*** By Savita ***

Project On Netflix

Description of data set

Dataset Description

- **Show_id**: Unique ID for every Movie / TV Show
- **Type**: Identifier - A Movie or TV Show
- **Title**: Title of the Movie / TV Show
- **Director**: Director of the Movie
- **Cast**: Actors involved in the movie/show
- **Country**: Country where the movie/show was produced
- **Date_added**: Date it was added on Netflix
- **Release_year**: Actual Release year of the movie/show
- **Rating**: TV Rating of the movie/show
- **Duration**: Total Duration (in minutes for movies or number of seasons for TV shows)
- **Listed_in**: Genre(s) of the movie/show
- **Description**: A brief summary or description of the movie/show

Project Objectives Conduct Exploratory Data Analysis (EDA) on the provided dataset to identify patterns, trends, and valuable insights. Offer actionable insights from the analysis to better understand the dataset's behavior and characteristics. Formulate business recommendations that can assist Netflix in expanding, enhancing its content strategy, and addressing regional preferences.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Load the dataset

```
# Load the dataset
df = pd.read_csv('netflix_titles.csv')
```

```
df.shape
```

```
(8807, 12)
```

Display the first few rows of the dataset

```
print("First few rows of the dataset:")
df.head(5)
```

```
First few rows of the dataset:
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...

```
#  information about the dataset
print("\nDataset information:")
print(df.info())
```

```
Dataset information:
<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
None
```

```
df.nunique()
```

```
show_id         8807
type               2
title           8807
director        4528
cast            7692
country          748
date_added      1767
release_year      74
rating            17
duration         220
listed_in        514
description     8775
dtype: int64
```

Info nd Describe Data set of Netfilx

```
df.describe()
```

```
       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
```

Missing values

```
# Check for missing values
print("\nMissing values in the dataset:")
print(df.isnull().sum())
```

```
Missing values in the dataset:
show_id            0
type               0
title              0
director        2634
cast             825
country          831
date_added        10
release_year       0
rating             4
duration           3
listed_in          0
description        0
dtype: int64
```

Data Cleaning

```
df[df['duration'].isna()]
```

```
     show_id   type                                 title    director
\
5541   s5542  Movie                        Louis C.K. 2017  Louis C.K.

5794   s5795  Movie                Louis C.K.: Hilarious  Louis C.K.

5813   s5814  Movie  Louis C.K.: Live at the Comedy Store  Louis C.K.


            cast        country          date_added  release_year
rating  \
5541  Louis C.K.  United States       April 4, 2017          2017  74
min
5794  Louis C.K.  United States  September 16, 2016          2010  84
min
5813  Louis C.K.  United States     August 15, 2016          2015  66
```

```
min

     duration listed_in
description
5541      NaN    Movies  Louis C.K. muses on religion, eternal love,
gi...
5794      NaN    Movies  Emmy-winning comedy writer Louis C.K. brings
h...
5813      NaN    Movies  The comic puts his trademark
hilarious/thought...
```

```
ind = df[df['duration'].isna()].index

df.loc[ind] = df.loc[ind].fillna(method = 'ffill' , axis = 1)

# replaced the wrong entries done in the rating column
df.loc[ind ,'rating'] = 'Not Available'

df.loc[ind]
```

```
     show_id   type                                    title    director
\
5541   s5542  Movie                     Louis C.K. 2017  Louis C.K.

5794   s5795  Movie                Louis C.K.: Hilarious  Louis C.K.

5813   s5814  Movie  Louis C.K.: Live at the Comedy Store  Louis C.K.


            cast        country          date_added release_year  \
5541  Louis C.K.  United States       April 4, 2017         2017
5794  Louis C.K.  United States  September 16, 2016         2010
5813  Louis C.K.  United States     August 15, 2016         2015

             rating duration listed_in  \
5541  Not Available   74 min    Movies
5794  Not Available   84 min    Movies
5813  Not Available   66 min    Movies

                                          description
5541  Louis C.K. muses on religion, eternal love, gi...
5794  Emmy-winning comedy writer Louis C.K. brings h...
5813  The comic puts his trademark hilarious/thought...
```

Fill the null values in rating column

```
df[df.rating.isna()]
```

```
     show_id     type
title  \
5989   s5990    Movie  13TH: A Conversation with Oprah Winfrey &
```

```
Ava ...
6827   s6828  TV Show                  Gargantia on the Verdurous
Planet
7312   s7313  TV Show                                      Little
Lunch
7537   s7538    Movie                               My Honor Was
Loyalty

             director
cast  \
5989              NaN                         Oprah Winfrey, Ava
DuVernay
6827              NaN  Kaito Ishikawa, Hisako Kanemoto, Ai Kayano,
Ka...
7312              NaN  Flynn Curry, Olivia Deeble, Madison Lu,
Oisín ...
7537  Alessandro Pepe  Leone Frisa, Paolo Vaccarino, Francesco
Miglio...

        country        date_added release_year rating  duration  \
5989        NaN  January 26, 2017         2017    NaN    37 min
6827      Japan  December 1, 2016         2013    NaN  1 Season
7312  Australia  February 1, 2018         2015    NaN  1 Season
7537      Italy     March 1, 2017         2015    NaN   115 min

                                listed_in  \
5989                              Movies
6827  Anime Series, International TV Shows
7312                 Kids' TV, TV Comedies
7537                              Dramas

                                          description
5989  Oprah Winfrey sits down with director Ava DuVe...
6827  After falling through a wormhole, a space-dwel...
7312  Adopting a child's perspective, this show take...
7537  Amid the chaos and horror of World War II, a c...
```

```
indices = df[df.rating.isna()].index
indices
```

```
Index([5989, 6827, 7312, 7537], dtype='int64')
```

```
df.loc[indices , 'rating'] = 'Not Available'
```

```
df.loc[indices]
```

```
     show_id     type
title  \
5989   s5990    Movie  13TH: A Conversation with Oprah Winfrey &
Ava ...
6827   s6828  TV Show                  Gargantia on the Verdurous
```

```
Planet
7312   s7313  TV Show                                          Little
Lunch
7537   s7538    Movie                                  My Honor Was
Loyalty

             director
cast  \
5989               NaN                          Oprah Winfrey, Ava
DuVernay
6827               NaN  Kaito Ishikawa, Hisako Kanemoto, Ai Kayano,
Ka...
7312               NaN  Flynn Curry, Olivia Deeble, Madison Lu,
Oisín ...
7537  Alessandro Pepe  Leone Frisa, Paolo Vaccarino, Francesco
Miglio...

        country        date_added release_year         rating
duration  \
5989        NaN  January 26, 2017         2017  Not Available    37
min
6827      Japan  December 1, 2016         2013  Not Available  1
Season
7312  Australia  February 1, 2018         2015  Not Available  1
Season
7537      Italy     March 1, 2017         2015  Not Available   115
min

                                  listed_in  \
5989                                 Movies
6827  Anime Series, International TV Shows
7312                 Kids' TV, TV Comedies
7537                                 Dramas

                                             description
5989  Oprah Winfrey sits down with director Ava DuVe...
6827  After falling through a wormhole, a space-dwel...
7312  Adopting a child's perspective, this show take...
7537  Amid the chaos and horror of World War II, a c...
```

```
df.rating.unique()
```

```
array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R',
       'TV-G', 'G', 'NC-17', 'Not Available', 'NR', 'TV-Y7-FV', 'UR'],
      dtype=object)
```

```
df.loc[df['rating'] == 'UR' , 'rating'] = 'NR'
df.rating.value_counts()
```

```
rating
TV-MA            3207
```

```
TV-14            2160
TV-PG             863
R                 799
PG-13             490
TV-Y7             334
TV-Y              307
PG                287
TV-G              220
NR                 83
G                  41
Not Available       7
TV-Y7-FV            6
NC-17               3
Name: count, dtype: int64
```

```
df.drop(df.loc[df['date_added'].isna()].index , axis = 0 , inplace = True)
```

```
df['date_added'].value_counts()
```

```
date_added
January 1, 2020      109
November 1, 2019      89
March 1, 2018         75
December 31, 2019     74
October 1, 2018       71
                    ...
December 4, 2016       1
November 21, 2016      1
November 19, 2016      1
November 17, 2016      1
January 11, 2020       1
Name: count, Length: 1767, dtype: int64
```

```
df['date_added'] = df['date_added'].astype(str).str.strip()
df['date_added'] = df['date_added'].replace('nan', pd.NA)
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')
```

```
# total null values in each column
df.isna().sum()
```

```
show_id            0
type               0
title              0
director        2624
cast             825
country          830
date_added         0
release_year       0
rating             0
duration           0
```

```
listed_in          0
description        0
dtype: int64
```

```
round((df.isna().sum()/ df.shape[0])*100)
```

```
show_id          0.0
type             0.0
title            0.0
director        30.0
cast             9.0
country          9.0
date_added       0.0
release_year     0.0
rating           0.0
duration         0.0
listed_in        0.0
description      0.0
dtype: float64
```

```
df['date_added'].value_counts()
```

```
date_added
2020-01-01    110
2019-11-01     91
2018-03-01     75
2019-12-31     74
2018-10-01     71
              ...
2017-02-21      1
2017-02-07      1
2017-01-29      1
2017-01-25      1
2020-01-11      1
Name: count, Length: 1714, dtype: int64
```

Data Exploration and Non Graphical Analysis

```
# 2 types of content present in dataset - either Movie or TV Show
df['type'].unique()
```

```
array(['Movie', 'TV Show'], dtype=object)
```

```
movies  = df.loc[df['type'] == 'Movie']
tv_shows = df.loc[df['type'] == 'TV Show']

movies.duration.value_counts()
```

```
duration
90 min     152
94 min     146
```

```
97 min     146
93 min     146
91 min     144
          ...
208 min      1
5 min        1
16 min       1
186 min      1
191 min      1
Name: count, Length: 205, dtype: int64
```

```
tv_shows.duration.value_counts()
```

```
duration
1 Season      1793
2 Seasons      421
3 Seasons      198
4 Seasons       94
5 Seasons       64
6 Seasons       33
7 Seasons       23
8 Seasons       17
9 Seasons        9
10 Seasons       6
13 Seasons       2
15 Seasons       2
12 Seasons       2
17 Seasons       1
11 Seasons       1
Name: count, dtype: int64
```

```
movies['duration'] = movies['duration'].str[:-3]
movies['duration'] = movies['duration'].astype('float')

tv_shows['duration'] = tv_shows.duration.str[:-7].apply(lambda x :
x.strip())
tv_shows['duration'] = tv_shows['duration'].astype('float')

tv_shows.rename({'duration': 'duration_in_seasons'} ,axis = 1 ,
inplace = True)
movies.rename({'duration': 'duration_in_minutes'} ,axis = 1 , inplace
= True)

tv_shows.duration_in_seasons
```

```
1       2.0
2       1.0
3       1.0
4       2.0
5       1.0
       ...
```

```
8795    2.0
8796    2.0
8797    3.0
8800    1.0
8803    2.0
Name: duration_in_seasons, Length: 2666, dtype: float64
```

```
movies.duration_in_minutes
```

```
0        90.0
6        91.0
7       125.0
9       104.0
12      127.0
        ...
8801     96.0
8802    158.0
8804     88.0
8805     88.0
8806    111.0
Name: duration_in_minutes, Length: 6131, dtype: float64
```

```
timeperiod = pd.Series((df['date_added'].min().strftime('%B %Y') ,
df['date_added'].max().strftime('%B %Y')))
timeperiod.index = ['first' , 'Most Recent']
timeperiod
```

```
first            January 2008
Most Recent    September 2021
dtype: object
```

```
df.release_year.min() , df.release_year.max()
```

```
(1925, 2021)
```

```
df.loc[(df.release_year == df.release_year.min()) | (df.release_year
== df.release_year.max())].sort_values('release_year')
```

```
     show_id     type                                          title  \
4250   s4251  TV Show           Pioneers: First Women Filmmakers*
966     s967    Movie                                 Get the Grift
967     s968  TV Show                   Headspace Guide to Sleep
968     s969  TV Show                                      Sexify
972     s973  TV Show                                       Fatma
...      ...      ...                                         ...
466     s467  TV Show                          My Unorthodox Life
467     s468    Movie  Private Network: Who Killed Manuel Buendía?
468     s469    Movie             The Guide to the Perfect Family
471     s472    Movie                                Day of Destiny
8437   s8438  TV Show                       The Netflix Afterparty
```

```
                     director  \
4250                      NaN
966             Pedro Antonio
967                       NaN
968                       NaN
972                       NaN
...                       ...
466                       NaN
467             Manuel Alcalá
468             Ricardo Trogi
471   Akay Mason, Abosi Ogba
8437                      NaN

                                                    cast        country
\
4250                                                 NaN            NaN

966   Marcus Majella, Samantha Schmütz, Caito Mainie...         Brazil

967                                  Evelyn Lewis Prieto            NaN

968   Aleksandra Skraba, Maria Sobocińska, Sandra Dr...         Poland

972   Burcu Biricik, Uğur Yücel, Mehmet Yılmaz Ak, H...         Turkey

...                                                  ...            ...

466                                                  NaN            NaN

467                                Daniel Giménez Cacho             NaN

468   Louis Morissette, Émilie Bierre, Catherine Cha...             NaN

471   Olumide Oworu, Denola Grey, Gbemi Akinlade, Ji...             NaN

8437       David Spade, London Hughes, Fortune Feimster  United States


     date_added release_year rating  duration  \
4250 2018-12-30         1925  TV-14  1 Season
966  2021-04-28         2021  TV-MA    95 min
967  2021-04-28         2021   TV-G  1 Season
968  2021-04-28         2021  TV-MA  1 Season
972  2021-04-27         2021  TV-MA  1 Season
...         ...          ...    ...       ...
466  2021-07-14         2021  TV-MA  1 Season
467  2021-07-14         2021  TV-MA   100 min
468  2021-07-14         2021  TV-MA   102 min
471  2021-07-13         2021  TV-PG   110 min
8437 2021-01-02         2021  TV-MA  1 Season
```

```
                                                  listed_in  \
4250                                               TV Shows
966                          Comedies, International Movies
967                        Docuseries, Science & Nature TV
968      International TV Shows, TV Comedies, TV Dramas
972     International TV Shows, TV Dramas, TV Thrillers
...                                                     ...
466                                           Reality TV
467                  Documentaries, International Movies
468              Comedies, Dramas, International Movies
471   Children & Family Movies, Dramas, Internationa...
8437          Stand-Up Comedy & Talk Shows, TV Comedies

                                                 description
4250  This collection restores films from women who ...
966   After a botched scam, Clóvis bumps into Lohane...
967   Learn how to sleep better with Headspace. Each...
968   To build an innovative sex app and win a tech ...
972   Reeling from tragedy, a nondescript house clea...
...                                                   ...
466   Follow Julia Haart, Elite World Group CEO and ...
467   A deep dive into the work of renowned Mexican ...
468   A couple in Québec deals with the pitfalls, pr...
471   With their family facing financial woes, two t...
8437  Hosts David Spade, Fortune Feimster and London...

[593 rows x 12 columns]
```

```
df.groupby(['type' , 'rating'])['show_id'].count()
```

```
type     rating
Movie    G                 41
         NC-17              3
         NR                78
         Not Available      5
         PG               287
         PG-13            490
         R                797
         TV-14           1427
         TV-G             126
         TV-MA           2062
         TV-PG            540
         TV-Y             131
         TV-Y7            139
         TV-Y7-FV           5
TV Show  NR                 4
         Not Available      2
         R                  2
         TV-14            730
         TV-G              94
```

```
         TV-MA            1143
         TV-PG             321
         TV-Y              175
         TV-Y7             194
         TV-Y7-FV            1
Name: show_id, dtype: int64
```

```
df['country'].value_counts()
```

```
country
United States                                2812
India                                         972
United Kingdom                                418
Japan                                         244
South Korea                                   199
                                             ...
Romania, Bulgaria, Hungary                      1
Uruguay, Guatemala                              1
France, Senegal, Belgium                        1
Mexico, United States, Spain, Colombia          1
United Arab Emirates, Jordan                    1
Name: count, Length: 748, dtype: int64
```

```
country_tb = df[['show_id' , 'type' , 'country']]
country_tb.dropna(inplace = True)
country_tb['country'] = country_tb['country'].apply(lambda x :
x.split(','))
country_tb = country_tb.explode('country')
country_tb
```

```
     show_id     type        country
0         s1    Movie  United States
1         s2  TV Show   South Africa
4         s5  TV Show          India
7         s8    Movie  United States
7         s8    Movie          Ghana
...      ...      ...            ...
8801   s8802    Movie         Jordan
8802   s8803    Movie  United States
8804   s8805    Movie  United States
8805   s8806    Movie  United States
8806   s8807    Movie          India

[10010 rows x 3 columns]
```

```
# some duplicate values are found, which have unnecessary spaces. some
empty strings found
country_tb['country'] = country_tb['country'].str.strip()
```

```
country_tb.loc[country_tb['country'] == '']
```

```
     show_id     type country
193     s194  TV Show
365     s366    Movie
1192   s1193    Movie
2224   s2225    Movie
4653   s4654    Movie
5925   s5926    Movie
7007   s7008    Movie
```

```
country_tb = country_tb.loc[country_tb['country'] != '']

country_tb['country'].nunique()
```

```
122
```

```
x = country_tb.groupby(['country' , 'type'])
['show_id'].count().reset_index()
x.pivot(index = ['country'] , columns = 'type' , values =
'show_id').sort_values('Movie',ascending = False)
```

```
type              Movie  TV Show
country
United States   2752.0    932.0
India            962.0     84.0
United Kingdom   534.0    271.0
Canada           319.0    126.0
France           303.0     90.0
...                ...      ...
Azerbaijan         NaN      1.0
Belarus            NaN      1.0
Cuba               NaN      1.0
Cyprus             NaN      1.0
Puerto Rico        NaN      1.0

[122 rows x 2 columns]
```

```
df['director'].value_counts()
```

```
director
Rajiv Chilaka                     19
Raúl Campos, Jan Suter            18
Marcus Raboy                      16
Suhas Kadav                       16
Jay Karas                         14
                                  ..
Raymie Muzquiz, Stu Livingston     1
Joe Menendez                       1
Eric Bross                         1
Will Eisenberg                     1
Mozez Singh                        1
Name: count, Length: 4528, dtype: int64
```

```
dir_tb = df[['show_id' , 'type' , 'director']]
dir_tb.dropna(inplace = True)
dir_tb['director'] = dir_tb['director'].apply(lambda x : x.split(','))
dir_tb
```

```
     show_id     type                          director
0         s1    Movie                 [Kirsten Johnson]
2         s3  TV Show                 [Julien Leclercq]
5         s6  TV Show                   [Mike Flanagan]
6         s7    Movie  [Robert Cullen,  José Luis Ucha]
7         s8    Movie                    [Haile Gerima]
...      ...      ...                               ...
8801   s8802    Movie                 [Majid Al Ansari]
8802   s8803    Movie                   [David Fincher]
8804   s8805    Movie                 [Ruben Fleischer]
8805   s8806    Movie                    [Peter Hewitt]
8806   s8807    Movie                     [Mozez Singh]

[6173 rows x 3 columns]
```

```
dir_tb = dir_tb.explode('director')
```

```
dir_tb['director'] = dir_tb['director'].str.strip()
```

```
# checking if empty stirngs are there in director column
dir_tb.director.apply(lambda x : True if len(x) == 0 else
False).value_counts()
```

```
director
False    6978
Name: count, dtype: int64
```

```
dir_tb
```

```
     show_id     type         director
0         s1    Movie  Kirsten Johnson
2         s3  TV Show  Julien Leclercq
5         s6  TV Show    Mike Flanagan
6         s7    Movie    Robert Cullen
6         s7    Movie   José Luis Ucha
...      ...      ...              ...
8801   s8802    Movie  Majid Al Ansari
8802   s8803    Movie    David Fincher
8804   s8805    Movie  Ruben Fleischer
8805   s8806    Movie     Peter Hewitt
8806   s8807    Movie      Mozez Singh

[6978 rows x 3 columns]
```

```
dir_tb['director'].nunique()
```

```
4993
```

```
x = dir_tb.groupby(['director' , 'type'])
['show_id'].count().reset_index()
x.pivot(index= ['director'] , columns = 'type' , values =
'show_id').sort_values('Movie' ,ascending = False)
```

```
type                  Movie  TV Show
director
Rajiv Chilaka          22.0      NaN
Jan Suter              21.0      NaN
Raúl Campos            19.0      NaN
Suhas Kadav            16.0      NaN
Marcus Raboy           15.0      1.0
...                     ...      ...
Vijay S. Bhanushali    NaN       1.0
Wouter Bouvijn         NaN       1.0
YC Tom Lee             NaN       1.0
Yasuhiro Irie          NaN       1.0
Yim Pilsung            NaN       1.0

[4993 rows x 2 columns]
```

```
genre_tb = df[['show_id' , 'type', 'listed_in']]

genre_tb['listed_in'] = genre_tb['listed_in'].apply(lambda x :
x.split(','))
genre_tb = genre_tb.explode('listed_in')
genre_tb['listed_in'] = genre_tb['listed_in'].str.strip()

genre_tb
```

```
     show_id     type                 listed_in
0         s1    Movie             Documentaries
1         s2  TV Show    International TV Shows
1         s2  TV Show                TV Dramas
1         s2  TV Show              TV Mysteries
2         s3  TV Show            Crime TV Shows
...      ...      ...                      ...
8805   s8806    Movie  Children & Family Movies
8805   s8806    Movie                 Comedies
8806   s8807    Movie                   Dramas
8806   s8807    Movie      International Movies
8806   s8807    Movie          Music & Musicals

[19303 rows x 3 columns]
```

```
genre_tb.listed_in.unique()

array(['Documentaries', 'International TV Shows', 'TV Dramas',
       'TV Mysteries', 'Crime TV Shows', 'TV Action & Adventure',
```

```
       'Docuseries', 'Reality TV', 'Romantic TV Shows', 'TV Comedies',
       'TV Horror', 'Children & Family Movies', 'Dramas',
       'Independent Movies', 'International Movies', 'British TV
Shows',
       'Comedies', 'Spanish-Language TV Shows', 'Thrillers',
       'Romantic Movies', 'Music & Musicals', 'Horror Movies',
       'Sci-Fi & Fantasy', 'TV Thrillers', "Kids' TV",
       'Action & Adventure', 'TV Sci-Fi & Fantasy', 'Classic Movies',
       'Anime Features', 'Sports Movies', 'Anime Series',
       'Korean TV Shows', 'Science & Nature TV', 'Teen TV Shows',
       'Cult Movies', 'TV Shows', 'Faith & Spirituality', 'LGBTQ
Movies',
       'Stand-Up Comedy', 'Movies', 'Stand-Up Comedy & Talk Shows',
       'Classic & Cult TV'], dtype=object)
```

```
genre_tb.listed_in.nunique()
```

```
42
```

```
df.merge(genre_tb , on = 'show_id' ).groupby(['type_y'])
['listed_in_y'].nunique()
```

```
type_y
Movie      20
TV Show    22
Name: listed_in_y, dtype: int64
```

```
# total movies/TV shows in each genre
x = genre_tb.groupby(['listed_in' , 'type'])
['show_id'].count().reset_index()
x.pivot(index = 'listed_in' , columns = 'type' , values =
'show_id').sort_index()
```

type	Movie	TV Show
listed_in		
Action & Adventure	859.0	NaN
Anime Features	71.0	NaN
Anime Series	NaN	175.0
British TV Shows	NaN	252.0
Children & Family Movies	641.0	NaN
Classic & Cult TV	NaN	26.0
Classic Movies	116.0	NaN
Comedies	1674.0	NaN
Crime TV Shows	NaN	469.0
Cult Movies	71.0	NaN
Documentaries	869.0	NaN
Docuseries	NaN	394.0
Dramas	2427.0	NaN
Faith & Spirituality	65.0	NaN
Horror Movies	357.0	NaN
Independent Movies	756.0	NaN

```
International Movies          2752.0      NaN
International TV Shows           NaN   1350.0
Kids' TV                         NaN    449.0
Korean TV Shows                  NaN    151.0
LGBTQ Movies                   102.0      NaN
Movies                          57.0      NaN
Music & Musicals               375.0      NaN
Reality TV                       NaN    255.0
Romantic Movies                616.0      NaN
Romantic TV Shows                NaN    370.0
Sci-Fi & Fantasy               243.0      NaN
Science & Nature TV              NaN     92.0
Spanish-Language TV Shows        NaN    173.0
Sports Movies                  219.0      NaN
Stand-Up Comedy                343.0      NaN
Stand-Up Comedy & Talk Shows     NaN     56.0
TV Action & Adventure            NaN    167.0
TV Comedies                      NaN    574.0
TV Dramas                        NaN    762.0
TV Horror                        NaN     75.0
TV Mysteries                     NaN     98.0
TV Sci-Fi & Fantasy              NaN     83.0
TV Shows                         NaN     16.0
TV Thrillers                     NaN     57.0
Teen TV Shows                    NaN     69.0
Thrillers                      577.0      NaN
```

```
cast_tb = df[['show_id' , 'type' ,'cast']]
cast_tb.dropna(inplace = True)
cast_tb['cast'] = cast_tb['cast'].apply(lambda x : x.split(','))
cast_tb = cast_tb.explode('cast')
cast_tb
```

```
     show_id     type                    cast
1         s2  TV Show              Ama Qamata
1         s2  TV Show            Khosi Ngema
1         s2  TV Show          Gail Mabalane
1         s2  TV Show         Thabang Molaba
1         s2  TV Show       Dillon Windvogel
...      ...      ...                     ...
8806   s8807    Movie       Manish Chaudhary
8806   s8807    Movie           Meghna Malik
8806   s8807    Movie          Malkeet Rauni
8806   s8807    Movie         Anita Shabdish
8806   s8807    Movie  Chittaranjan Tripathy

[64057 rows x 3 columns]
```

```
cast_tb['cast'] = cast_tb['cast'].str.strip()
```

```
# checking empty strings
cast_tb[cast_tb['cast'] == '']
```

```
Empty DataFrame
Columns: [show_id, type, cast]
Index: []
```

```
# Total actors on the Netflix
cast_tb.cast.nunique()
```

```
36403
```

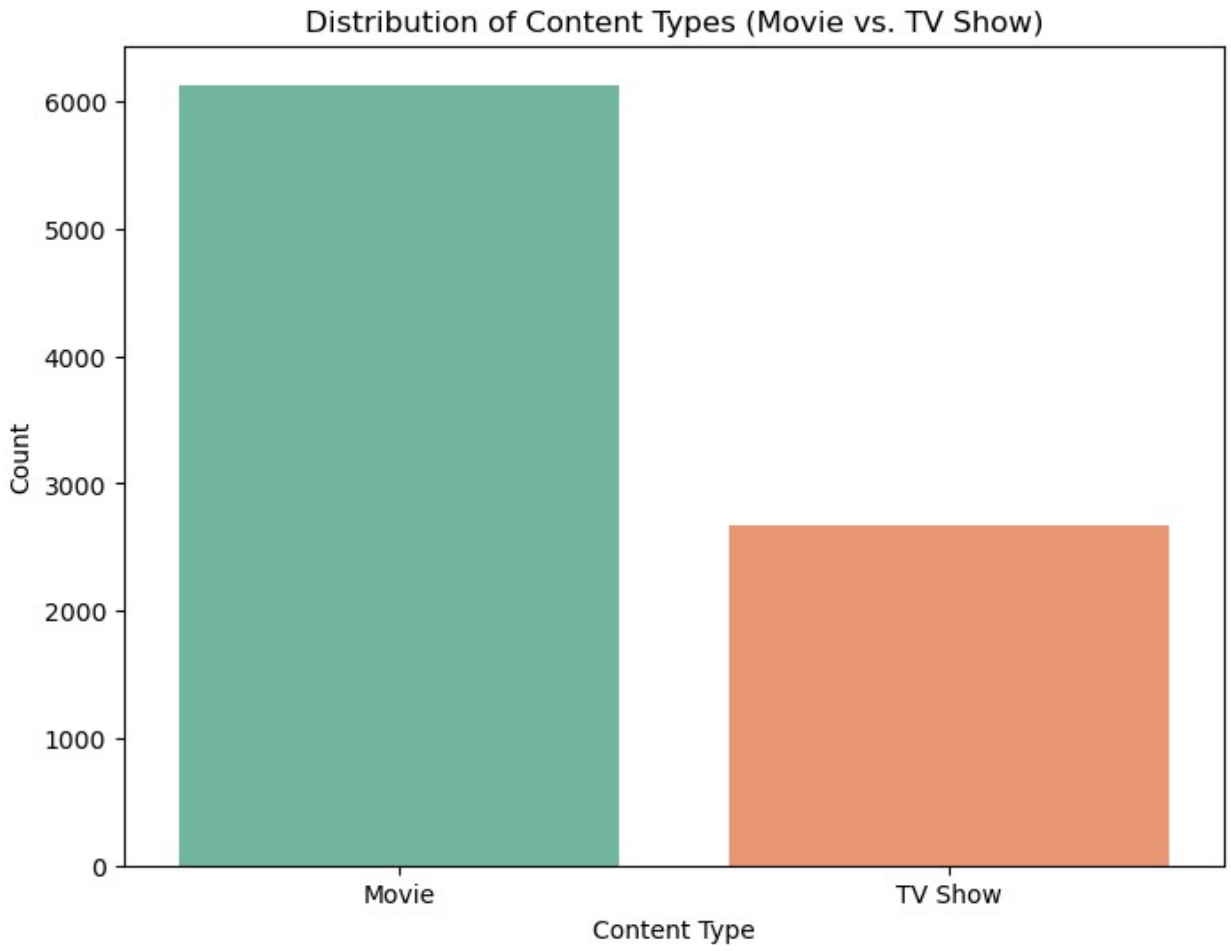
```
# Total movies/TV shows by each actor
x = cast_tb.groupby(['cast' , 'type'])
['show_id'].count().reset_index()
x.pivot(index = 'cast' , columns = 'type' , values =
'show_id').sort_values('TV Show' , ascending = False)
```

```
type               Movie  TV Show
cast
Takahiro Sakurai     7.0     25.0
Yuki Kaji           10.0     19.0
Junichi Suwabe       4.0     17.0
Daisuke Ono          5.0     17.0
Ai Kayano            2.0     17.0
...                  ...      ...
Şerif Sezer          1.0      NaN
Şevket Çoruh         1.0      NaN
Şinasi Yurtsever     3.0      NaN
Şükran Ovalı         1.0      NaN
Ṣọpẹ́ Dìrísù        1.0      NaN

[36403 rows x 2 columns]
```

Data Visualization

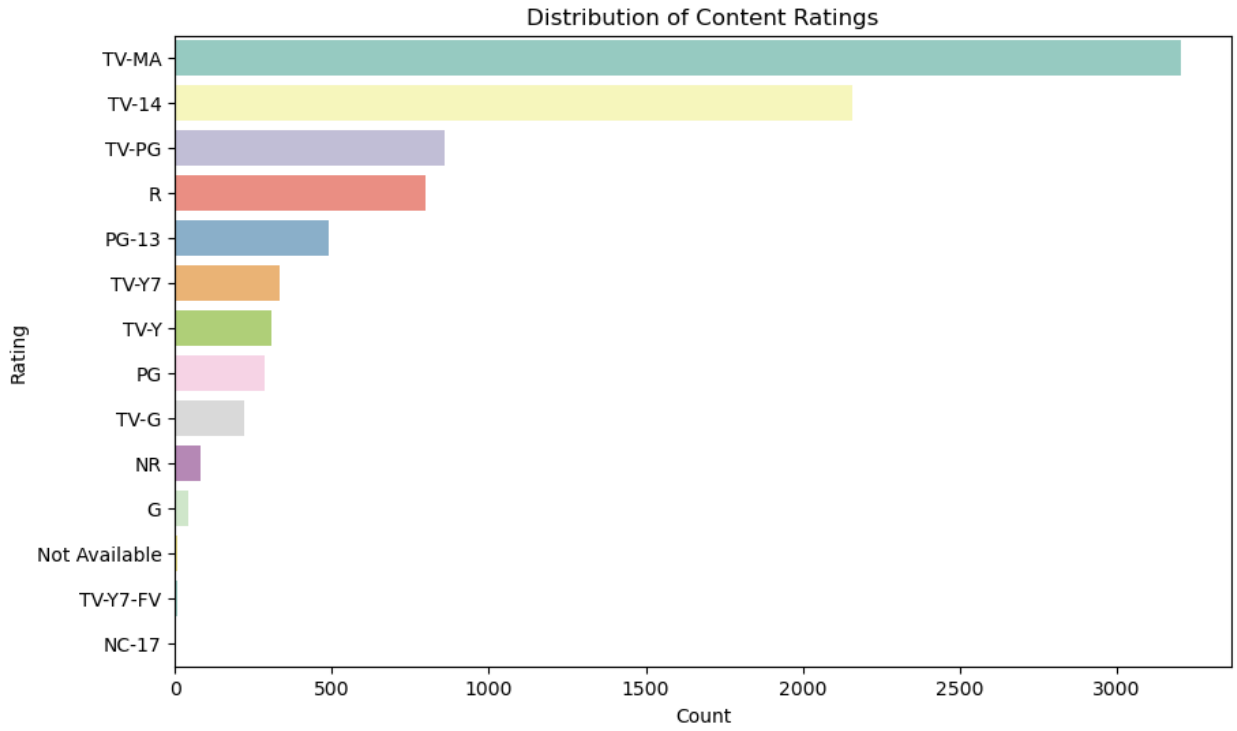
1. Distribution By Types :-

```
# 1. Distribution of content types (Movie vs. TV Show)
plt.figure(figsize=(8, 6))
sns.countplot(x='type', hue='type', data=df, palette='Set2',
legend=False)
plt.title('Distribution of Content Types (Movie vs. TV Show)')
plt.xlabel('Content Type')
plt.ylabel('Count')
plt.show()
```



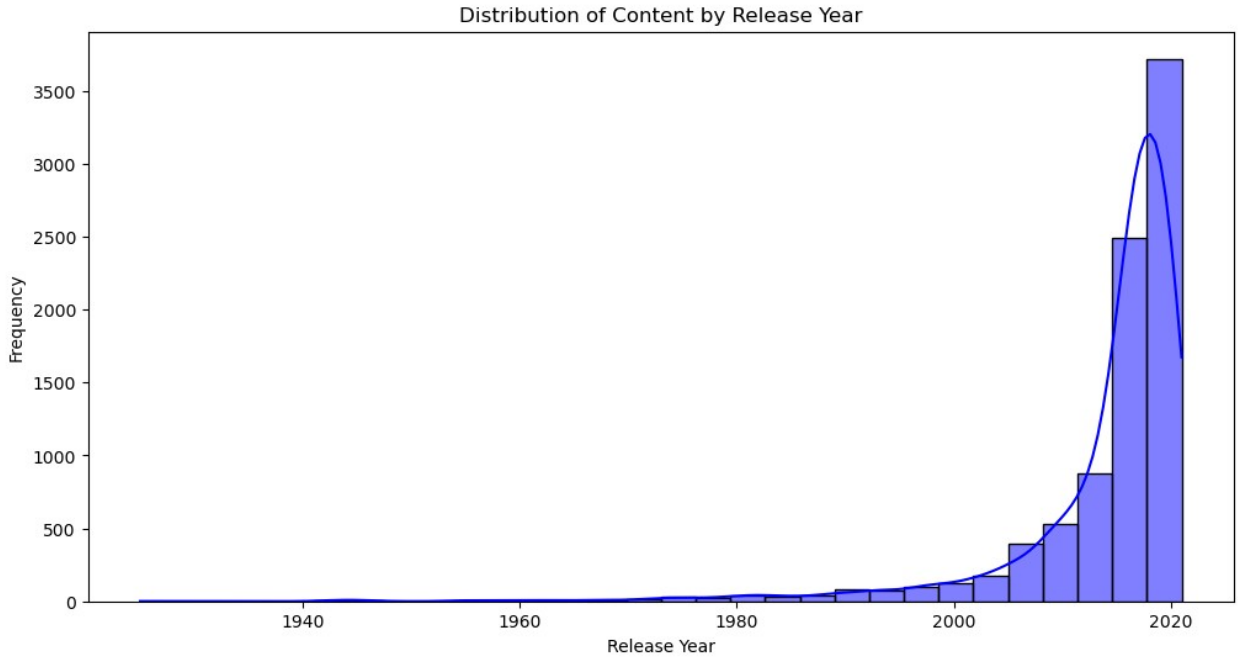
2. Distribution of content ratings

```
plt.figure(figsize=(10, 6))
sns.countplot(y='rating', data=df, palette='Set3',
order=df['rating'].value_counts().index)
plt.title('Distribution of Content Ratings')
plt.xlabel('Count')
plt.ylabel('Rating')
plt.show()
```



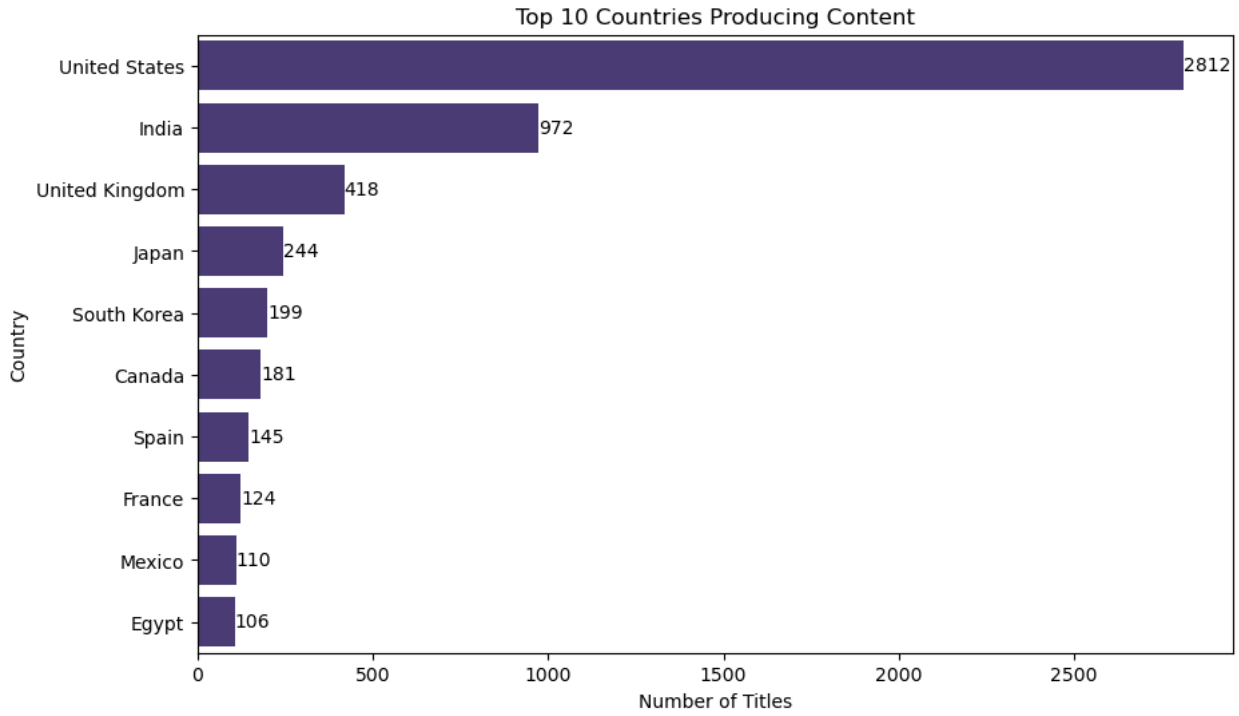
3. Distribution of content by release year

```
plt.figure(figsize=(12, 6))
sns.histplot(df['release_year'], bins=30, kde=True, color='blue')
plt.title('Distribution of Content by Release Year')
plt.xlabel('Release Year')
plt.ylabel('Frequency')
plt.show()
```



4. Top 10 countries producing content

```
top_countries = df['country'].value_counts().head(10)  # Get the top
10 countries producing content


plt.figure(figsize=(10, 6))
ax = sns.barplot(x=top_countries.values, y=top_countries.index,
color=sns.color_palette('viridis')[0])
plt.title('Top 10 Countries Producing Content')
plt.xlabel('Number of Titles')
plt.ylabel('Country')

# Add count labels on top of each bar
for i, v in enumerate(top_countries.values):
    ax.text(v + 0.5, i, str(v), color='black', va='center', ha='left')

plt.show()
```



5.Number of movies released per year changed

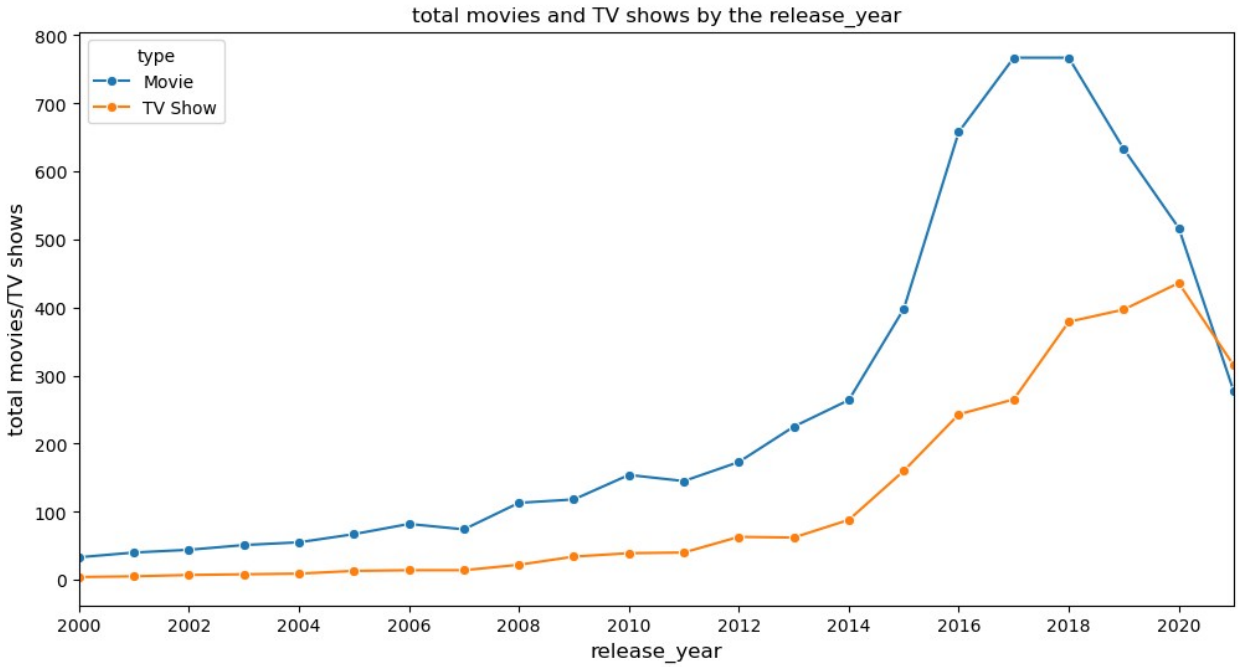
```
d = df.groupby(['type' , 'release_year'])
['show_id'].count().reset_index()
d.rename({'show_id' : 'total movies/TV shows'}, axis = 1 , inplace =
True)
d
```

	type	release_year	total movies/TV shows
0	Movie	1942	2
1	Movie	1943	3
2	Movie	1944	3
3	Movie	1945	3
4	Movie	1946	1
..	...	...	...
114	TV Show	2017	265
115	TV Show	2018	379
116	TV Show	2019	397
117	TV Show	2020	436
118	TV Show	2021	315

[119 rows x 3 columns]

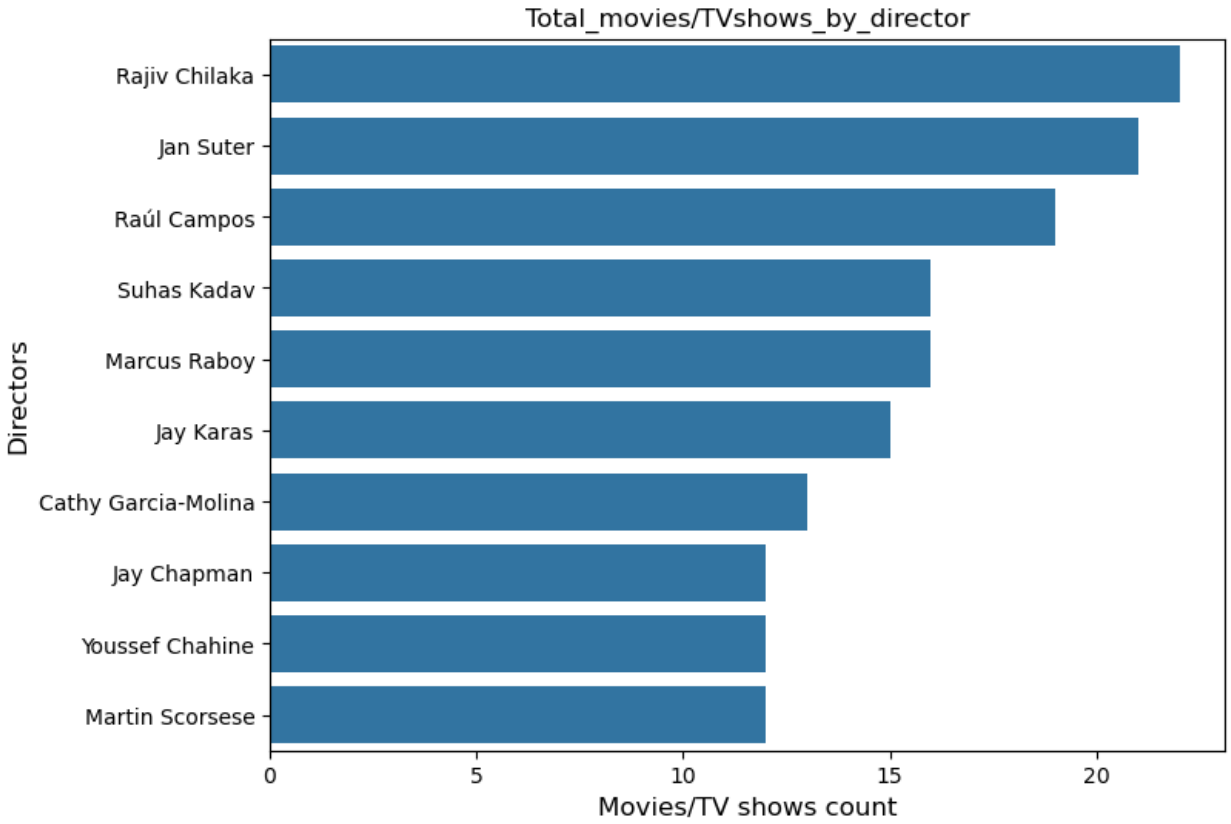
```
plt.figure(figsize = (12,6))
sns.lineplot(data = d , x = 'release_year' , y = 'total movies/TV
shows' , hue = 'type' , marker = 'o'  , ms = 6 )
plt.xlabel('release_year' , fontsize = 12)
plt.ylabel('total movies/TV shows' , fontsize = 12)
```

```
plt.title('total movies and TV shows by the release_year' , fontsize =
12)
plt.xlim( left = 2000 , right = 2021)
plt.xticks(np.arange(2000 , 2021 , 2))
plt.show()
```



6.Total shows

```
# total Movies directed by top 10 directors
top_10_dir = dir_tb.director.value_counts().head(10).index
df_new = dir_tb.loc[dir_tb['director'].isin(top_10_dir)]

plt.figure(figsize= (8 , 6))
sns.countplot(data = df_new , y = 'director' , order = top_10_dir ,
orient = 'v')
plt.xlabel('total_movies/TV shows' , fontsize = 12)
plt.xlabel('Movies/TV shows count')
plt.ylabel('Directors' , fontsize = 12)
plt.title('Total_movies/TVshows_by_director')
plt.show()
```



7. Outliers Checking

```
x = dir_tb.director.value_counts()
x
```

```
director
Rajiv Chilaka     22
Jan Suter         21
Raúl Campos       19
Suhas Kadav       16
Marcus Raboy      16
                  ..
Raymie Muzquiz     1
Stu Livingston     1
Joe Menendez       1
Eric Bross         1
Mozez Singh        1
Name: count, Length: 4993, dtype: int64
```

```
def calculate_outliers(data):
    # Calculate the first quartile (Q1)
    q1 = np.percentile(data, 25)

    # Calculate the third quartile (Q3)
    q3 = np.percentile(data, 75)
```

```
    # Calculate the interquartile range (IQR)
    iqr = q3 - q1

    # Determine the lower and upper bounds for outliers
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    # Identify outliers in the dataset
    outliers = [value for value in data if value < lower_bound or
value > upper_bound]

    return outliers


def calculate_max_occurred_value(data):
    # Calculate the unique values and their counts in the dataset
    unique_values, value_counts = np.unique(data, return_counts=True)

    # Find the index of the maximum count
    max_count_index = np.argmax(value_counts)

    # Retrieve the corresponding unique value with the maximum count
    max_occurred_value = unique_values[max_count_index]

    return max_occurred_value

outliers = calculate_outliers(x)  # Implement your outlier calculation
method
max_occurred_value = calculate_max_occurred_value(x)  # Implement your
method to find the maximum-occurred value
set(outliers)
```

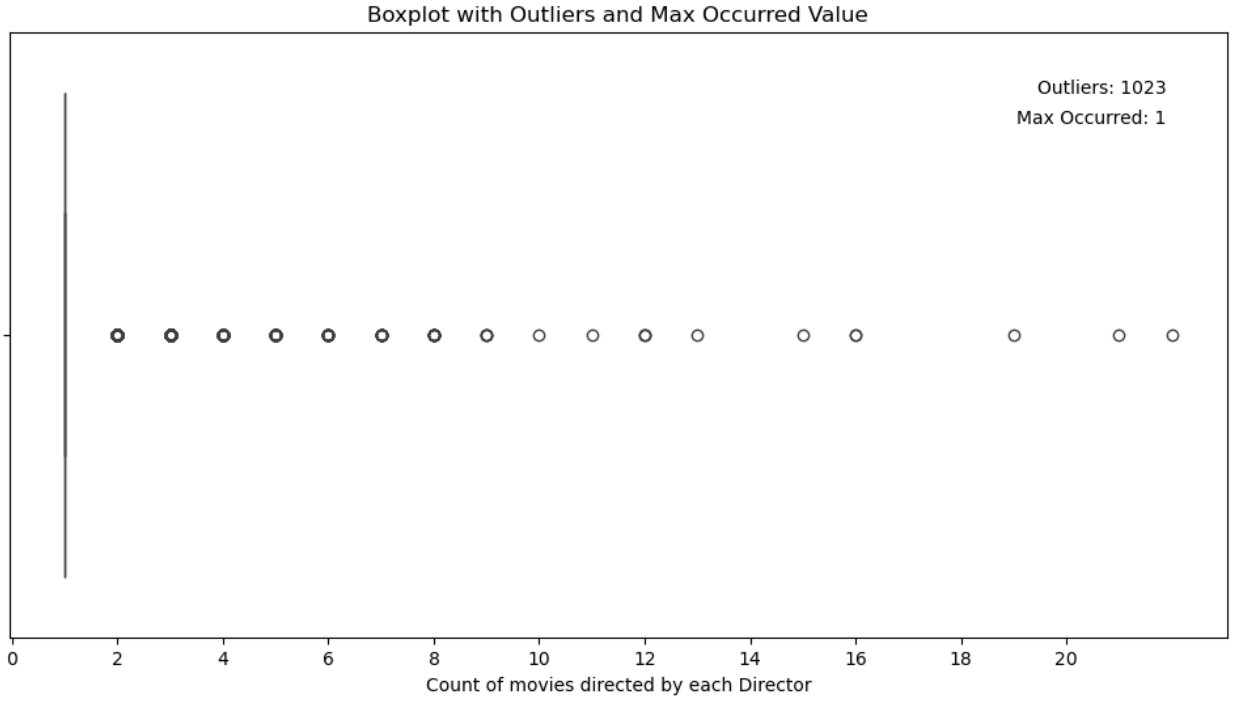
```
{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 19, 21, 22}
```

```
max_occurred_value
```

```
1
```

```
plt.figure(figsize = (12,6))
sns.boxplot(data=x, showfliers=True, whis=1.5 , orient = 'h')

# Calculate the outliers and maximum-occurred value
outliers = calculate_outliers(x)  # Implement your outlier calculation
method
max_occurred_value = calculate_max_occurred_value(x)  # Implement your
method to find the maximum-occurred value

# Annotate the plot
plt.text(0.95, 0.9, f"Outliers: {len(outliers)}",
transform=plt.gca().transAxes, ha='right')
```

```
plt.text(0.95, 0.85, f"Max Occurred: {max_occurred_value}",
transform=plt.gca().transAxes, ha='right')


plt.xlabel("Count of movies directed by each Director")
plt.xticks(np.arange(0,22,2))
plt.title("Boxplot with Outliers and Max Occurred Value")

# Show the plot
plt.show()
```



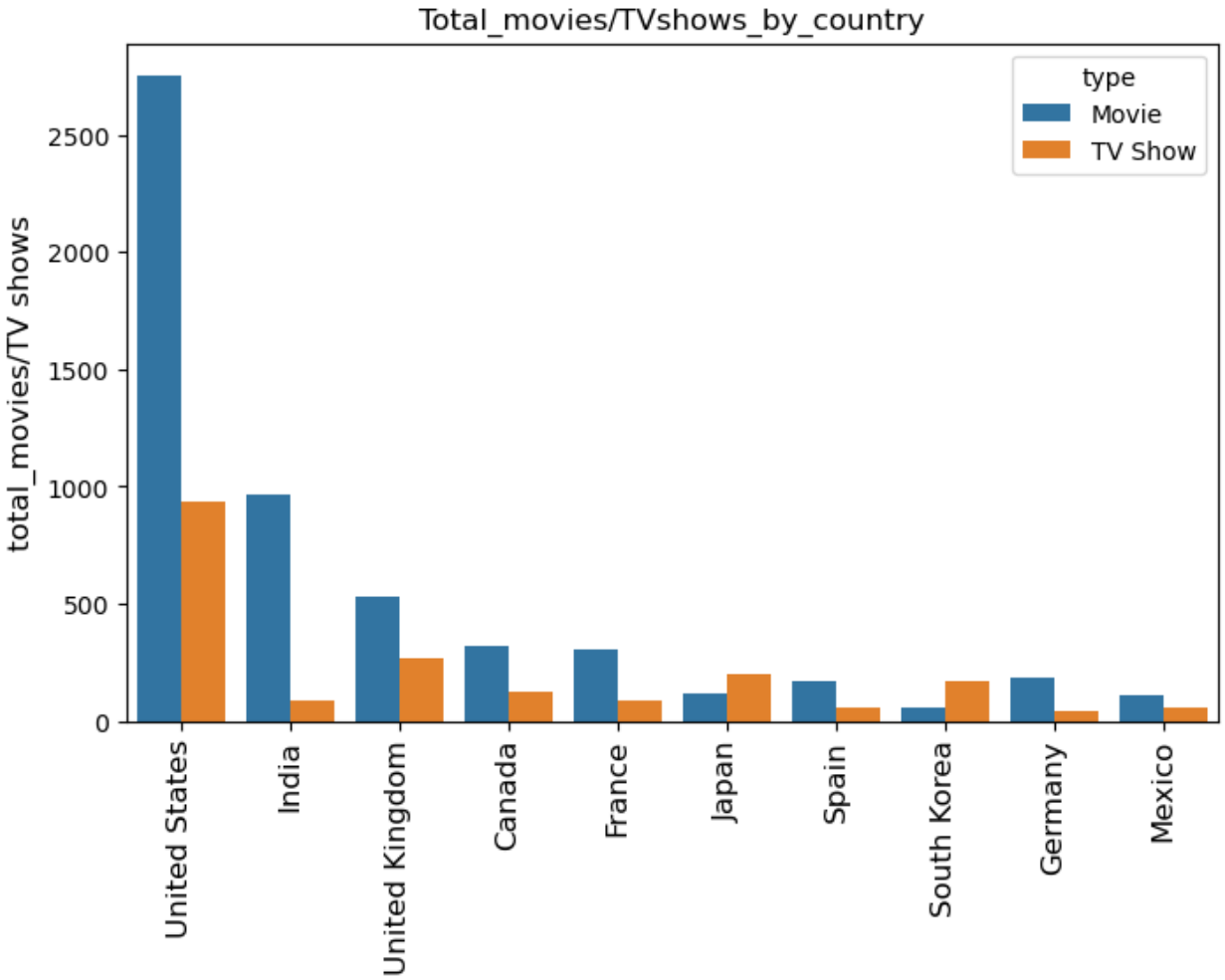
8. Total shows by country

```
# Lets check for top 10 countries
top_10_country = country_tb.country.value_counts().head(10).index
df_new = country_tb.loc[country_tb['country'].isin(top_10_country)]

x = df_new.groupby(['country' , 'type'])
['show_id'].count().reset_index()
x.pivot(index = 'country' , columns = 'type' , values =
'show_id').sort_values('Movie',ascending = False)
```

type	Movie	TV Show
country		
United States	2752	932
India	962	84
United Kingdom	534	271
Canada	319	126

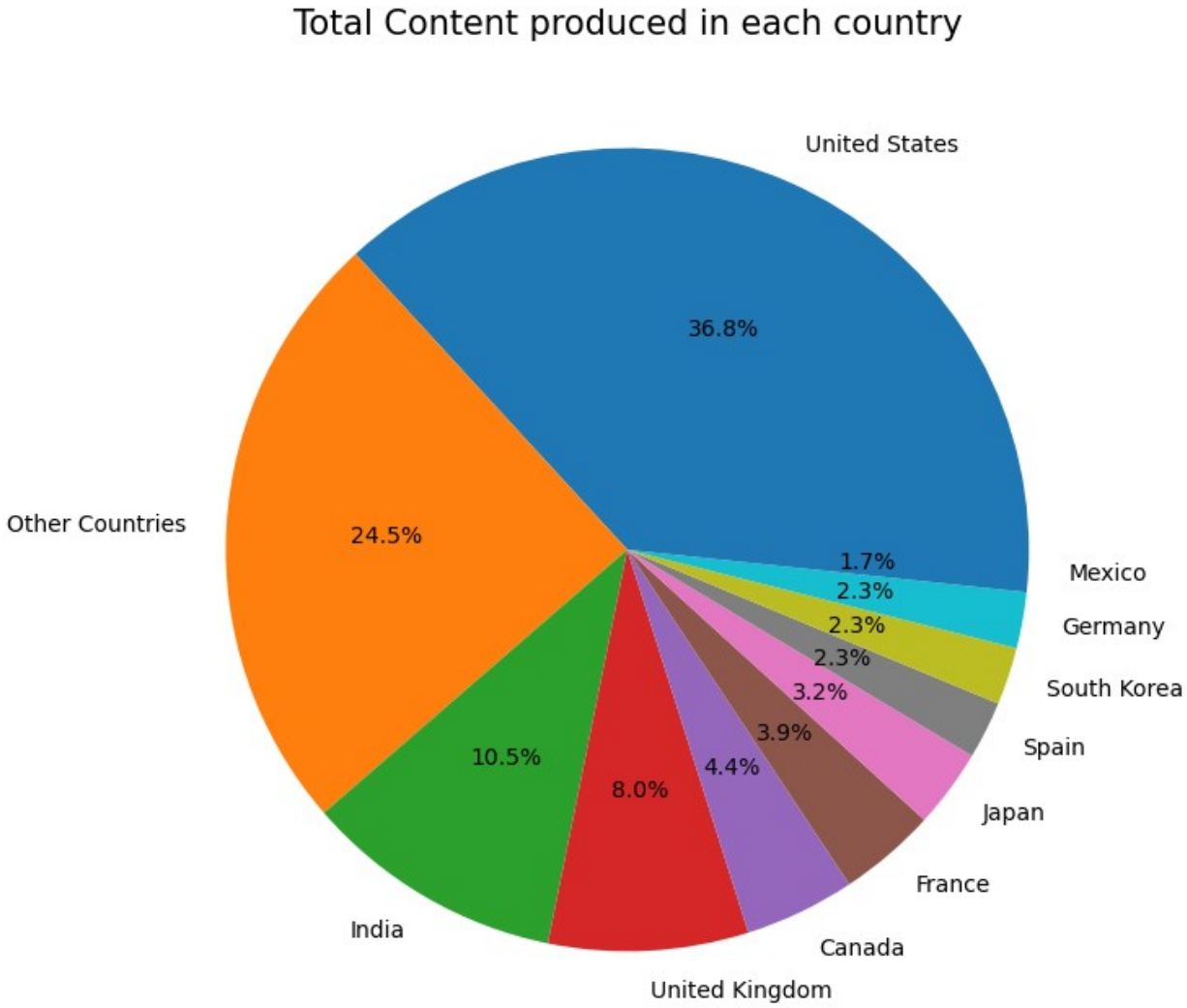
```
France              303       90
Germany             182       44
Spain               171       61
Japan               119      198
Mexico              111       58
South Korea          61      170
```

```
plt.figure(figsize= (8,5))
sns.countplot(data = df_new , x = 'country' , order = top_10_country ,
hue = 'type')
plt.xticks(rotation = 90 , fontsize = 12)
plt.ylabel('total_movies/TV shows' , fontsize = 12)
plt.xlabel('')
plt.title('Total_movies/TVshows_by_country')
plt.show()
```



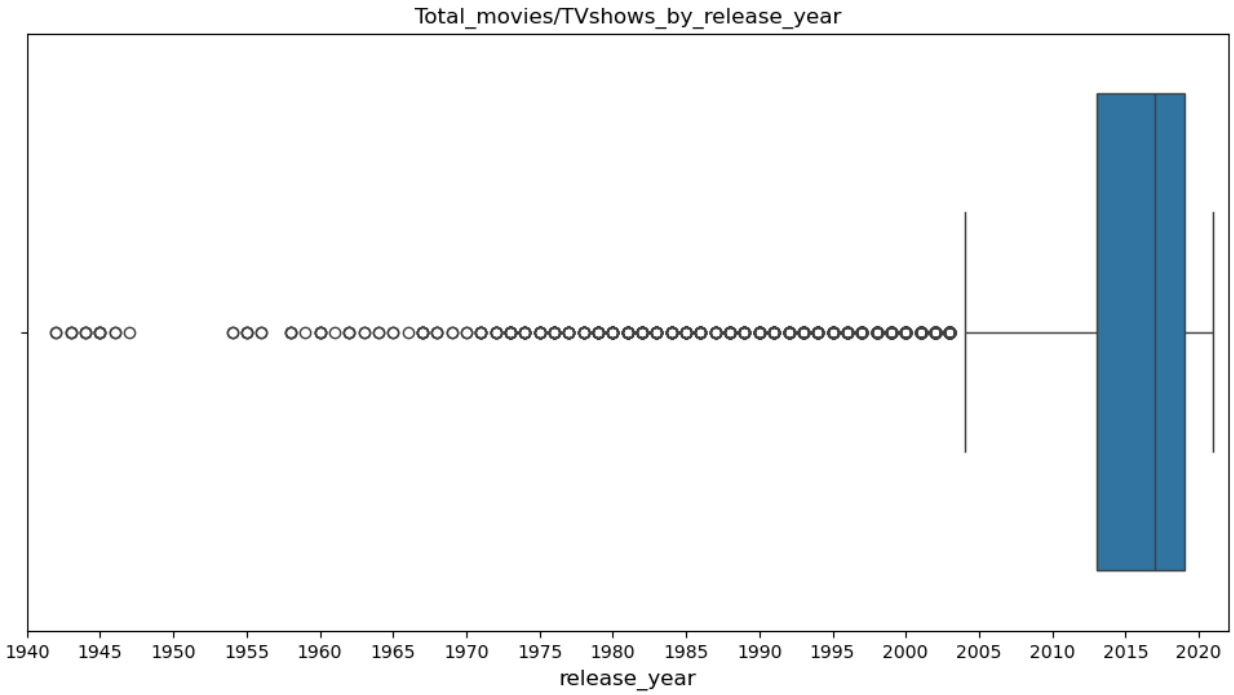

```
top_10_country = country_tb.country.value_counts().head(10).index
country_tb['cat'] = country_tb['country'].apply(lambda x : x if x in
top_10_country else 'Other Countries' )
```

```
x = country_tb.cat.value_counts()

plt.figure(figsize = (8,8))
plt.pie(x , labels = x.index, autopct='%1.1f%%')
plt.title('Total Content produced in each country' , fontsize = 15)
plt.show()
```



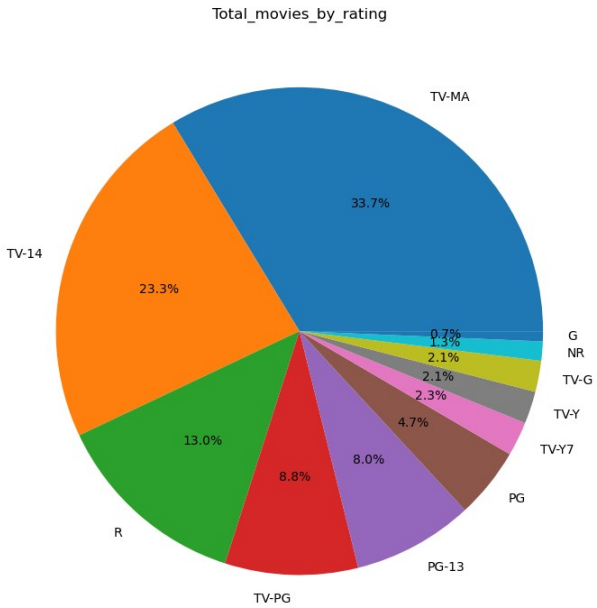
9. Distribution :- total content by release year

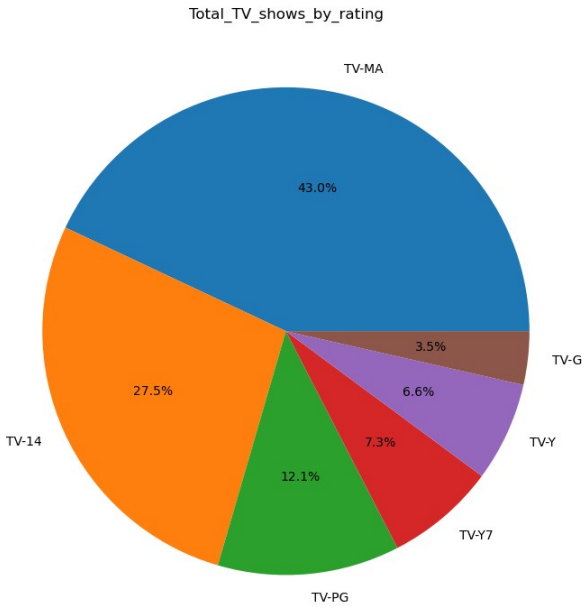
```
plt.figure(figsize= (12,6))
sns.boxplot(data = df , x = 'release_year')
plt.xlabel('release_year' , fontsize = 12)
plt.title('Total_movies/TVshows_by_release_year')
plt.xticks(np.arange(1940 , 2021 , 5))
plt.xlim((1940 , 2022))
plt.show()
```



10.Total shows distribution by rating of the content

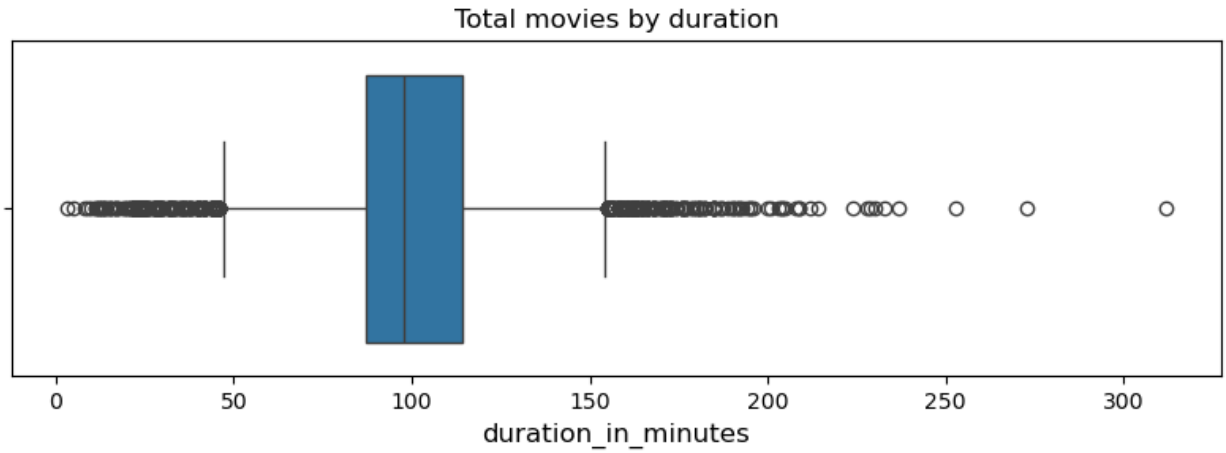
```
m = movies.loc[~movies.rating.isin(['Not Available' , 'NC-17' , 'TV-
Y7-FV'])]
m = m.rating.value_counts()
t = tv_shows.loc[~tv_shows.rating.isin(['Not Available' , 'R' , 'NR',
'TV-Y7-FV'])]
t = t.rating.value_counts()


fig, ax = plt.subplots(1,2, figsize=(14,8))
ax[0].pie(m , labels = m.index, autopct='%1.1f%%')
ax[0].set_title('Total_movies_by_rating')

ax[1].pie(t , labels = t.index, autopct='%1.1f%%')
ax[1].set_title('Total_TV_shows_by_rating')

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

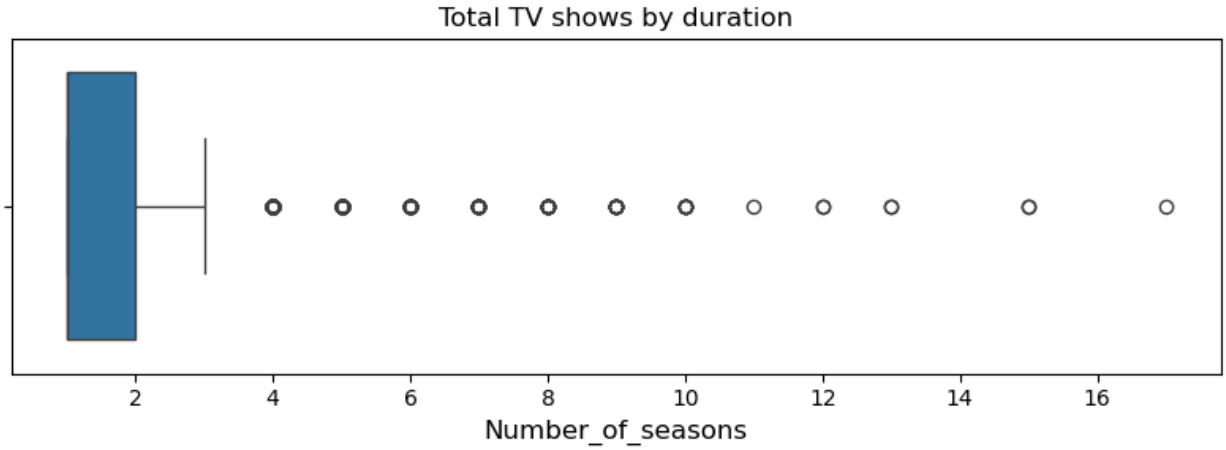




11. Total shows distribution by duration of the content

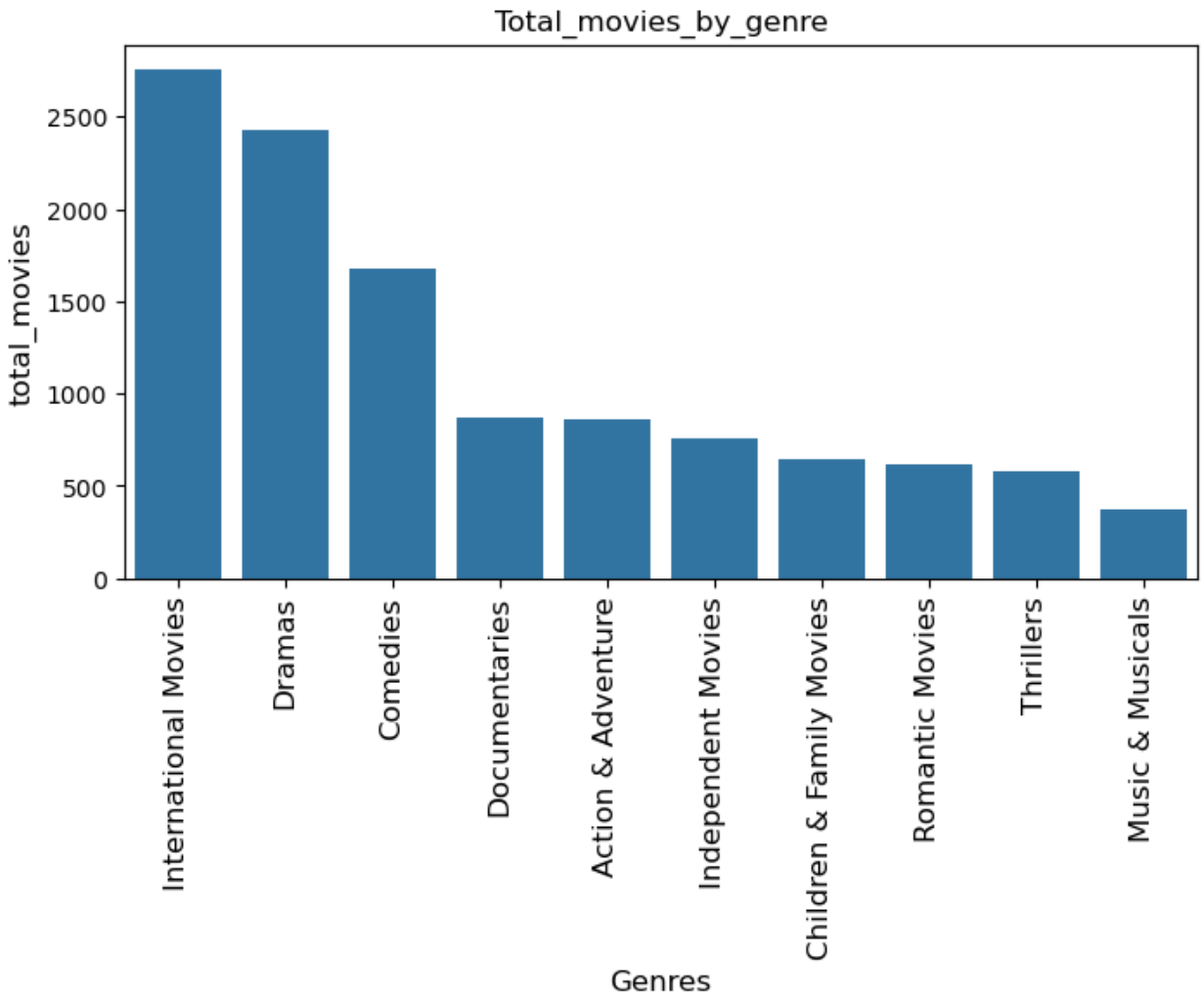
```
fig, ax = plt.subplots(2,1, figsize=(8,6))

sns.boxplot (data = movies , x = 'duration_in_minutes' ,ax =ax[0])
ax[0].set_xlabel('duration_in_minutes' ,  fontsize = 12)
ax[0].set_title('Total movies by duration')

sns.boxplot (data = tv_shows , x = 'duration_in_seasons' , ax = ax[1])
ax[1].set_xlabel('Number_of_seasons' ,  fontsize = 12)
ax[1].set_title('Total TV shows by duration')

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



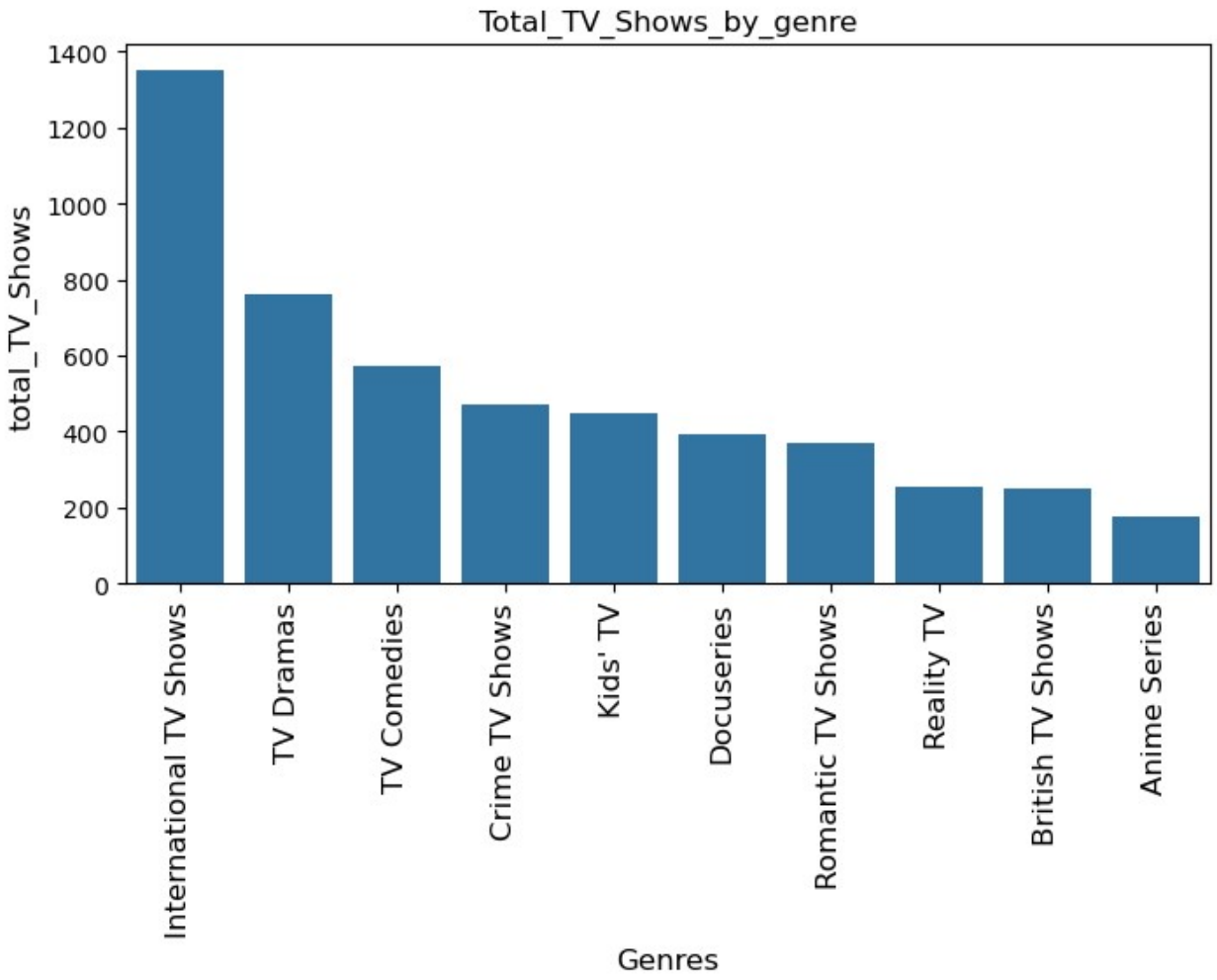


Total shows in each Genre

```
# Lets check the count for top 10 genres in Movies and TV_shows
top_10_movie_genres = genre_tb[genre_tb['type'] ==
'Movie'].listed_in.value_counts().head(10).index
df_movie =
genre_tb.loc[genre_tb['listed_in'].isin(top_10_movie_genres)]
top_10_TV_genres = genre_tb[genre_tb['type'] == 'TV
Show'].listed_in.value_counts().head(10).index
df_tv = genre_tb.loc[genre_tb['listed_in'].isin(top_10_TV_genres)]

plt.figure(figsize= (8,4))
sns.countplot(data = df_movie , x = 'listed_in' , order =
top_10_movie_genres)
plt.xticks(rotation = 90 , fontsize = 12)
plt.ylabel('total_movies' , fontsize = 12)
plt.xlabel('Genres' , fontsize = 12)
plt.title('Total_movies_by_genre')
plt.show()
```



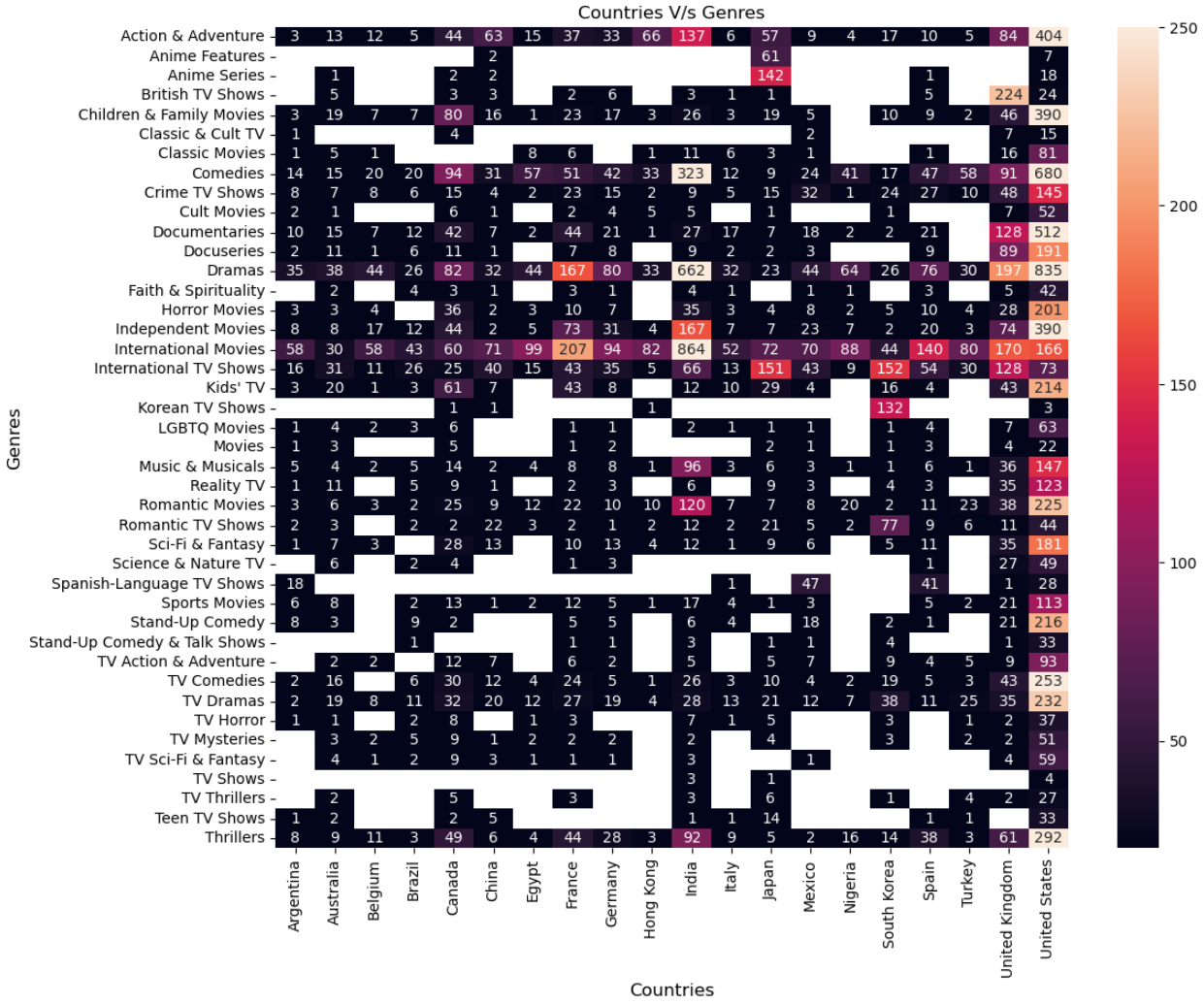

```
plt.figure(figsize= (8,4))
sns.countplot(data = df_tv , x = 'listed_in' , order =
top_10_TV_genres)
plt.xticks(rotation = 90 , fontsize = 12)
plt.ylabel('total_TV_Shows' , fontsize = 12)
plt.xlabel('Genres' , fontsize = 12)
plt.title('Total_TV_Shows_by_genre')
plt.show()
```



12. Lets check popular genres in top 20 countries

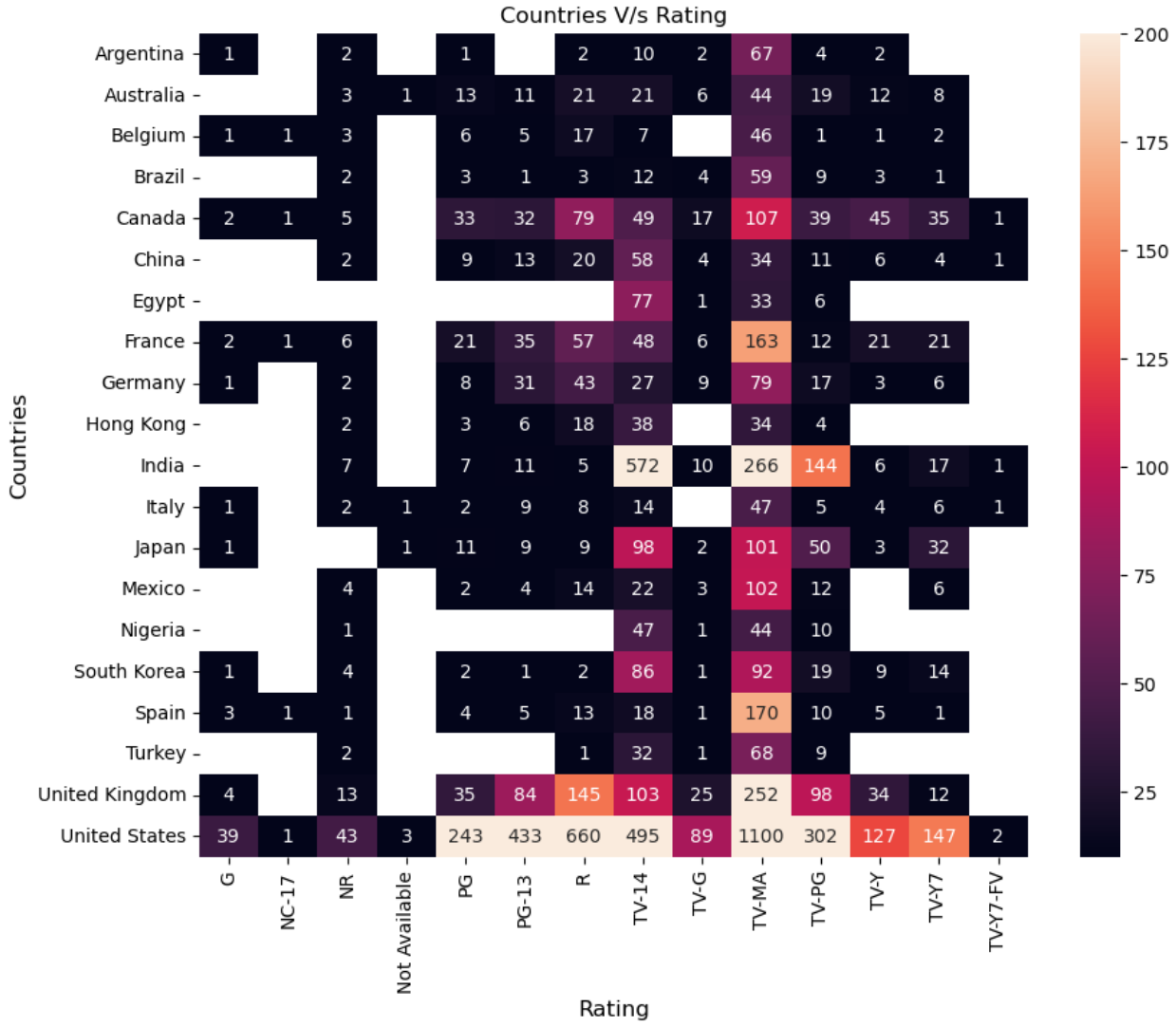
```
top_20_country = country_tb.country.value_counts().head(20).index
top_20_country =
country_tb.loc[country_tb['country'].isin(top_20_country)]
x = top_20_country.merge(genre_tb , on = 'show_id').drop_duplicates()
country_genre = x.groupby([ 'country' , 'listed_in'])
['show_id'].count().sort_values(ascending = False).reset_index()
country_genre = country_genre.pivot(index = 'listed_in' , columns =
'country' , values = 'show_id')
plt.figure(figsize = (12,10))
sns.heatmap(data = country_genre , annot = True , fmt=".0f" , vmin =
20 , vmax = 250 )
plt.xlabel('Countries' , fontsize = 12)
plt.ylabel('Genres' , fontsize = 12)
plt.title('Countries V/s Genres' , fontsize = 12)
```

```
Text(0.5, 1.0, 'Countries V/s Genres')
```



13. Country-wise Rating of Content

```
x = top_20_country.merge(df , on = 'show_id').groupby(['country_x' , 'rating'])['show_id'].count().reset_index()

country_rating = x.pivot(index = ['country_x'] , columns = 'rating' , values = 'show_id')

plt.figure(figsize = (10,8))
sns.heatmap(data = country_rating , annot = True , fmt=".0f"  , vmin = 10 , vmax=200)
plt.ylabel('Countries' , fontsize = 12)
plt.xlabel('Rating' , fontsize = 12)
plt.title('Countries V/s Rating' , fontsize = 12)

Text(0.5, 1.0, 'Countries V/s Rating')
```



14. The top actors by country

```
x = cast_tb.merge(country_tb , on = 'show_id').drop_duplicates()
x = x.groupby(['country' , 'cast'])['show_id'].count().reset_index()
x.loc[x['country'].isin(['United States'])].sort_values('show_id' ,
ascending = False).head(5)
```

	country	cast	show_id
49405	United States	Tara Strong	22
48330	United States	Samuel L. Jackson	22
40463	United States	Fred Tatasciore	21
35733	United States	Adam Sandler	20
41672	United States	James Franco	19

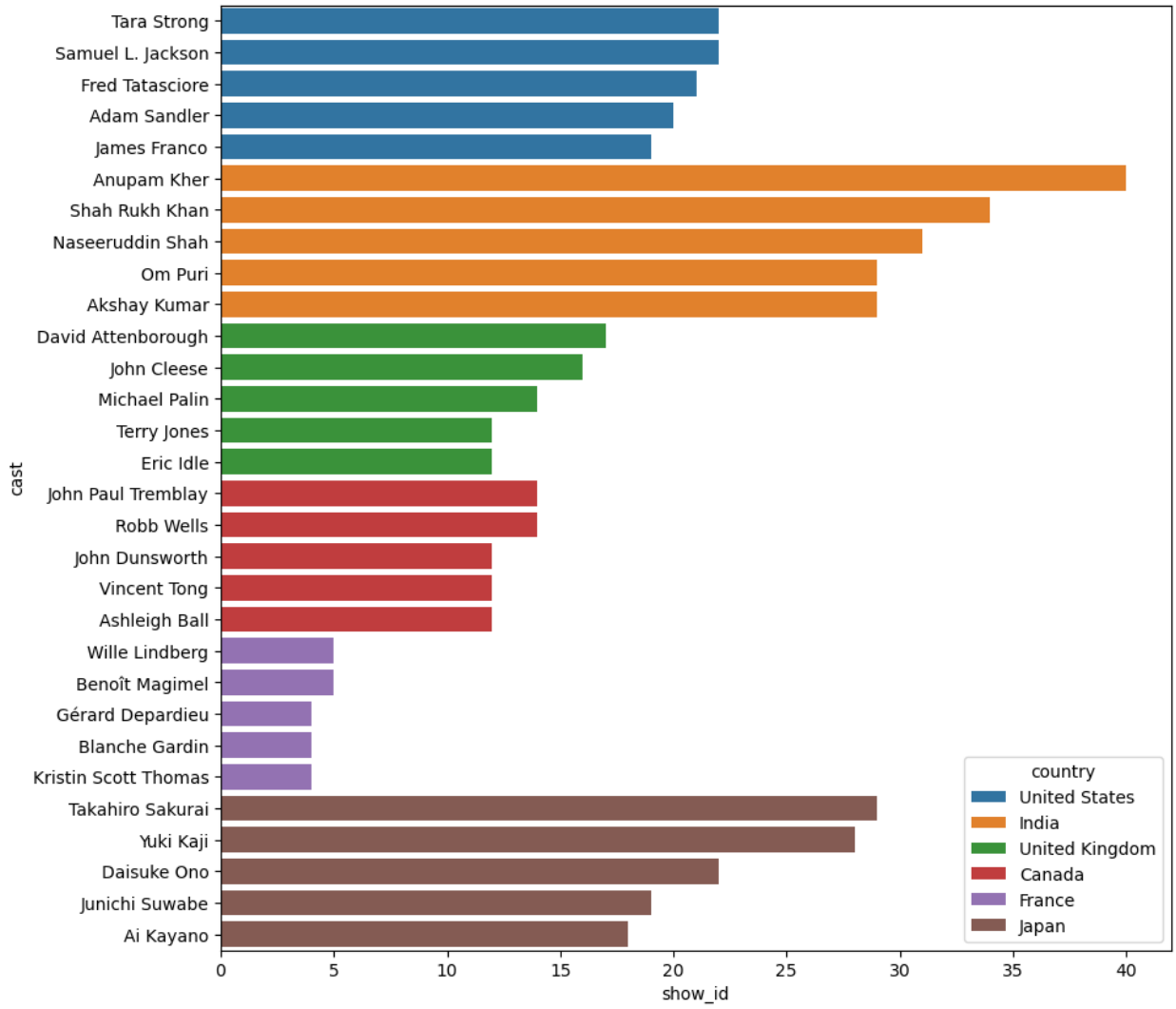
```
country_list = ['India'  , 'United Kingdom' , 'Canada' , 'France' ,
'Japan']
```

```
top_5_actors = x.loc[x['country'].isin(['United
States'])].sort_values('show_id' , ascending = False).head(5)

for i in country_list:
    new = x.loc[x['country'].isin([i])].sort_values('show_id' ,
ascending = False).head(5)
    top_5_actors = pd.concat( [top_5_actors , new] , ignore_index =
True)


#top 5 actors in top countries and their movies/tv shows count
top_5_actors
```

	country	cast	show_id
0	United States	Tara Strong	22
1	United States	Samuel L. Jackson	22
2	United States	Fred Tatasciore	21
3	United States	Adam Sandler	20
4	United States	James Franco	19
5	India	Anupam Kher	40
6	India	Shah Rukh Khan	34
7	India	Naseeruddin Shah	31
8	India	Om Puri	29
9	India	Akshay Kumar	29
10	United Kingdom	David Attenborough	17
11	United Kingdom	John Cleese	16
12	United Kingdom	Michael Palin	14
13	United Kingdom	Terry Jones	12
14	United Kingdom	Eric Idle	12
15	Canada	John Paul Tremblay	14
16	Canada	Robb Wells	14
17	Canada	John Dunsworth	12
18	Canada	Vincent Tong	12
19	Canada	Ashleigh Ball	12
20	France	Wille Lindberg	5
21	France	Benoît Magimel	5
22	France	Gérard Depardieu	4
23	France	Blanche Gardin	4
24	France	Kristin Scott Thomas	4
25	Japan	Takahiro Sakurai	29
26	Japan	Yuki Kaji	28
27	Japan	Daisuke Ono	22
28	Japan	Junichi Suwabe	19
29	Japan	Ai Kayano	18

```
plt.figure(figsize = (10,10))
sns.barplot(data = top_5_actors , y = 'cast' , x = 'show_id' , hue =
'country')
```

```
<Axes: xlabel='show_id', ylabel='cast'>
```



15.Top 5 directors by Genre

```
genre_list = [ 'Children & Family Movies', 'Comedies','Dramas',
'International Movies', 'Documentaries' ,
              'International TV Shows', 'Sci-Fi & Fantasy',
'Thrillers', 'Horror Movies']

x = dir_tb.merge(genre_tb , on = 'show_id').groupby([ 'listed_in' ,
'director',])['show_id'].count().reset_index()

top_5_dir = x.loc[x['listed_in'] == 'Action &
Adventure'].sort_values('show_id' , ascending = False).head()

for i in genre_list:
    new = x.loc[x['listed_in'] == i].sort_values('show_id' , ascending
= False).head()
    top_5_dir = pd.concat([top_5_dir , new])
```

top_5_dir

	listed_in	director	show_id
147	Action & Adventure	Don Michael Paul	9
550	Action & Adventure	S.S. Rajamouli	7
651	Action & Adventure	Toshiya Shinohara	7
215	Action & Adventure	Hidenori Inoue	7
606	Action & Adventure	Steven Spielberg	5
1215	Children & Family Movies	Rajiv Chilaka	22
1303	Children & Family Movies	Suhas Kadav	16
1211	Children & Family Movies	Prakash Satam	7
1241	Children & Family Movies	Robert Rodriguez	7
1288	Children & Family Movies	Steve Ball	6
1756	Comedies	David Dhawan	9
1905	Comedies	Hakan Algül	8
2686	Comedies	Suhas Kadav	8
2456	Comedies	Prakash Satam	7
1663	Comedies	Cathy Garcia-Molina	7
5935	Dramas	Youssef Chahine	12
4254	Dramas	Cathy Garcia-Molina	9
5099	Dramas	Martin Scorsese	9
4590	Dramas	Hanung Bramantyo	8
5544	Dramas	S.S. Rajamouli	7
7509	International Movies	Cathy Garcia-Molina	13
9330	International Movies	Youssef Chahine	10
9340	International Movies	Yılmaz Erdoğan	9
7620	International Movies	David Dhawan	8
8208	International Movies	Kunle Afolayan	8
3834	Documentaries	Vlad Yudin	6
3799	Documentaries	Thierry Donard	5
3217	Documentaries	Edward Cotterill	4
3262	Documentaries	Frank Capra	4
3075	Documentaries	Barry Avrich	4
9373	International TV Shows	Alastair Fothergill	3
9419	International TV Shows	Hsu Fu-chun	2
9436	International TV Shows	Jung-ah Im	2
9501	International TV Shows	Shin Won-ho	2
9478	International TV Shows	Pali Yahya	1
10752	Sci-Fi & Fantasy	Lilly Wachowski	4
10744	Sci-Fi & Fantasy	Lana Wachowski	4
10684	Sci-Fi & Fantasy	Guillermo del Toro	3
10790	Sci-Fi & Fantasy	Paul W.S. Anderson	3
10635	Sci-Fi & Fantasy	Barry Sonnenfeld	3
11974	Thrillers	Rathindran R Prasad	4
11698	Thrillers	David Fincher	4
11612	Thrillers	Anurag Kashyap	3
11636	Thrillers	Brad Anderson	3
11754	Thrillers	Gregory Hoblit	3
6280	Horror Movies	Rocky Soraya	6

```
6260              Horror Movies              Poj Arnon         5
6267              Horror Movies    Rathindran R Prasad         4
6191              Horror Movies           Leigh Janiak         3
6052              Horror Movies  Banjong Pisanthanakun         3
```

16. Variation in duration of movies by Release year

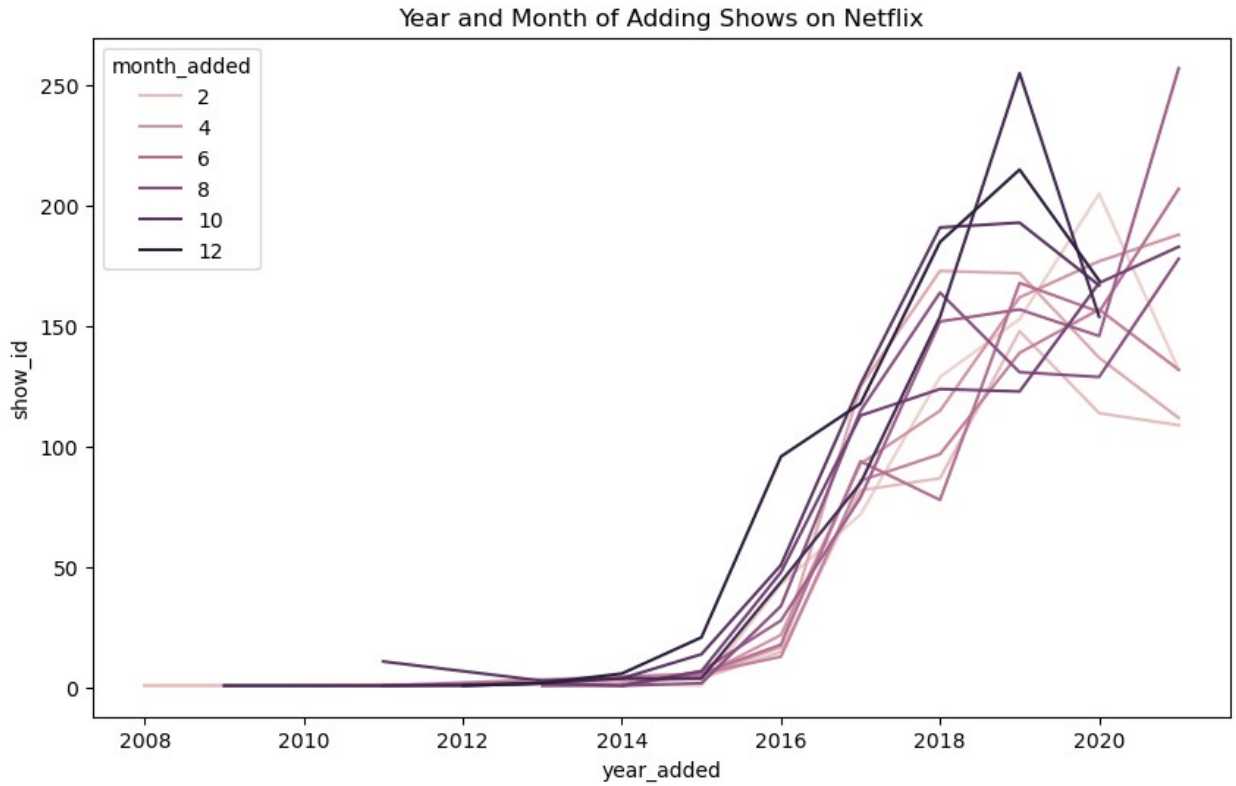
```
plt.figure(figsize=(12, 8))
sns.scatterplot(x=movies['duration_in_minutes'],
y=movies['release_year'], alpha=0.5)
plt.xlim((0, 200))
plt.xlabel('Duration (in minutes)')
plt.ylabel('Release Year')
plt.title('Scatterplot of Movie Duration vs Release Year')
plt.show()
```

17 The best time of the year when maximum content get added on the Netflix

```
# Assuming there's a 'date_added' column, let's extract 'year_added'
and 'month_added'
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')
```

```
# Convert to datetime format
df['year_added'] = df['date_added'].dt.year
df['month_added'] = df['date_added'].dt.month

month_year = df.groupby(['year_added', 'month_added'])
['show_id'].count().reset_index()

plt.figure(figsize = (10,6))
sns.lineplot(data=month_year, x = 'year_added', y = 'show_id',
hue='month_added')
plt.title('Year and Month of Adding Shows on Netflix')
```

```
Text(0.5, 1.0, 'Year and Month of Adding Shows on Netflix')
```

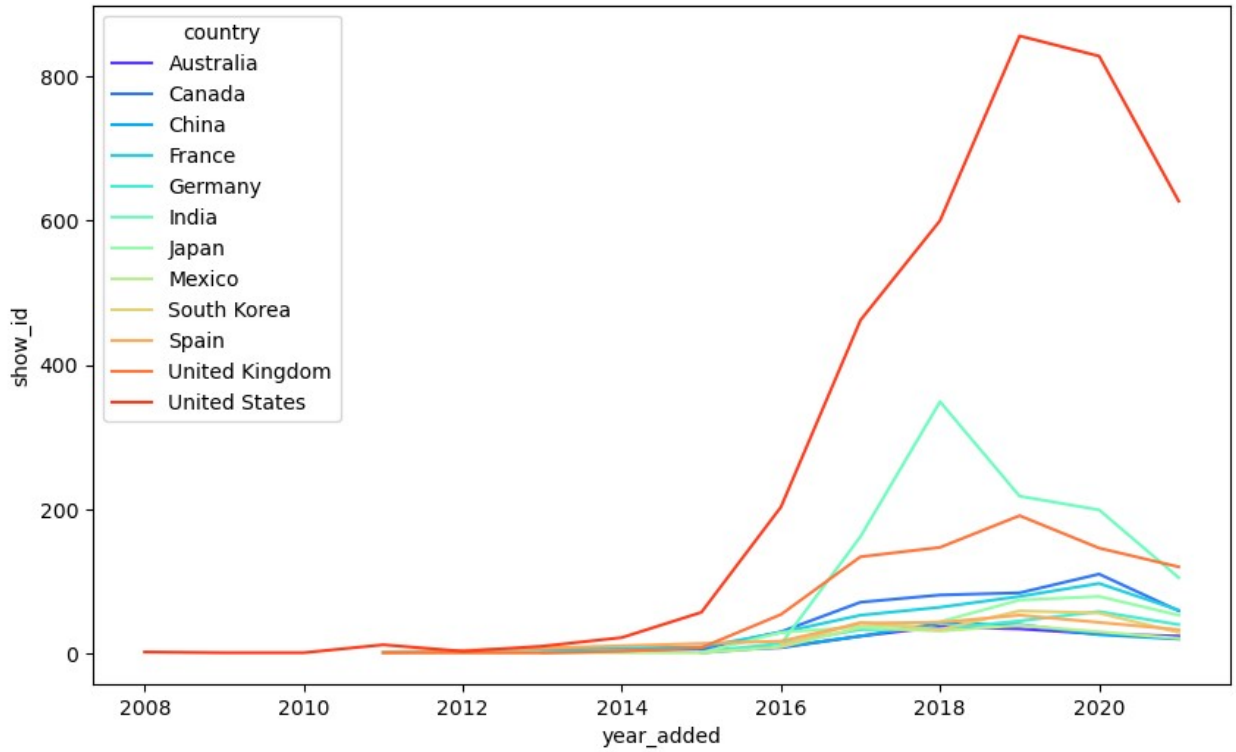


18. Countries are adding more number of content over the time Shown Below

```
country_list = country_tb.country.value_counts().head(12).index
top_12_country =
country_tb.loc[country_tb['country'].isin(country_list)]
country_year = top_12_country.merge(df , on = 'show_id')
[['show_id','country_x' ,'type_x' , 'year_added' ]]
country_year.columns = ['show_id', 'country', 'type', 'year_added']
```

```
country_year = country_year.groupby(['country' , 'year_added'])
['show_id'].count().reset_index()

plt.figure(figsize = (10,6))
sns.lineplot(data = country_year , x = 'year_added' , y = 'show_id' ,
hue = 'country' , palette ='rainbow' )
```

```
<Axes: xlabel='year_added', ylabel='show_id'>
```



Observations:

Target Audience: The majority of Netflix content is geared towards Mature and Adult viewers, with most movies rated for these age groups. Around 20% of content is aimed at children under 13, indicating that Netflix primarily caters to Mature audiences and Children with parental guidance.

Popular Genres: The top genres on Netflix include:

International Movies and TV Shows Dramas Comedies Action & Adventure Children & Family Movies Thrillers Content Distribution: Around 75% of Netflix's content comes from the top 10 countries, with the remaining 25% originating from other regions. This indicates a potential opportunity for Netflix to expand its content production in more countries to further grow its global presence.

Rising Popularity of Short Content: There is an increasing demand for shorter content, such as movies ranging from 75 to 150 minutes and seasons 1 to 3. This trend should be considered when planning future content production for Netflix.

Impact of the Pandemic: A notable decline in content production was observed across all countries and content types during 2020 and 2021, likely due to the pandemic and its disruption of production schedules.

Recommendations

Expand Genre Diversity: Most countries, apart from the US, have a narrow focus on specific genres. It appears that the current selection works well for the US and a few other countries, but many regions would benefit from a broader range of genres tailored to their local tastes. For example, Indian Mythological content is extremely popular, and creating more region-specific genres could appeal to local audiences and even gain global popularity, much like how Japanese Anime has become internationally beloved.

Tailor Content to Country Demographics: Content should be curated based on the demographics of each country. Netflix could produce more content in specific rating categories that align with the local audience's age group and preferences. For instance:

In a country like India, where the population is large and diverse, the content primarily falls under three ratings: TV-MA, TV-14, and TV-PG. This suggests a need to focus more on content suitable for teenagers and adults, while also considering options for younger audiences under 14 and adults over 35.

Conclusion

To continue expanding and catering to a global audience, Netflix should diversify its genre offerings based on regional preferences and demographics. Tailoring content to suit local tastes, while also exploring the creation of unique country-specific genres, will allow Netflix to appeal to a broader and more diverse viewership. Additionally, producing content that aligns with the age and preference distribution in each country can enhance viewer engagement and retention.