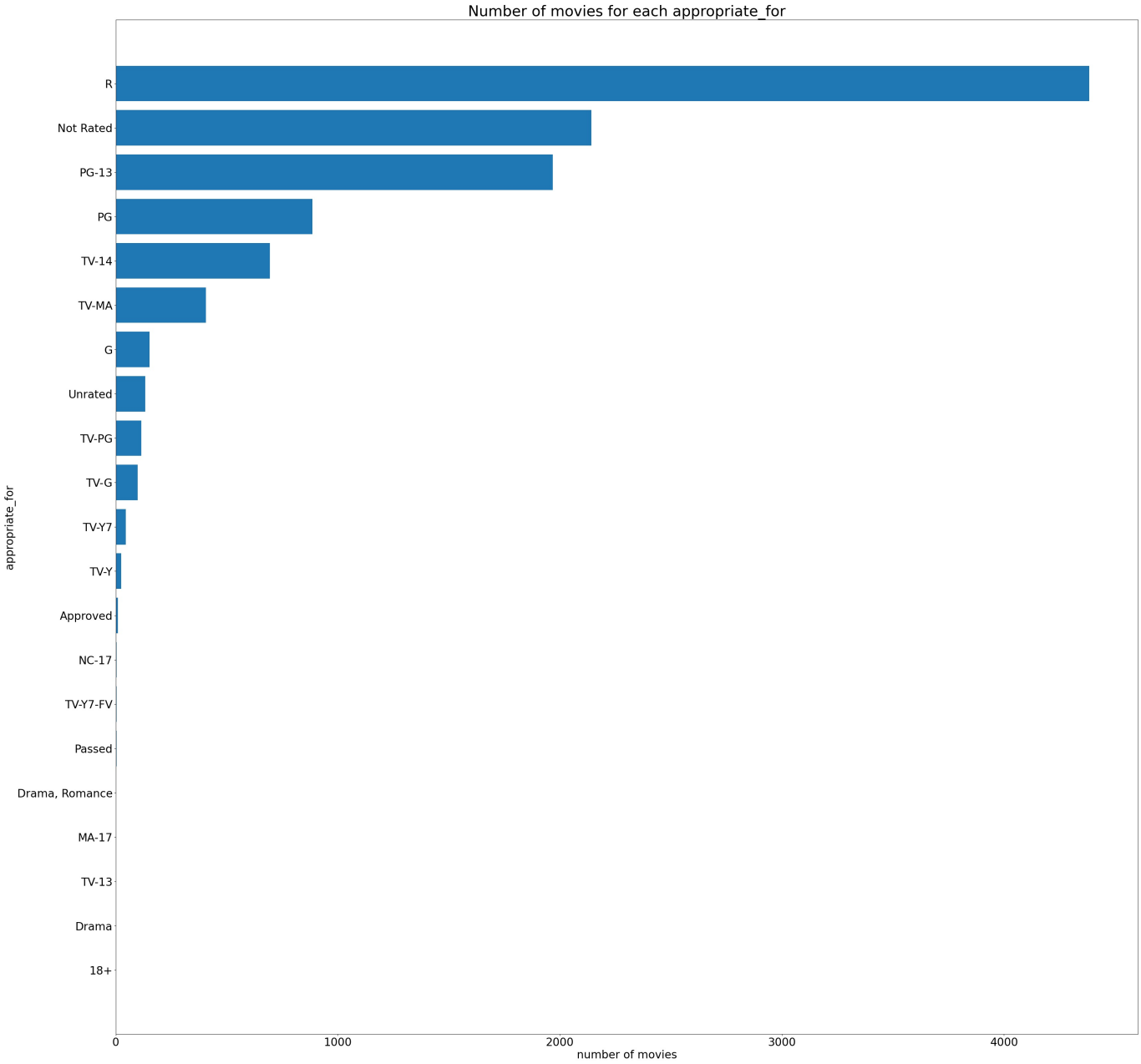
barh,hist

In [ ]:
```
# 定义画图函数
def draw_barh(attribute_name,size=0):
    attribute_count = pd.DataFrame(mov_data[attribute_name].value_counts()).sort_values(by='count', ascending=T
    plt.figure(figsize=(40, 40))
    if size == 0:
        plt.barh(attribute_count.index, width=attribute_count['count'])
    else :
        plt.barh(attribute_count.tail(size).index, width=attribute_count['count'].tail(size))
    plt.yticks(fontsize=24)
    plt.xticks(fontsize=24)
    plt.ylabel(attribute_name, fontsize=24)
    plt.xlabel('number of movies', fontsize=24)
    plt.title('Number of movies for each {}'.format(attribute_name), fontsize=32, loc='center')
    plt.show()
```

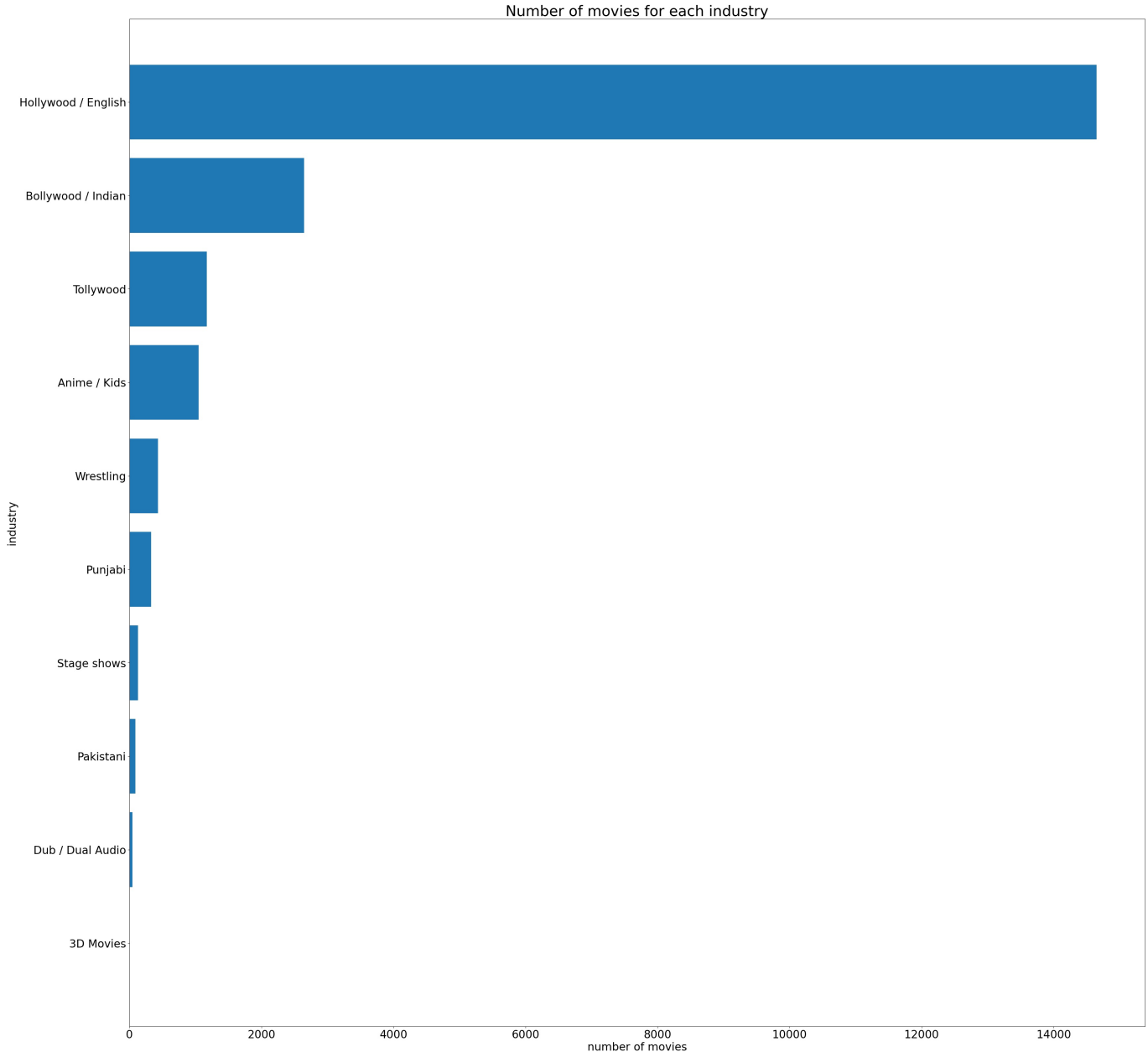
In [ ]:
```
draw_barh('industry')
```



In [ ]:
```
draw_barh('appropriate_for')
```

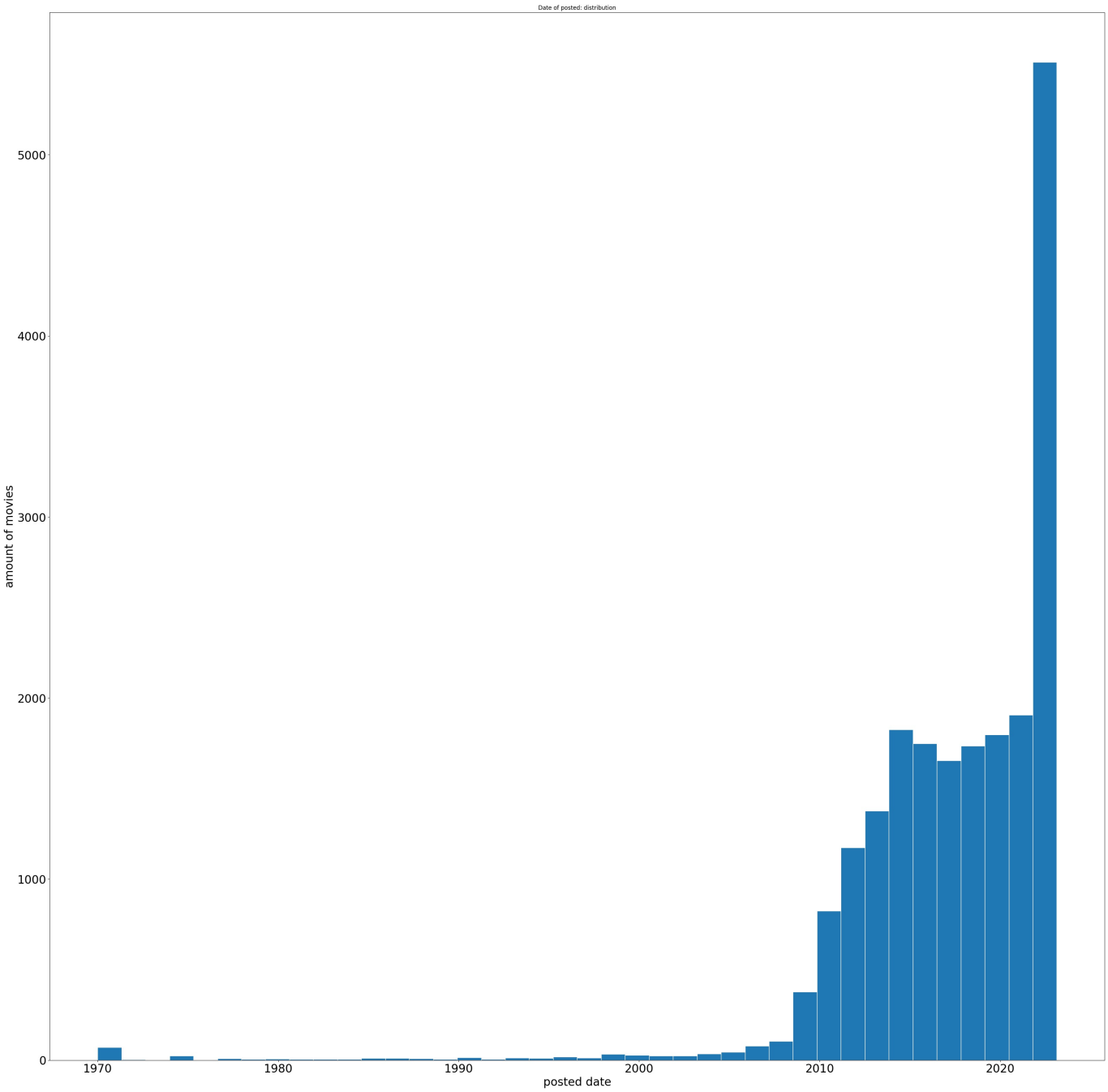


In [ ]:
```
draw_barh('industry')
```

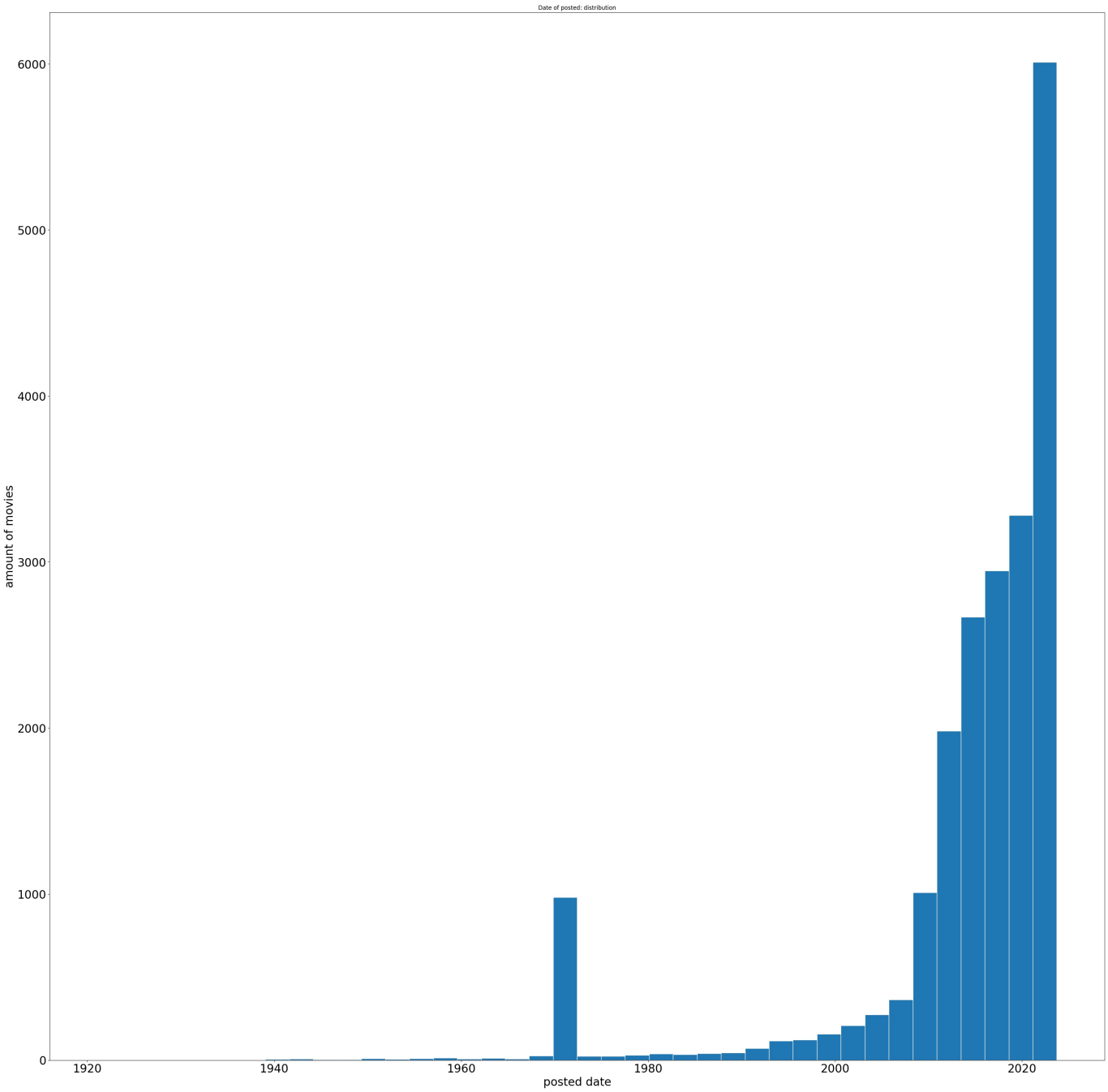


In [ ]:
```
# 构建画图函数
def draw_hist(attribute_name):
    plt.figure(figsize=(40, 40))
    plt.hist(mov_data[attribute_name], bins=40, edgecolor='white')
    plt.title('Date of posted: distribution')
    plt.yticks(fontsize=24)
    plt.xticks(fontsize=24)
    plt.ylabel('amount of movies', fontsize=24)
    plt.xlabel('posted date', fontsize=24)
    plt.show()
```

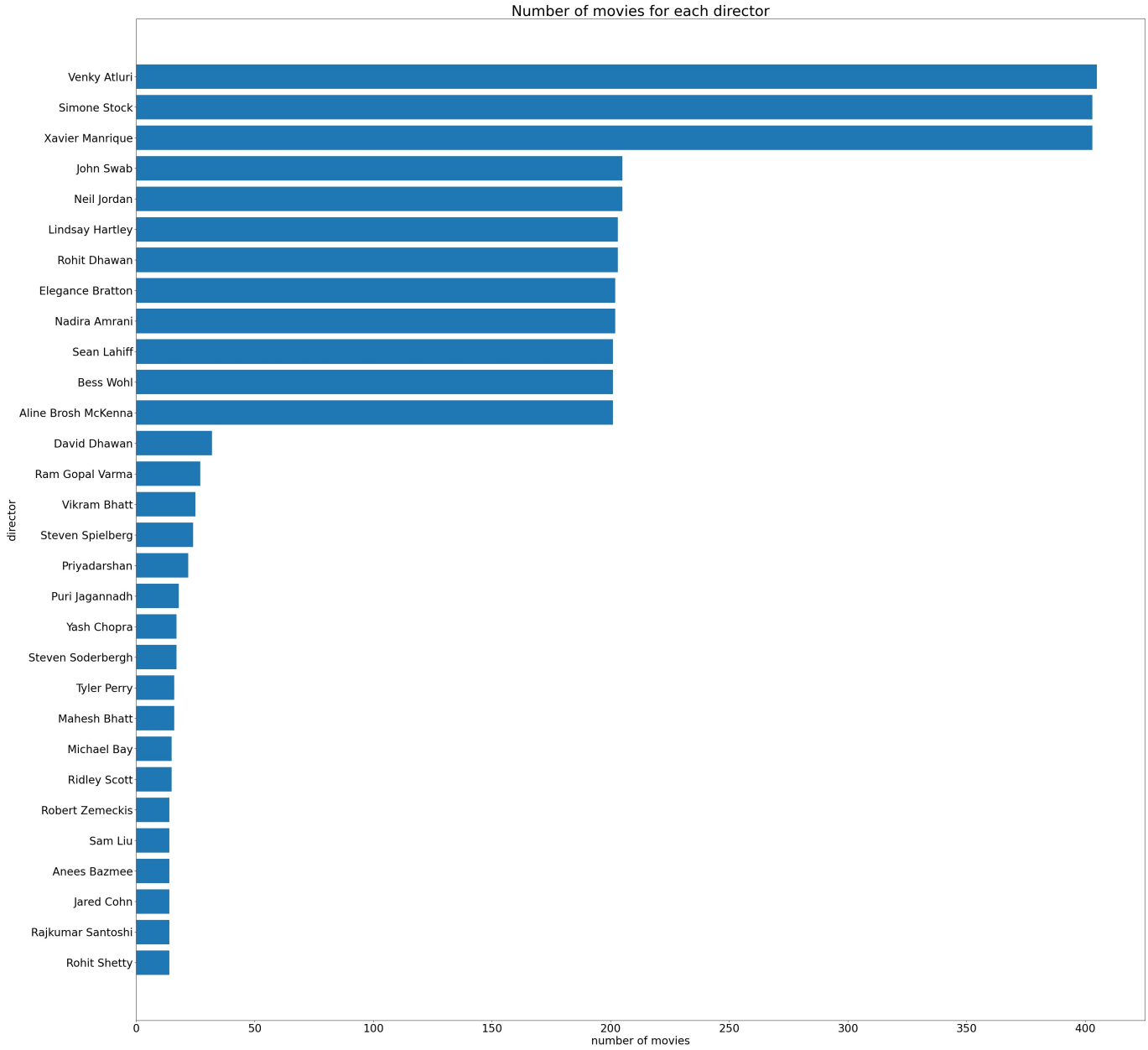
In [ ]:
```
draw_hist('posted_date')
```



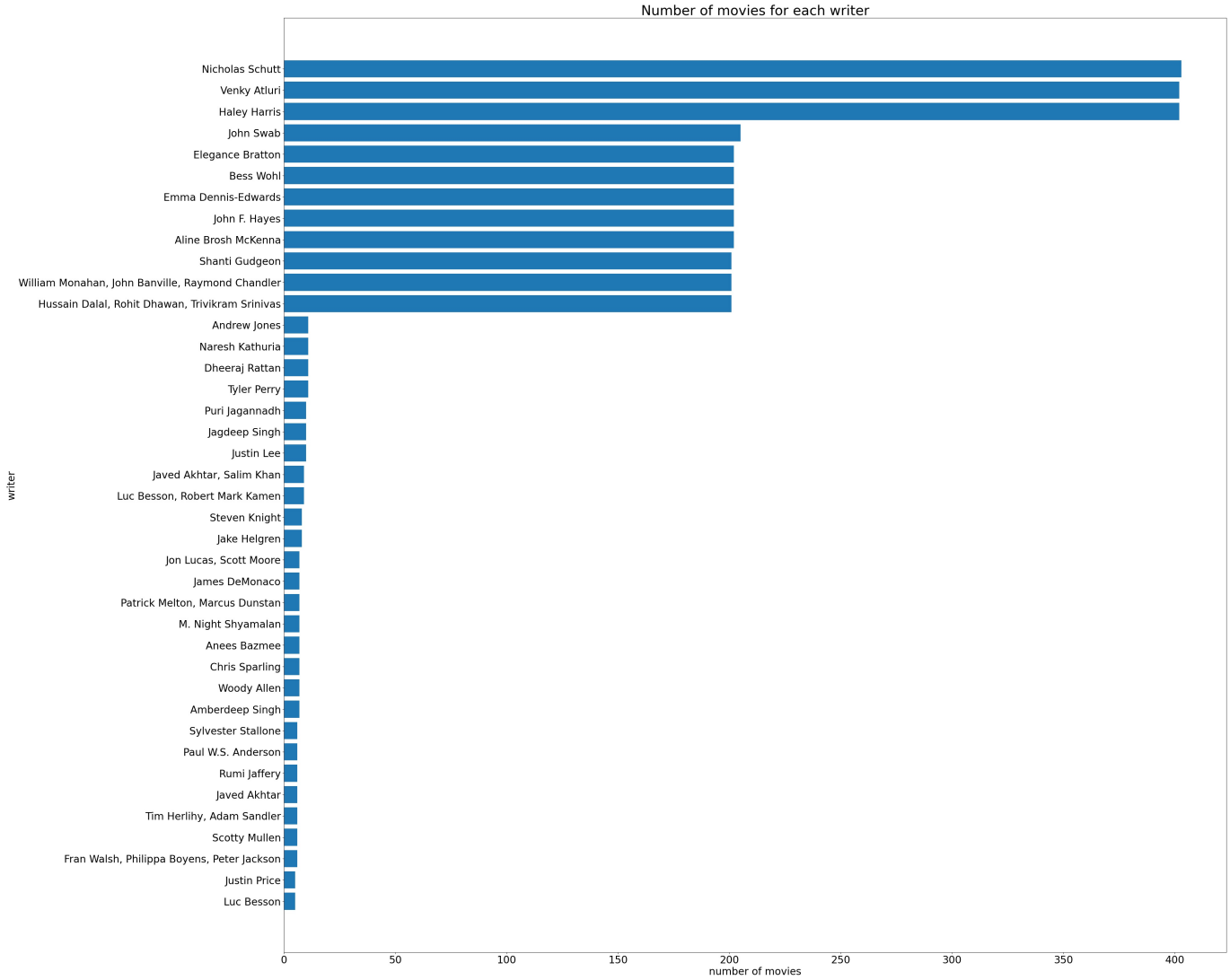
In [ ]:
```
draw_hist('release_date')
```



In [ ]:
```
draw_barh('director',30)
```



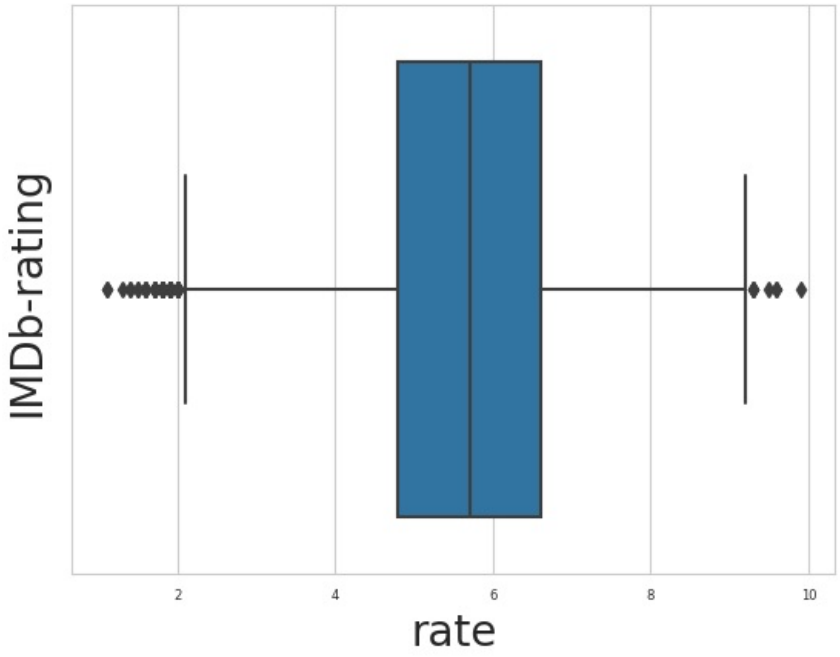
In [ ]:
```
draw_barh('writer',40)
```



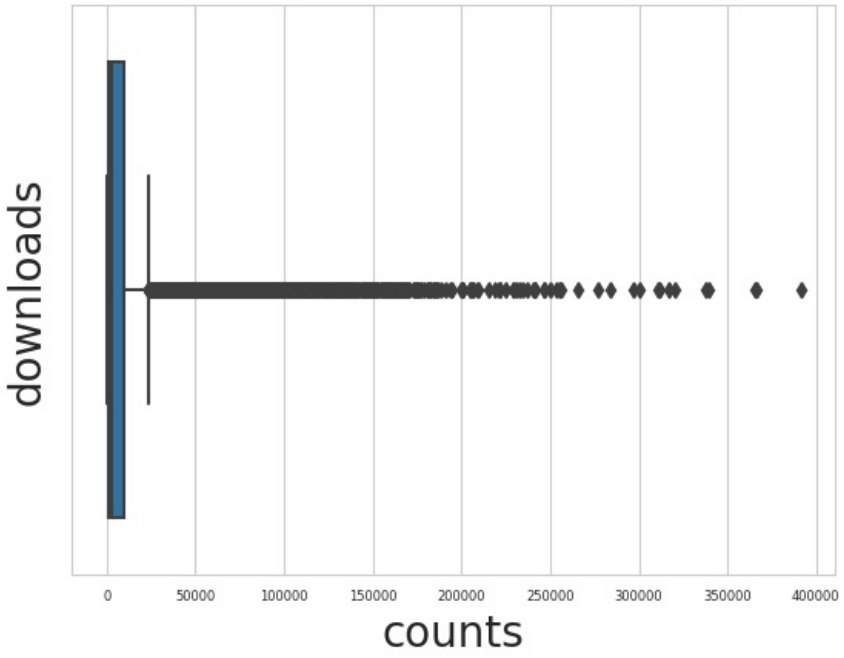
盒图

In [ ]:
```
def draw_box(attribute_name,x_label='counts',showFliers=True,data_draw=mov_data):
    sns.set_style("whitegrid")
    # operations_count_box = sns.catplot(data=operation_counts, kind='box', y='operation', x='counts', height=20
    sns.boxplot(x=attribute_name,data=data_draw,showfliers=showFliers)
    plt.yticks(fontsize=16)
    plt.xticks(fontsize=6)
    plt.ylabel(attribute_name, fontsize=20)
    plt.xlabel(x_label, fontsize=20)
    plt.show()
```

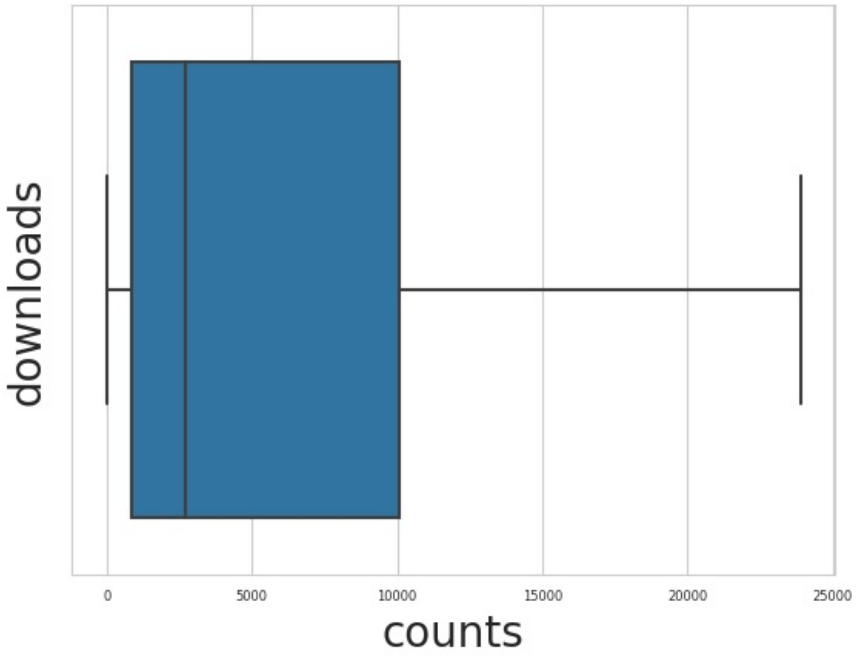
In [ ]:
```
draw_box('IMDb-rating',x_label='rate')
```



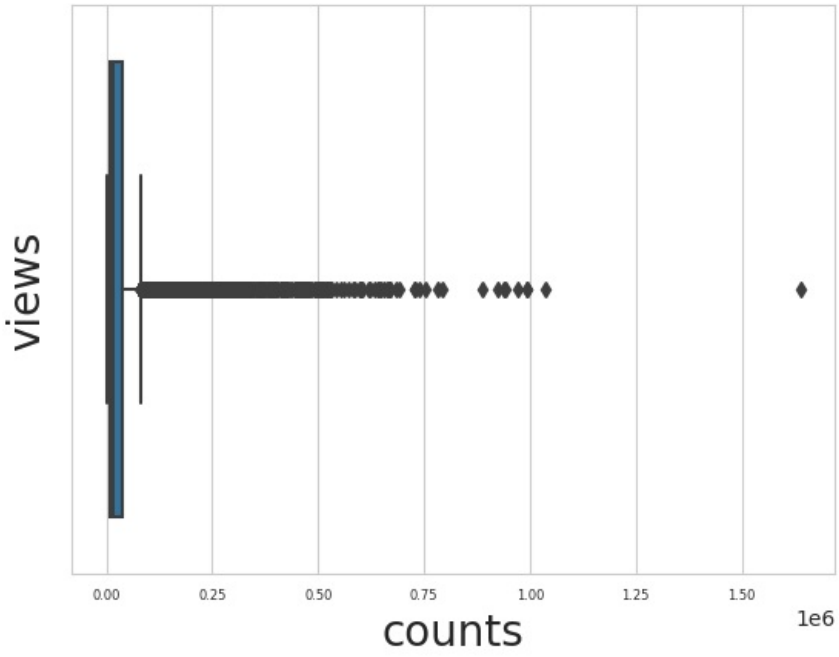
In [ ]:
```
draw_box('downloads',showFliers=True)
```



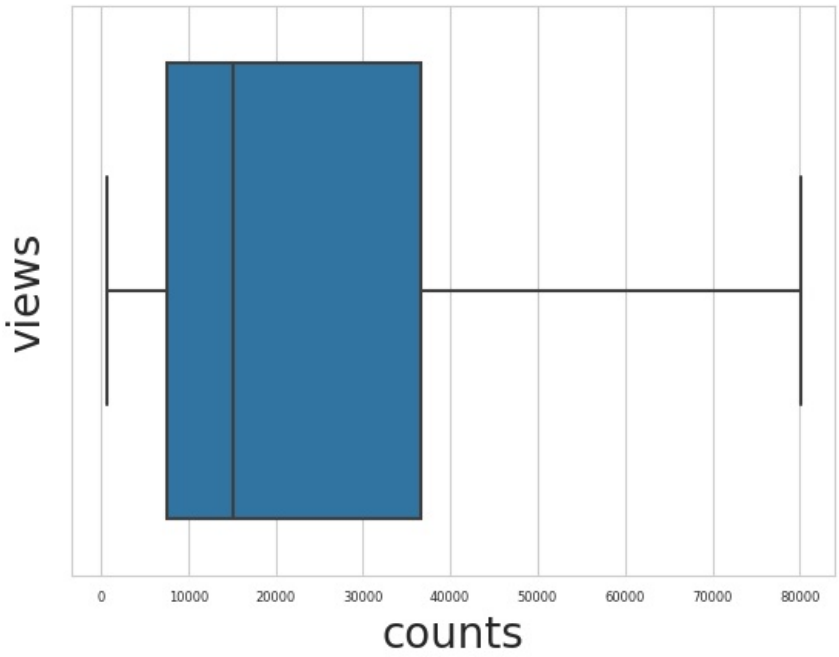
In [ ]:
```
draw_box('downloads',showFliers=False)
```



In [ ]:
```
draw_box('views')
```



In [ ]:
```
draw_box('views',showFliers=False)
```



3.数据缺失值处理

缺失值分析

In [ ]:
```
dtype_uni_miss(mov_data)
```

Out[ ]:

	dtype	eg.	num_unique	null_share
IMDb-rating	float64	4.8	85	4.09%
appropriate_for	object	R	21	46.12%
director	object	John Swab	9672	9.43%
downloads	float64	304.0	10625	0.00%
id	object	372092	17086	0.00%
industry	object	Hollywood / English	10	0.00%
language	object	English	1167	2.66%
posted_date	datetime64[ns]	2023-02-20 00:00:00	4123	0.00%
release_date	datetime64[ns]	2023-01-28 00:00:00	4886	0.00%
run_time	object	105	415	8.60%
storyline	object	Doc\r\n facilitates a fragile truce between th...	15748	8.28%
title	object	Little Dixie	16572	0.00%
views	float64	2794.0	16821	0.00%
writer	object	John Swab	13603	10.67%

In [ ]:
```
cols, nulls = [], []

for col in mov_data:
    cols.append(col)
    null = mov_data[col].isna().sum()
    nulls.append(null)

df = pd.DataFrame(nulls, index=cols, columns=['num_null']).T
df
```

Out[ ]:

	IMDb-rating	appropriate_for	director	downloads	id	industry	language	posted_date	release_date	run_time	storyline	title
num_null	841	9476	1938	1	0	1	546	1	1	1768	1701	1

删除缺失值

In [ ]:
```
from numpy import nan as NA
```

In [ ]:
```
mov_data_cleaned = mov_data.dropna()
mov_data_cleaned
```

Out[ ]:

	IMDb-rating	appropriate_for	director	downloads	id	industry	language	posted_date	release_date
0	4.8	R	John Swab	304.0	372092	Hollywood / English	English	2023-02-20	2023-01-28
1	6.4	TV-PG	Paul Ziller	73.0	372091	Hollywood / English	English	2023-02-20	2023-02-05
2	5.2	R	Ben Wheatley	1427.0	343381	Hollywood / English	English,Hindi	2021-04-20	2021-06-18
7	6.5	R	Benjamin Caron	1781.0	371751	Hollywood / English	English	2023-02-13	2023-02-17
8	6.9	PG-13	Ravi Kapoor	458.0	372042	Hollywood / English	English	2023-02-18	2022-12-02
...	...	...	...	...	...	...	...	...	...
20522	7.1	Not Rated	Biren Nag	1932.0	23825	Bollywood / Indian	Hindi	1970-01-01	1962-05-11
20525	7.0	G	Guy Hamilton	2544.0	25548	Hollywood / English	English,German,Polish,French	1970-01-01	1969-09-17
20533	5.6	R	Barbara Topsøe-Rothenborg	12284.0	1173	Hollywood / English	Spanish,German,English	2016-05-26	1970-01-01
20537	7.1	Not Rated	Biren Nag	1932.0	23825	Bollywood / Indian	Hindi	1970-01-01	1962-05-11
20540	7.0	G	Guy Hamilton	2544.0	25548	Hollywood / English	English,German,Polish,French	1970-01-01	1969-09-17

9902 rows × 14 columns

appropriate_for

In [ ]:
```
# def draw_comparison(attribute_name):
#     attribute_count = pd.DataFrame(mov_data[attribute_name].value_counts()).sort_values(by='count', ascending=

#     attribute_count_cleaned = attribute_count
#     attribute_count_cleaned['count_cleaned'] = [0] * len(attribute_count)

#     for level in list(attribute_count.index):
#         if level in list(mov_data_cleaned[attribute_name].value_counts().index):
#             attribute_count_cleaned.loc[[level], ['count_cleaned']] = mov_data_cleaned[attribute_name].value_

#     plt.figure(figsize=(40, 40))
#     plt.yticks(fontsize=24)
#     plt.xticks(fontsize=24)
#     plt.barh(attribute_count_cleaned.index, width=attribute_count_cleaned['count'], label='attribute_count')
#     plt.barh(attribute_count_cleaned.index, width=attribute_count_cleaned['count_cleaned'], label='attribute_
#     plt.ylabel(attribute_name, fontsize=24)
#     plt.xlabel('number of movies', fontsize=24)
#     plt.title('Number of movies for each attribute', fontsize=32, loc='center')
```

```
#     plt.legend(fontsize=32, loc='lower right')
#     plt.show()
```

In [ ]:
```
def draw_comparison(attribute_name, data_1=mov_data, data_2=mov_data_cleaned, top_n=None):
    """
    绘制属性对比图

    Args:
        attribute_name (str): 属性名称
        top_n (int, optional): 要展示的前n个属性。默认为None，表示展示所有属性。

    Returns:
        None
    """
    attribute_count = pd.DataFrame(data_1[attribute_name].value_counts()).sort_values(by='count', ascending=Tru

    if top_n is not None:
#         attribute_count_cleaned = attribute_count_cleaned.iloc[-top_n:]
        attribute_count = attribute_count.iloc[-top_n:]

    attribute_count_cleaned = attribute_count
    attribute_count_cleaned['count_cleaned'] = [0] * len(attribute_count)



    for level in list(attribute_count.index):
        if level in list(data_2[attribute_name].value_counts().index):
            attribute_count_cleaned.loc[[level], ['count_cleaned']] = data_2[attribute_name].value_counts().loc


    plt.figure(figsize=(40, 40))
    plt.yticks(fontsize=24)
    plt.xticks(fontsize=24)
    plt.barh(attribute_count_cleaned.index, width=attribute_count_cleaned['count'], label='attribute_count')
    plt.barh(attribute_count_cleaned.index, width=attribute_count_cleaned['count_cleaned'], label='attribute_co
    plt.ylabel(attribute_name, fontsize=24)
    plt.xlabel('number of movies', fontsize=24)
    plt.title(f'Number of movies for each {attribute_name}', fontsize=32, loc='center')
    plt.legend(fontsize=32, loc='lower right')
    plt.show()
```

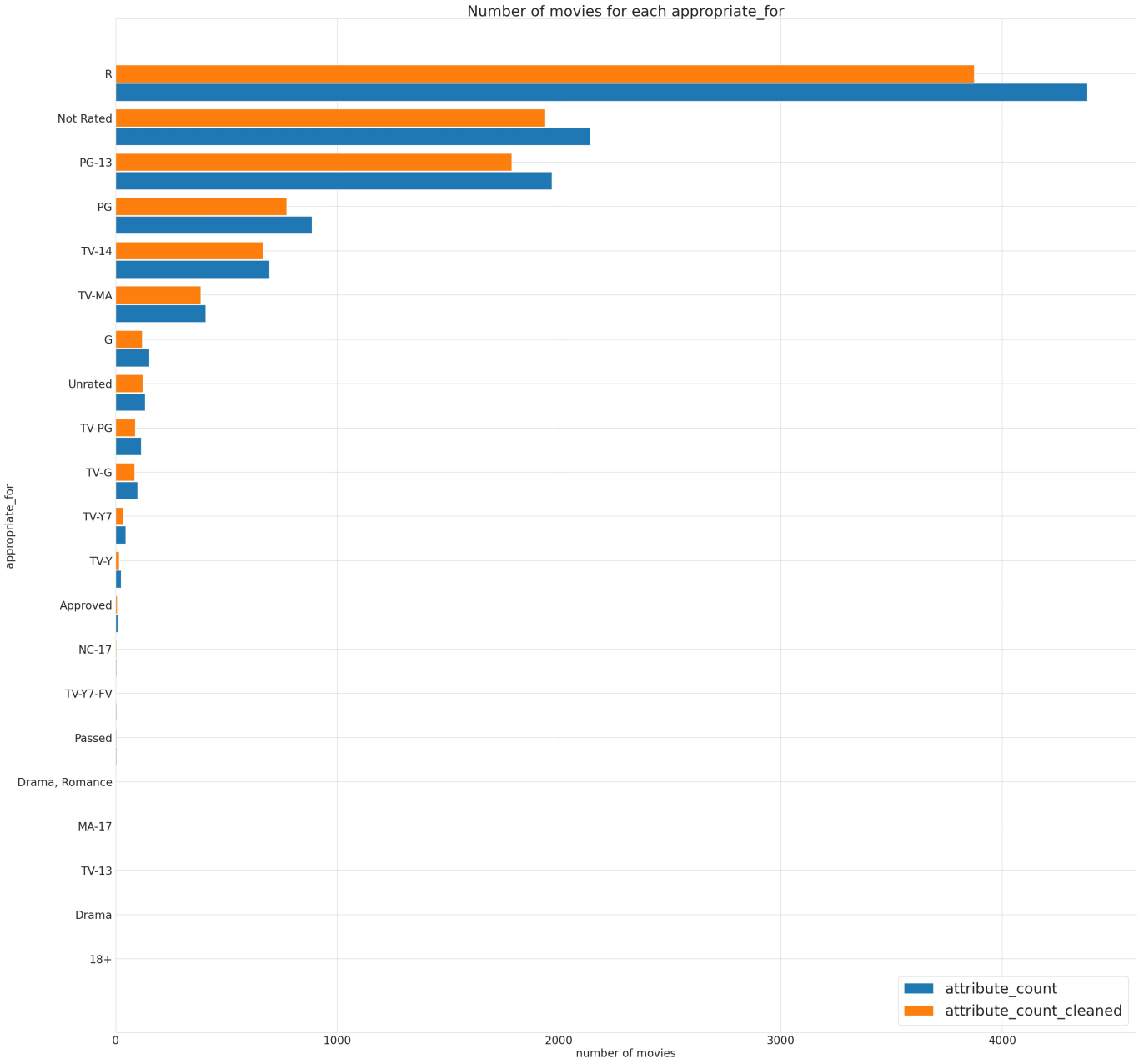
In [ ]:
```
def draw_comparison_1(attribute_name, data_1=mov_data, data_2=mov_data_cleaned, top_n=None):
    """
    绘制属性对比图

    Args:
        attribute_name (str): 属性名称
        top_n (int, optional): 要展示的前n个属性。默认为None，表示展示所有属性。

    Returns:
        None
    """
    attribute_count = pd.DataFrame(data_1[attribute_name].value_counts()).sort_values(by='count', ascending=Tru

    if top_n is not None:
#         attribute_count_cleaned = attribute_count_cleaned.iloc[-top_n:]
        attribute_count = attribute_count.iloc[-top_n:]

    attribute_count_cleaned = attribute_count
    attribute_count_cleaned['count_cleaned'] = [0] * len(attribute_count)



    for level in list(attribute_count.index):
        if level in list(data_2[attribute_name].value_counts().index):
            attribute_count_cleaned.loc[[level], ['count_cleaned']] = data_2[attribute_name].value_counts().loc


    plt.figure(figsize=(40, 40))
    plt.yticks(fontsize=24)
    plt.xticks(fontsize=24)
    plt.barh(y=list(range(len(attribute_count_cleaned))), tick_label=attribute_count_cleaned.index, width=attri
    plt.barh(y=[d+0.42 for d in list(range(len(attribute_count_cleaned)))], tick_label=attribute_count_cleaned.
    plt.ylabel(attribute_name, fontsize=24)
    plt.xlabel('number of movies', fontsize=24)
    plt.title(f'Number of movies for each {attribute_name}', fontsize=32, loc='center')
    plt.legend(fontsize=32, loc='lower right')
    plt.show()
```

In [ ]:
```
appropriate_count = pd.DataFrame(mov_data['appropriate_for'].value_counts()).sort_values(by='count', ascending=
appropriate_count
```

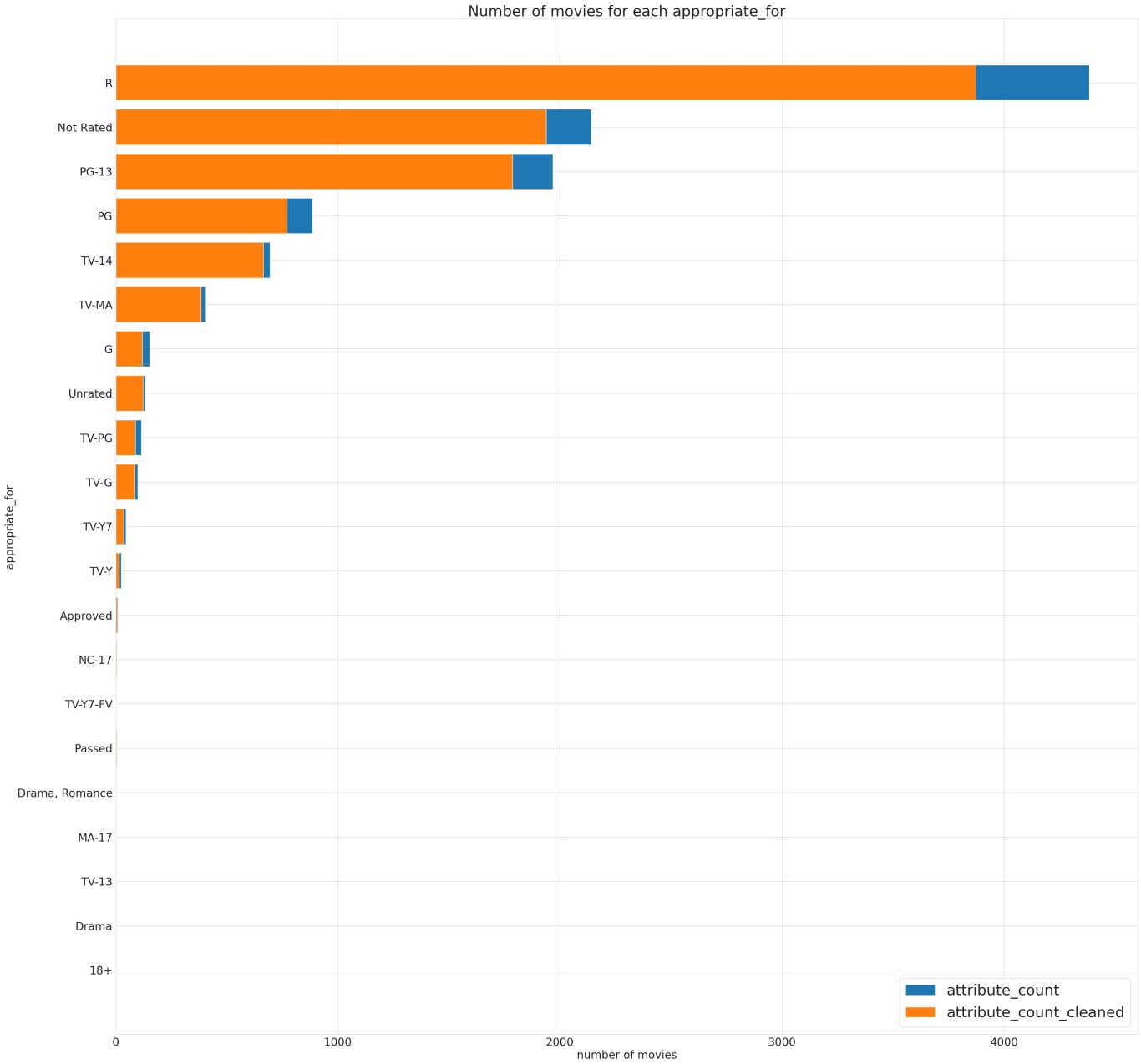
Out[ ]:

appropriate_for	count
18+	1
Drama	1
TV-13	1
MA-17	1
Drama, Romance	1
Passed	3
TV-Y7-FV	3
NC-17	4
Approved	9
TV-Y	25
TV-Y7	45
TV-G	99
TV-PG	115
Unrated	132
G	152
TV-MA	406
TV-14	694
PG	886
PG-13	1968
Not Rated	2142
R	4384

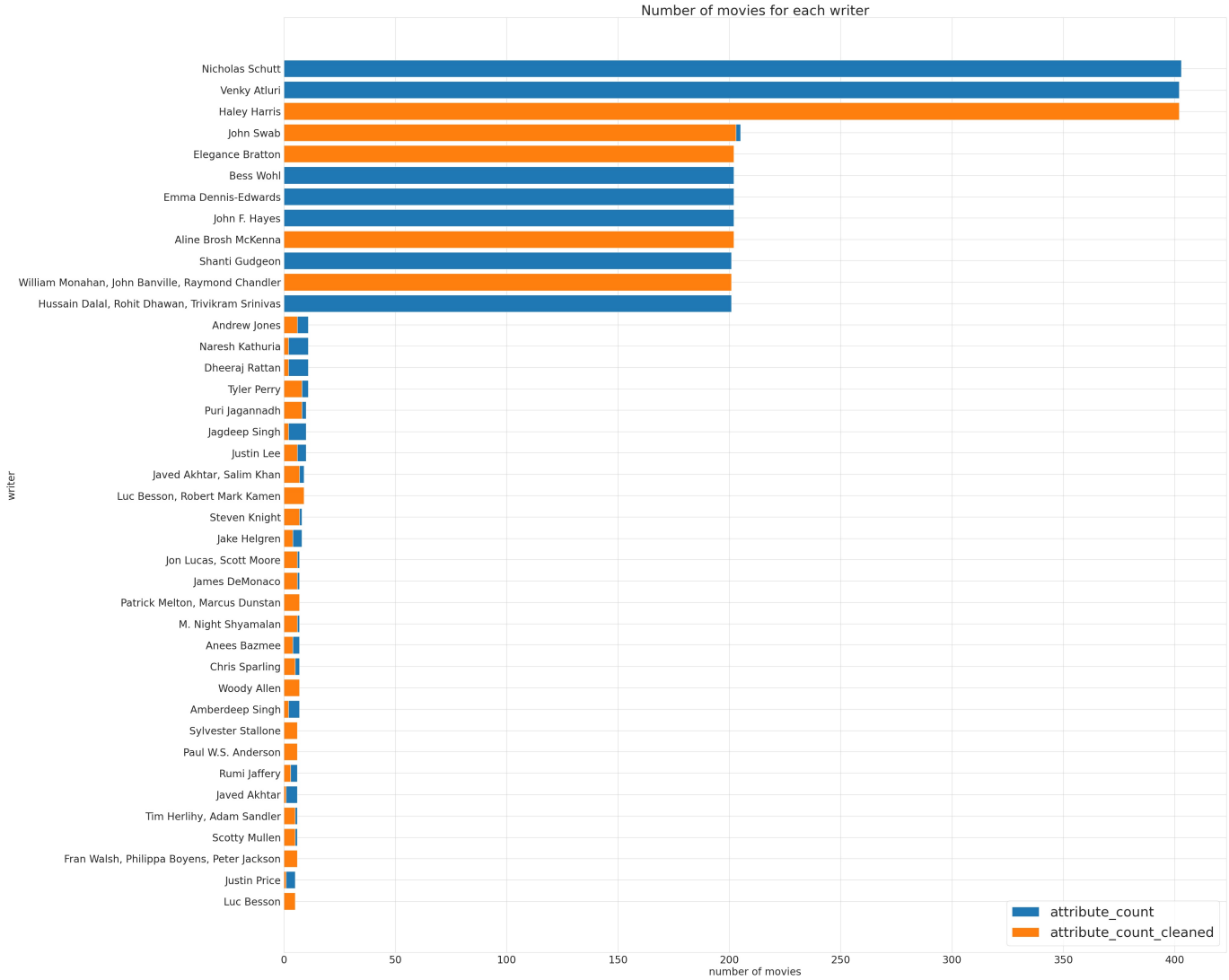
In [ ]:
```
draw_comparison_1('appropriate_for')
```



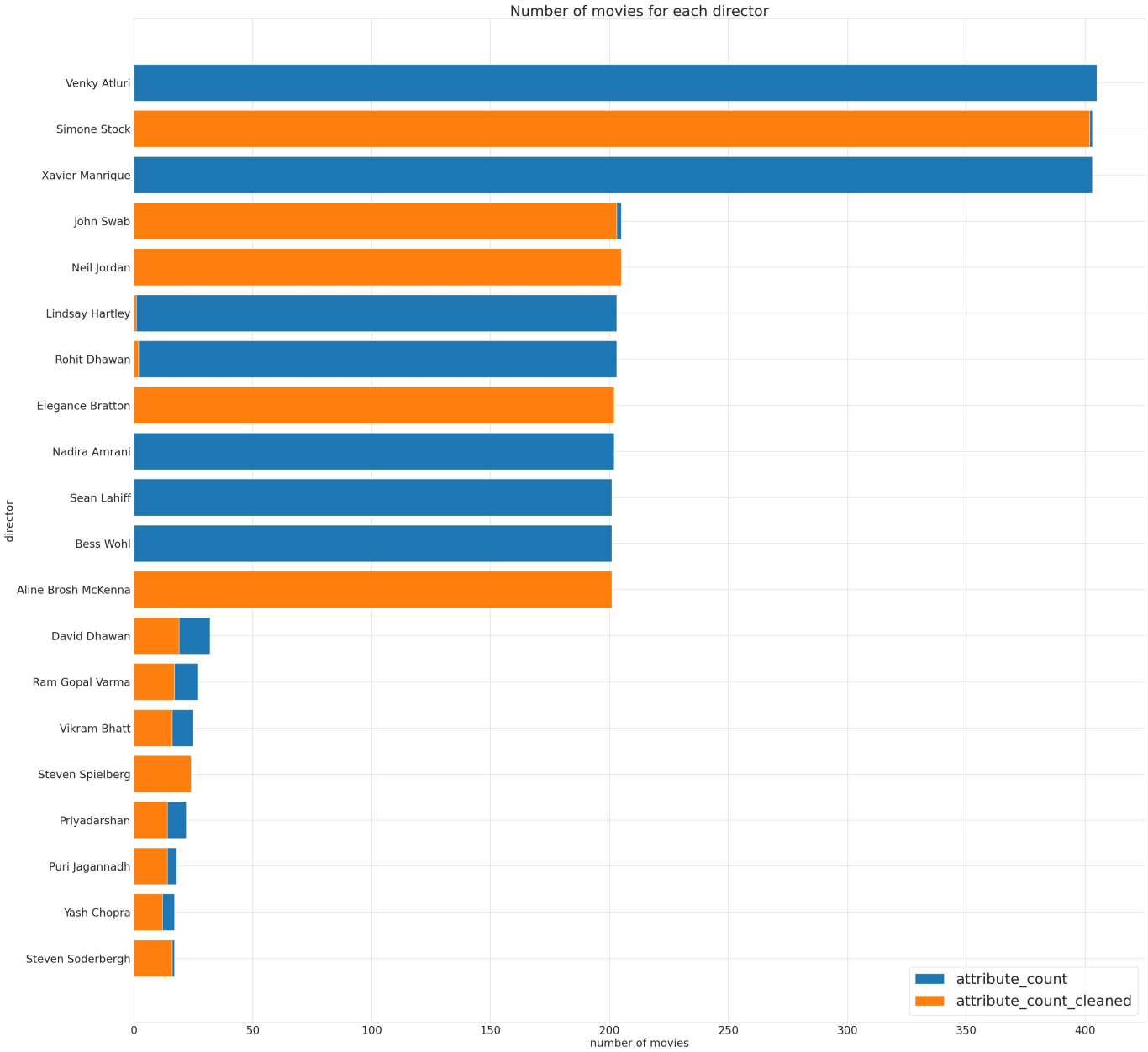
In [ ]:
```
draw_comparison('appropriate_for')
```



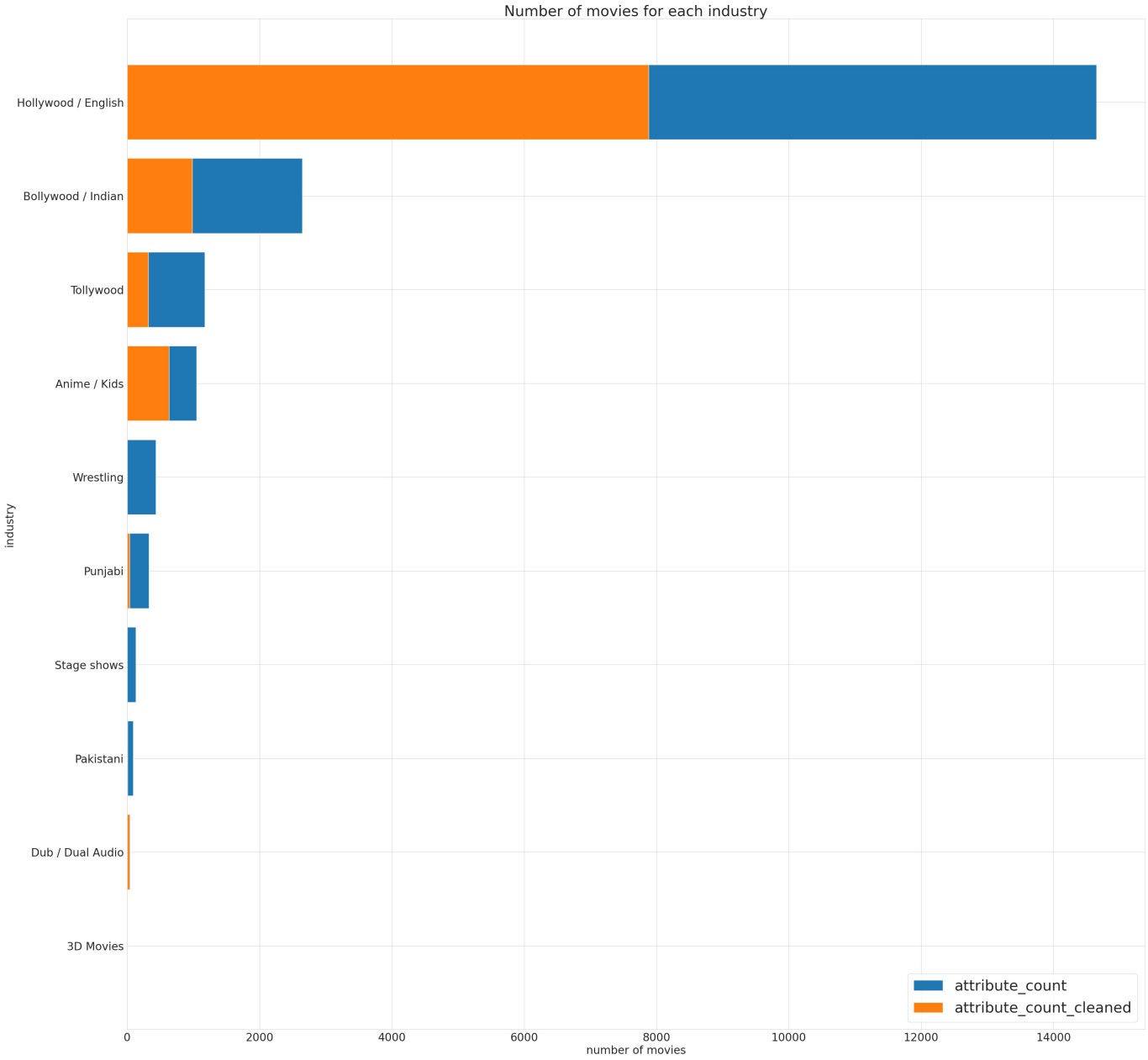
In [ ]:
```
draw_comparison('writer',top_n=40)
```



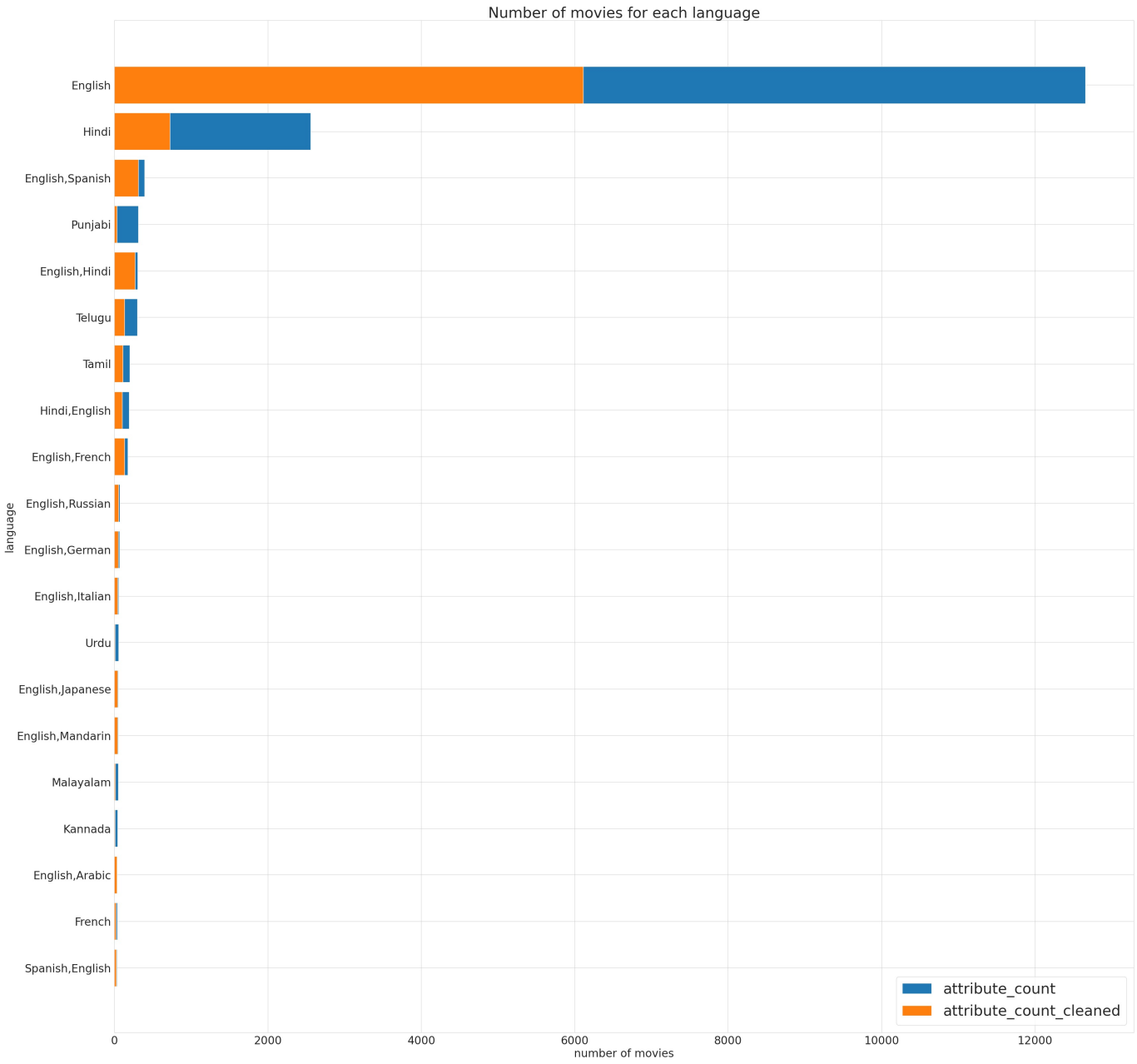
In [ ]:
```
draw_comparison('director',top_n=20)
```



In [ ]:
```
draw_comparison('industry')
```



In [ ]:
```
draw_comparison('language',top_n=20)
```

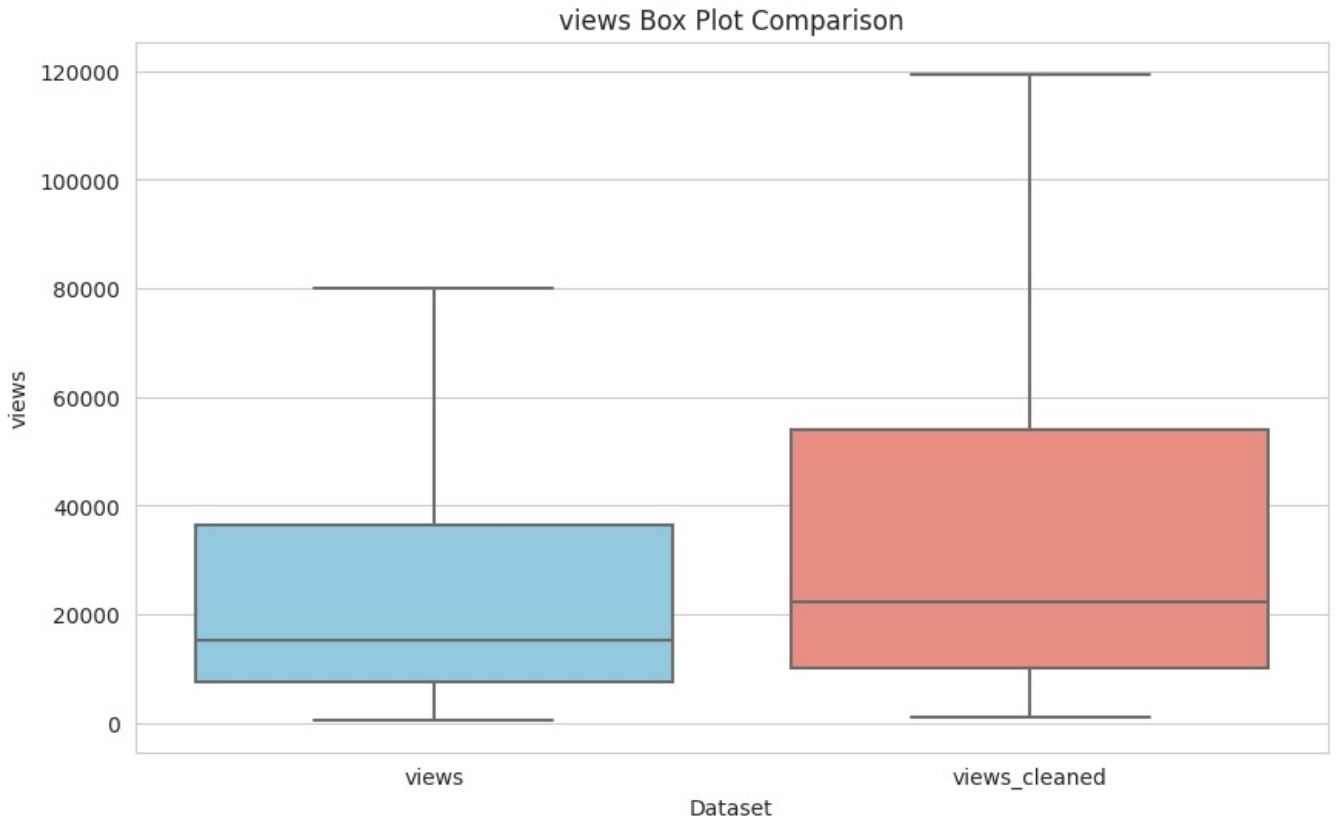


In [ ]:
```
#  def draw_box_comparison(attribute_name,x_label='counts',showFliers=True,data_1=mov_data,data_2=mov_data):

#     sns.set_style("whitegrid")
#     # operations_count_box = sns.catplot(data=operation_counts, kind='box', y='operation', x='counts', height=
#     sns.boxplot(x=attribute_name,data=data_1,showfliers=showFliers)
#     sns.boxplot(x=attribute_name,data=data_2,showfliers=showFliers,color='red')
#     plt.yticks(fontsize=16)
#     plt.xticks(fontsize=6)
#     plt.ylabel(attribute_name, fontsize=20)
#     plt.xlabel(x_label, fontsize=20)
#     plt.show()
```

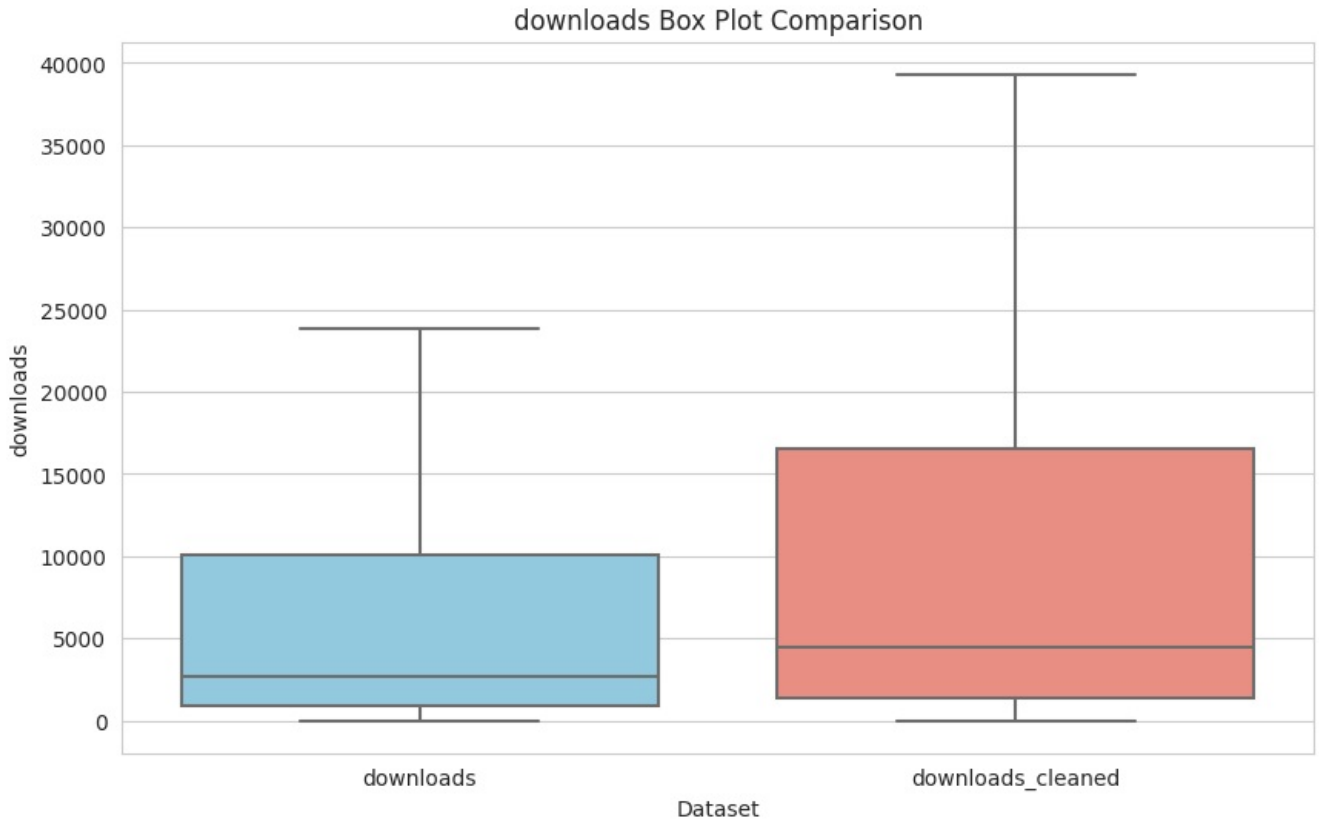
数值属性的对比

In [ ]:
```
def draw_box_comparison(attribute_name,showFliers=True,data_1=mov_data,data_2=mov_data):
# Assuming your datasets are stored in DataFrames named mov_data and mov_data_cleaned
    attribute_data = data_1[attribute_name]
    attribute_data_cleaned = data_2[attribute_name]

    plt.figure(figsize=(10, 6))
    sns.boxplot(data=[attribute_data, attribute_data_cleaned], showfliers=showFliers, palette=['skyblue', 'salm
    plt.title(attribute_name+' Box Plot Comparison')
    plt.xlabel('Dataset')
    plt.ylabel(attribute_name)
    plt.xticks(ticks=[0, 1], labels=[attribute_name, attribute_name+'_cleaned'])
    plt.show()
```

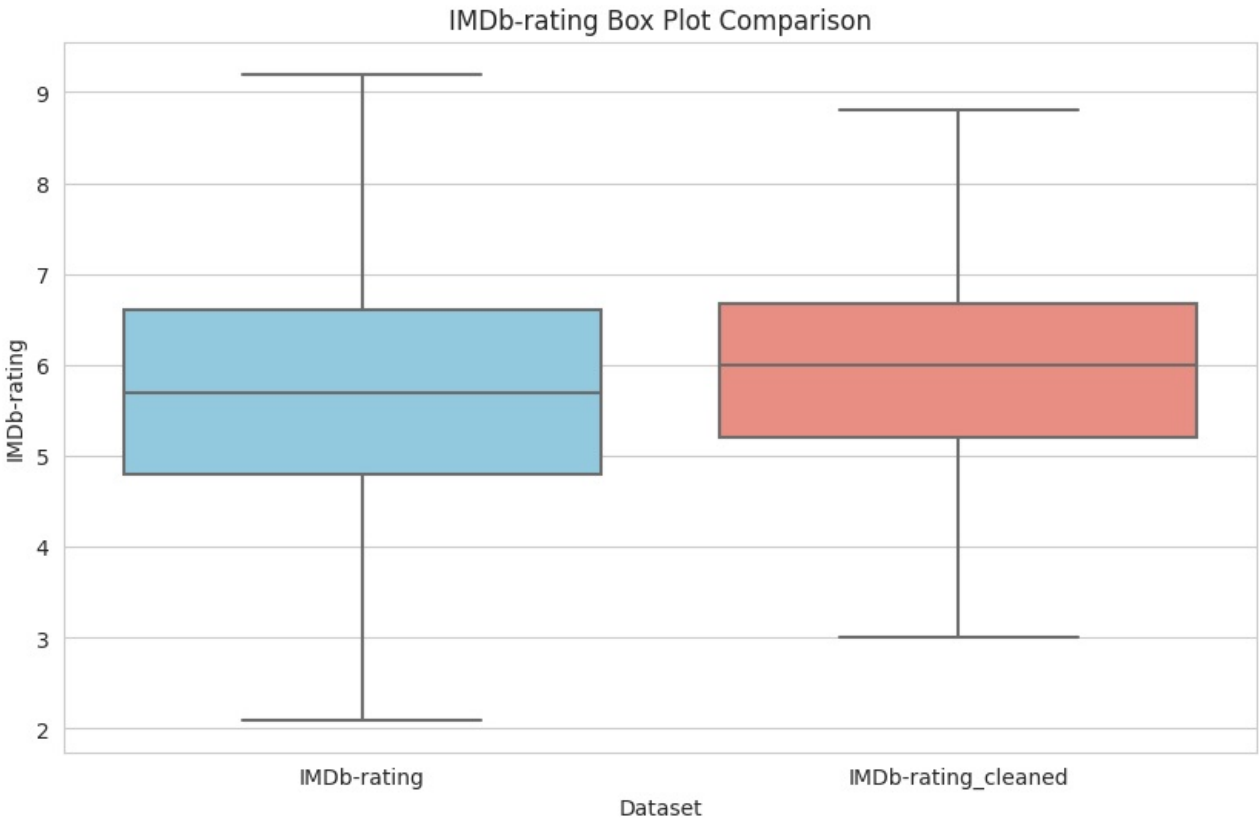
In [ ]:
```
draw_box_comparison('views',showFliers=False,data_1=mov_data,data_2=mov_data_cleaned)
```



In [ ]:
```
draw_box_comparison('downloads',showFliers=False,data_1=mov_data,data_2=mov_data_cleaned)
```



In [ ]:
```
draw_box_comparison('IMDb-rating',showFliers=False,data_1=mov_data,data_2=mov_data_cleaned)
```



将缺失值替换成频率最高的值

记为HF--Highest Frequency

In [ ]:
```
import copy
HF = {}
mov_data_HF_replaced = copy.deepcopy(mov_data)
for col in mov_data_HF_replaced:
    HF[col] = mov_data_HF_replaced[col].value_counts().index[0]
# HF['language'] = 'English'

# 取每一列的频率最高值
HF
```

Out[ ]:
```
{'IMDb-rating': 6.6,
 'appropriate_for': 'R',
 'director': 'Venky Atluri',
 'downloads': 75.0,
 'id': '372090',
 'industry': 'Hollywood / English',
 'language': 'English',
 'posted_date': Timestamp('2023-02-13 00:00:00'),
 'release_date': Timestamp('1970-01-01 00:00:00'),
 'run_time': '93',
 'storyline': 'The life of a young man and his struggles against the privatization of education.',
 'title': 'The Girl Who Escaped: The Kara Robinson Story',
 'views': 6259.0,
 'writer': 'Nicholas Schutt'}
```

In [ ]:
```
mov_data.isna().sum()
```

Out[ ]:
```
IMDb-rating         841
appropriate_for    9476
director           1938
downloads             1
id                    0
industry              1
language            546
posted_date           1
release_date          1
run_time           1768
storyline          1701
title                 1
views                 1
writer             2192
dtype: int64
```

In [ ]:
```
# 代替空值
```

```
for col in mov_data_HF_replaced:
    mov_data_HF_replaced[col] = mov_data_HF_replaced[col].replace(NA, HF[col])
mov_data_HF_replaced.isna().sum()
```

Out[ ]:
```
IMDb-rating        0
appropriate_for    0
director           0
downloads          0
id                 0
industry           0
language           0
posted_date        0
release_date       0
run_time           0
storyline          0
title              0
views              0
writer             0
dtype: int64
```

In [ ]:
```
mov_data_HF_replaced.head(10)
```

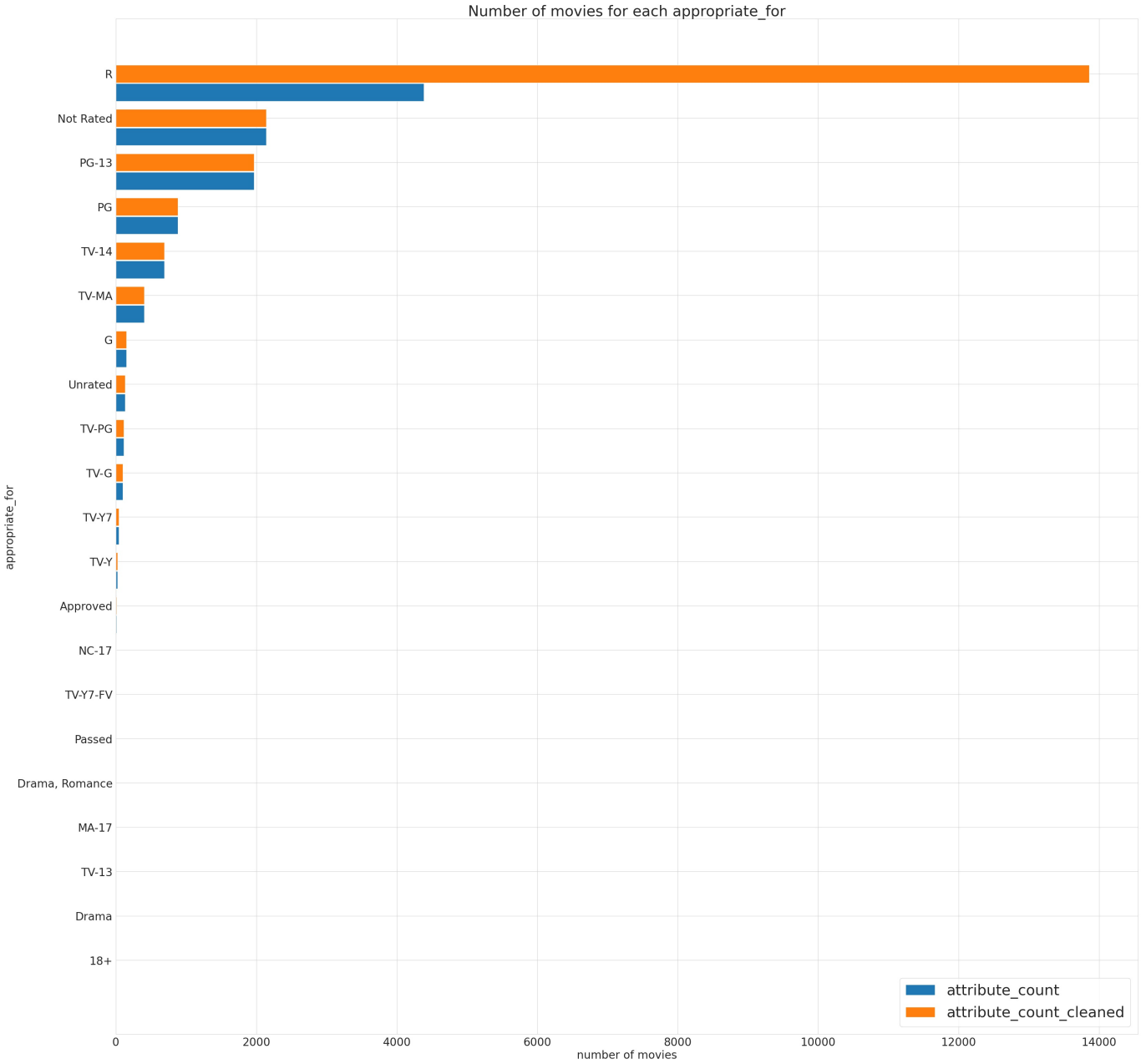
Out[ ]:

	IMDb-rating	appropriate_for	director	downloads	id	industry	language	posted_date	release_date	run_time	storyli
0	4.8	R	John Swab	304.0	372092	Hollywood / English	English	2023-02-20	2023-01-28	105	Doc\r facilitates fragile tru between th
1	6.4	TV-PG	Paul Ziller	73.0	372091	Hollywood / English	English	2023-02-20	2023-02-05	84	Caterer\r Goldy Ber reunites w detectiv
2	5.2	R	Ben Wheatley	1427.0	343381	Hollywood / English	English,Hindi	2021-04-20	2021-06-18	1h 47min	As the wo searches for cure to disastro
3	8.1	R	Venky Atluri	1549.0	372090	Tollywood	Hindi	2023-02-20	2023-02-17	139	The life o young ma and h struggl aga
4	4.6	R	Shaji Kailas	657.0	372089	Tollywood	Hindi	2023-02-20	2023-01-26	122	A man nam Kalidas ge stranded d to the p
5	5.4	R	Srinivas Gavireddy	746.0	372088	Tollywood	Hindi	2023-02-20	2021-11-26	131	Bagaram,\r after inheriti h grandfather
6	6.6	TV-PG	Venky Atluri	5332.0	372059	Wrestling	English	2023-02-19	2023-02-18	200	Undisputed\r WW Universal tit Reigns vs
7	6.5	R	Benjamin Caron	1781.0	371751	Hollywood / English	English	2023-02-13	2023-02-17	116	Motivatio are suspe a expectatio are
8	6.9	PG-13	Ravi Kapoor	458.0	372042	Hollywood / English	English	2023-02-18	2022-12-02	80	An\r unmotivat South Asi Americ rappe
9	4.2	R	Danny LeGare	1965.0	372041	Hollywood / English	English	2023-02-18	2023-02-07	80	A\r\n fam moves back a farm in t mothe

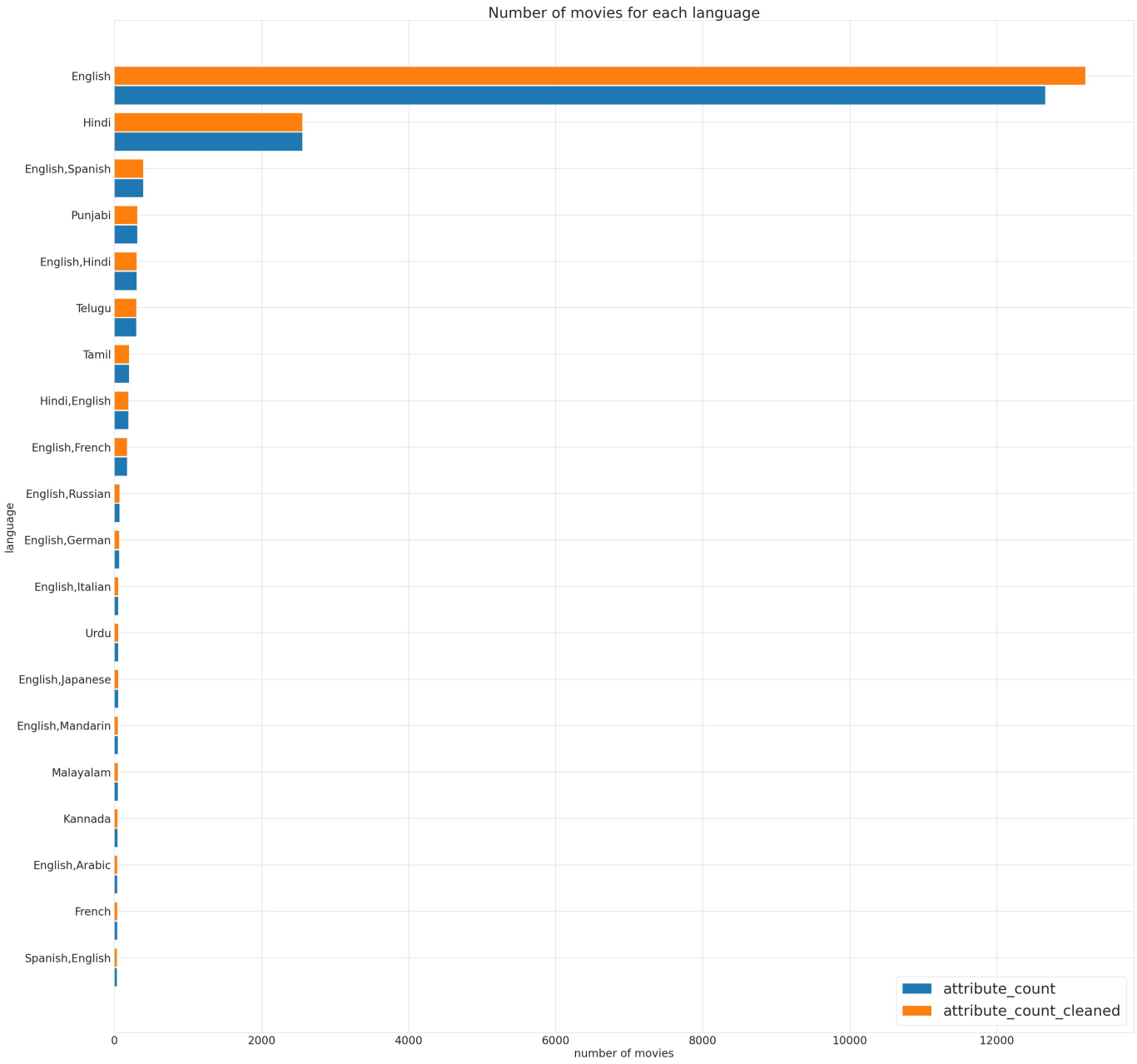
In [ ]:
```
def draw_comparison_2(attribute_name, data_1=mov_data, data_2=mov_data_cleaned, top_n=None):
    """
    绘制属性对比图

    Args:
        attribute_name (str): 属性名称
        top_n (int, optional): 要展示的前n个属性。默认为None，表示展示所有属性。
```

```
    Returns:
        None
    """
    attribute_count = pd.DataFrame(data_1[attribute_name].value_counts()).sort_values(by='count', ascending=Tru

    if top_n is not None:
        attribute_count = attribute_count.iloc[-top_n:]

    attribute_count_cleaned = attribute_count.copy()
    attribute_count_cleaned['count_cleaned'] = 0

    for level in attribute_count.index:
        if level in data_2[attribute_name].value_counts().index:
            attribute_count_cleaned.loc[level, 'count_cleaned'] = data_2[attribute_name].value_counts().loc[lev

    # Create separate subplots for the two attributes
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))

    # Plot the first attribute
    ax1.barh(attribute_count.index, width=attribute_count['count'], label='attribute_count')
    ax1.set_ylabel(attribute_name, fontsize=14)
    ax1.set_xlabel('Number of movies', fontsize=14)
    ax1.set_title(f'Number of movies for each {attribute_name}', fontsize=16)
    ax1.legend(loc='lower right')

    # Plot the second attribute
    ax2.barh(attribute_count_cleaned.index, width=attribute_count_cleaned['count_cleaned'], label='attribute_co
    ax2.set_ylabel(attribute_name, fontsize=14)
    ax2.set_xlabel('Number of movies', fontsize=14)
    ax2.set_title(f'Number of cleaned movies for each {attribute_name}', fontsize=16)
    ax2.legend(loc='lower right')

    plt.tight_layout()
    plt.show()
```

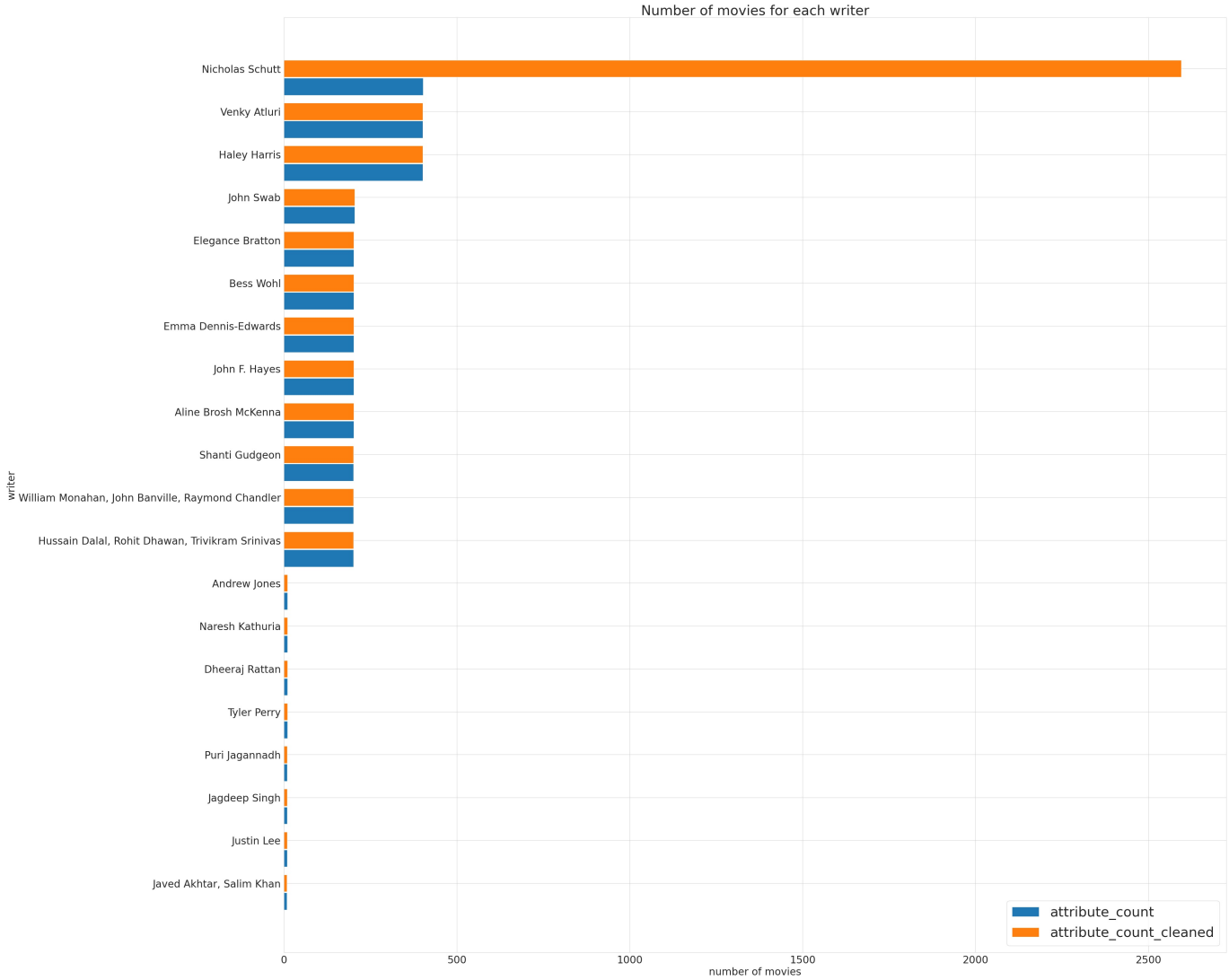
In [ ]:
```
draw_comparison_1('appropriate_for',data_2=mov_data_HF_replaced)
```



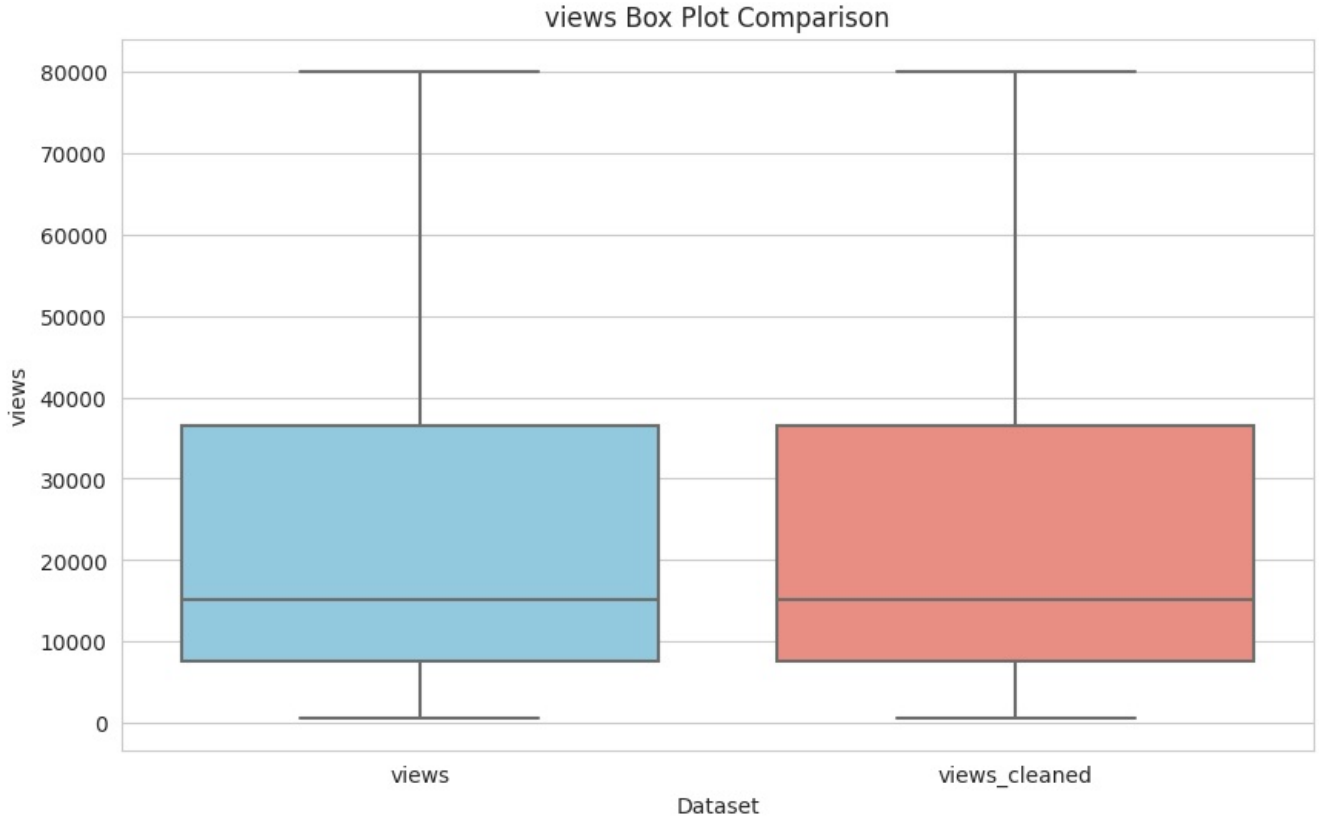
In [ ]:
```
draw_comparison_1('language', data_2=mov_data_HF_replaced,top_n=20)
```



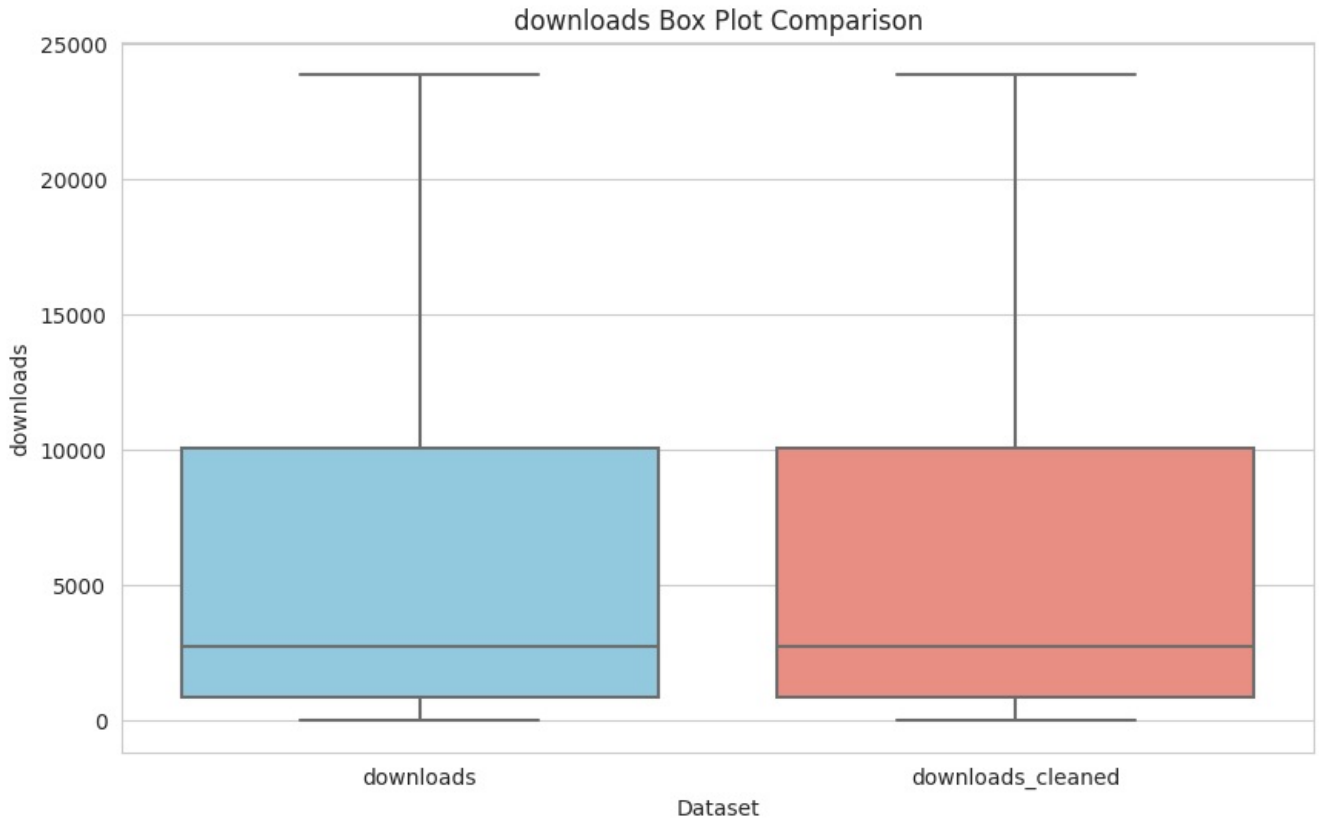
In [ ]:
```
draw_comparison_1('writer',data_2=mov_data_HF_replaced,top_n=20)
```



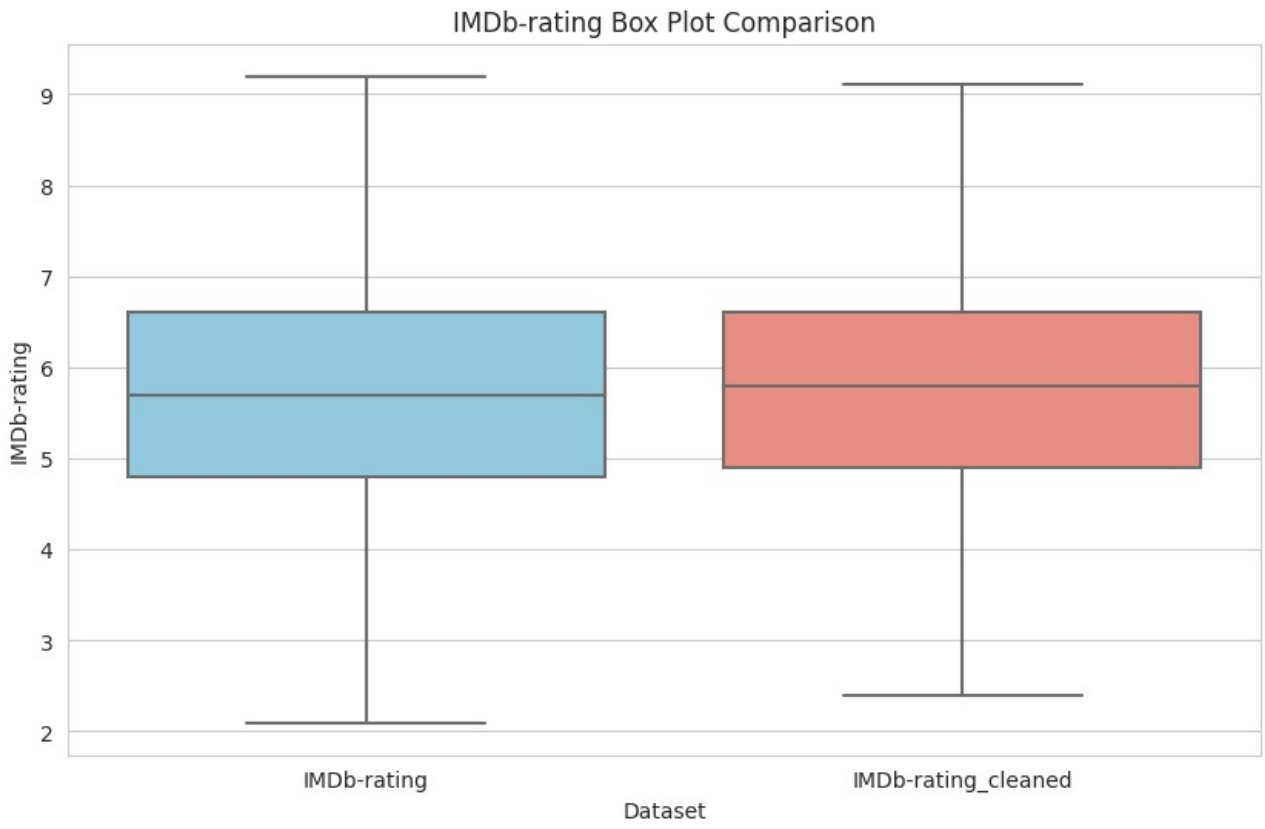
In [ ]:
```
draw_box_comparison('views',showFliers=False,data_1=mov_data,data_2=mov_data_HF_replaced)
```



In [ ]:
```
draw_box_comparison('downloads',showFliers=False,data_1=mov_data,data_2=mov_data_HF_replaced)
```



In [ ]:
```
draw_box_comparison('IMDb-rating',showFliers=False,data_1=mov_data,data_2=mov_data_HF_replaced)
```



通过属性的相关关系来填补缺失值

In [ ]:
```
mov_data_attr_corr = copy.deepcopy(mov_data)
```

In [ ]:
```
df_coded = pd.get_dummies(mov_data_attr_corr, columns=['appropriate_for'], dummy_na=True, drop_first=True)
df_coded.head()
```

Out[ ]:

	IMDb-rating	director	downloads	id	industry	language	posted_date	release_date	run_time	storyline	...	appropriate_fo
0	4.8	John Swab	304.0	372092	Hollywood / English	English	2023-02-20	2023-01-28	105	Doc\r\n facilitates a fragile truce between th...	...	
1	6.4	Paul Ziller	73.0	372091	Hollywood / English	English	2023-02-20	2023-02-05	84	Caterer\r\n Goldy Berry reunites with detectiv...	...	
2	5.2	Ben Wheatley	1427.0	343381	Hollywood / English	English,Hindi	2021-04-20	2021-06-18	1h 47min	As the world searches for a cure to a disastro...	...	
3	8.1	Venky Atluri	1549.0	372090	Tollywood	Hindi	2023-02-20	2023-02-17	139	The life of a young man and his struggles agai...	...	
4	4.6	Shaji Kailas	657.0	372089	Tollywood	Hindi	2023-02-20	2023-01-26	122	A man named Kalidas gets stranded due to the p...	...	

5 rows × 34 columns

In [ ]:
```
df_coded.corr(method='spearman', numeric_only=True)
```

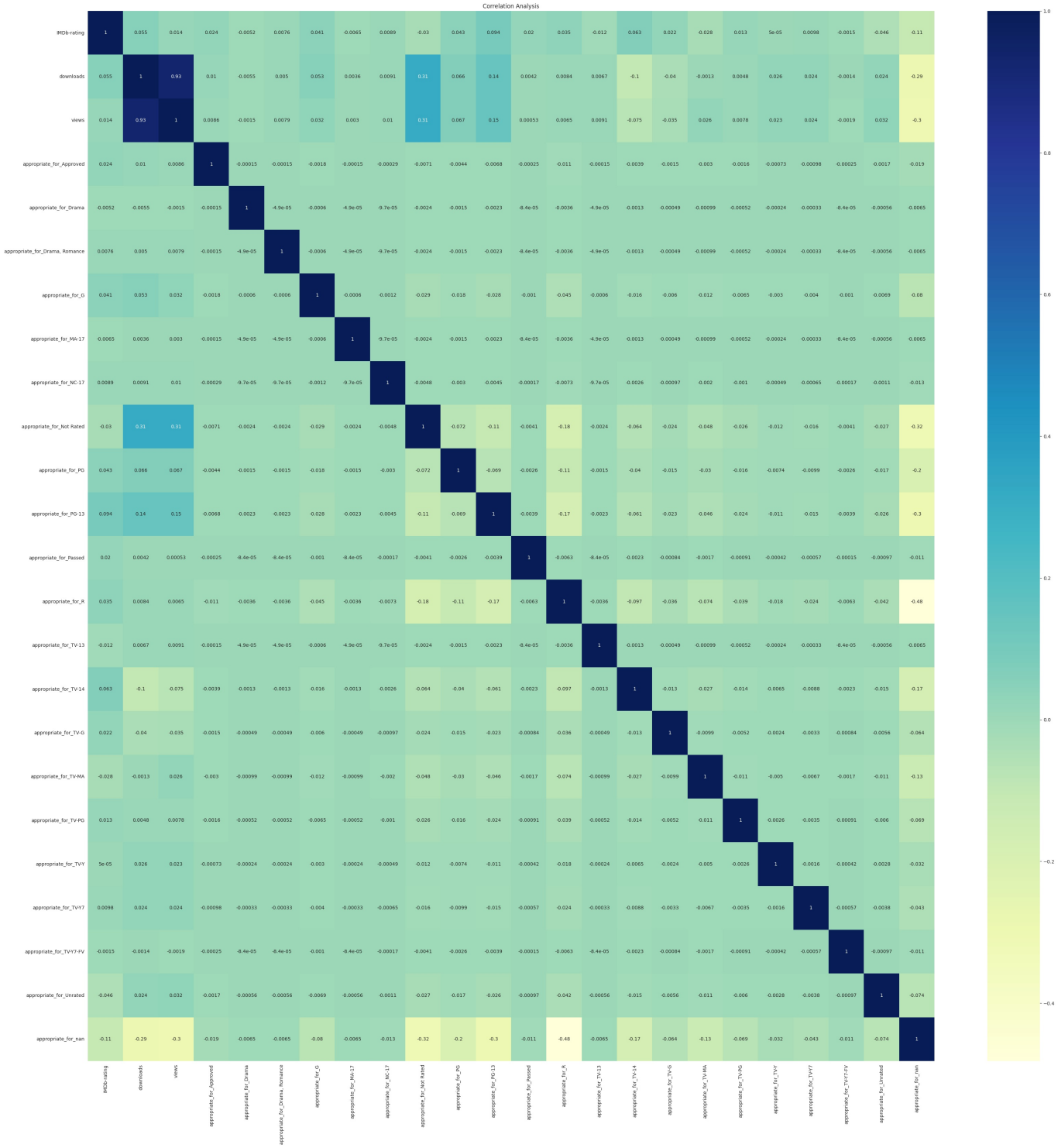
Out[ ]:

	IMDb-rating	downloads	views	appropriate_for_Approved	appropriate_for_Drama	appropriate_for_Dram Romanc
IMDb-rating	1.000000	0.055112	0.014251	0.023950	-0.005219	0.0075
downloads	0.055112	1.000000	0.932468	0.010468	-0.005535	0.0049
views	0.014251	0.932468	1.000000	0.008640	-0.001526	0.0078
appropriate_for_Approved	0.023950	0.010468	0.008640	1.000000	-0.000146	-0.0001
appropriate_for_Drama	-0.005219	-0.005535	-0.001526	-0.000146	1.000000	-0.0000
appropriate_for_Drama, Romance	0.007588	0.004985	0.007899	-0.000146	-0.000049	1.0000
appropriate_for_G	0.041430	0.052592	0.031512	-0.001807	-0.000602	-0.0006
appropriate_for_MA-17	-0.006480	0.003614	0.003000	-0.000146	-0.000049	-0.0000
appropriate_for_NC-17	0.008884	0.009131	0.010189	-0.000292	-0.000097	-0.0000
appropriate_for_Not Rated	-0.030294	0.313128	0.313190	-0.007141	-0.002380	-0.0023
appropriate_for_PG	0.042933	0.066343	0.066869	-0.004444	-0.001481	-0.0014
appropriate_for_PG-13	0.094077	0.143987	0.148471	-0.006813	-0.002270	-0.0022
appropriate_for_Passed	0.019590	0.004211	0.000530	-0.000253	-0.000084	-0.0000
appropriate_for_R	0.035372	0.008365	0.006465	-0.010902	-0.003633	-0.0036
appropriate_for_TV-13	-0.012301	0.006711	0.009146	-0.000146	-0.000049	-0.0000
appropriate_for_TV-14	0.062572	-0.100569	-0.075212	-0.003914	-0.001304	-0.0013
appropriate_for_TV-G	0.022403	-0.039781	-0.035488	-0.001457	-0.000485	-0.0004
appropriate_for_TV-MA	-0.028109	-0.001295	0.025848	-0.002972	-0.000990	-0.0009
appropriate_for_TV-PG	0.012746	0.004771	0.007788	-0.001570	-0.000523	-0.0005
appropriate_for_TV-Y	0.000050	0.025847	0.023056	-0.000731	-0.000243	-0.0002
appropriate_for_TV-Y7	0.009807	0.023954	0.024461	-0.000981	-0.000327	-0.0003
appropriate_for_TV-Y7-FV	-0.001535	-0.001365	-0.001946	-0.000253	-0.000084	-0.0000
appropriate_for_Unrated	-0.045874	0.024269	0.032467	-0.001683	-0.000561	-0.0005
appropriate_for_nan	-0.108043	-0.287122	-0.303708	-0.019366	-0.006454	-0.0064

24 rows × 24 columns

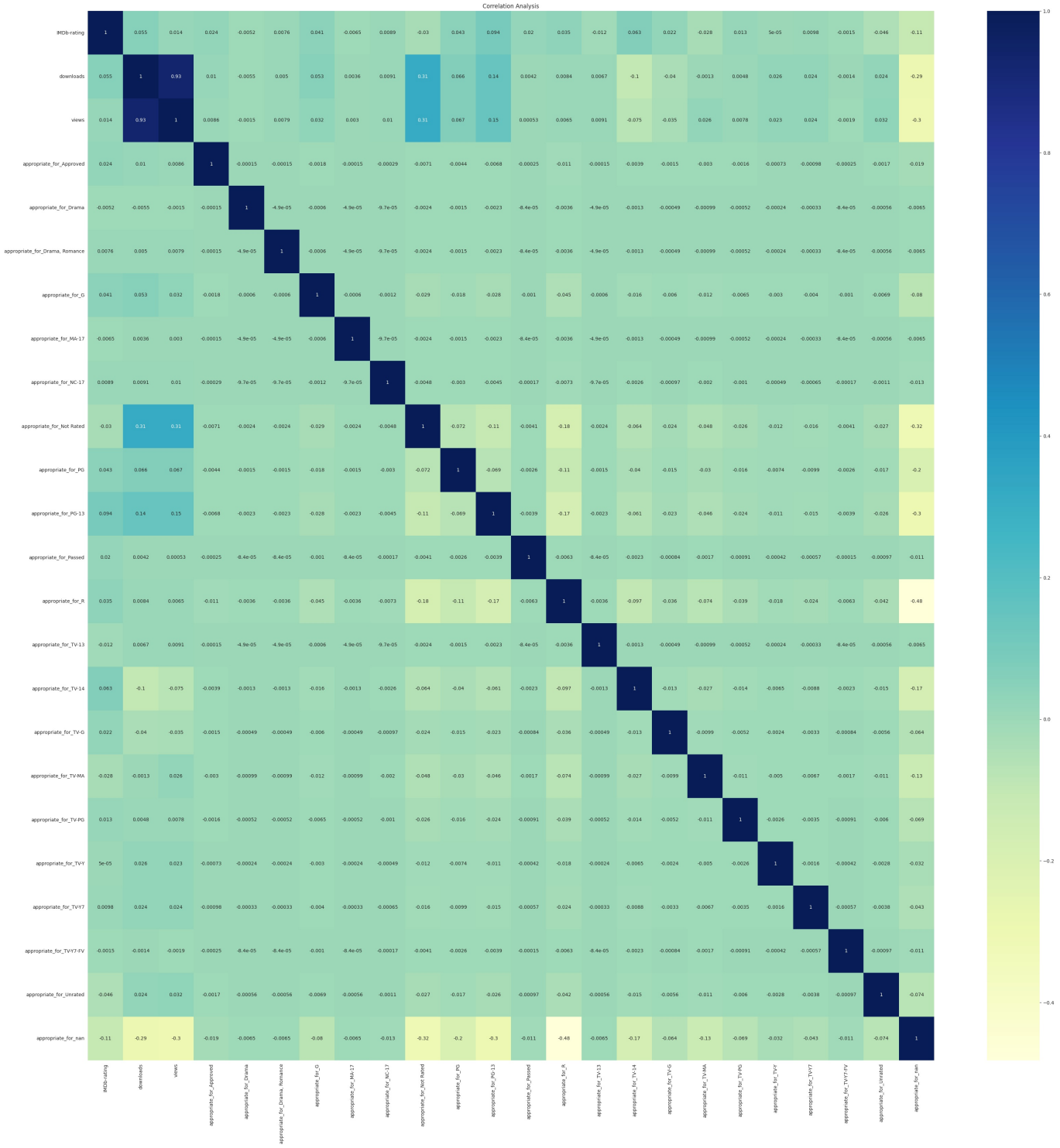
In [ ]:
```
plt.figure(figsize=(40, 40))
sns.heatmap(df_coded.corr(method='spearman', numeric_only=True), cmap='YlGnBu', annot=True)
plt.title('Correlation Analysis')
```

Out[ ]:
```
Text(0.5, 1.0, 'Correlation Analysis')
```



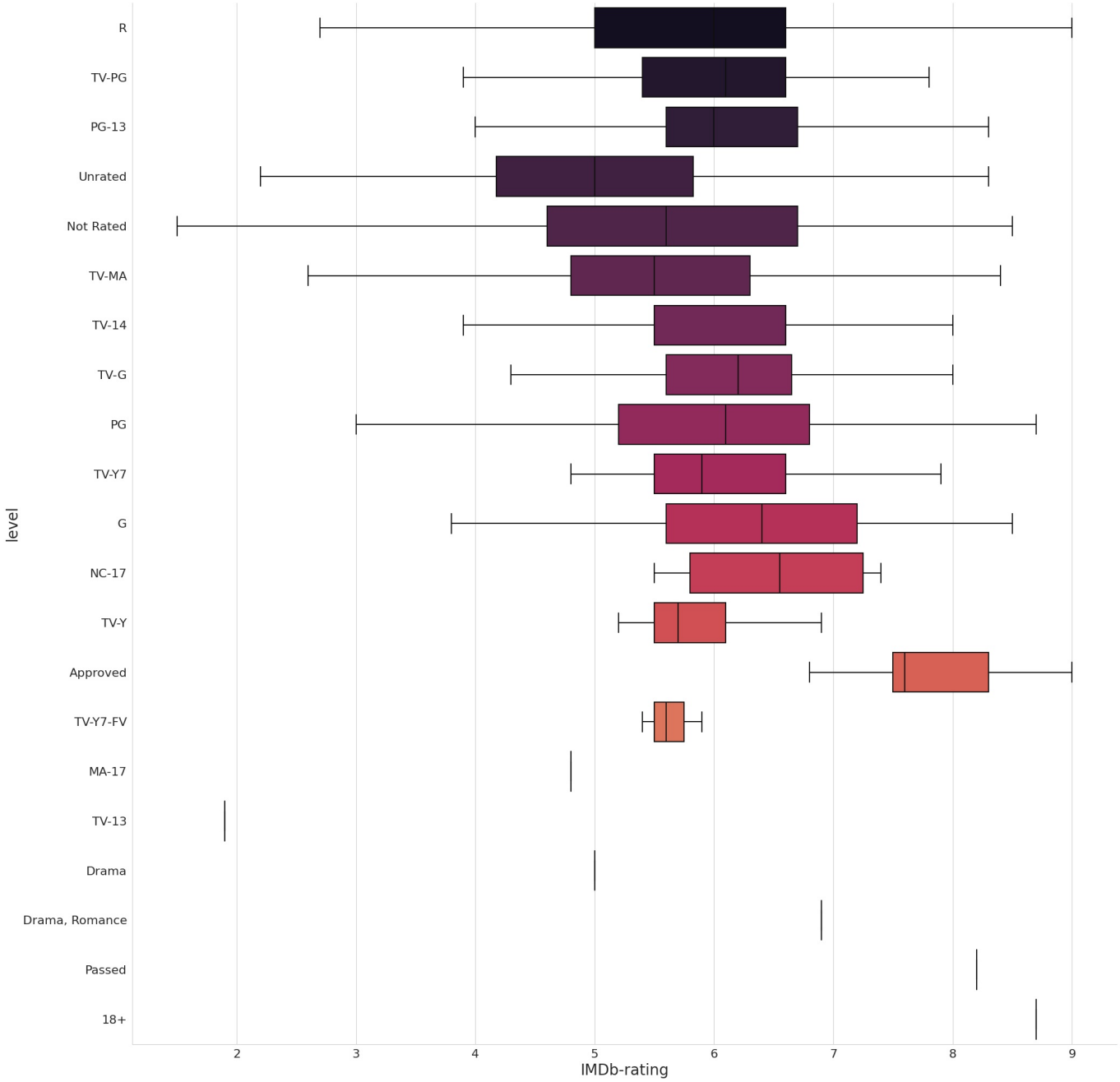
In [ ]:
```
def Correlation_analysis(dataset=mov_data_attr_corr, attribute_name='appropriate_for'):
    df_coded = pd.get_dummies(dataset, columns=[attribute_name], dummy_na=True, drop_first=True)
    df_coded.head()

    plt.figure(figsize=(40, 40))
    sns.heatmap(df_coded.corr(method='spearman', numeric_only=True), cmap='YlGnBu', annot=True)
    plt.title('Correlation Analysis')
```

In [ ]:
```
Correlation_analysis(attribute_name='appropriate_for')
```



可以看出，appropriate_for属性间的相关性较弱，与其他数值属性的相关性也较弱，因此不适合使用相关关系对空值进行填充。

In [ ]:
```
sns.set_style("whitegrid")
level_views_box = sns.catplot(data=mov_data, kind='box', y='appropriate_for', x='IMDb-rating', height=20, palet
plt.yticks(fontsize=16)
plt.xticks(fontsize=16)
plt.ylabel('level', fontsize=20, )
plt.xlabel('IMDb-rating', fontsize=20)
plt.show()
```



In [ ]:
```
df1 = mov_data_attr_corr.groupby('appropriate_for').agg(avg=('IMDb-rating', 'mean'))
df1
```

Out[ ]:

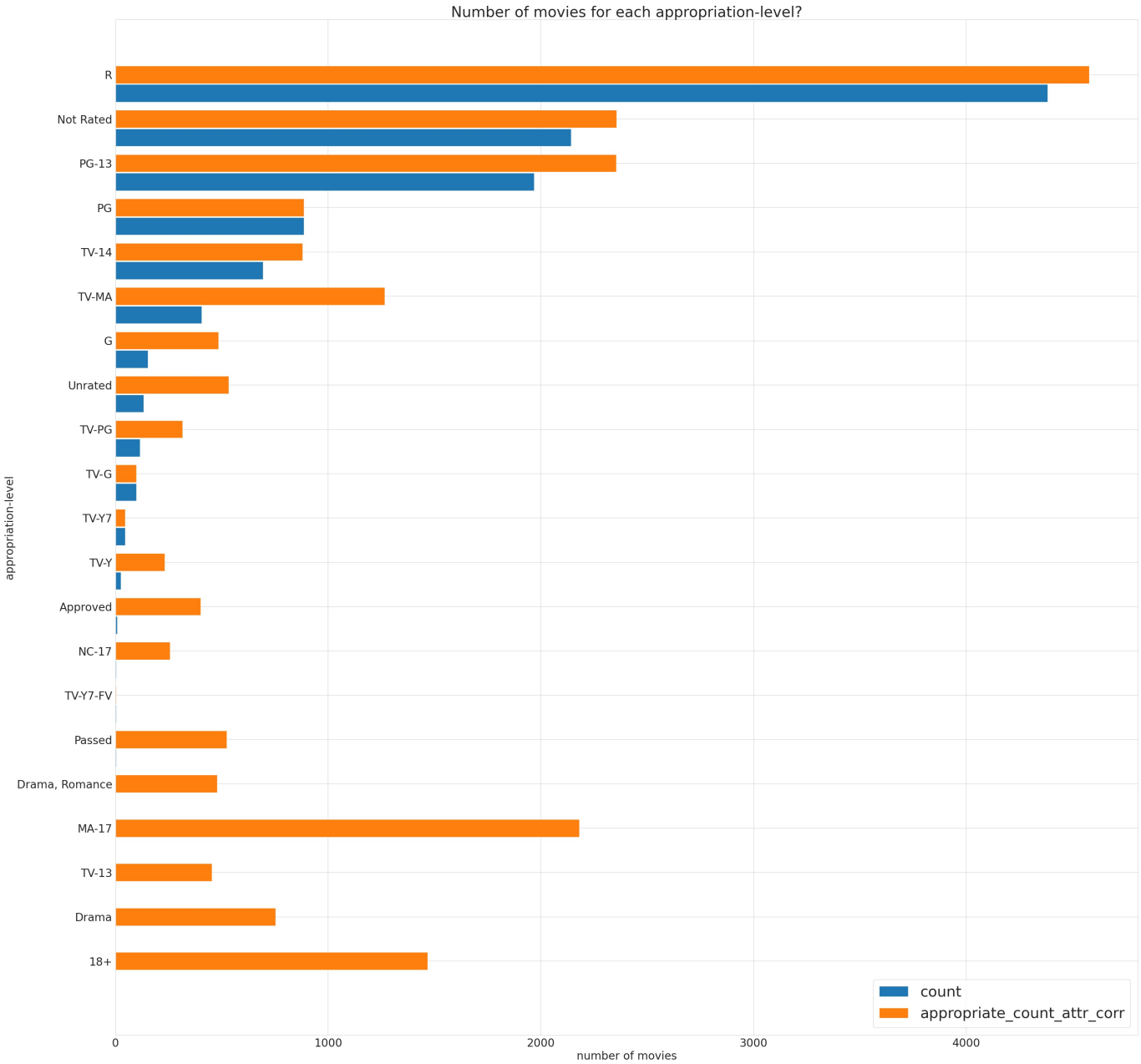
appropriate_for	avg
18+	8.700000
Approved	7.411111
Drama	5.000000
Drama, Romance	6.900000
G	6.336842
MA-17	4.800000
NC-17	6.500000
Not Rated	5.592997
PG	5.974179
PG-13	6.077863
Passed	8.200000
R	5.805686
TV-13	1.900000
TV-14	6.023775
TV-G	6.076768
TV-MA	5.477833
TV-PG	5.922807
TV-Y	5.708000
TV-Y7	5.964444
TV-Y7-FV	5.633333
Unrated	5.002273

In [ ]:
```
mov_data_attr_corr = mov_data.copy()  # 创建一个副本以避免在原始DataFrame上直接修改
for i in range(len(mov_data_attr_corr)):
    if pd.isna(mov_data_attr_corr['appropriate_for'].iloc[i]):  # 使用pd.isna来检查NA值
        rate = mov_data_attr_corr['IMDb-rating'].iloc[i]
        dist = []
        for j in range(len(df1)):
            dist.append(abs(df1.iloc[j]['avg']-rate))
        idx = dist.index(min(dist))
        mov_data_attr_corr.loc[i, 'appropriate_for'] = df1.index[idx]  # 使用.loc来确保在原始DataFrame上赋值

# 计算'appropriate_for'属性的值的计数
value_counts = mov_data_attr_corr['appropriate_for'].value_counts()
print(value_counts)
```

```
appropriate_for
R                4579
Not Rated        2357
PG-13            2355
MA-17            2181
18+              1468
TV-MA            1266
PG                886
TV-14             880
Drama             753
Unrated           533
Passed            523
G                 484
Drama, Romance    479
TV-13             453
Approved          400
TV-PG             316
NC-17             257
TV-Y              231
TV-G               99
TV-Y7              45
TV-Y7-FV            3
Name: count, dtype: int64
```

In [ ]:
```
appropriate_count_attr_corr = appropriate_count
appropriate_count_attr_corr['appropriate_count_attr_corr'] = [0] * len(appropriate_count)

for level in list(appropriate_count.index):
    if level in list(mov_data_attr_corr['appropriate_for'].value_counts().index):
        appropriate_count_attr_corr.loc[[level], ['appropriate_count_attr_corr']] = mov_data_attr_corr['appropr
```

```
plt.figure(figsize=(40, 40))
plt.yticks(fontsize=24)
plt.xticks(fontsize=24)
plt.barh(list(range(len(appropriate_count_attr_corr))), tick_label=appropriate_count_attr_corr.index, width=app
plt.barh([d+0.42 for d in list(range(len(appropriate_count_attr_corr)))], tick_label=appropriate_count_attr_cor
plt.ylabel('appropriation-level', fontsize=24)
plt.xlabel('number of movies', fontsize=24)
plt.title('Number of movies for each appropriation-level?', fontsize=32, loc='center')
plt.legend(fontsize=32, loc='lower right')
plt.show()
```



通过数据对象之间的相似性来填补缺失值

In [ ]:
```
mov_data_sample_corr = copy.deepcopy(mov_data)
```

In [ ]:
```
def regularit(df):
    new_df = pd.DataFrame(index=df.index)
    columns = ['IMDb-rating', 'downloads', 'views']
    for c in columns:
        d = df[c]
        MAX = d.max()
        MIN = d.min()
        new_df[c] = ((d - MIN) / (d - MAX))
    return new_df
```

In [ ]:
```
normal_mov_data = regularit(mov_data)
normal_mov_data
```

Out[ ]:

	IMDb-rating	downloads	views
0	-0.725490	-0.000778	-0.001300
1	-1.514286	-0.000187	-0.000205
2	-0.872340	-0.003660	-0.008467
3	-3.888889	-0.003975	-0.002578
4	-0.660377	-0.001682	-0.001082
...	...	...	...
20543	NaN	-0.005133	-0.003401
20544	-3.000000	-0.015784	-0.009684
20545	-3.631579	-0.008443	-0.004017
20546	NaN	-0.000790	-0.000459
20547	NaN	-0.006723	-0.003695

20548 rows × 3 columns

In [ ]:
```
normal_appropriate = pd.concat([normal_mov_data, mov_data['appropriate_for']], axis=1)
normal_appropriate
```

Out[ ]:

	IMDb-rating	downloads	views	appropriate_for
0	-0.725490	-0.000778	-0.001300	R
1	-1.514286	-0.000187	-0.000205	TV-PG
2	-0.872340	-0.003660	-0.008467	R
3	-3.888889	-0.003975	-0.002578	Passed
4	-0.660377	-0.001682	-0.001082	MA-17
...	...	...	...	...
20543	NaN	-0.005133	-0.003401	NaN
20544	-3.000000	-0.015784	-0.009684	NaN
20545	-3.631579	-0.008443	-0.004017	NaN
20546	NaN	-0.000790	-0.000459	NaN
20547	NaN	-0.006723	-0.003695	NaN

20548 rows × 4 columns