In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
!wget 'https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/or
```

```
'wget' is not recognized as an internal or external command,
operable program or batch file.
```

In [3]:
```
df = pd.read_csv('netflix.csv')
```

In [4]:
```
# 2. Observations on the shape of data, data types of all the attributes,
# conversion of categorical attributes to 'category' (If required),
# missing value detection, statistical summary
df.shape
```

Out[4]: (8807, 12)

In [5]:
```
df.info()
# only the release year has int data type except all are object that is string
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

In [6]:
```
# testing column wise to which column can be converted to category or to another
df[['show_id',
 'type',
 'title',
 'director',
 'cast',
 'country',
 'date_added',
 'rating',
 'duration',
 'listed_in',
 'description']].nunique()
```

Out[6]:
```
show_id         8807
type               2
title           8807
director        4528
cast            7692
country          748
date_added      1767
rating             17
duration          220
listed_in         514
description      8775
dtype: int64
```

In [7]:
```
df[['type','rating','country','listed_in']].nunique()
# i found that only type, rating, country and listed_in are the ones with limite
```

Out[7]:
```
type           2
rating        17
country      748
listed_in    514
dtype: int64
```

In [8]:
```
df[['type','rating','country','listed_in']]=df[['type','rating','country','liste
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8807 non-null   object
 1   type          8807 non-null   category
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   category
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64
 8   rating        8803 non-null   category
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   category
 11  description   8807 non-null   object
dtypes: category(4), int64(1), object(7)
memory usage: 645.2+ KB
```

In [9]:
```
# missing value detection
df.isna().sum()
percentage_of_missing_value=round(df.isnull().mean()*100,2).reset_index()
percentage_of_missing_value.columns = ['Column Name', 'Missing Percentage']
percentage_of_missing_value
```

Out[9]:

	Column Name	Missing Percentage
0	show_id	0.00
1	type	0.00
2	title	0.00
3	director	29.91
4	cast	9.37
5	country	9.44
6	date_added	0.11
7	release_year	0.00
8	rating	0.05
9	duration	0.03
10	listed_in	0.00
11	description	0.00

In [10]:
```
# statistical summary
numerical_dtype=df.describe()
print(numerical_dtype)
# df['release_year'].nunique()---> only 74
# here min year 1925 and max year 2021 means data for 74 years
```

```
       release_year
count   8807.000000
mean    2014.180198
std        8.819312
min     1925.000000
25%     2013.000000
50%     2017.000000
75%     2019.000000
max     2021.000000
```

In [11]:
```
categorical_dtype=df.describe(include='object')
categorical_dtype
```

Out[11]:

	show_id	title	director	cast	date_added	duration	description
count	8807	8807	6173	7982	8797	8804	8807
unique	8807	8807	4528	7692	1767	220	8775
top	s1	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	January 1, 2020	1 Season	Paranormal activity at a lush, abandoned prope...
freq	1	1	19	19	109	1793	4

Rajiv Chilaka has been the most frequent in terms of directing more number of movies

paranormal description apeared most frquently

David Attenborough appered maximum in the cast

Non-Graphical Analysis: Value counts and unique attributes (10 Points

In [12]:
```
# df['country'].value_counts()
df['country'].value_counts()
```

Out[12]:
```
country
United States
2818
India
972
United Kingdom
419
Japan
245
South Korea
199

...
Ireland, Canada, Luxembourg, United States, United Kingdom, Philippines, India
1
Ireland, Canada, United Kingdom, United States
1
Ireland, Canada, United States, United Kingdom
1
Ireland, France, Iceland, United States, Mexico, Belgium, United Kingdom, Hong
Kong       1
Zimbabwe
1
Name: count, Length: 748, dtype: int64
```

In [13]:
```
df['country'].value_counts()
df['rating'].value_counts()
df['duration'].value_counts(ascending=False)
duration_no_of_season=df[df['type']=='TV Show'].groupby('duration')['title'].cou
```

In [14]:
```
# we have seen the data that nested columns are nested with comma and
def find_nested_col(df):
  names_of_nested=[]
  for i in df.columns:
        # Check if any value in the column is a string and contains a comma
```

```
        if df[i].apply(lambda x: isinstance(x, str) and "," in x).any():
            names_of_nested.append(i)
    return names_of_nested

find=find_nested_col(df)
print(find)
```

```
['title', 'director', 'cast', 'country', 'date_added', 'listed_in', 'descriptio
n']
```

Need to make dataframe of column wise which has nested data in them and make them

In [15]:
```
df_cast_split=df['cast'].str.split(', ') # don't expand we need all data in sing
df_cast=pd.concat([df['title'],df_cast_split],axis=1)
df_cast=df_cast.explode(['cast'])
df_cast.rename(columns={'cast':'Individual_actor'},inplace=True)
df_cast
# df_cast_split.explode('cast')# for explode to work data must be in the list fi
```

Out[15]:

	title	Individual_actor
0	Dick Johnson Is Dead	NaN
1	Blood & Water	Ama Qamata
1	Blood & Water	Khosi Ngema
1	Blood & Water	Gail Mabalane
1	Blood & Water	Thabang Molaba
...	...	...
8806	Zubaan	Manish Chaudhary
8806	Zubaan	Meghna Malik
8806	Zubaan	Malkeet Rauni
8806	Zubaan	Anita Shabdish
8806	Zubaan	Chittaranjan Tripathy

64951 rows × 2 columns

In [16]:
```
# for director ---> use str split function
df_director_split=df['director'].str.split(', ') # don't expand we need all data
df_director=pd.concat([df['title'],df_director_split],axis=1)
df_director=df_director.explode(['director'])
df_director
```

Out[16]:

	title	director
0	Dick Johnson Is Dead	Kirsten Johnson
1	Blood & Water	NaN
2	Ganglands	Julien Leclercq
3	Jailbirds New Orleans	NaN
4	Kota Factory	NaN
...	...	...
8802	Zodiac	David Fincher
8803	Zombie Dumb	NaN
8804	Zombieland	Ruben Fleischer
8805	Zoom	Peter Hewitt
8806	Zubaan	Mozez Singh

9612 rows × 2 columns

In [17]:
```
# for country df
df_country_split=df['country'].str.split(', ') # don't expand we need all data i
df_country=pd.concat([df['title'],df_country_split],axis=1)
df_country=df_country.explode(['country'])
df_country
```

Out[17]:

	title	country
0	Dick Johnson Is Dead	United States
1	Blood & Water	South Africa
2	Ganglands	NaN
3	Jailbirds New Orleans	NaN
4	Kota Factory	India
...	...	...
8802	Zodiac	United States
8803	Zombie Dumb	NaN
8804	Zombieland	United States
8805	Zoom	United States
8806	Zubaan	India

10845 rows × 2 columns

In [18]:
```
# for the listed_in df
df_listed_in_split=df['listed_in'].str.split(', ') # don't expand we need all da
df_listed_in=pd.concat([df['title'],df_listed_in_split],axis=1)
```

```
df_listed_in=df_listed_in.explode(['listed_in'])
df_listed_in
```

Out[18]:

	title	listed_in
0	Dick Johnson Is Dead	Documentaries
1	Blood & Water	International TV Shows
1	Blood & Water	TV Dramas
1	Blood & Water	TV Mysteries
2	Ganglands	Crime TV Shows
...	...	...
8805	Zoom	Children & Family Movies
8805	Zoom	Comedies
8806	Zubaan	Dramas
8806	Zubaan	International Movies
8806	Zubaan	Music & Musicals

19323 rows × 2 columns

In [19]:

```
#merging all the relevant dataframe with each other
dfs=[df_cast,df_country,df_director,df_listed_in]

merged_df=dfs[0]
for i in dfs[1:]:
  merged_df=pd.merge(merged_df,i,how='inner',on='title')
merged_df
```

Out[19]:

	title	Individual_actor	country	director	listed_in
0	Dick Johnson Is Dead	NaN	United States	Kirsten Johnson	Documentaries
1	Blood & Water	Ama Qamata	South Africa	NaN	International TV Shows
2	Blood & Water	Ama Qamata	South Africa	NaN	TV Dramas
3	Blood & Water	Ama Qamata	South Africa	NaN	TV Mysteries
4	Blood & Water	Khosi Ngema	South Africa	NaN	International TV Shows
...	...	...	...	...	...
201986	Zubaan	Anita Shabdish	India	Mozez Singh	International Movies
201987	Zubaan	Anita Shabdish	India	Mozez Singh	Music & Musicals
201988	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Dramas
201989	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	International Movies
201990	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Music & Musicals

201991 rows × 5 columns

In [20]:
```
# now merge with the main df with relevant columns
df_final=merged_df.merge(df[['show_id', 'type', 'title', 'date_added',
       'release_year', 'rating', 'duration']])
df_final
```

Out[20]:

	title	Individual_actor	country	director	listed_in	show_id	type	d
0	Dick Johnson Is Dead	NaN	United States	Kirsten Johnson	Documentaries	s1	Movie	
1	Blood & Water	Ama Qamata	South Africa	NaN	International TV Shows	s2	TV Show	
2	Blood & Water	Ama Qamata	South Africa	NaN	TV Dramas	s2	TV Show	
3	Blood & Water	Ama Qamata	South Africa	NaN	TV Mysteries	s2	TV Show	
4	Blood & Water	Khosi Ngema	South Africa	NaN	International TV Shows	s2	TV Show	
...	...	...	...	...	...	...	...	
201986	Zubaan	Anita Shabdish	India	Mozez Singh	International Movies	s8807	Movie	
201987	Zubaan	Anita Shabdish	India	Mozez Singh	Music & Musicals	s8807	Movie	
201988	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Dramas	s8807	Movie	
201989	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	International Movies	s8807	Movie	
201990	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Music & Musicals	s8807	Movie	

201991 rows × 11 columns

In [21]:
```
# now need to check the nulls and deal with nulls and missing values
df_final.isnull().sum() # for column wise null values
# df.isnull().sum(axis=1).sort_values(ascending=False)---> not much of use row w
```

Out[21]:
```
title                 0
Individual_actor     2146
country             11897
director            50643
listed_in               0
show_id                 0
type                    0
date_added            158
release_year            0
rating                 67
duration                3
dtype: int64
```

In [22]:
```
# to fill the null values in director's column it is better to fill with unknown
df_final['director']= df_final['director'].fillna('Unknown_director')
```

In [23]:
```
df_final['director'].unique()
# we still have nan of string
```

Out[23]:
```
array(['Kirsten Johnson', 'Unknown_director', 'Julien Leclercq', ...,
       'Majid Al Ansari', 'Peter Hewitt', 'Mozez Singh'], dtype=object)
```

In [24]:
```
df_final['director']=df_final['director'].replace(['nan'], ['Unknown_director'])
```

In [25]:
```
df_final['Individual_actor']=df_final['Individual_actor'].replace(['nan'], ['Unk
df_final['country']=df_final['country'].replace(['nan'], ['Unknown_country'])
```

In [26]:
```
df_final['country']=df_final['country'].fillna('Unknown_country')
```

In [27]:
```
df_final['Individual_actor']=df_final['Individual_actor'].replace(['NaN',np.nan]
# Replace both np.nan and the string 'NaN' if unsure about the values
```

In [28]:
```
df_final.isna().sum()
x=df_final.loc[df_final['rating'].isna()]
x
# print(x.to_string()) # this function to print all the values
```

Out[28]:

	title	Individual_actor	country	director	listed_in
135125	13TH: A Conversation with Oprah Winfrey & Ava ...	Oprah Winfrey	Unknown_country	Unknown_director	Movies
135126	13TH: A Conversation with Oprah Winfrey & Ava ...	Ava DuVernay	Unknown_country	Unknown_director	Movies
154377	Gargantia on the Verdurous Planet	Kaito Ishikawa	Japan	Unknown_director	Anime Series
154378	Gargantia on the Verdurous Planet	Kaito Ishikawa	Japan	Unknown_director	International TV Shows
154379	Gargantia on the Verdurous Planet	Hisako Kanemoto	Japan	Unknown_director	Anime Series
...	...	...	...	...	...
171942	My Honor Was Loyalty	Francesco Migliore	Italy	Alessandro Pepe	Dramas
171943	My Honor Was Loyalty	Albrecht Weimer	Italy	Alessandro Pepe	Dramas
171944	My Honor Was Loyalty	Giulia Dichiaro	Italy	Alessandro Pepe	Dramas
171945	My Honor Was Loyalty	Alessandra Oriti Niosi	Italy	Alessandro Pepe	Dramas
171946	My Honor Was Loyalty	Andreas Segeritz	Italy	Alessandro Pepe	Dramas

67 rows × 11 columns

In [29]:
```
df_final.loc[df_final['duration'].isna()]
# excel is the best tool to see that kind of values where values substitiuted in
```

Out[29]:

	title	Individual_actor	country	director	listed_in	show_id	type	date_a
126537	Louis C.K. 2017	Louis C.K.	United States	Louis C.K.	Movies	s5542	Movie	Ap
131603	Louis C.K.: Hilarious	Louis C.K.	United States	Louis C.K.	Movies	s5795	Movie	Septe 16,
131737	Louis C.K.: Live at the Comedy Store	Louis C.K.	United States	Louis C.K.	Movies	s5814	Movie	Augu

In [30]:
```
df_final.loc[df_final['duration'].isna()].index # to get the row number
# excel is the best tool to see that kind of values where values substitiuted in
```

Out[30]:
```
Index([126537, 131603, 131737], dtype='int64')
```

In [31]:
```
# filling the null values in duration but they were present in the rating column
df_final.loc[df_final['duration'].isnull()]
df_final['duration']=df_final['duration'].fillna(df_final['rating'])
```

In [32]:
```
df_final['rating']=df_final['rating'].astype(str) # no need to convert rating to
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201991 entries, 0 to 201990
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   title             201991 non-null  object
 1   Individual_actor  201991 non-null  object
 2   country           201991 non-null  object
 3   director          201991 non-null  object
 4   listed_in         201991 non-null  object
 5   show_id           201991 non-null  object
 6   type              201991 non-null  category
 7   date_added        201833 non-null  object
 8   release_year      201991 non-null  int64
 9   rating            201991 non-null  object
 10  duration          201991 non-null  object
dtypes: category(1), int64(1), object(9)
memory usage: 15.6+ MB
```

In [33]:
```
# but also need to remove these values of rating where they contains the values
# if ever try to masking on the column containing null values do specify na=fals
df_final.loc[df_final['rating'].str.contains('min', na=False),'rating']='Not_ava
```

In [34]:
```
df_final.loc[df_final['rating'].str.contains('Not')]
```

Out[34]:

	title	Individual_actor	country	director	listed_in	show_id	type	date_a
126537	Louis C.K. 2017	Louis C.K.	United States	Louis C.K.	Movies	s5542	Movie	Ap
131603	Louis C.K.: Hilarious	Louis C.K.	United States	Louis C.K.	Movies	s5795	Movie	Septe 16,
131737	Louis C.K.: Live at the Comedy Store	Louis C.K.	United States	Louis C.K.	Movies	s5814	Movie	Augu

In [35]:
```
df_final['date_added'].nunique()
```

Out[35]:
```
1767
```

In [36]:
```
# df_final['date_added'].fillna(mode,inplace=True)
```

In [37]:
```
df_final.isna().sum()

# till here removed all the null values now need to correct the columns values
```

Out[37]:
```
title               0
Individual_actor    0
country             0
director            0
listed_in           0
show_id             0
type                0
date_added        158
release_year        0
rating              0
duration            0
dtype: int64
```

In [38]:
```
# here replacing the null values of date added column with  mode of the release_
# when release year was 2013.So below piece of code just checks the mode of date
years=df_final.loc[df_final['date_added'].isnull()]['release_year'].unique()
for year in years:
  mode_val=df_final[df_final['release_year']==year]['date_added'].mode().values[
  if mode_val:
    df_final.loc[(df_final['release_year'] == year) & (df_final['date_added'].is
# mode
# # don't worry that year_added and the release has sufficient gap in them
```

In [39]:
```
df_final.loc[df_final['date_added'].isnull(),'date_added'] # before it was 158 v
```

Out[39]:
```
Series([], Name: date_added, dtype: object)
```

In [40]:
```
#now we are checking before replacing director's country mode value with unknown
df_final[df_final['country']=="Unknown_country"]['country']
```

Out[40]:
```
58        Unknown_country
59        Unknown_country
60        Unknown_country
61        Unknown_country
62        Unknown_country
              ...
201424    Unknown_country
201425    Unknown_country
201932    Unknown_country
201933    Unknown_country
201934    Unknown_country
Name: country, Length: 11897, dtype: object
```

In [41]:
```
# Compute the mode of the 'country' column for each director
mode_mapping = df_final[df_final['country'] != 'Unknown_country'].groupby('direc

# Update the 'country' column in rows where the country is 'Unknown_country'
df_final.loc[(df_final['country'] == 'Unknown_country') & (df_final['director'].
```

In [42]:
```
# to check whether value is filling or not before using imputation it was 11897
df_final[df_final['country']=="Unknown_country"]['country']
```

Out[42]:
```
159       Unknown_country
160       Unknown_country
161       Unknown_country
162       Unknown_country
163       Unknown_country
              ...
199919    Unknown_country
199920    Unknown_country
199921    Unknown_country
199922    Unknown_country
200738    Unknown_country
Name: country, Length: 4276, dtype: object
```

In [43]:
```
mode_mapping = df_final[df_final['country'] != 'Unknown_country'].groupby('Indiv

# Update the 'country' column in rows where the country is 'Unknown_country'
df_final.loc[
    (df_final['country'] == 'Unknown_country') & (df_final['Individual_actor'].i
    'country'
] = df_final['Individual_actor'].map(mode_mapping)
```

In [44]:
```
df_final
```

Out[44]:

	title	Individual_actor	country	director	listed_in	show_id
0	Dick Johnson Is Dead	Unknown_actor	United States	Kirsten Johnson	Documentaries	s1
1	Blood & Water	Ama Qamata	South Africa	Unknown_director	International TV Shows	s2
2	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Dramas	s2
3	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Mysteries	s2
4	Blood & Water	Khosi Ngema	South Africa	Unknown_director	International TV Shows	s2
...	...	...	...	...	...	...
201986	Zubaan	Anita Shabdish	India	Mozez Singh	International Movies	s8807
201987	Zubaan	Anita Shabdish	India	Mozez Singh	Music & Musicals	s8807
201988	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Dramas	s8807
201989	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	International Movies	s8807
201990	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Music & Musicals	s8807

201991 rows × 11 columns

In [45]:
```
# now need to work on duration column
df_final['duration_movies']=df_final['duration'].copy()
df_final
```

Out[45]:

	title	Individual_actor	country	director	listed_in	show_id
0	Dick Johnson Is Dead	Unknown_actor	United States	Kirsten Johnson	Documentaries	s1
1	Blood & Water	Ama Qamata	South Africa	Unknown_director	International TV Shows	s2
2	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Dramas	s2
3	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Mysteries	s2
4	Blood & Water	Khosi Ngema	South Africa	Unknown_director	International TV Shows	s2
...	...	...	...	...	...	...
201986	Zubaan	Anita Shabdish	India	Mozez Singh	International Movies	s8807
201987	Zubaan	Anita Shabdish	India	Mozez Singh	Music & Musicals	s8807
201988	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Dramas	s8807
201989	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	International Movies	s8807
201990	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Music & Musicals	s8807

201991 rows × 12 columns

In [46]:
```
df_final.loc[df_final['duration_movies'].str.contains('Season'),'duration_movies
# to make the diffentiation between movies and shows
```

In [47]:
```
df_final[df_final['type']=='Movie']['duration_movies'].unique()
```

Out[47]:
```
array(['90 min', '91 min', '125 min', '104 min', '127 min', '67 min',
       '94 min', '161 min', '61 min', '166 min', '147 min', '103 min',
       '97 min', '106 min', '111 min', '110 min', '105 min', '96 min',
       '124 min', '116 min', '98 min', '23 min', '115 min', '122 min',
       '99 min', '88 min', '100 min', '102 min', '93 min', '95 min',
       '85 min', '83 min', '113 min', '13 min', '182 min', '48 min',
       '145 min', '87 min', '92 min', '80 min', '117 min', '128 min',
       '119 min', '143 min', '114 min', '118 min', '108 min', '63 min',
       '121 min', '142 min', '154 min', '120 min', '82 min', '109 min',
       '101 min', '86 min', '229 min', '76 min', '89 min', '156 min',
       '112 min', '107 min', '129 min', '135 min', '136 min', '165 min',
       '150 min', '133 min', '70 min', '84 min', '140 min', '78 min',
       '64 min', '59 min', '139 min', '69 min', '148 min', '189 min',
       '141 min', '130 min', '138 min', '81 min', '132 min', '123 min',
       '65 min', '68 min', '66 min', '62 min', '74 min', '131 min',
       '39 min', '46 min', '38 min', '126 min', '155 min', '159 min',
       '137 min', '12 min', '273 min', '36 min', '34 min', '77 min',
       '60 min', '49 min', '58 min', '72 min', '204 min', '212 min',
       '25 min', '73 min', '29 min', '47 min', '32 min', '35 min',
       '71 min', '149 min', '33 min', '15 min', '54 min', '224 min',
       '162 min', '37 min', '75 min', '79 min', '55 min', '158 min',
       '164 min', '173 min', '181 min', '185 min', '21 min', '24 min',
       '51 min', '151 min', '42 min', '22 min', '134 min', '177 min',
       '52 min', '14 min', '53 min', '8 min', '57 min', '28 min',
       '50 min', '9 min', '26 min', '45 min', '171 min', '27 min',
       '44 min', '146 min', '20 min', '157 min', '17 min', '203 min',
       '41 min', '30 min', '194 min', '233 min', '237 min', '230 min',
       '195 min', '253 min', '152 min', '190 min', '160 min', '208 min',
       '180 min', '144 min', '5 min', '174 min', '170 min', '192 min',
       '209 min', '187 min', '172 min', '16 min', '186 min', '11 min',
       '193 min', '176 min', '56 min', '169 min', '40 min', '10 min',
       '3 min', '168 min', '312 min', '153 min', '214 min', '31 min',
       '163 min', '19 min', '179 min', '43 min', '200 min', '196 min',
       '167 min', '178 min', '228 min', '18 min', '205 min', '201 min',
       '191 min'], dtype=object)
```

In [48]:
```
df_final['duration_movies'] = df_final['duration_movies'].str.extract('(\d+)').a
# to convert to int
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Rohit\AppData\Local\Temp\ipykernel_11272\2435843079.py:1: SyntaxWarning:
invalid escape sequence '\d'
  df_final['duration_movies'] = df_final['duration_movies'].str.extract('(\d+)').
astype(float)
```

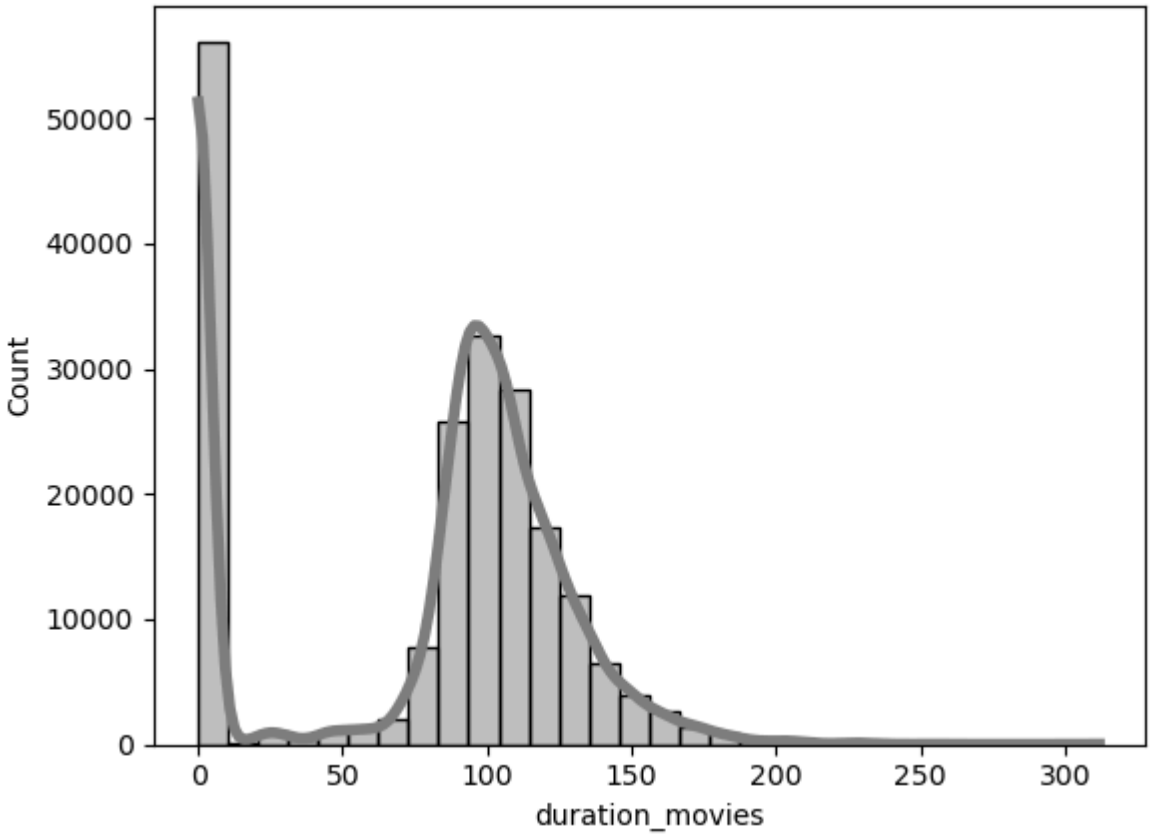
In [49]:
```
df_final['duration_movies']=df_final['duration_movies'].fillna(0)
```

In [50]:
```
df_final['duration_movies']=df_final['duration_movies'].astype(int)
```

In [51]:
```
df_final['duration_movies'].describe()
```

Out[51]:
```
count    201991.000000
mean         77.152789
std          52.269154
min           0.000000
25%           0.000000
50%          95.000000
75%         112.000000
max         312.000000
Name: duration_movies, dtype: float64
```

In [52]:
```
import seaborn as sns
import matplotlib.pyplot as plt

ax=sns.histplot(x=df_final['duration_movies'],kde=True,color='grey',
             bins=30)
for line in ax.lines:
    line.set_linewidth(4)
```



most of the movies are under the 112 minutes (aprox 30k). that is the most preferable time run for the movies.

In [53]:
```
df_final['duration_movies'].max()
```

Out[53]:
```
312
```

numerical to categorical conversion to make it more efficient for analysis.

In [54]:
```
bins1=[-1,1,50,100,150,200,250,312]
labels1=['<1','1-50','50-100','100-150','150-200','200-250','250-312']
df_final['duration_movies']=pd.cut(df_final['duration_movies'],bins=bins1,labels
```

In [55]:
```
# here we replacing the values where duration doesnot not contain tv shows durat
df_final['duration']=np.where( df_final['duration'].str.contains('Season',na=Fal
df_final
```

Out[55]:

	title	Individual_actor	country	director	listed_in	show_id
0	Dick Johnson Is Dead	Unknown_actor	United States	Kirsten Johnson	Documentaries	s1
1	Blood & Water	Ama Qamata	South Africa	Unknown_director	International TV Shows	s2
2	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Dramas	s2
3	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Mysteries	s2
4	Blood & Water	Khosi Ngema	South Africa	Unknown_director	International TV Shows	s2
...	...	...	...	...	...	...
201986	Zubaan	Anita Shabdish	India	Mozez Singh	International Movies	s8807
201987	Zubaan	Anita Shabdish	India	Mozez Singh	Music & Musicals	s8807
201988	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Dramas	s8807
201989	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	International Movies	s8807
201990	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Music & Musicals	s8807

201991 rows × 12 columns

In [56]:
```
df_final['modified_date_added']=pd.to_datetime(df_final['date_added'].str.strip(
df_final['month']=df_final['modified_date_added'].dt.month
df_final['week']=df_final['modified_date_added'].dt.isocalendar().week
df_final['year']=df_final['modified_date_added'].dt.year
df_final.head()
```

Out[56]:

	title	Individual_actor	country	director	listed_in	show_id	type
0	Dick Johnson Is Dead	Unknown_actor	United States	Kirsten Johnson	Documentaries	s1	Movie
1	Blood & Water	Ama Qamata	South Africa	Unknown_director	International TV Shows	s2	TV Show
2	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Dramas	s2	TV Show
3	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Mysteries	s2	TV Show
4	Blood & Water	Khosi Ngema	South Africa	Unknown_director	International TV Shows	s2	TV Show

In [57]:
```
np.all(df_final['modified_date_added'].isnull())
#df_final['modified_date_added'].isnull().sum()
```

Out[57]:
```
False
```

univariate analysis on the basis of numerical and categorical values

In [58]:
```
df_final['year'].value_counts()
df_final[df_final['year']==2019]['month'].value_counts()
df_final[df_final['year']==2020]['month'].value_counts()
```

Out[58]:
```
month
1     6139
4     4204
6     4190
10    4186
9     3971
5     3928
12    3844
11    3563
7     3363
2     3051
3     2860
8     2726
Name: count, dtype: int64
```

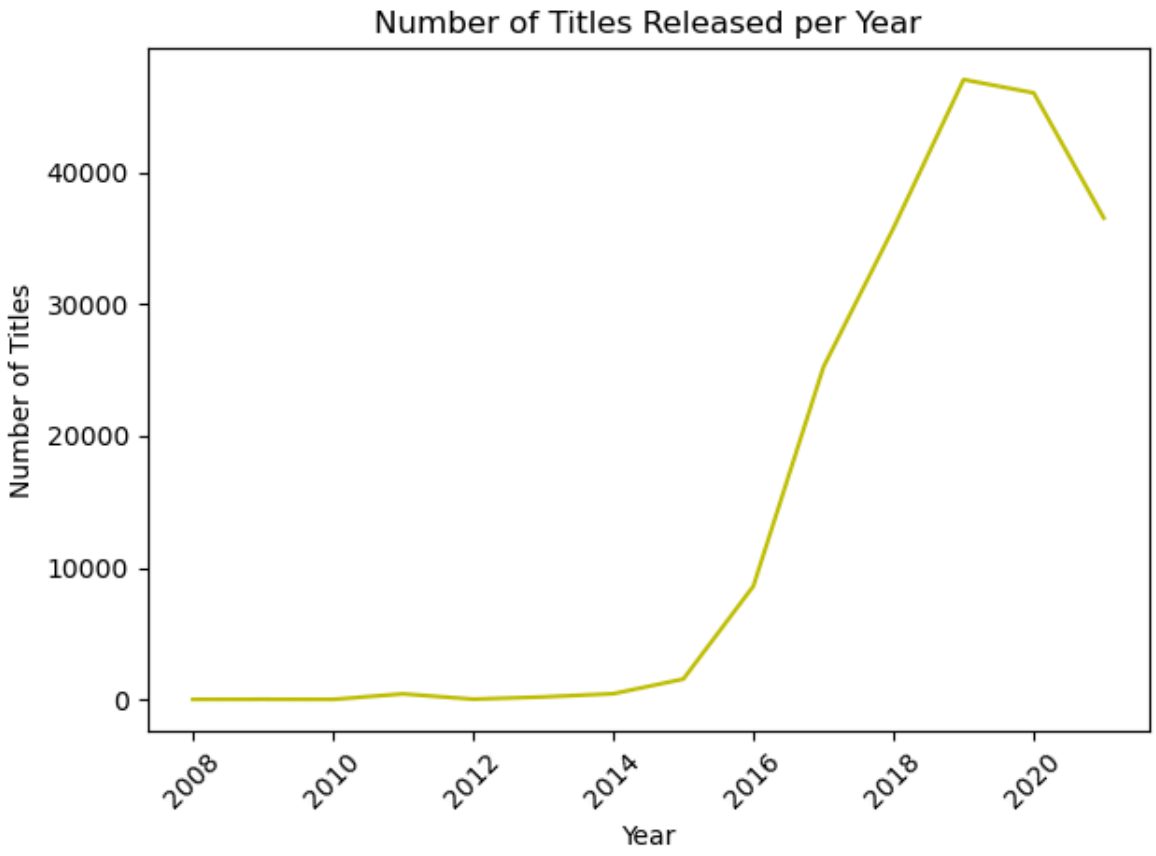
In [59]:
```
df_final[df_final['year']==2020]['listed_in'].value_counts().reset_index()
```

Out[59]:

	listed_in	count
0	Dramas	6202
1	International Movies	5829
2	Comedies	5226
3	International TV Shows	3121
4	Children & Family Movies	2921
5	Action & Adventure	2546
6	TV Dramas	2205
7	Independent Movies	1865
8	Romantic Movies	1755
9	Thrillers	1641
10	Kids' TV	1145
11	TV Comedies	1136
12	Crime TV Shows	1085
13	Horror Movies	972
14	Sci-Fi & Fantasy	819
15	Anime Series	623
16	Music & Musicals	621
17	TV Action & Adventure	619
18	Romantic TV Shows	589
19	Classic Movies	454
20	Spanish-Language TV Shows	438
21	TV Mysteries	422
22	Documentaries	415
23	British TV Shows	383
24	Sports Movies	324
25	TV Horror	305
26	TV Sci-Fi & Fantasy	289
27	LGBTQ Movies	273
28	Korean TV Shows	252
29	Reality TV	225
30	Teen TV Shows	209
31	Anime Features	201
32	TV Thrillers	192

	listed_in	count
33	Cult Movies	178
34	Docuseries	165
35	Faith & Spirituality	100
36	Stand-Up Comedy	89
37	Classic & Cult TV	62
38	TV Shows	52
39	Movies	31
40	Stand-Up Comedy & Talk Shows	24
41	Science & Nature TV	22

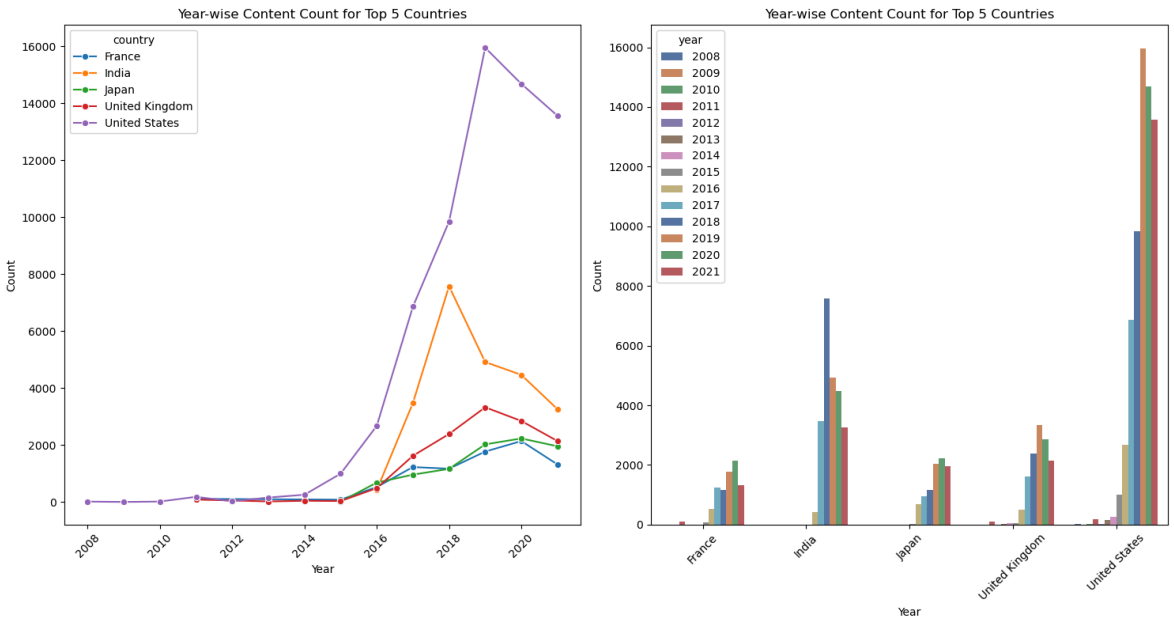
In [60]:
```
# Count number of titles per year
yearly_counts = df_final.groupby('year')['title'].count().reset_index()

sns.lineplot(data=yearly_counts, x='year', y='title',color='y')
plt.title('Number of Titles Released per Year')
plt.xlabel('Year')
plt.ylabel('Number of Titles')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



year 2019(especially in the month of oct,nov,dec) and 2020 has the most titles added to the netflix must be in high demand

and most content was Dramas, international movies, comedies

In [61]:
```
top_5_count=df_final['country'].value_counts().head(5).index
top_5_count=df_final[df_final['country'].isin(top_5_count)]
 # top_5_count.groupby(['country','year']).agg({'year':'count'})
top_5_count= top_5_count.groupby(['country', 'year']).size().reset_index(name='c

plt.figure(figsize=(15, 8))

plt.subplot(1,2,1)
sns.lineplot(data=top_5_count, x='year', y='count', hue='country', marker='o')
plt.title('Year-wise Content Count for Top 5 Countries')
plt.xlabel('Year')
plt.ylabel('Count')
plt.xticks(rotation=45)

plt.subplot(1,2,2)
sns.barplot(data=top_5_count, x='country', y='count', hue='year',palette='deep')
plt.title('Year-wise Content Count for Top 5 Countries')
plt.xlabel('Year')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



there is quite declining in the Us and UK from the year 2019 must be hit by covid too and in india it is declining after 2018

In [62]:
```
# Safe way to handle even if df has 1 column
for dtype in pd.Series(df_final.dtypes).unique():
    cols = df_final.columns[df_final.dtypes == dtype].tolist()
    print(f"{dtype}: {cols}")
```

```
object: ['title', 'Individual_actor', 'country', 'director', 'listed_in', 'show_i
d', 'date_added', 'rating', 'duration']
category: ['type']
int64: ['release_year']
category: ['duration_movies']
datetime64[ns]: ['modified_date_added']
int32: ['month', 'year']
UInt32: ['week']
```

In [63]:
```
# for particular dtype like only want to check int and object
# import pandas as pd

def get_column_types(df):
    """
    Returns a dictionary of column names categorized by data types.

    Parameters:
        df (pd.DataFrame): The input DataFrame.

    Returns:
        dict: Dictionary with keys like 'int', 'float', 'object', etc.,
              and values as lists of column names.
    """
    return {
        'int': df.select_dtypes(include='int').columns.tolist(),
#         'float': df.select_dtypes(include='float').columns.tolist(),
        'object': df.select_dtypes(include='object').columns.tolist(),
#         'bool': df.select_dtypes(include='bool').columns.tolist(),
#         'datetime': df.select_dtypes(include='datetime').columns.tolist(),
      'category': df.select_dtypes(include='category').columns.tolist(),
#         'other': df.select_dtypes(exclude=['int', 'float', 'object', 'bool', '
    }
# df_final=df_final.apply(get_column_types) not working because no for loop
get_column_types(df_final)
```

Out[63]:
```
{'int': ['release_year', 'month', 'year'],
 'object': ['title',
  'Individual_actor',
  'country',
  'director',
  'listed_in',
  'show_id',
  'date_added',
  'rating',
  'duration'],
 'category': ['type', 'duration_movies']}
```

In [64]:
```
df_final.rename(columns={'listed_in':'Genre'},inplace=True)
df_final
```

Out[64]:

	title	Individual_actor	country	director	Genre	show_id
0	Dick Johnson Is Dead	Unknown_actor	United States	Kirsten Johnson	Documentaries	s1
1	Blood & Water	Ama Qamata	South Africa	Unknown_director	International TV Shows	s2
2	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Dramas	s2
3	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Mysteries	s2
4	Blood & Water	Khosi Ngema	South Africa	Unknown_director	International TV Shows	s2
...	...	...	...	...	...	...
201986	Zubaan	Anita Shabdish	India	Mozez Singh	International Movies	s8807
201987	Zubaan	Anita Shabdish	India	Mozez Singh	Music & Musicals	s8807
201988	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Dramas	s8807
201989	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	International Movies	s8807
201990	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Music & Musicals	s8807

201991 rows × 16 columns

In [65]:
```
df_genre=df_final.groupby('Genre').agg({'title':'nunique'}).sort_values(by='titl
df_genre
# df_final.groupby('Genre')['title'].nunique().sort_values(ascending=False) also
```

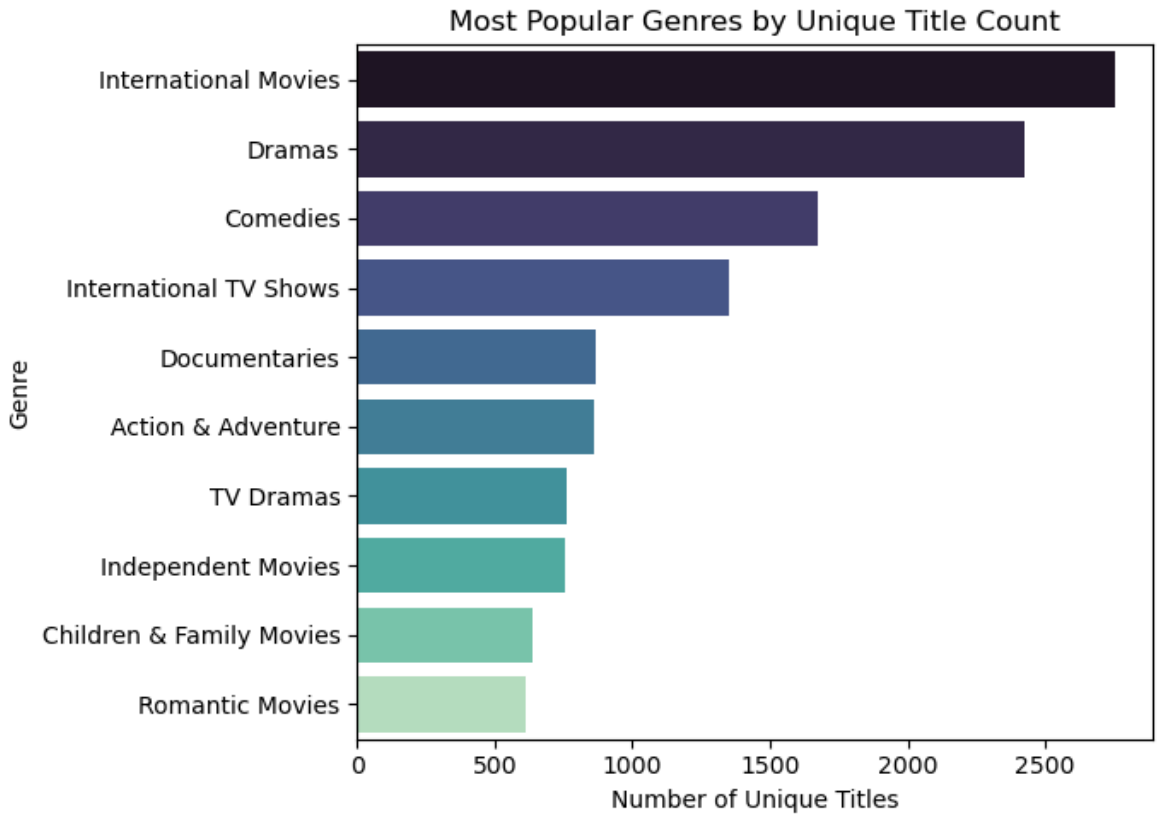
Out[65]:

	Genre	title
0	International Movies	2752
1	Dramas	2427
2	Comedies	1674
3	International TV Shows	1351
4	Documentaries	869
5	Action & Adventure	859
6	TV Dramas	763
7	Independent Movies	756
8	Children & Family Movies	641
9	Romantic Movies	616

international movies,Dramas, comedies and international tv shows are the most popular title coz added in huge demand and children and romantic movies are added in less number as compared

In [66]:

```
# countplot won't work which needs simple values_count on cat basis and genre ti
# Plot barplot with the number of unique titles per genre(cat-num case)
plt.figure(figsize=(7, 5))
sns.barplot(
    y='Genre',
    x='title',
    data=df_genre,
    palette='mako',
    hue='Genre'
)
plt.xlabel('Number of Unique Titles')
plt.ylabel('Genre')
plt.title('Most Popular Genres by Unique Title Count')
plt.tight_layout()
plt.show()
```



In [67]:
```
df_genre_Tv=df_final[df_final['type']=='TV Show'].groupby('Genre').agg({'title':
df_genre_Tv
```

Out[67]:

	Genre	title
0	International TV Shows	1351
1	TV Dramas	763
2	TV Comedies	581
3	Crime TV Shows	470
4	Kids' TV	451
5	Docuseries	395
6	Romantic TV Shows	370
7	Reality TV	255
8	British TV Shows	253
9	Anime Series	176

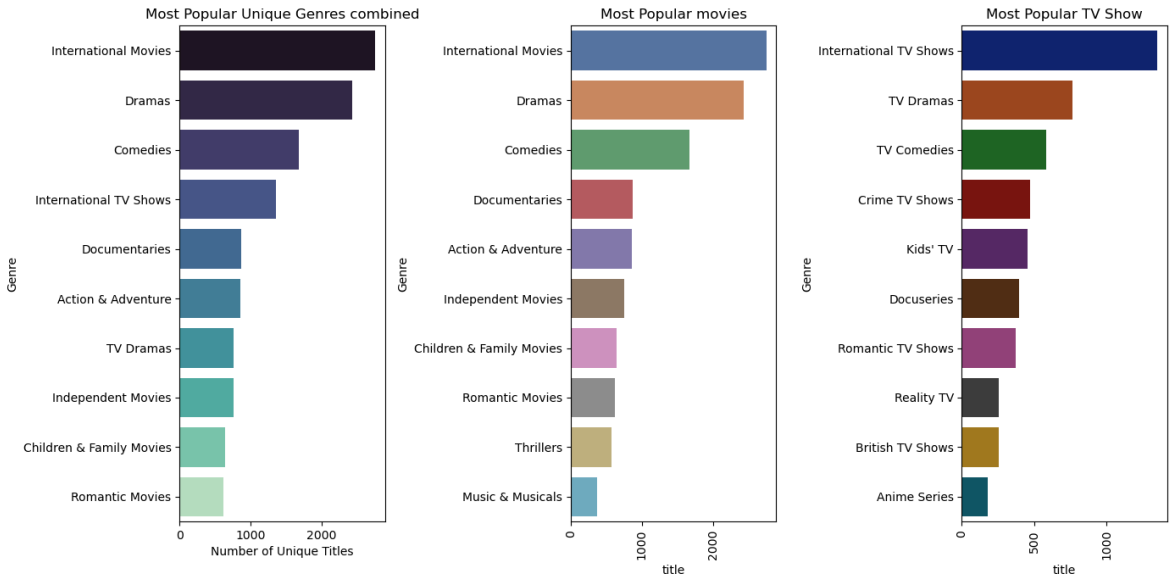
In [68]:
```
#pd.reset_option('display.max_rows')
df_genre_movies=df_final[df_final['type']=='Movie'].groupby('Genre').agg({'title
df_genre_movies
```

Out[68]:

	Genre	title
0	International Movies	2752
1	Dramas	2427
2	Comedies	1674
3	Documentaries	869
4	Action & Adventure	859
5	Independent Movies	756
6	Children & Family Movies	641
7	Romantic Movies	616
8	Thrillers	577
9	Music & Musicals	375

In [69]:

```
plt.figure(figsize=(14, 7))
plt.subplot(1,3,1)
sns.barplot(y='Genre',x='title',data=df_genre,palette='mako',hue='Genre')
plt.xlabel('Number of Unique Titles')
plt.ylabel('Genre')
plt.title('Most Popular Unique Genres combined')

plt.subplot(1,3,2)
sns.barplot(df_genre_movies,y='Genre',x='title',palette='deep',hue='Genre') # ne
plt.xticks(rotation =90) # here frequency as numerical for barplot
plt.title('Most Popular movies')

plt.subplot(1,3,3)
sns.barplot(df_genre_Tv,y='Genre',x='title',palette='dark',hue='Genre') # need t
plt.xticks(rotation =90) # here frequency as numerical for barplot
plt.title('Most Popular TV Show')

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



we can say that international movies and tv shows are are on the top charts as followed by the dramas and comedies in both movies and tv shows, crime tv shows is quite popular in tv shows and in tv shows content for child is prefered while in movies where stuff for family&children is less.

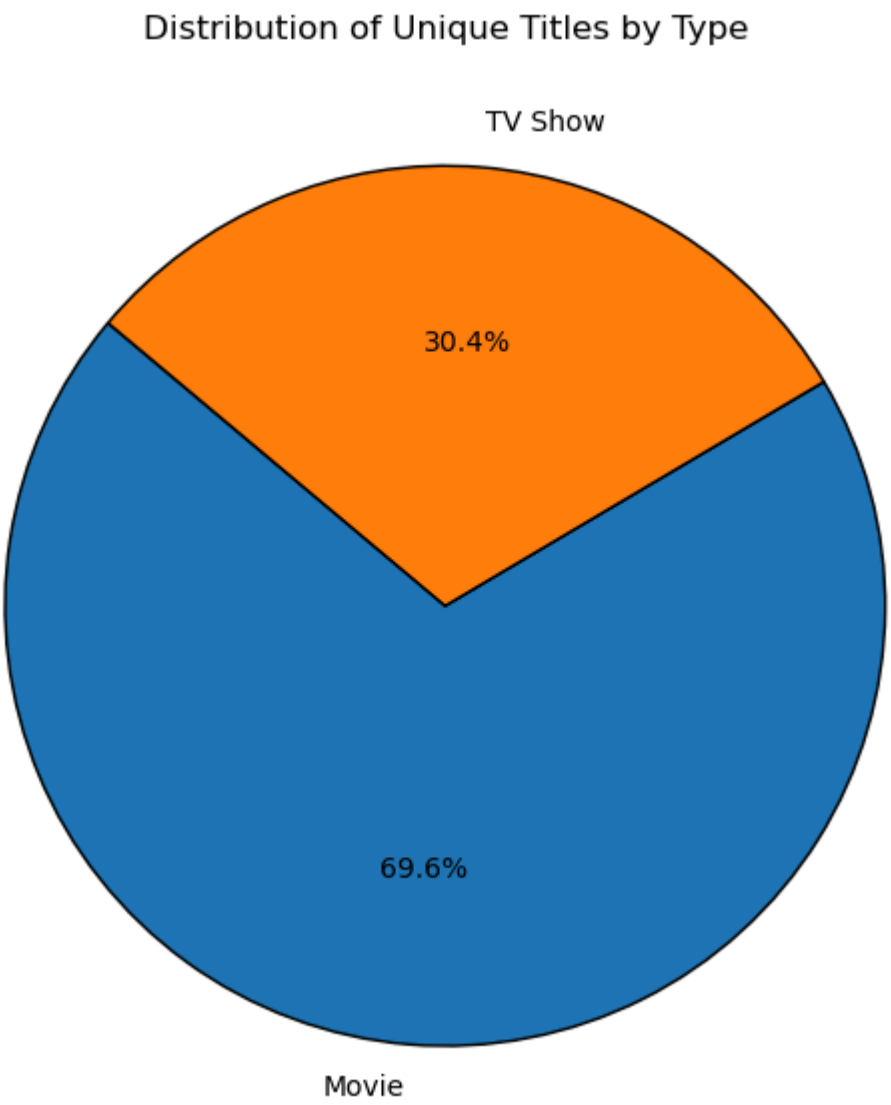
In [70]:
```
# now we can see the category column type , in which proportion content on the n
df_final.groupby('type')['title'].nunique()
```

```
C:\Users\Rohit\AppData\Local\Temp\ipykernel_11272\3762917087.py:2: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a futu
re version of pandas. Pass observed=False to retain current behavior or observed=
True to adopt the future default and silence this warning.
  df_final.groupby('type')['title'].nunique()
```

Out[70]:
```
type
Movie      6131
TV Show    2676
Name: title, dtype: int64
```

In [71]:
```
type_counts = df_final.groupby('type')['title'].nunique()

# Plot pie chart
plt.figure(figsize=(6, 6))
plt.pie(
    type_counts,
    labels=type_counts.index,
    autopct='%1.1f%%',
    startangle=140,
    wedgeprops={'edgecolor': 'black'}
)
plt.title('Distribution of Unique Titles by Type')
plt.tight_layout()
plt.show()
```

```
C:\Users\Rohit\AppData\Local\Temp\ipykernel_11272\1991886078.py:1: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a futu
re version of pandas. Pass observed=False to retain current behavior or observed=
True to adopt the future default and silence this warning.
  type_counts = df_final.groupby('type')['title'].nunique()
```



so we can say that movie content is more in number so the audience may preffered to movies as it is approx 70% of the whole

In [72]:
```
count=df_final.groupby('country')['title'].nunique().sort_values(ascending=False
count=count.reset_index()  # to make this a dataframe
count.columns = ['country', 'count'] # now we can provide labels
count.sort_values(by='count',ascending=False)
# df_final1.groupby(['country']).agg({"title":"nunique"}) can use this can direc

# as we see all countries data there is , after them so we need to clean them be
```

Out[72]:

	country	count
0	United States	4248
1	India	1138
2	United Kingdom	829
3	Canada	460
4	France	409
...	...	...
103	Armenia	1
104	Vatican City	1
105	Mozambique	1
106	Nicaragua	1
127	Ethiopia	1

128 rows × 2 columns

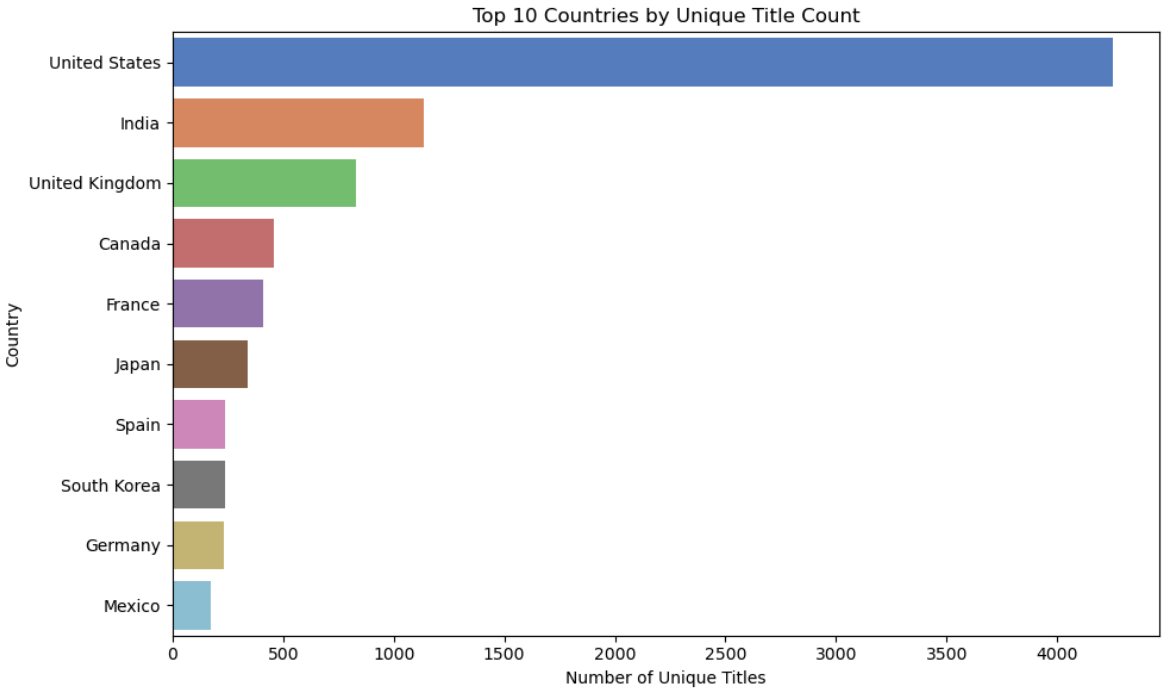
In [73]:
```
df_final['country']=df_final['country'].str.replace(',','')
```

In [74]:
```
# now lets find out which country more productive in terms of  producing more mo

df_final.groupby('country')['title'].nunique().sort_values(ascending=False)
```

Out[74]:
```
country
United States     4249
India             1138
United Kingdom     831
Canada             460
France             409
                  ...
Jamaica              1
Slovakia             1
Somalia              1
Ethiopia             1
Lithuania            1
Name: title, Length: 124, dtype: int64
```

In [75]:
```
df_country=df_final.groupby('country')['title'].nunique().sort_values(ascending=
plt.figure(figsize=(10, 6))
sns.barplot(
    x='title',   # WE KNOW THAT HER TITLE IS NUMERICAL NO CATEGORICAL
    y='country',
    data=df_country,
    palette="muted"
)
plt.xlabel('Number of Unique Titles')
plt.ylabel('Country')
plt.title('Top 10 Countries by Unique Title Count')
plt.tight_layout()
plt.show()
```

```
C:\Users\Rohit\AppData\Local\Temp\ipykernel_11272\3608049499.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(
```



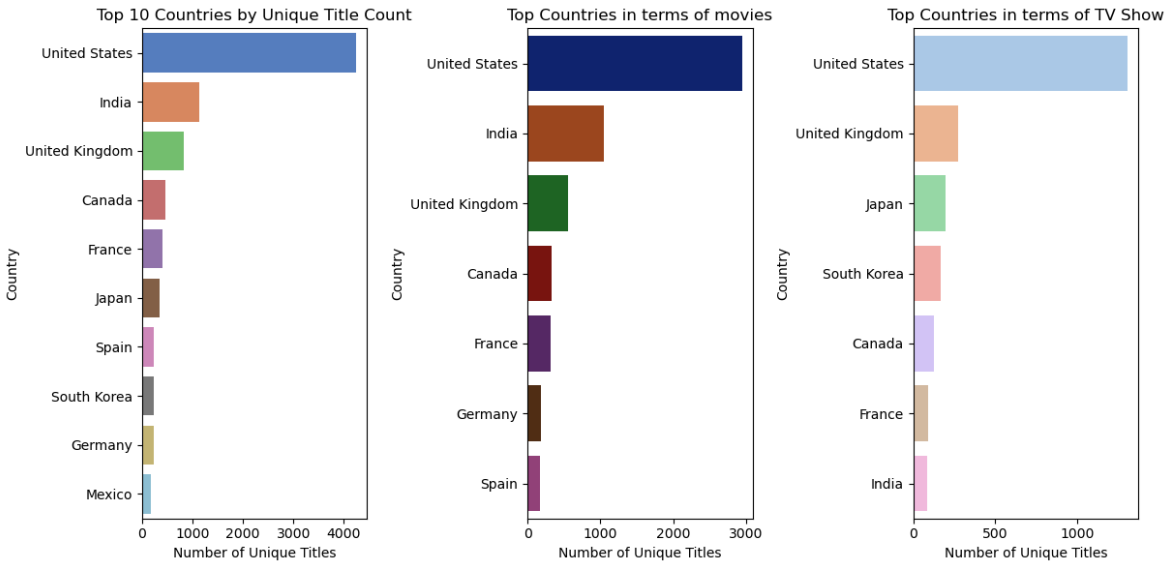
so it is obvious that UNITED STATES is most producitve in terms of producing followed by india, UK, Canada , France

UNITED STATES has huge gap interms of producing movies production must have share in economy as well.

In [76]:
```
df_country_movie=df_final[df_final['type']=='Movie'].groupby('country')['title']
df_country_tv=df_final[df_final['type']=='TV Show'].groupby('country')['title'].
```

In [77]:
```
plt.figure(figsize=(12, 6))
plt.subplot(1,3,1)
sns.barplot( x='title',   y='country',data=df_country,palette="muted",hue='count
plt.xlabel('Number of Unique Titles')
plt.ylabel('Country')
plt.title('Top 10 Countries by Unique Title Count')

plt.subplot(1,3,2)
sns.barplot( x='title',   y='country',data=df_country_movie,palette="dark",hue='
plt.xlabel('Number of Unique Titles')
plt.ylabel('Country')
plt.title('Top Countries in terms of movies')

plt.subplot(1,3,3)
sns.barplot( x='title',   y='country',data=df_country_tv,palette="pastel",hue='c
plt.xlabel('Number of Unique Titles')
```

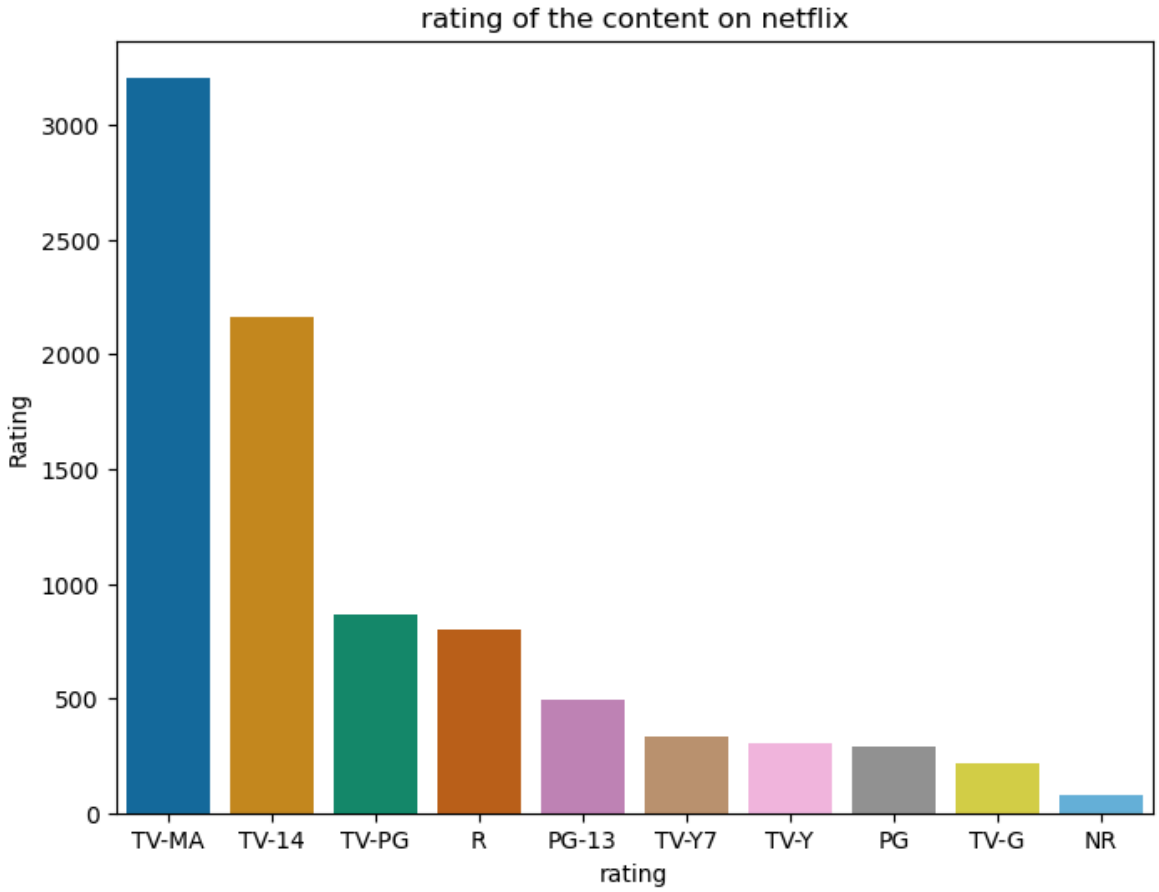
```
plt.ylabel('Country')
plt.title('Top Countries in terms of TV Show')
plt.tight_layout()

plt.show()
```



we can say tht united states,India, UK leading here in all the segment india is runner up interms of producing movies while UK in TVShows Japan and korean shows here is unexpected winner in TV Shows which was not polpular ealrier in this industry

In [78]:
```
# now let's take the column rating for analysis
df_final.groupby('rating')['title'].agg('nunique')
# or
df_rating=df_final.groupby('rating').agg({'title':'nunique'}).reset_index().sort
```

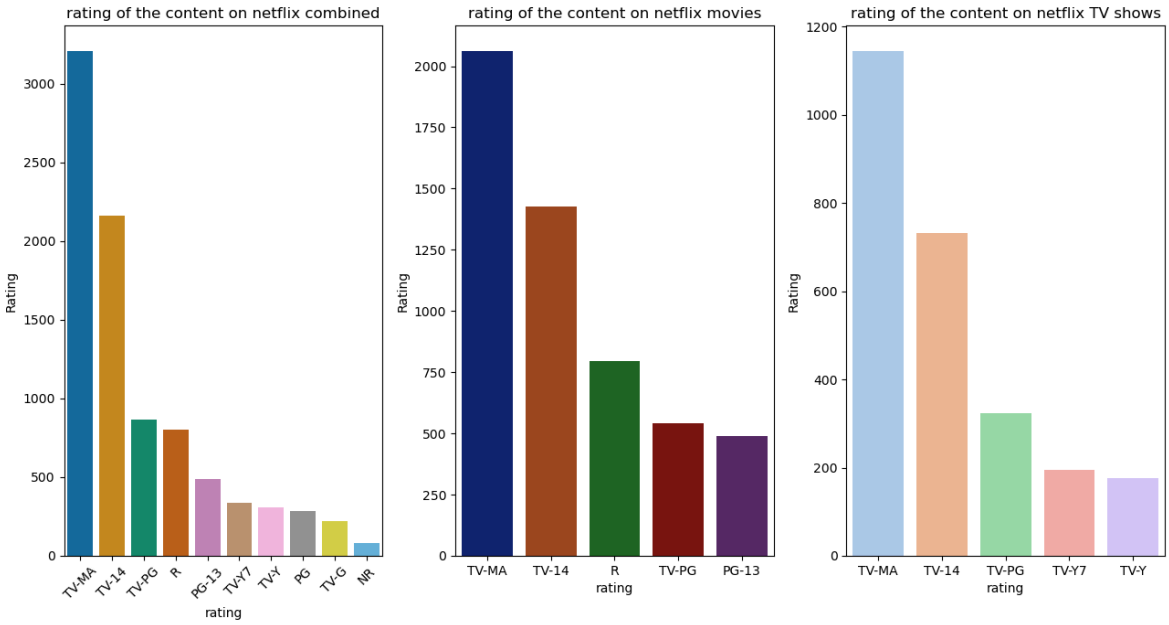
In [79]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=df_rating['rating'],y=df_rating['title'],palette="colorblind",hue=
plt.title('rating of the content on netflix')
plt.ylabel('Rating',fontsize=10)
plt.show()
```



most content which is highly watched is matured audience and after it follows by tv-14 and tv-pg for teens and adults which required parents guidance but for movies R-rating is the most highest.

In [80]:
```
df_rating_movie=df_final[df_final['type']=="Movie"].groupby('rating').agg({'titl
df_rating_tv=df_final[df_final['type']=="TV Show"].groupby('rating').agg({'title
```

In [81]:
```
plt.figure(figsize=(13,7))
plt.subplot(1,3,1)
sns.barplot(x=df_rating['rating'],y=df_rating['title'],palette="colorblind",hue=
plt.xticks(rotation=45)
plt.title('rating of the content on netflix combined')
plt.ylabel('Rating',fontsize=10)

plt.subplot(1,3,2)
sns.barplot(x=df_rating_movie['rating'],y=df_rating_movie['title'],palette="dark
plt.title('rating of the content on netflix movies')
plt.ylabel('Rating',fontsize=10)

plt.subplot(1,3,3)
sns.barplot(x=df_rating_tv['rating'],y=df_rating_tv['title'],palette="pastel",hu
plt.title('rating of the content on netflix TV shows')
plt.ylabel('Rating',fontsize=10)

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



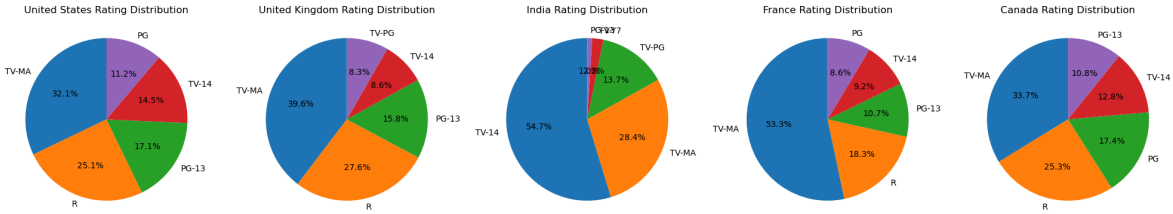
most content which is highly watched is matured audience in all and after it follows by tv-14 and tv-pg for teens and adults which required parents guidance but for movies R-rating is the most highest.

In [82]:
```
# df_rating_country=df_final['country']==['United States','India','United Kingdo
top_5_count= df_final.groupby('country')['title'].nunique().sort_values(ascendin
top_5_count=df_final[df_final['country'].isin(top_5_count)]

rating_counts = top_5_count.groupby(['country', 'rating']).size().reset_index(na
rating_counts_sorted = rating_counts.sort_values(by=['country', 'count'], ascend
top_5_ratings_by_country = rating_counts_sorted.groupby('country').head(5).reset
top_5_ratings_by_country
```

Out[82]:

	country	rating	count
0	United States	TV-MA	17337
1	United States	R	13554
2	United States	PG-13	9212
3	United States	TV-14	7818
4	United States	PG	6066
5	United Kingdom	TV-MA	4397
6	United Kingdom	R	3063
7	United Kingdom	PG-13	1755
8	United Kingdom	TV-14	959
9	United Kingdom	TV-PG	924
10	India	TV-14	12867
11	India	TV-MA	6668
12	India	TV-PG	3222
13	India	TV-Y7	522
14	India	PG-13	233
15	France	TV-MA	3989
16	France	R	1366
17	France	PG-13	798
18	France	TV-14	686
19	France	PG	643
20	Canada	TV-MA	2018
21	Canada	R	1514
22	Canada	PG	1038
23	Canada	TV-14	764
24	Canada	PG-13	647

In [83]:
```
fig, axes = plt.subplots(1, 5, figsize=(20, 6))  # Create 1 row, 5 columns of pi

for i, country in enumerate(top_5_ratings_by_country['country'].unique()):
    country_data = top_5_ratings_by_country[top_5_ratings_by_country['country']
    axes[i].pie(country_data['count'], labels=country_data['rating'], autopct='%
    axes[i].set_title(f'{country} Rating Distribution')

# Display the pie charts
plt.tight_layout()
plt.show()
```



TV-MA is the most watched content across the globe except India which is tv-14 inlclined , so can say that audience likes the maturity and R- rated content the most.

In [84]:
```
# duration column which consist of both tv shows and movies
df_duration_tv=df_final[df_final['duration'].str.contains('Season')].groupby('du
df_duration_tv
```

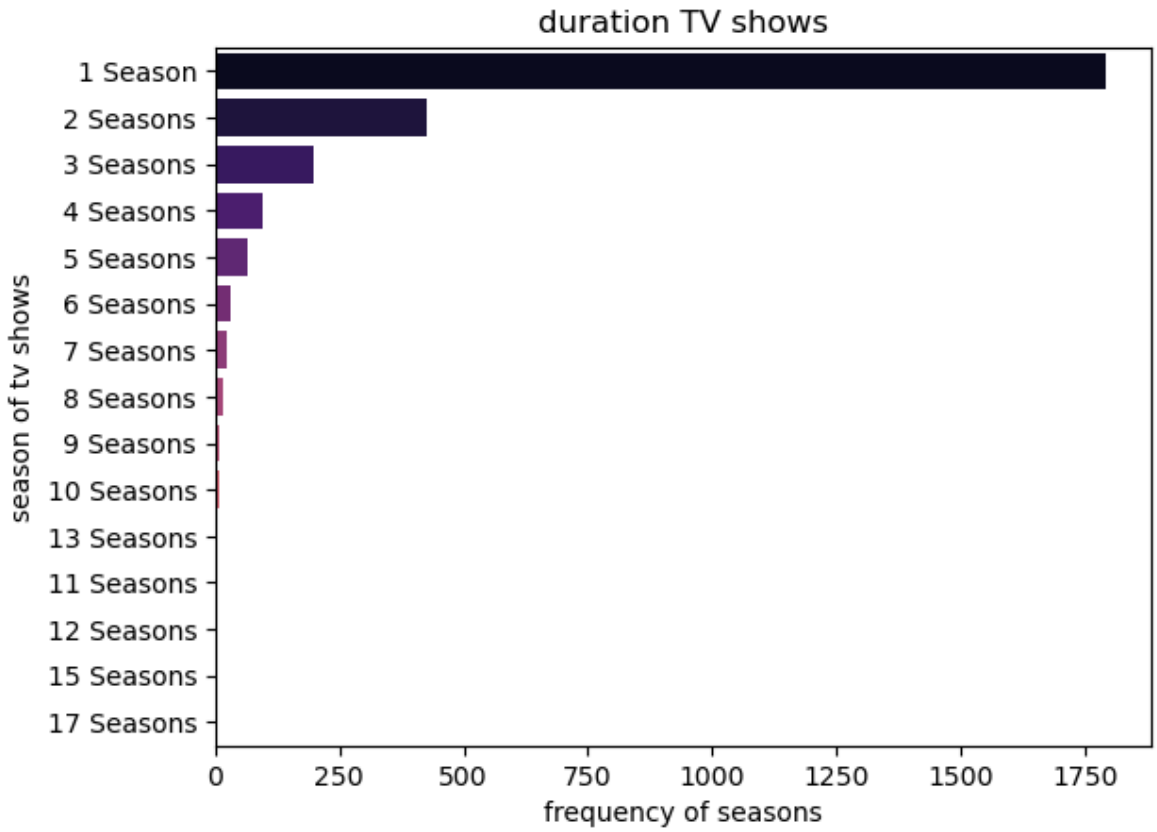
Out[84]:

	duration	title
0	1 Season	1793
7	2 Seasons	425
8	3 Seasons	199
9	4 Seasons	95
10	5 Seasons	65
11	6 Seasons	33
12	7 Seasons	23
13	8 Seasons	17
14	9 Seasons	9
1	10 Seasons	7
4	13 Seasons	3
2	11 Seasons	2
3	12 Seasons	2
5	15 Seasons	2
6	17 Seasons	1

In [85]:
```
sns.barplot(data=df_duration_tv,y='duration',x='title',palette='magma')
plt.title('duration TV shows')
plt.xlabel('frequency of seasons')
plt.ylabel('season of tv shows')
```

```
C:\Users\Rohit\AppData\Local\Temp\ipykernel_11272\758331560.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(data=df_duration_tv,y='duration',x='title',palette='magma')
```

Out[85]:
```
Text(0, 0.5, 'season of tv shows')
```



i can say that most tv seasons concluded after season 5 and which still continued must be supported by audience views and interest

In [86]:
```
tv_shows= df_final[df_final['type']=='TV Show'].copy()
tv_shows['num_seasons'] = tv_shows['duration'].str.extract('(\d+)').astype(float
tv_shows_gt_10=tv_shows[tv_shows['num_seasons']>10]
tv_shows_gt_10['title'].unique()
```

```
<>:2: SyntaxWarning: invalid escape sequence '\d'
<>:2: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Rohit\AppData\Local\Temp\ipykernel_11272\2620119772.py:2: SyntaxWarning:
invalid escape sequence '\d'
  tv_shows['num_seasons'] = tv_shows['duration'].str.extract('(\d+)').astype(floa
t)
```

Out[86]:
```
array(["Grey's Anatomy", 'Heartland', 'Supernatural',
       'COMEDIANS of the world', 'NCIS', 'Trailer Park Boys',
       'Criminal Minds', 'Cheers', 'Frasier', 'Red vs. Blue'],
      dtype=object)
```

In [87]:
```
tv_shows_gt_10[['title','Genre']]
tv_shows_gt_10.groupby('title').agg({'Genre':'unique'})
# if want to explode the list so can use expolode() to make unnested data in gen
```

Out[87]:

title	Genre
COMEDIANS of the world	[Stand-Up Comedy & Talk Shows, TV Comedies]
Cheers	[Classic & Cult TV, TV Comedies]
Criminal Minds	[Crime TV Shows, TV Dramas, TV Mysteries]
Frasier	[Classic & Cult TV, TV Comedies]
Grey's Anatomy	[Romantic TV Shows, TV Dramas]
Heartland	[TV Dramas]
NCIS	[Crime TV Shows, TV Dramas, TV Mysteries]
Red vs. Blue	[TV Action & Adventure, TV Comedies, TV Sci-Fi...
Supernatural	[Classic & Cult TV, TV Action & Adventure, TV ...
Trailer Park Boys	[Classic & Cult TV, Crime TV Shows, Internatio...

'Greys Anatomy', 'Heartland', 'Supernatural', 'Comedians Of The World', 'Ncis', 'Trailer Park Boys', 'Criminal Minds', 'Cheers', 'Frasier', 'Red Vs Blue'] these are the tv shows which are which continued after season 10 must be popular and fascinating genre it comprises are classic & cult , tv comedies, crime tv shows , tv dramas, tv science, that genre audience wants to see them that they should never ending feeling.

In [88]:
```
df_duration_movies=df_final[df_final['duration'].str.contains('-')].groupby('dur
df_duration_movies
```

Out[88]:

	duration	title
5	50-100	3030
1	100-150	2569
0	1-50	287
2	150-200	226
3	200-250	16
4	250-312	3

In [89]:
```
df_duration_movies_ctry=df_final[df_final['duration'].str.contains('-')].groupby
pd.set_option('display.max_colwidth', None) # to see the complete list
df_duration_movies_ctry
```

Out[89]:

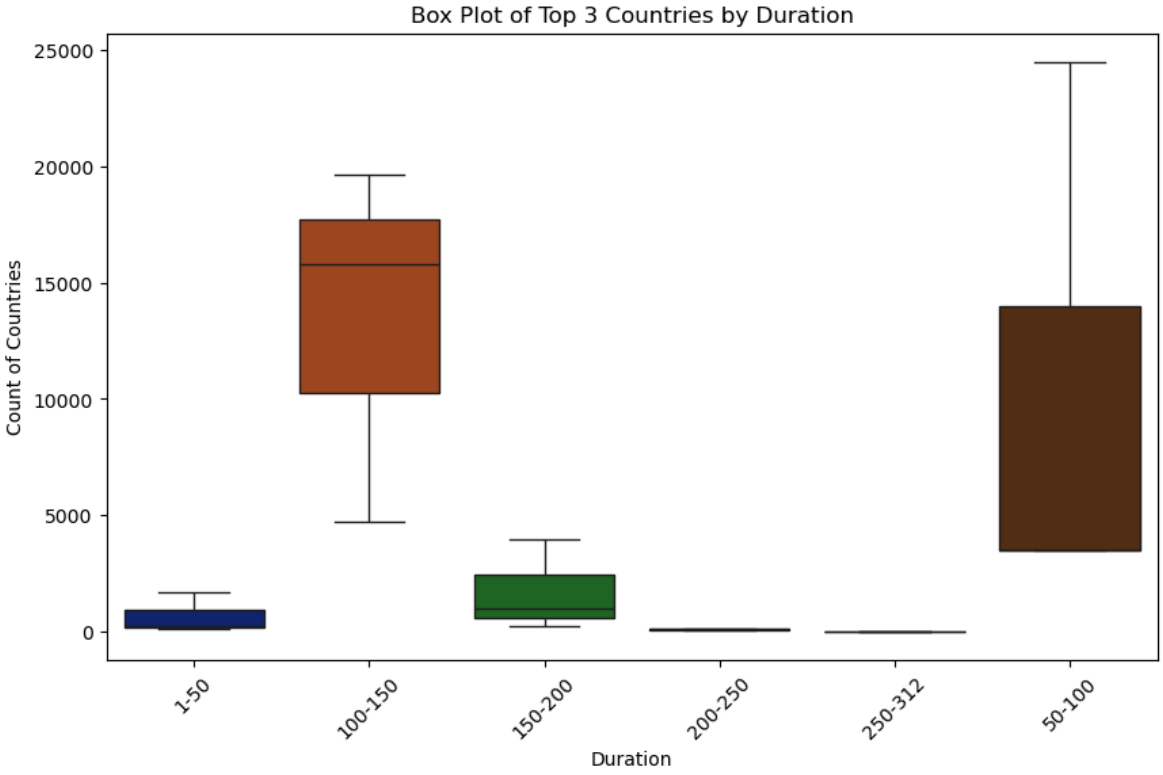
	duration	country	
		unique	count
0	1-50	[United States, Unknown_country, India, South Korea, Lebanon, Syria, Mexico, Egypt, Nigeria, United Kingdom, Canada, Spain, Namibia, Kenya, South Africa, France, Japan, Germany, Pakistan, Poland, Sweden, Netherlands, Ireland, Georgia, Argentina, Greece, Denmark, China, Australia, Italy, Brazil]	2530
1	100-150	[United States, Ghana, Burkina Faso, United Kingdom, Germany, Ethiopia, Czech Republic, India, France, Unknown_country, Belgium, South Africa, Japan, Nigeria, Canada, Australia, Mexico, Italy, China, Hong Kong, Taiwan, Philippines, New Zealand, Brazil, Greece, Switzerland, Argentina, Spain, , Algeria, Turkey, Israel, Denmark, Thailand, Indonesia, Kuwait, Egypt, Malaysia, South Korea, Vietnam, Hungary, Lebanon, Romania, Sweden, Finland, Netherlands, Cameroon, Poland, Ireland, Russia, Chile, Colombia, Cambodia, Bangladesh, Portugal, Norway, Iceland, Singapore, Serbia, Malta, Kenya, Saudi Arabia, Bulgaria, Angola, Peru, Mozambique, United Arab Emirates, Jordan, Senegal, Luxembourg, Pakistan, Austria, Malawi, Paraguay, Uruguay, Iran, Albania, Qatar, Soviet Union, Georgia, Morocco, Slovakia, West Germany, Armenia, Mongolia, Bahamas, Latvia, Venezuela, Nicaragua, Nepal, Jamaica, Somalia, Sudan, Namibia, Zimbabwe]	75415
2	150-200	[India, Unknown_country, Nigeria, United States, Italy, Romania, United Kingdom, Japan, Egypt, Kuwait, Mexico, Indonesia, France, Switzerland, Germany, Taiwan, Denmark, Pakistan, Soviet Union, Malaysia, Belgium, Spain, Hong Kong, Singapore, Turkey, Liechtenstein, Netherlands, Canada, Sweden, Norway, Morocco, New Zealand, Croatia, Slovenia, Serbia, Montenegro]	6737
3	200-250	[Italy, United States, Japan, India, United Kingdom, Kuwait, Unknown_country, Egypt, Morocco, New Zealand]	481
4	250-312	[United States, Egypt]	43
5	50-100	[United States, Brazil, United Kingdom, France, Unknown_country, China, Canada, Japan, Germany, Spain, Sweden, Philippines, Australia, Argentina, Venezuela, India, Egypt, Italy, Nepal, Nigeria, Colombia, Belgium, South Africa, Bulgaria, Taiwan, , Poland, Mexico, South Korea, New Zealand, Saudi Arabia, Denmark, Switzerland, Hong Kong, Cameroon, Netherlands, Singapore, Indonesia, Lebanon, United Arab Emirates, Syria, Qatar, Mauritius, Austria, Russia, Turkey, Palestine, Cuba, Ireland, Kenya, Chile, Uruguay, Portugal, Thailand, Cayman Islands, Algeria, Malaysia, Iceland, Luxembourg, Norway, Czech Republic, Serbia, Jordan, Vietnam, Cambodia, Zimbabwe, Kuwait, Romania, Hungary, Ghana, Guatemala, Finland, Peru, Iraq, Pakistan, Bangladesh, Israel, Iran, Ukraine, Bermuda, Ecuador, Sri Lanka, Morocco, Greece, Croatia, Slovenia, Dominican Republic, Senegal, Samoa, West Germany, Botswana, Vatican City, Kazakhstan, Lithuania, Afghanistan, Panama, Uganda, East Germany]	60637

In [90]:
```
top3_countries = (df_final[df_final['duration'].str.contains('-')].explode('coun
print(top3_countries)
```

	duration	country	count
0	1-50	United States	1678
1	1-50	Canada	221
2	1-50	United Kingdom	146
3	100-150	United States	19658
4	100-150	India	15771
5	100-150	United Kingdom	4701
6	150-200	India	3964
7	150-200	United States	980
8	150-200	United Kingdom	247
9	200-250	India	131
10	200-250	United States	102
11	200-250	United Kingdom	68
12	250-312	United States	22
13	250-312	Egypt	21
14	50-100	United States	24488
15	50-100	Canada	3529
16	50-100	United Kingdom	3493

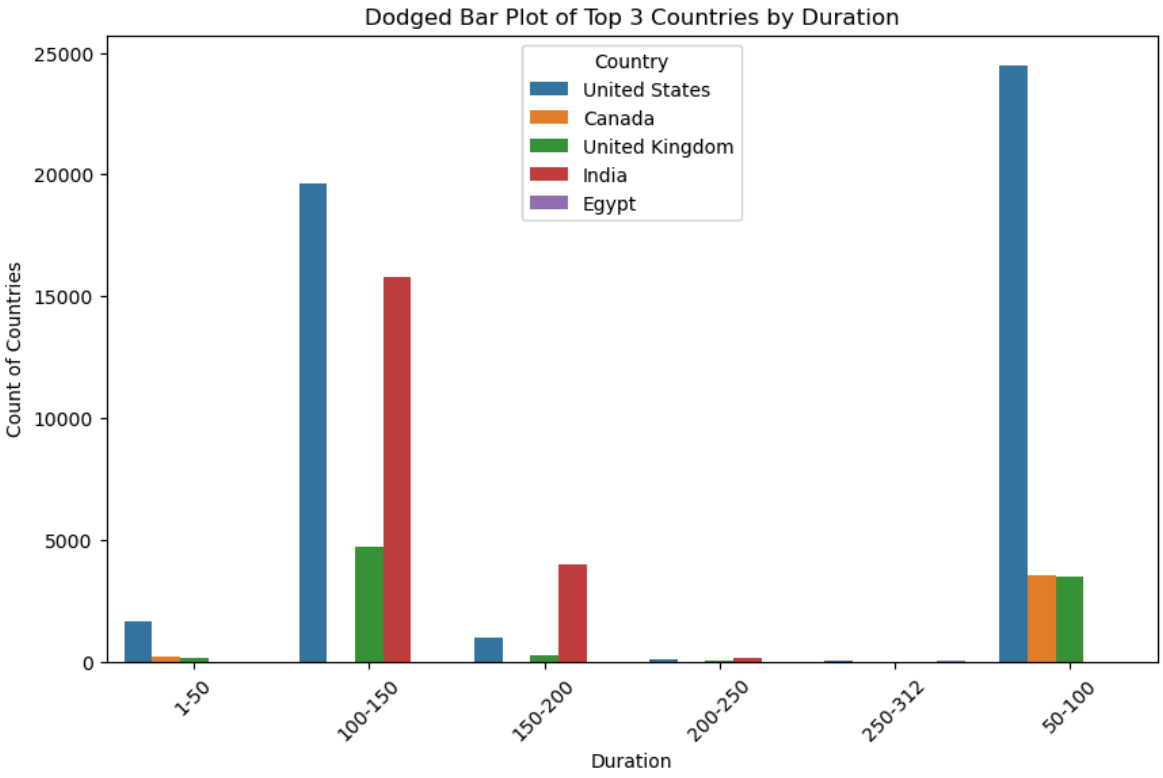
In [91]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='duration', y='count', data=top3_countries,palette='dark')
plt.title('Box Plot of Top 3 Countries by Duration')
plt.xlabel('Duration')
plt.ylabel('Count of Countries')
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\Rohit\AppData\Local\Temp\ipykernel_11272\2250370667.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot(x='duration', y='count', data=top3_countries,palette='dark')
```



In [92]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='duration', y='count', hue='country', data=top3_countries, dodge=T
```

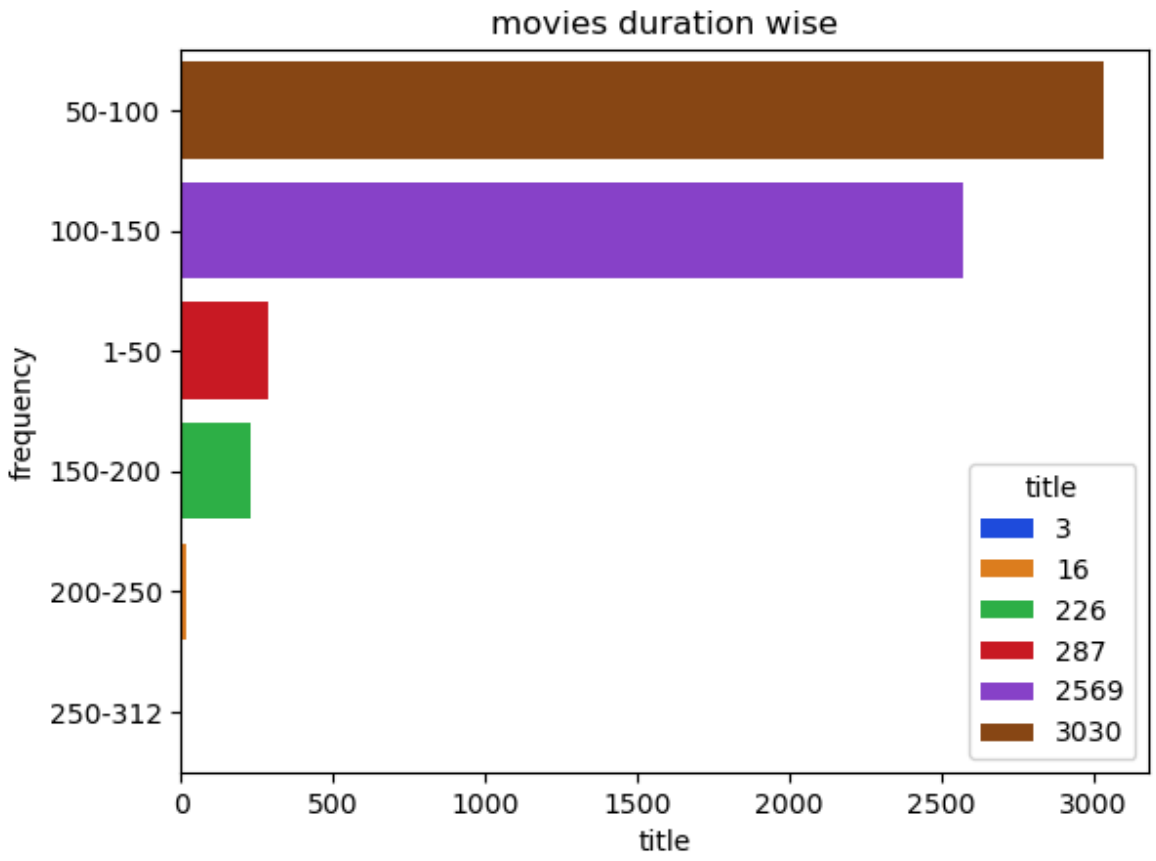
```
plt.title('Dodged Bar Plot of Top 3 Countries by Duration')
plt.xlabel('Duration')
plt.ylabel('Count of Countries')
plt.xticks(rotation=45)
plt.legend(title='Country')
plt.show()
```



united states prodcing highest titles in almost every segment of duration of movies while and india mostly in 150-200 and 200-250 segment which between 2 too 3.5 hr movie segment.

In [93]:
```
sns.barplot(data=df_duration_movies,x='title',y='duration',palette='bright',hue=
plt.title('movies duration wise')
plt.ylabel('frequency')
```

Out[93]:
```
Text(0, 0.5, 'frequency')
```



most of the movies are in the range of 50-100 or 100-150 minutes that must be most standard time for most movies and must be the director's choice as well

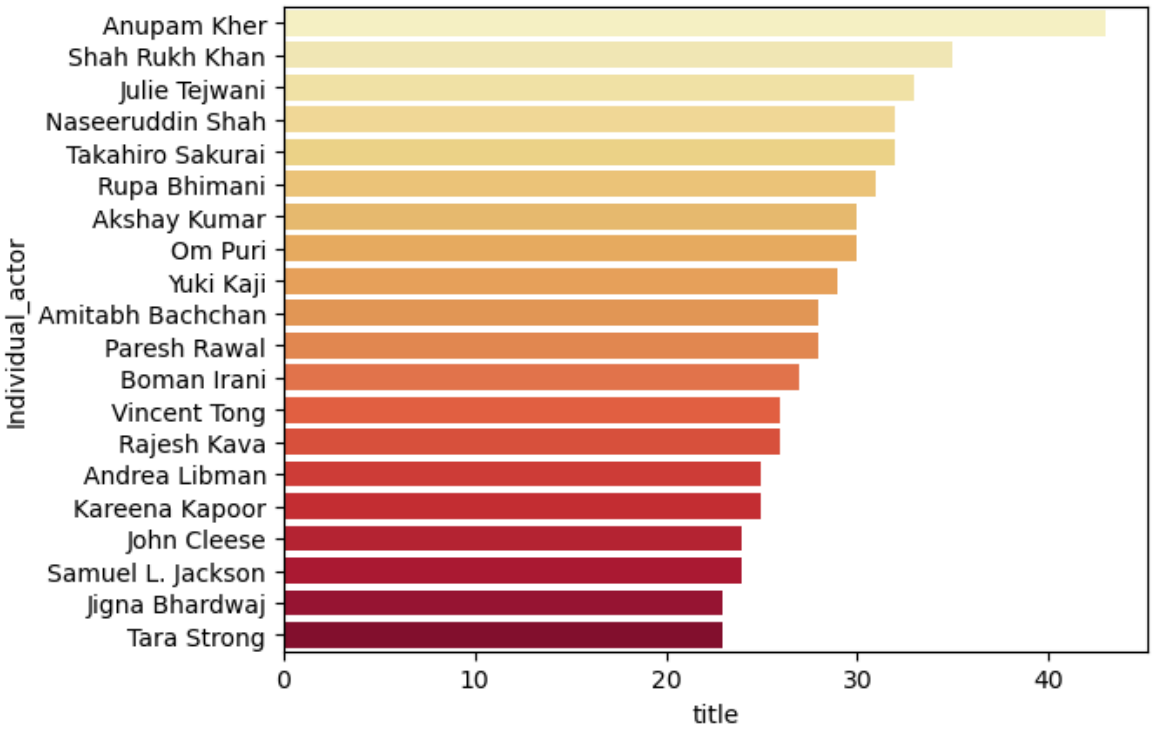
In [94]:
```
# we can do on actors now
df_actors=df_final[df_final['Individual_actor']!='Unknown_actor'].groupby('Indiv
df_actors=df_actors.sort_values(by='title',ascending=False).head(20)
df_actors
```

Out[94]:

	Individual_actor	title
2833	Anupam Kher	43
30489	Shah Rukh Khan	35
16697	Julie Tejwani	33
24215	Naseeruddin Shah	32
32591	Takahiro Sakurai	32
28974	Rupa Bhimani	31
846	Akshay Kumar	30
25424	Om Puri	30
35880	Yuki Kaji	29
1774	Amitabh Bachchan	28
25782	Paresh Rawal	28
4528	Boman Irani	27
34717	Vincent Tong	26
27355	Rajesh Kava	26
2063	Andrea Libman	25
17221	Kareena Kapoor	25
15690	John Cleese	24
29600	Samuel L. Jackson	24
15232	Jigna Bhardwaj	23
32795	Tara Strong	23

In [95]:
```
sns.barplot(data=df_actors,x='title',y='Individual_actor',hue='Individual_actor'
```

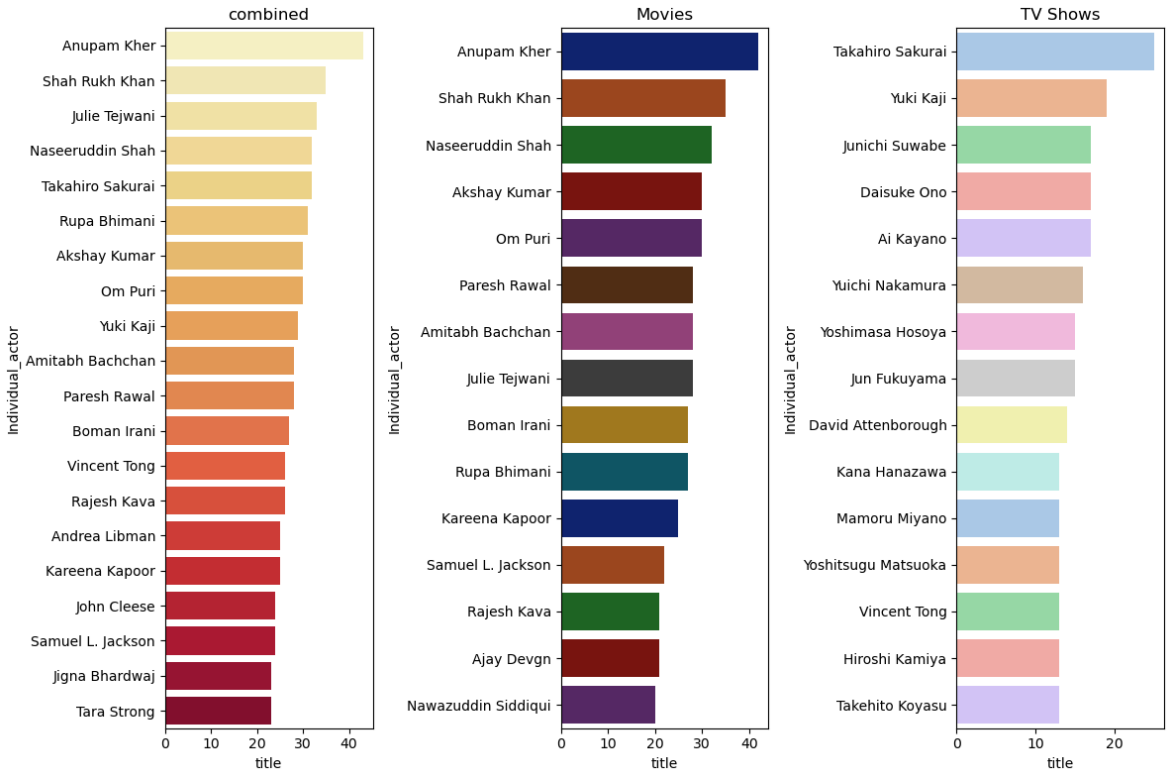
Out[95]:
```
<Axes: xlabel='title', ylabel='Individual_actor'>
```



Aumpam Kher has been the most active actors in terms of number of movies follwed by Shahrukh khan, julie Tejawani and Naseeruddin Shah but we can't say that doing more movies is tantamount to Best actor also.but we can conclude he must be director's choice who wants to work with that actor.

In [96]:
```
#pd.reset_option("display.max_rows")
df_actors_movie=df_final[(df_final['Individual_actor']!='Unknown_actor')& (df_fi
df_actors_TV=df_final[(df_final['Individual_actor']!='Unknown_actor')& (df_final
```

In [97]:
```
plt.figure(figsize=(12,8))
plt.subplot(1,3,1)
sns.barplot(data=df_actors,x='title',y='Individual_actor',hue='Individual_actor'
plt.title('combined')

plt.subplot(1,3,2)
sns.barplot(data=df_actors_movie,x='title',y='Individual_actor',hue='Individual_
plt.title('Movies')

plt.subplot(1,3,3)
sns.barplot(data=df_actors_TV,x='title',y='Individual_actor',hue='Individual_act
plt.title('TV Shows')

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



Aumpam Kher has been the most active actors in terms of number of movies follwed by Shahrukh khan, julie Tejawani and Naseeruddin Shah, but in tv show scenario is different Takahiro Sakurai, yuki kaji and Junichi Suwabe dominating the tv industry.

In [98]:
```
# let's find the director with whom these actors worked with
director_actor_title=df_final[df_final['Individual_actor'].isin(['Anupam Kher',
```

In [99]:
```
director_actor_title= df_final[df_final['Individual_actor'].isin(['Anupam Kher',
director_actor_title=director_actor_title.groupby(['Individual_actor','director'
director_actor_title
```

Out[99]:

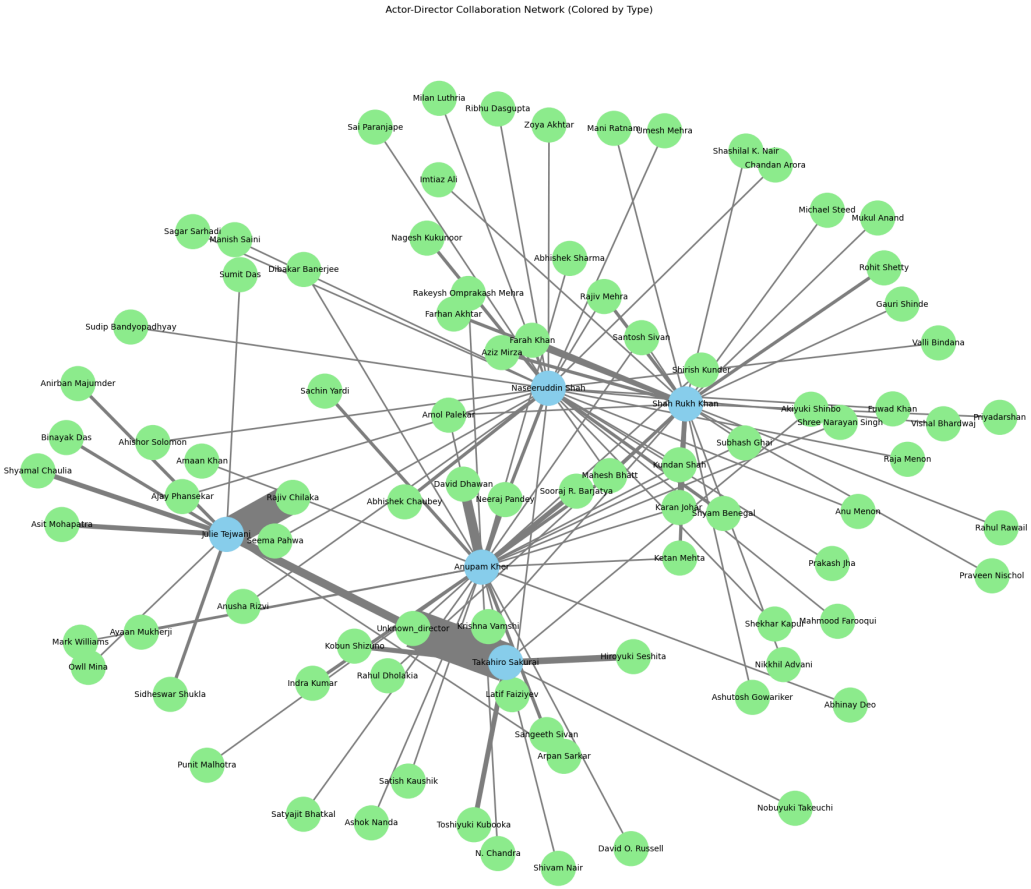
	Individual_actor	director	title
0	Anupam Kher	Abhinay Deo	[Game]
1	Anupam Kher	Abhishek Sharma	[The Shaukeens]
2	Anupam Kher	Amaan Khan	[Mahabharat]
3	Anupam Kher	Amol Palekar	[Paheli]
4	Anupam Kher	Ashok Nanda	[One Day: Justice Delivered]
...	...	...	...
96	Takahiro Sakurai	Hiroyuki Seshita	[GODZILLA The Planet Eater, GODZILLA City on the Edge of Battle, Godzilla, BLAME!]
97	Takahiro Sakurai	Kobun Shizuno	[GODZILLA The Planet Eater, GODZILLA City on the Edge of Battle, Godzilla]
98	Takahiro Sakurai	Nobuyuki Takeuchi	[Fireworks]
99	Takahiro Sakurai	Toshiyuki Kubooka	[Berserk: The Golden Age Arc II - The Battle for Doldrey, Berserk: The Golden Age Arc III - The Advent, Berserk: The Golden Age Arc I - The Egg of the King]
100	Takahiro Sakurai	Unknown_director	[Record of Ragnarok, JoJo's Bizarre Adventure, Thus Spoke Kishibe Rohan, Demon Slayer: Kimetsu no Yaiba, Food Wars!: Shokugeki no Soma, Zoids Wild, DRIFTING DRAGONS, 7SEEDS, Cagaster of an Insect Cage, SAINT SEIYA: Knights of the Zodiac, Cells at Work!, CAROLE & TUESDAY, Levius, Mobile Suit Gundam: Iron-Blooded Orphans, Anohana: The Flower We Saw That Day, Record of Grancrest War, Magi: The Labyrinth of Magic, March Comes in Like a Lion, Code Geass: Lelouch of the Rebellion, Sirius the Jaeger, AJIN: Demi-Human, K, Knights of Sidonia, Magi: Adventure of Sinbad]

101 rows × 3 columns

In [100…

```
import networkx as nx
import matplotlib.pyplot as plt

# Step 1: Filter and group data
actor_list = ['Anupam Kher', 'Shah Rukh Khan', 'Julie Tejwani', 'Naseeruddin Sha
filtered_df = df_final[df_final['Individual_actor'].isin(actor_list)][['Individu
grouped = filtered_df.groupby(['Individual_actor', 'director'])['title'].nunique

# Step 2: Build graph
G = nx.Graph()

for _, row in grouped.iterrows():
    actor = row['Individual_actor']
    director = row['director']
    count = row['title_count']
    G.add_node(actor, type='actor')
    G.add_node(director, type='director')
```

```
    G.add_edge(actor, director, weight=count)

# Step 3: Assign colors
node_colors = []
for node in G.nodes(data=True):
    if node[1]['type'] == 'actor':
        node_colors.append('skyblue')
    else:
        node_colors.append('lightgreen')

# Step 4: Draw graph
plt.figure(figsize=(20, 16))
pos = nx.spring_layout(G, k=0.6)

edges = G.edges(data=True)
weights = [d['weight'] * 2 for (_, _, d) in edges]  # scale line thickness by we

nx.draw(
    G, pos, with_labels=True, node_color=node_colors,
    edge_color='gray', width=weights, node_size=1800, font_size=10
)

plt.title("Actor-Director Collaboration Network (Colored by Type)")
plt.axis('off')
plt.show()
```



In [101…

```
#pd.reset_option("display.max_rows")
from datetime import datetime
current_year = datetime.now().year
last_5_years = current_year - 5
df_actors_5=df_final[(df_final['Individual_actor']!='Unknown_actor') & (df_final
```

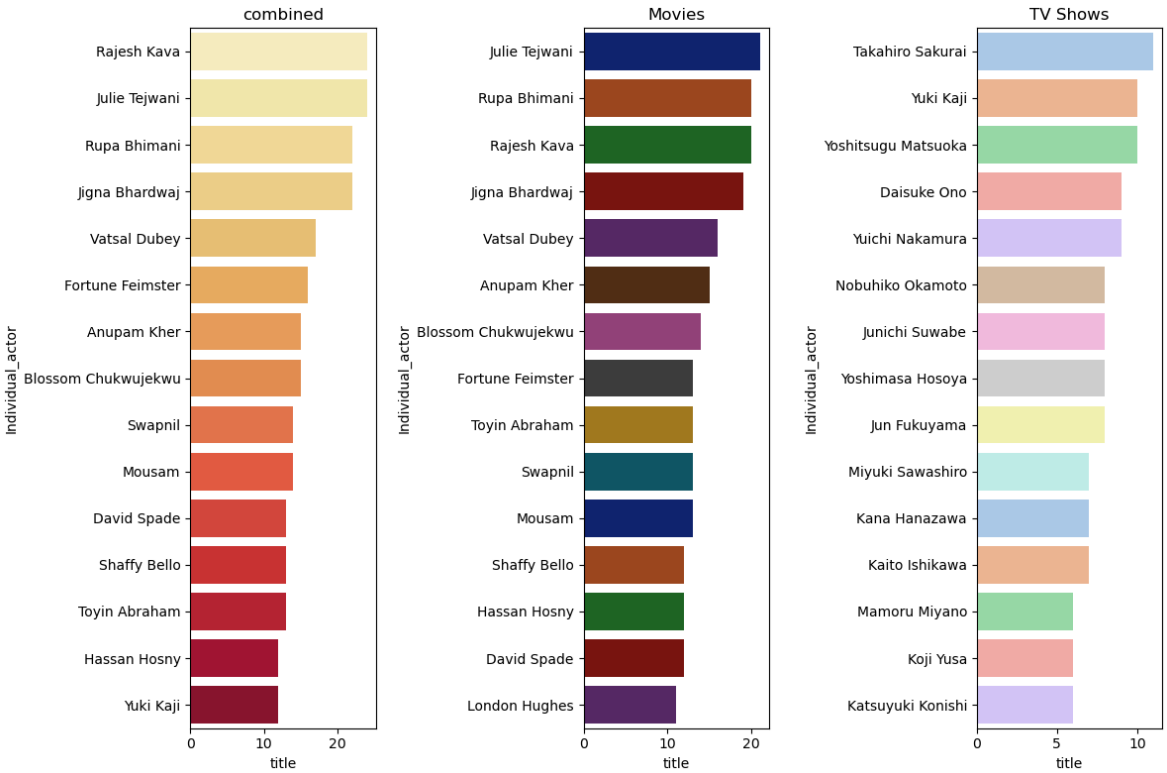
```
df_actors_movie_5=df_final[(df_final['Individual_actor']!='Unknown_actor')& (df_
df_actors_TV_5=df_final[(df_final['Individual_actor']!='Unknown_actor')& (df_fin
df_actors_5
```

Out[101…

	Individual_actor	title
14115	Rajesh Kava	24
8559	Julie Tejwani	24
14963	Rupa Bhimani	22
7833	Jigna Bhardwaj	22
17690	Vatsal Dubey	17
5521	Fortune Feimster	16
1482	Anupam Kher	15
2263	Blossom Chukwujekwu	15
16630	Swapnil	14
12325	Mousam	14
4128	David Spade	13
15708	Shaffy Bello	13
17418	Toyin Abraham	13
6386	Hassan Hosny	12
18363	Yuki Kaji	12

In [102…

```
# actors datat on the last five years to test who is most popular in the recent
plt.figure(figsize=(12,8))
plt.subplot(1,3,1)
sns.barplot(data=df_actors_5,x='title',y='Individual_actor',hue='Individual_acto
plt.title('combined')

plt.subplot(1,3,2)
sns.barplot(data=df_actors_movie_5,x='title',y='Individual_actor',hue='Individua
plt.title('Movies')

plt.subplot(1,3,3)
sns.barplot(data=df_actors_TV_5,x='title',y='Individual_actor',hue='Individual_a
plt.title('TV Shows')

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



the names i see in this data julie tejwani, rajesh kava and jigna bhardwaj they are the voice actors of the anime and cartoons it means i should test the data in this genre too.

In [103…

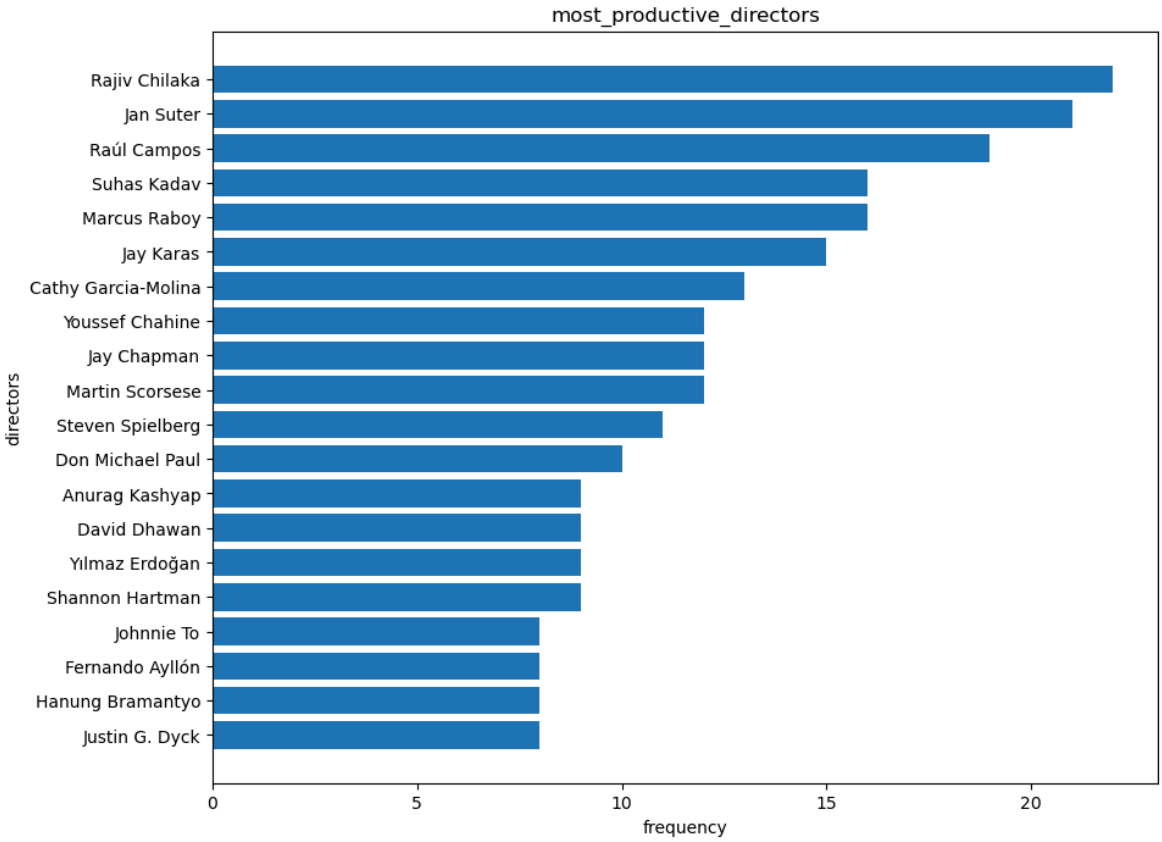
```
# now the most important one is the director
df_directors= df_final[df_final['director']!='Unknown_director'].groupby('direct
df_directors=df_directors.sort_values(by='title',ascending=False).head(20)
df_directors
```

Out[103…

	director	title
3749	Rajiv Chilaka	22
1906	Jan Suter	21
3800	Raúl Campos	19
4457	Suhas Kadav	16
2866	Marcus Raboy	16
1954	Jay Karas	15
755	Cathy Garcia-Molina	13
4941	Youssef Chahine	12
1951	Jay Chapman	12
2945	Martin Scorsese	12
4425	Steven Spielberg	11
1217	Don Michael Paul	10
403	Anurag Kashyap	9
1075	David Dhawan	9
4952	Yılmaz Erdoğan	9
4243	Shannon Hartman	9
2189	Johnnie To	8
1404	Fernando Ayllón	8
1671	Hanung Bramantyo	8
2353	Justin G. Dyck	8

In [104…

```
plt.figure(figsize=(10,8))
plt.barh(df_directors[::-1]['director'],df_directors[::-1]['title']) # barh to t
plt.title('most_productive_directors')
plt.ylabel('directors')
plt.xlabel('frequency')
plt.show()
```



In [105…

```
df_directors_movie= df_final[(df_final['director']!='Unknown_director') & (df_fi
df_directors_tv= df_final[(df_final['director']!='Unknown_director') & (df_final
df_directors_movie
```

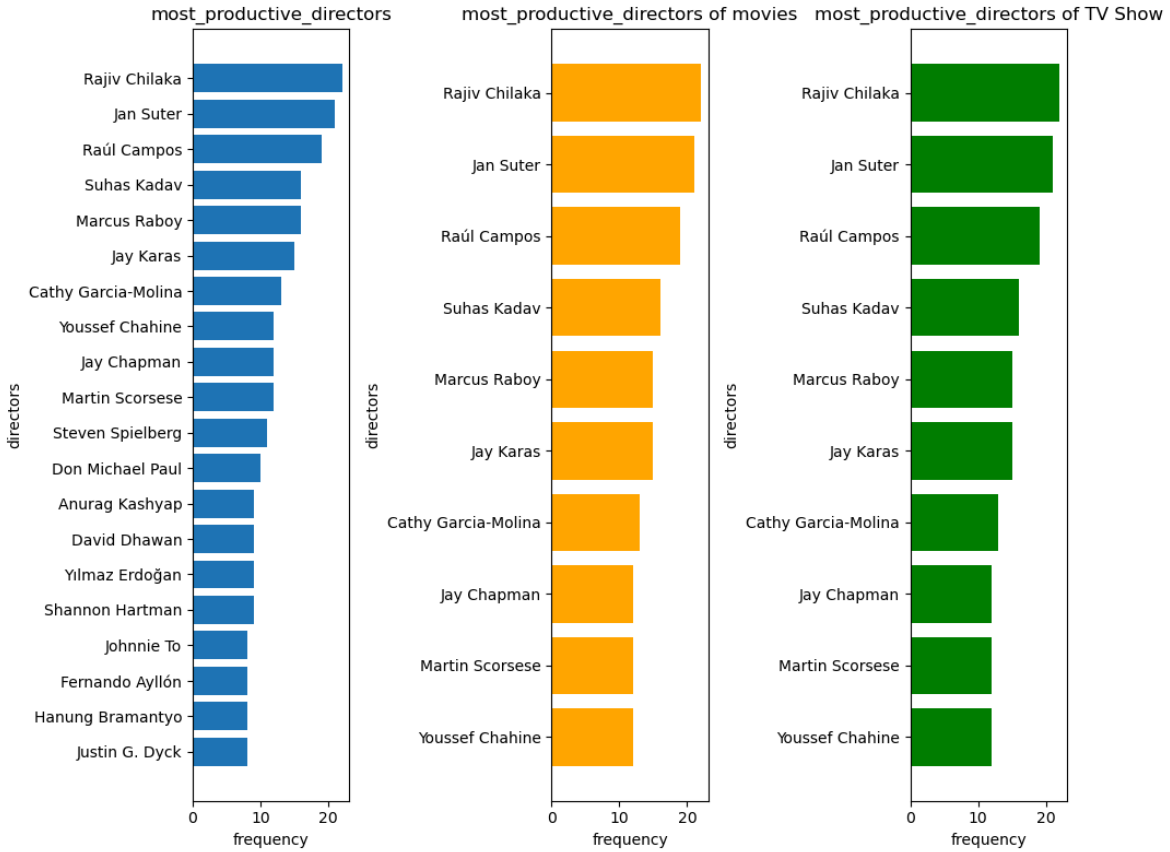
Out[105…

	director	title
3582	Rajiv Chilaka	22
1817	Jan Suter	21
3633	Raúl Campos	19
4261	Suhas Kadav	16
2739	Marcus Raboy	15
1862	Jay Karas	15
727	Cathy Garcia-Molina	13
1859	Jay Chapman	12
2815	Martin Scorsese	12
4725	Youssef Chahine	12

In [106…

```
plt.figure(figsize=(10,8))
plt.subplot(1,3,1)
plt.barh(df_directors[::-1]['director'],df_directors[::-1]['title']) # barh to t
plt.title('most_productive_directors')
plt.ylabel('directors')
plt.xlabel('frequency')
plt.subplot(1,3,2)
plt.barh(df_directors_movie[::-1]['director'],df_directors_movie[::-1]['title'],
plt.title('most_productive_directors of movies')
```

```
plt.ylabel('directors')
plt.xlabel('frequency')
plt.subplot(1,3,3)
plt.barh(df_directors_tv[::-1]['director'],df_directors_tv[::-1]['title'],color=
plt.title('most_productive_directors of TV Show')
plt.ylabel('directors')
plt.xlabel('frequency')

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



the names in the director is common throughout the combine as well as in movies and tv shows as well Rajiv Chilakia, Jan Sutter ,Raul Campos should check what kind of content these director create that is popular among the audiences.

In [107…

```
pd.reset_option("display.max_rows")
filtered_df = df_final[
    df_final['director'].isin([
        'Rajiv Chilaka', 'Jan Suter', 'Raúl Campos',
        'Suhas Kadav', 'Marcus Raboy', 'Jay Karas'
    ])
][['director', 'title']].groupby('director')['title'].apply(list).reset_index()

filtered_df
```

Out[107...

	director	title
0	Jan Suter	[Coco y Raulito: Carrusel de ternura, Coco y Raulito: Carrusel de ternura, Luciano Mellera: Infantiloide, Jani Dueñas: Grandes fracasos de ayer y hoy, Fernando Sanjiao: Hombre, Carlos Ballarta: Furia Ñera, Todo lo que sería Lucas Lauriente, Sofía Niño de Rivera: Selección Natural, Malena Pichot: Estupidez compleja, Natalia Valdebenito: El especial, Sebastián Marcelo Wainraich, Ricardo Quevedo: Hay gente así, Arango y Sanint: Ríase el show, Arango y Sanint: Ríase el show, Mea Culpa, El Especial de Alex Fernández, el Especial, Alan Saldaña: Mi vida de pobre, Simplemente Manu NNa, Daniel Sosa: Sosafado, Ricardo O'Farrill: Abrazo navideño, Ricardo O'Farrill: Abrazo navideño, Ricardo O'Farrill: Abrazo navideño, Ricardo O'Farrill: Abrazo navideño, Ricardo O'Farrill: Abrazo navideño, Carlos Ballarta: El amor es de putos, Sofía Niño de Rivera: Exposed, Ricardo O'Farrill Abrazo Genial]
1	Jay Karas	[The Main Event, Demetri Martin: The Overthinker, Adam Devine: Best Time of Our Lives, Bill Burr: You People Are All the Same, Ali Wong: Hard Knock Wife, Tom Segura: Disgraceful, Christina P: Mother Inferior, Bill Burr: Walk Your Way Out, Jeff Foxworthy and Larry the Cable Guy: We've Been Thinking..., Jeff Foxworthy and Larry the Cable Guy: We've Been Thinking..., Jim Gaffigan: Mr. Universe, Ali Wong: Baby Cobra, Tom Segura: Mostly Stories, Anjelah Johnson: Not Fancy, Demetri Martin: Live (At the Time), Bill Burr: I'm Sorry You Feel That Way]
2	Marcus Raboy	[Patton Oswalt: I Love Everything, Patton Oswalt: I Love Everything, Patton Oswalt: I Love Everything, Patton Oswalt: I Love Everything, Taylor Tomlinson: Quarter-Life Crisis, Whitney Cummings: Can I Touch It?, Miranda Sings Live... Your Welcome, Anthony Jeselnik: Fire in the Maternity Ward, Vir Das: Losing It, Katt Williams: Kattpacalypse, Steve Martin and Martin Short: An Evening You Will Forget for the Rest of Your Life, Steve Martin and Martin Short: An Evening You Will Forget for the Rest of Your Life, Marlon Wayans: Woke-ish, Judd Apatow: The Return, DeRay Davis: How to Act Black, Ryan Hamilton: Happy Face, Lynne Koplitz: Hormonal Beast, Vir Das: Abroad Understanding, Cristela Alonzo: Lower Classy, Dana Carvey: Straight White Male, 60]
3	Rajiv Chilaka	[Chhota Bheem - Neeli Pahaadi, Chhota Bheem - Neeli Pahaadi, Chhota Bheem - Neeli Pahaadi, Chhota Bheem - Neeli Pahaadi, Chhota Bheem - Neeli Pahaadi, Chhota Bheem - Neeli Pahaadi, Chhota Bheem - Neeli Pahaadi, Chhota Bheem - Neeli Pahaadi, Chhota Bheem & Ganesh, Chhota Bheem & Ganesh, Chhota Bheem & Ganesh, Chhota Bheem & Ganesh, Chhota Bheem & Ganesh, Chhota Bheem & Ganesh, Chhota Bheem & Ganesh, Chhota Bheem & Krishna: Mayanagari, Chhota Bheem & Krishna: Mayanagari, Chhota Bheem & Krishna: Mayanagari, Chhota Bheem & Krishna: Mayanagari, Chhota Bheem & Krishna: Mayanagari, Chhota Bheem & Krishna: Mayanagari, Chhota Bheem & Krishna: Pataliputra- City of the Dead, Chhota Bheem & Krishna: Pataliputra- City of the Dead, Chhota Bheem & Krishna: Pataliputra- City of the Dead, Chhota Bheem & Krishna: Pataliputra- City of the Dead, Chhota Bheem & Krishna: Pataliputra- City of the Dead, Chhota Bheem & Krishna: Pataliputra- City of the Dead, Chhota Bheem & Krishna: Pataliputra- City of the Dead, Chhota Bheem And The Broken Amulet, Chhota Bheem And The Broken Amulet, Chhota Bheem And The Broken Amulet, Chhota Bheem And The Broken Amulet, Chhota Bheem And The Broken Amulet, Chhota Bheem And The Broken Amulet, Chhota Bheem And The Crown of Valhalla, Chhota Bheem And The Crown of Valhalla, Chhota Bheem And The Crown of Valhalla, Chhota Bheem And The Crown of Valhalla, Chhota Bheem And The Crown of Valhalla, Chhota

	director	title
		Bheem And The Crown of Valhalla, Chhota Bheem And The Crown of Valhalla, Chhota Bheem and the Incan Adventure, Chhota Bheem and the Incan Adventure, Chhota Bheem and the Incan Adventure, Chhota Bheem and the Incan Adventure, Chhota Bheem and the Incan Adventure, Chhota Bheem and the Incan Adventure, Chhota Bheem and the Incan Adventure, Chhota Bheem and The ShiNobi Secret, Chhota Bheem and The ShiNobi Secret, Chhota Bheem and The ShiNobi Secret, Chhota Bheem and The ShiNobi Secret, Chhota Bheem and The ShiNobi Secret, Chhota Bheem and The ShiNobi Secret, Chhota Bheem and The ShiNobi Secret, Chhota Bheem Aur Hanuman, Chhota Bheem Aur Hanuman, Chhota Bheem Aur Hanuman, Chhota Bheem Aur Hanuman, Chhota Bheem Aur Hanuman, Chhota Bheem Aur Hanuman, Chhota Bheem Aur Hanuman, Chhota Bheem aur Krishna, Chhota Bheem aur Krishna, Chhota Bheem aur Krishna, Chhota Bheem aur Krishna, Chhota Bheem aur Krishna, Chhota Bheem aur Krishna, Chhota Bheem aur Krishna, Chhota Bheem aur Krishna vs Zimbara, Chhota Bheem aur Krishna vs Zimbara, Chhota Bheem aur Krishna vs Zimbara, Chhota Bheem aur Krishna vs Zimbara, Chhota Bheem aur Krishna vs Zimbara, Chhota Bheem aur Krishna vs Zimbara, Chhota Bheem aur Krishna vs Zimbara, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Bheem vs Aliens, Chhota Bheem: Dholakpur to Kathmandu, Chhota Bheem: Dholakpur to Kathmandu, Chhota Bheem: Dholakpur to Kathmandu, Chhota Bheem: Dholakpur to Kathmandu, Chhota Bheem: Dholakpur to Kathmandu, Chhota Bheem: Dholakpur to Kathmandu, Chhota Bheem: Dholakpur to Kathmandu, Chhota Bheem: Dus Pe Dus, Chhota Bheem: Dus Pe Dus, ...]
4	Raúl Campos	[Coco y Raulito: Carrusel de ternura, Coco y Raulito: Carrusel de ternura, Luciano Mellera: Infantiloide, Jani Dueñas: Grandes fracasos de ayer y hoy, Fernando Sanjiao: Hombre, Carlos Ballarta: Furia Ñera, Todo lo que sería Lucas Lauriente, Sofía Niño de Rivera: Selección Natural, Malena Pichot: Estupidez compleja, Natalia Valdebenito: El especial, Sebastián Marcelo Wainraich, Ricardo Quevedo: Hay gente así, Arango y Sanint: Ríase el show, Arango y Sanint: Ríase el show, Mea Culpa, Alan Saldaña: Mi vida de pobre, Simplemente Manu NNa, Daniel Sosa: Sosafado, Ricardo O'Farrill: Abrazo navideño, Ricardo O'Farrill: Abrazo navideño, Ricardo O'Farrill: Abrazo navideño, Ricardo O'Farrill: Abrazo navideño, Ricardo O'Farrill: Abrazo navideño, Sofía Niño de Rivera: Exposed, Ricardo O'Farrill Abrazo Genial]
5	Suhas Kadav	[Motu Patlu in Dragon's World, Motu Patlu in Dragon's World, Motu Patlu in the Game of Zones, Motu Patlu in the Game of Zones, Motu Patlu in the Game of Zones, Motu Patlu in Wonderland, Motu Patlu: Deep Sea Adventure, Motu Patlu: Deep Sea Adventure, Motu Patlu: Deep Sea Adventure, Motu Patlu: Deep Sea Adventure, Motu Patlu: Deep Sea Adventure, Motu Patlu: Deep Sea Adventure, Motu Patlu: Deep Sea Adventure, Motu Patlu: Deep Sea Adventure, Motu Patlu: Deep Sea Adventure, Motu Patlu: Deep Sea Adventure, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu:

director	title
	Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu: Mission Moon, Motu Patlu Dino Invasion, Motu Patlu Dino Invasion, Motu Patlu Dino Invasion, Motu Patlu Dino Invasion, Motu Patlu Dino Invasion, Motu Patlu Dino Invasion, Motu Patlu in Octupus World, Motu Patlu in Octupus World, Motu Patlu in Octupus World, Motu Patlu in Octupus World, Motu Patlu in Octupus World, Motu Patlu in Octupus World, Motu Patlu in Octupus World, Motu Patlu in Octupus World, Motu Patlu in Octupus World, Motu Patlu in Octupus World, Motu Patlu in Octupus World, Motu Patlu in Octupus World, Motu Patlu VS Robo Kids, Motu Patlu VS Robo Kids, Motu Patlu in Hong Kong: Kung Fu Kings 3, Motu Patlu in Hong Kong: Kung Fu Kings 3, Motu Patlu in Hong Kong: Kung Fu Kings 3, Motu Patlu in Hong Kong: Kung Fu Kings 3, Motu Patlu in Hong Kong: Kung Fu Kings 3, Motu Patlu in Hong Kong: Kung Fu Kings 3, Motu Patlu in Hong Kong: Kung Fu Kings 3, Motu Patlu in Hong Kong: Kung Fu Kings 3, Motu Patlu in Hong Kong: Kung Fu Kings 3, Motu Patlu in the City of Gold, Motu Patlu in the City of Gold, Motu Patlu in the City of Gold, Motu Patlu in the City of Gold, Motu Patlu in the City of Gold, Motu Patlu in the City of Gold, Motu Patlu in the City of Gold, Motu Patlu in the City of Gold, Motu Patlu in the City of Gold, Motu Patlu in the City of Gold, ...]

In [108…

```
df_final[
    df_final['director'].isin([
        'Rajiv Chilaka', 'Jan Suter', 'Raúl Campos',
        'Suhas Kadav', 'Marcus Raboy', 'Jay Karas'
    ])
].groupby(['director', 'Genre'])['title'].nunique().reset_index(name='unique_tit
```

Out[108...

	director	Genre	unique_title_count
0	Jan Suter	Stand-Up Comedy	21
1	Jay Karas	Children & Family Movies	1
2	Jay Karas	Comedies	1
3	Jay Karas	Sports Movies	1
4	Jay Karas	Stand-Up Comedy	14
5	Marcus Raboy	Stand-Up Comedy	15
6	Marcus Raboy	Stand-Up Comedy & Talk Shows	1
7	Marcus Raboy	TV Comedies	1
8	Rajiv Chilaka	Children & Family Movies	22
9	Rajiv Chilaka	Sports Movies	2
10	Raúl Campos	Stand-Up Comedy	19
11	Suhas Kadav	Children & Family Movies	16
12	Suhas Kadav	Comedies	8
13	Suhas Kadav	Music & Musicals	5

so i can Rajiv Chilaka who topped the list of most poular director is making movies of Children and Family(Chhota Bheem) that means this genre has huge audience as well, while jan sutter and raul campos is known for stand-up comedy(Coco Y Raulito Carrusel De Ternura).

In [109...

```
top_actor_genre= df_final[df_final['Individual_actor'].isin(['Anupam Kher', 'Sha
top_actor_genre=top_actor_genre.groupby(['Individual_actor'])['Genre'].unique().
top_actor_genre

# to plot the graph need to explode the list here
```

Out[109...

	Individual_actor	Genre
0	Anupam Kher	[Action & Adventure, Comedies, International Movies, Music & Musicals, Sci-Fi & Fantasy, Dramas, Independent Movies, Thrillers, Romantic Movies, Classic Movies, Children & Family Movies, Crime TV Shows, International TV Shows, TV Comedies]
1	Julie Tejwani	[Kids' TV, Children & Family Movies, Sports Movies, Movies]
2	Naseeruddin Shah	[Comedies, Dramas, International Movies, Independent Movies, Romantic Movies, Children & Family Movies, Documentaries, Thrillers, Action & Adventure, Music & Musicals, Sci-Fi & Fantasy]
3	Shah Rukh Khan	[Dramas, International Movies, Thrillers, Action & Adventure, Comedies, Romantic Movies, Music & Musicals, Sci-Fi & Fantasy]
4	Takahiro Sakurai	[Anime Series, International TV Shows, Teen TV Shows, Kids' TV, Anime Features, Romantic Movies, Action & Adventure, International Movies, TV Thrillers, TV Shows, Crime TV Shows]

In [110…

```
#pd.reset_option("display.max_rows")
top_actor_genre= df_final[df_final['Individual_actor'].isin(['Anupam Kher', 'Sha
top_actor_genre
```

Out[110…

	Individual_actor	Genre
990	Julie Tejwani	Kids' TV
2491	Shah Rukh Khan	Dramas
2492	Shah Rukh Khan	International Movies
2493	Shah Rukh Khan	Thrillers
4722	Anupam Kher	Action & Adventure
4723	Anupam Kher	Comedies
4724	Anupam Kher	International Movies
5029	Anupam Kher	Music & Musicals
5082	Anupam Kher	Sci-Fi & Fantasy
5404	Naseeruddin Shah	Comedies
5405	Naseeruddin Shah	Dramas
5406	Naseeruddin Shah	International Movies
7327	Shah Rukh Khan	Action & Adventure
7328	Shah Rukh Khan	Comedies
10055	Julie Tejwani	Children & Family Movies
10162	Julie Tejwani	Sports Movies
18072	Takahiro Sakurai	Anime Series
18073	Takahiro Sakurai	International TV Shows
18116	Anupam Kher	Dramas
18117	Anupam Kher	Independent Movies
24222	Naseeruddin Shah	Independent Movies
45609	Takahiro Sakurai	Teen TV Shows
46347	Naseeruddin Shah	Romantic Movies
48253	Naseeruddin Shah	Children & Family Movies
50893	Takahiro Sakurai	Kids' TV
54235	Anupam Kher	Thrillers
55280	Anupam Kher	Romantic Movies
65214	Shah Rukh Khan	Romantic Movies
70473	Anupam Kher	Classic Movies
74110	Takahiro Sakurai	Anime Features
74111	Takahiro Sakurai	Romantic Movies
75458	Shah Rukh Khan	Music & Musicals
78814	Naseeruddin Shah	Documentaries

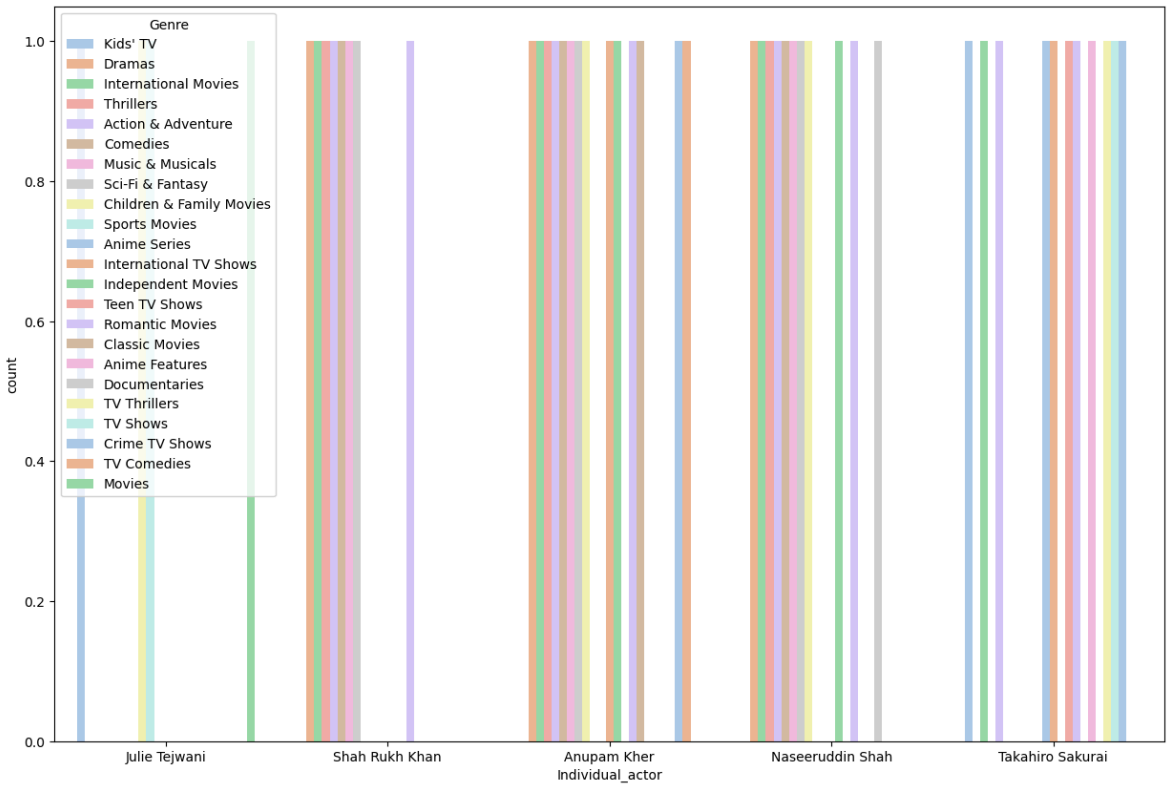
	Individual_actor	Genre
98582	Takahiro Sakurai	Action & Adventure
98584	Takahiro Sakurai	International Movies
99913	Takahiro Sakurai	TV Thrillers
104187	Anupam Kher	Children & Family Movies
109847	Naseeruddin Shah	Thrillers
112625	Naseeruddin Shah	Action & Adventure
117110	Takahiro Sakurai	TV Shows
117915	Anupam Kher	Crime TV Shows
117916	Anupam Kher	International TV Shows
117917	Anupam Kher	TV Comedies
118288	Naseeruddin Shah	Music & Musicals
125410	Shah Rukh Khan	Sci-Fi & Fantasy
125413	Naseeruddin Shah	Sci-Fi & Fantasy
145744	Julie Tejwani	Movies
162528	Takahiro Sakurai	Crime TV Shows

In [111…

```
plt.figure(figsize=(15,10))
sns.countplot(data=top_actor_genre,x='Individual_actor',hue='Genre',palette='pas
```

Out[111…

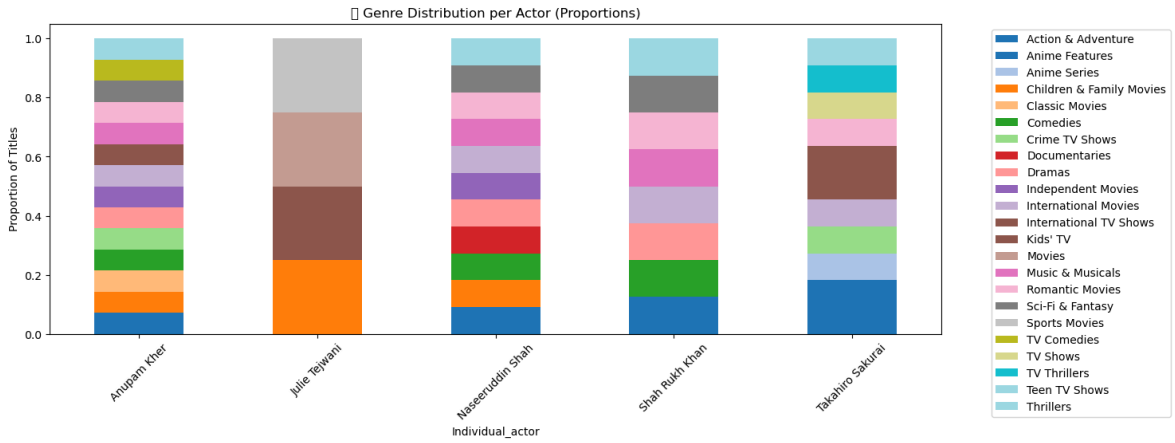
```
<Axes: xlabel='Individual_actor', ylabel='count'>
```



In [112…

```
# Count genres per actor
genre_counts = df_final[
    df_final['Individual_actor'].isin([
        'Anupam Kher', 'Shah Rukh Khan', 'Julie Tejwani',
        'Naseeruddin Shah', 'Takahiro Sakurai'
    ])
][['Individual_actor', 'Genre']].drop_duplicates()

# Pivot for stacked bar and size is to calculate number of rows and
genre_counts = genre_counts.groupby(['Individual_actor', 'Genre']).size().reset_
pivot_df = genre_counts.pivot(index='Individual_actor', columns='Genre', values=

pivot_df_norm = pivot_df.div(pivot_df.sum(axis=1), axis=0)

# Plot normalized bar
pivot_df_norm.plot(kind='bar', stacked=True, figsize=(12, 6), colormap='tab20')
plt.title(' Genre Distribution per Actor (Proportions)')
plt.ylabel('Proportion of Titles')
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

```
C:\Users\Rohit\AppData\Local\Temp\ipykernel_11272\1141695466.py:20: UserWarning:
Glyph 128202 (\N{BAR CHART}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
C:\Users\Rohit\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:170: UserWa
rning: Glyph 128202 (\N{BAR CHART}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



here we can say Anupam Kher and Naseeruddin shah have been more versatile actor who worked in so many genre so did Shah Rukh Khan and Takahiro Sakurai while Julie Tejwani who is voice artitst for anime sticked to children & Family Movie and sports and tv Gnere that was his forte.

In [113…

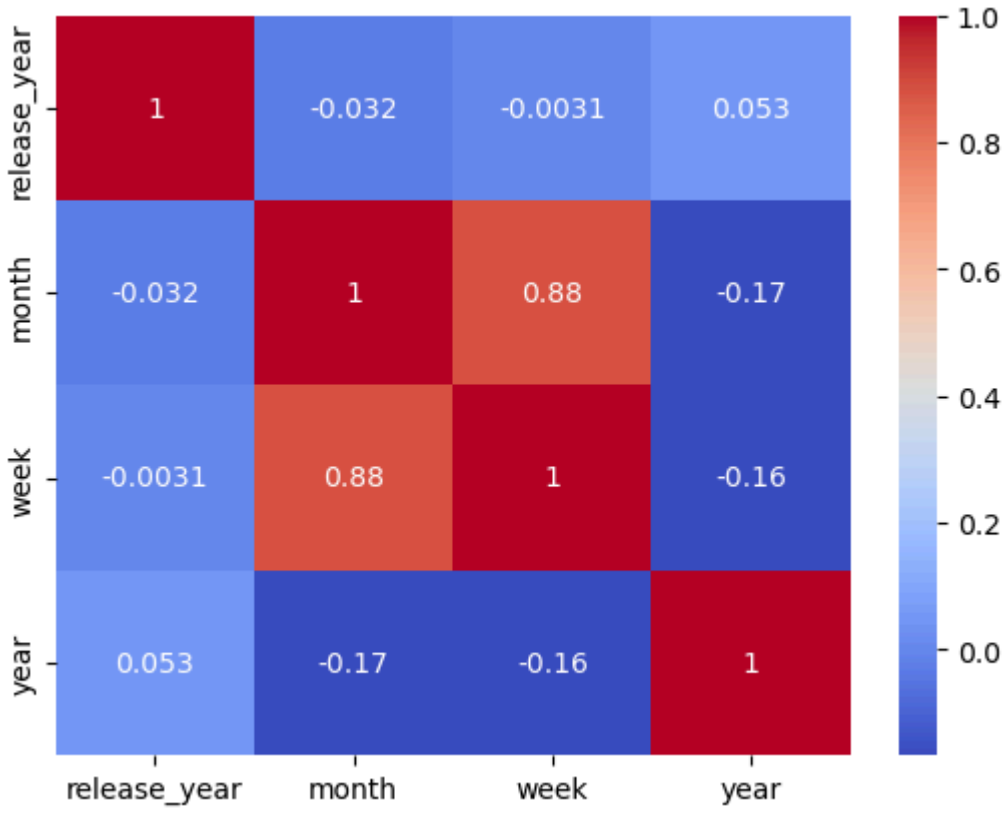
```
df_corr=df_final.select_dtypes(include=np.number).corr()
```

In [114…

```
sns.heatmap(df_corr,cmap='coolwarm',annot=True)
# these year week  month and  release_year from the date added
```

Out[114…

```
<Axes: >
```



In [115…

```
pd.reset_option("display.max_rows")
df_final
```

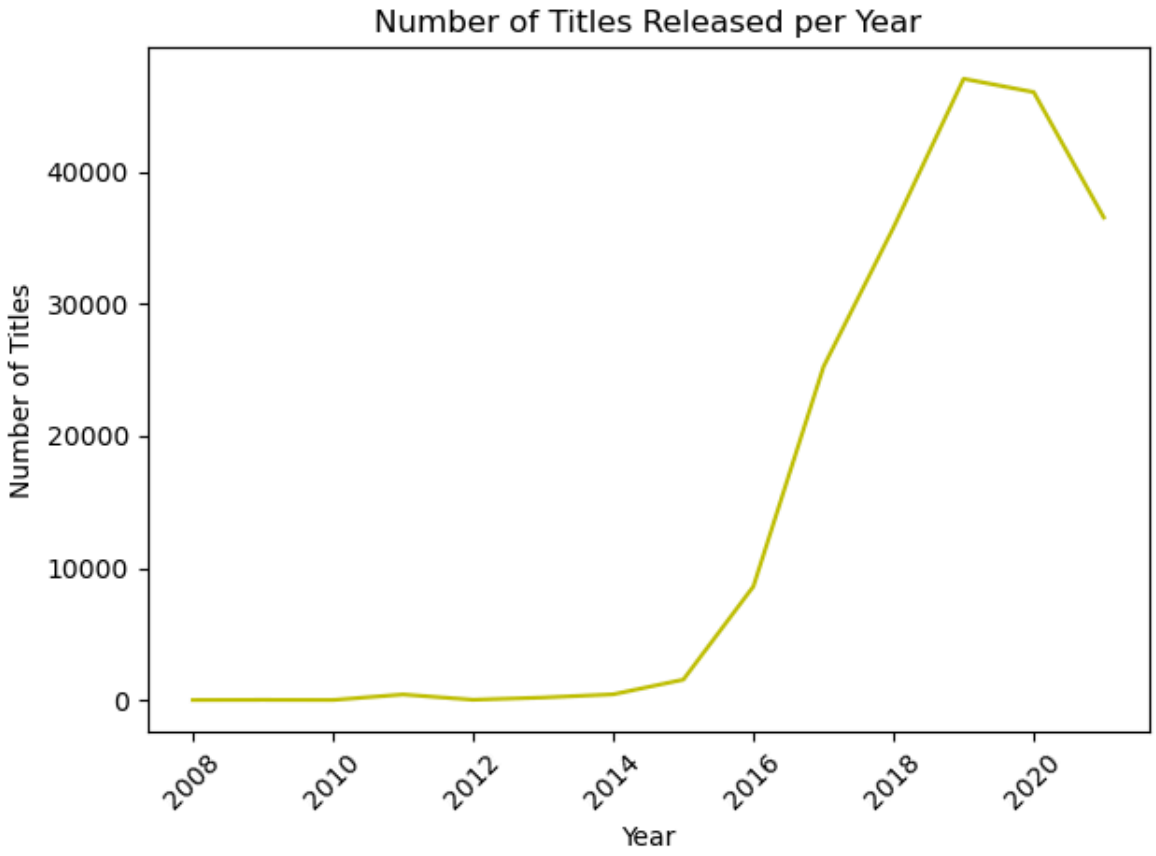
Out[115...

	title	Individual_actor	country	director	Genre	show_id
0	Dick Johnson Is Dead	Unknown_actor	United States	Kirsten Johnson	Documentaries	s1
1	Blood & Water	Ama Qamata	South Africa	Unknown_director	International TV Shows	s2
2	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Dramas	s2
3	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Mysteries	s2
4	Blood & Water	Khosi Ngema	South Africa	Unknown_director	International TV Shows	s2
...	...	...	...	...	...	...
201986	Zubaan	Anita Shabdish	India	Mozez Singh	International Movies	s8807
201987	Zubaan	Anita Shabdish	India	Mozez Singh	Music & Musicals	s8807
201988	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Dramas	s8807
201989	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	International Movies	s8807
201990	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Music & Musicals	s8807

201991 rows × 16 columns

In [116...

```
# Count number of titles per year
yearly_counts = df_final.groupby('year')['title'].count().reset_index()

sns.lineplot(data=yearly_counts, x='year', y='title',color='y')
plt.title('Number of Titles Released per Year')
plt.xlabel('Year')
plt.ylabel('Number of Titles')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



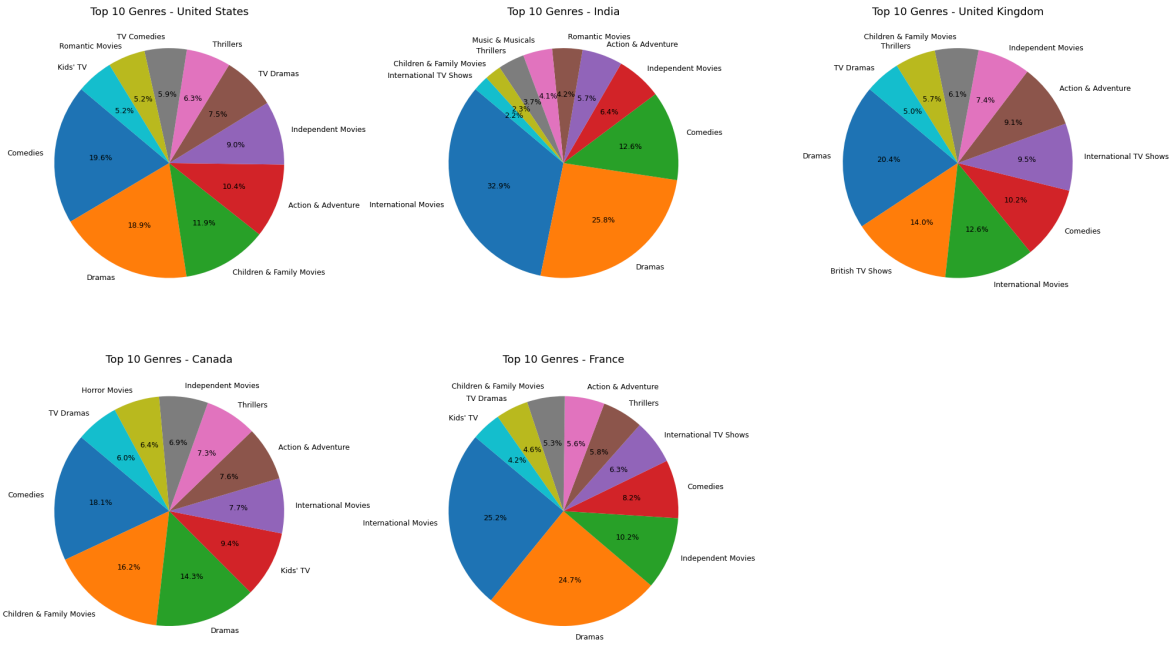
In [117…

```
# want to test top 5 countries genre distribution
df_country_genre_us=df_final[df_final['country']=='United States'].groupby('Genr
df_country_genre_India=df_final[df_final['country']=='India'].groupby('Genre').a
df_country_genre_UK=df_final[df_final['country']=='United Kingdom'].groupby('Gen
df_country_genre_Canada=df_final[df_final['country']=='Canada'].groupby('Genre')
df_country_genre_france=df_final[df_final['country']=='France'].groupby('Genre')
```

In [118…

```
import matplotlib.pyplot as plt

# Create figure with 2 rows & 3 columns (one will be empty)
fig, axes = plt.subplots(2, 3, figsize=(20, 12))

# List of your dataframes and titles
genre_data = [
    (df_country_genre_us, "United States"),
    (df_country_genre_India, "India"),
    (df_country_genre_UK, "United Kingdom"),
    (df_country_genre_Canada, "Canada"),
    (df_country_genre_france, "France")
]

# Loop through each subplot and assign pie chart
for ax, (df, country) in zip(axes.flat, genre_data):
    ax.pie(
        df['title'],
        labels=df['Genre'],
        autopct='%1.1f%%',
        startangle=140,
        textprops={'fontsize': 9}
    )
    ax.set_title(f"Top 10 Genres - {country}", fontsize=13)

# Hide the 6th (empty) subplot
```

```
if len(genre_data) < 6:
    axes.flat[-1].axis('off')

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



we can see that how different genres popular across diffrent countries in US COMEDY, DRAMAS , CHILDREN & FAMILY MOVIES ACTION AND ADVENTURE dominating the market share. In India surplrisingly it is INTERNATIONAL MOVIES, DRAMAS CONSTITUTE MORE THAN 55% followed by comdies by 12%. almost similar to France where people also majorly watching these two. In United Kingdom there is dramas and their own british shows is popular among them .

we have already seen in the non graphic analysis that from the year 2018 the number of titles got added is highest till 2021

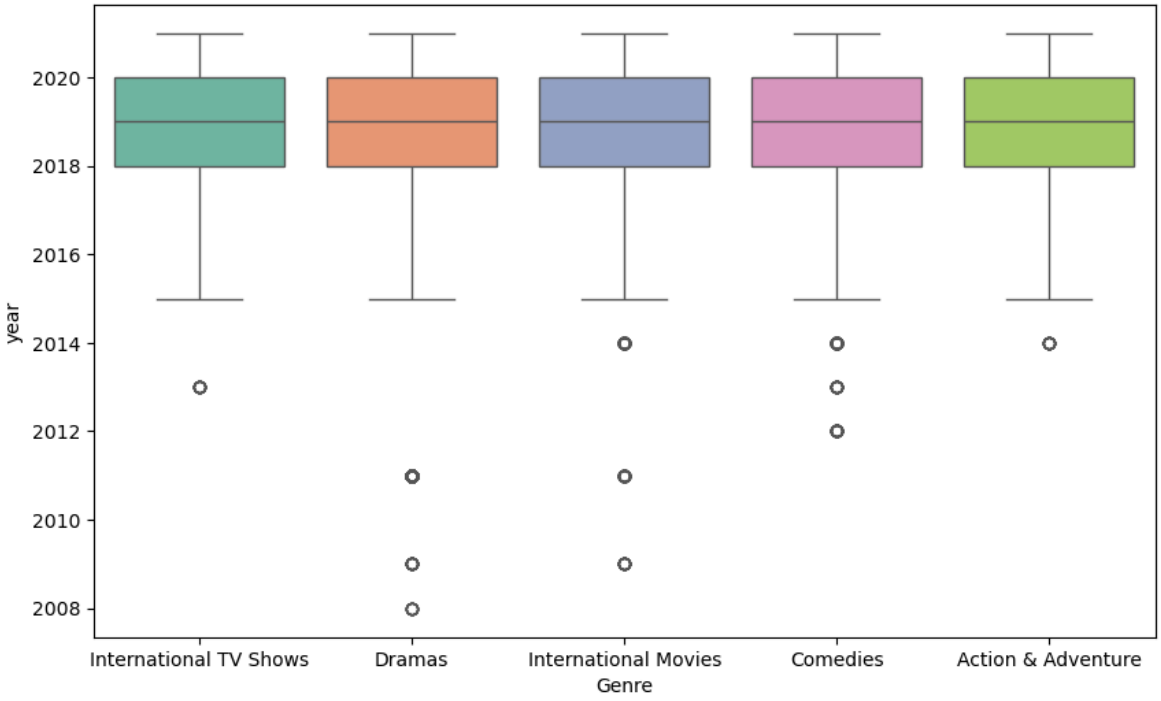
In [119…

```
# want to test the top 7 genre across the years
top_5_genre=df_final['Genre'].value_counts().head(5).index  # can also do the
top_5_genre=df_final[df_final['Genre'].isin(top_5_genre)]
top_5_genre

plt.figure(figsize=(10, 6))
sns.boxplot(data=top_5_genre,x='Genre',y='year',palette='Set2')
```

```
C:\Users\Rohit\AppData\Local\Temp\ipykernel_11272\2064681025.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot(data=top_5_genre,x='Genre',y='year',palette='Set2')
```

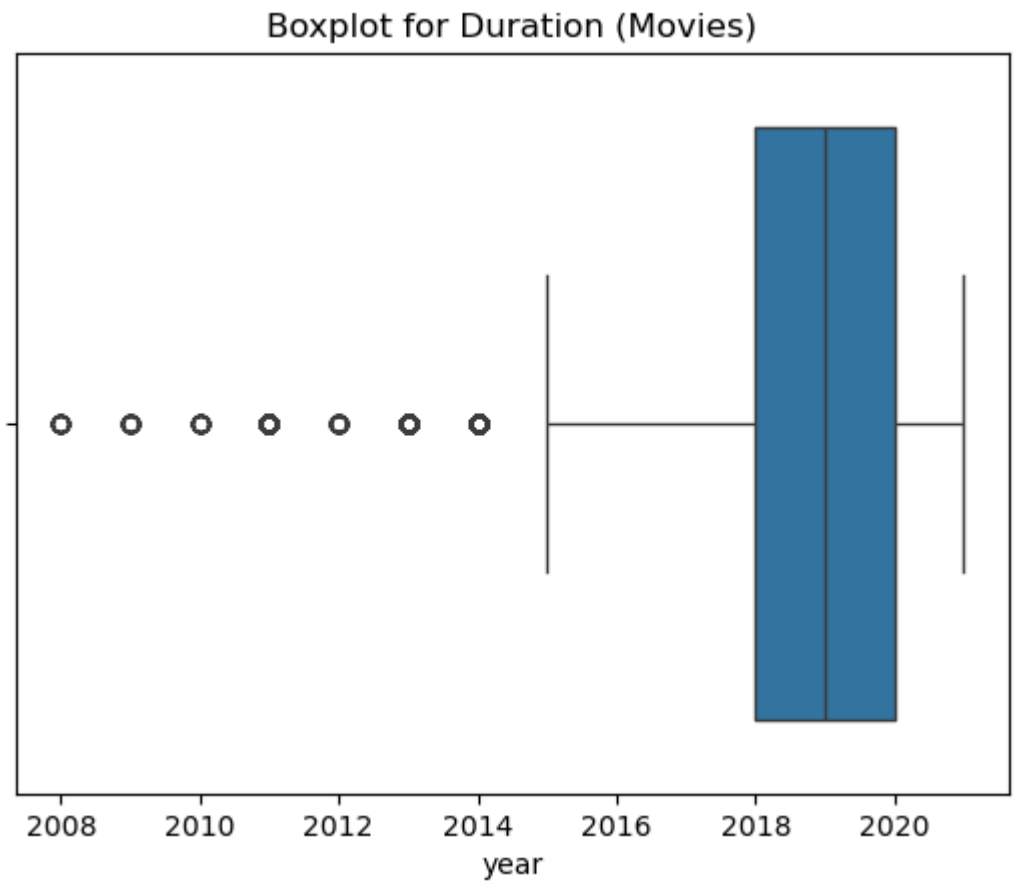
Out[119…

```
<Axes: xlabel='Genre', ylabel='year'>
```



In [120…

```
sns.boxplot(x=df_final['year'])
plt.title('Boxplot for Duration (Movies)')
plt.show()
```



In [121…

```
Q1 = df_final['year'].quantile(0.25)
Q3 = df_final['year'].quantile(0.75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
```

```
upper_bound = Q3 + 1.5 * IQR

df_cleaned=df_final[(df_final['year'] >= lower_bound) &  (df_final['year'] <= up
df_cleaned
```

Out[121...

	title	Individual_actor	country	director	Genre	show_id
0	Dick Johnson Is Dead	Unknown_actor	United States	Kirsten Johnson	Documentaries	s1
1	Blood & Water	Ama Qamata	South Africa	Unknown_director	International TV Shows	s2
2	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Dramas	s2
3	Blood & Water	Ama Qamata	South Africa	Unknown_director	TV Mysteries	s2
4	Blood & Water	Khosi Ngema	South Africa	Unknown_director	International TV Shows	s2
...	...	...	...	...	...	...
201986	Zubaan	Anita Shabdish	India	Mozez Singh	International Movies	s8807
201987	Zubaan	Anita Shabdish	India	Mozez Singh	Music & Musicals	s8807
201988	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Dramas	s8807
201989	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	International Movies	s8807
201990	Zubaan	Chittaranjan Tripathy	India	Mozez Singh	Music & Musicals	s8807

200791 rows × 16 columns

recomendations for the review

• most liked and evergreen content on the both the platfrom is both mvoies and tv shows Dramas , International movies and TVs shows and comedies,in tv - crime tv shows that must be updated in the libraries on monnthly basis. while poplar actors mostly folowing the trend of action adventure which is quite popular in movies.

• most audience prefer movies in the [50-150]minutes segment. This must be also be a directors's choice as well.

• audience likes to watch more versatile and most famous directors movies, library of content must include them on bi-montly basis.

• recent popular actor and directors combo cannot be overlooked for audience, which even surpassing the best actors of the decade.

• content for child& family is less as conmpared to other content. • audience prefers the tv-MA and R-rating content the most while in India they are more inclined to tv-14 content.

• library content is india is almost half from 2018 to 2021.

• most popular tv shows among the public are of Genre classic & cult , tv comedies, crime tv shows , tv dramas, tv science, audiencde are more baised towards them.

• recently korean and Japaneese shows are getting popular audience has a new region to their liking especialy the anima and korean dramas.

• in the month of july and december and is the best month to add the most awaited content or exlusive titles due to summer season and christmas holiday.

In [ ]: