Sclaer_Netflix_Analysis

April 24, 2024

```
[ ]: #importing Libraries
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import warnings
     warnings.filterwarnings('ignore')
```

```
[ ]: #Loding the dataset
     from google.colab import files
     files.upload()
```

```
<IPython.core.display.HTML object>
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-90-dafbd9b6b182> in <cell line: 3>()
      1 #Loding the dataset
      2 from google.colab import files
----> 3 files.upload()

/usr/local/lib/python3.10/dist-packages/google/colab/files.py in upload()
     67   """
     68
---> 69   uploaded_files = _upload_files(multiple=True)
     70   # Mapping from original filename to filename as saved locally.
     71   local_filenames = dict()

/usr/local/lib/python3.10/dist-packages/google/colab/files.py in␣
 ↪_upload_files(multiple)
    154
    155   # First result is always an indication that the file picker has␣
 ↪completed.
--> 156   result = _output.eval_js(
    157       'google.colab._files._uploadFiles("{input_id}", "{output_id}")'.
 ↪format(
    158           input_id=input_id, output_id=output_id
```

```
/usr/local/lib/python3.10/dist-packages/google/colab/output/_js.py in␣
 ↪eval_js(script, ignore_result, timeout_sec)
     38   if ignore_result:
     39     return
---> 40   return _message.read_reply_from_input(request_id, timeout_sec)
     41
     42

/usr/local/lib/python3.10/dist-packages/google/colab/_message.py in␣
 ↪read_reply_from_input(message_id, timeout_sec)
     94     reply = _read_next_input_message()
     95     if reply == _NOT_READY or not isinstance(reply, dict):
---> 96       time.sleep(0.025)
     97       continue
     98     if (

KeyboardInterrupt:
```

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

[ ]:
```
  show_id     type                 title         director  \
0      s1    Movie  Dick Johnson Is Dead  Kirsten Johnson
1      s2  TV Show         Blood & Water              NaN
2      s3  TV Show             Ganglands  Julien Leclercq
3      s4  TV Show  Jailbirds New Orleans              NaN
4      s5  TV Show          Kota Factory              NaN

                                                cast        country  \
0                                                NaN  United States
1  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…   South Africa
2  Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi…            NaN
3                                                NaN            NaN
4  Mayur More, Jitendra Kumar, Ranjan Raj, Alam K…          India

          date_added  release_year rating   duration  \
0  September 25, 2021          2020  PG-13     90 min
1  September 24, 2021          2021  TV-MA  2 Seasons
2  September 24, 2021          2021  TV-MA   1 Season
3  September 24, 2021          2021  TV-MA   1 Season
4  September 24, 2021          2021  TV-MA  2 Seasons

                                          listed_in  \
0                                     Documentaries
1    International TV Shows, TV Dramas, TV Mysteries
2  Crime TV Shows, International TV Shows, TV Act…
```

```
3                              Docuseries, Reality TV
4  International TV Shows, Romantic TV Shows, TV …

                                         description
0  As her father nears the end of his life, filmm…
1  After crossing paths at a party, a Cape Town t…
2  To protect his family from a powerful drug lor…
3  Feuds, flirtations and toilet talk go down amo…
4  In a city of coaching centers known to train I…
```

[ ]:
```
df.shape
```

[ ]:
```
(8807, 12)
```

[ ]:
```
#Descriptive Statistics
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
```

Only single column having numerical values. It gives idea of release year of the content ranges between what timeframe. Rest all the columns are having categorical data.

[ ]:
```
#concise summary
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8797 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8797 non-null   object
 1   type          8797 non-null   category
 2   title         8797 non-null   object
 3   director      8797 non-null   object
 4   cast          8797 non-null   object
 5   country       8797 non-null   category
 6   date_added    8797 non-null   object
 7   release_year  8797 non-null   int64
 8   rating        8797 non-null   object
 9   duration      8797 non-null   object
 10  listed_in     8797 non-null   object
 11  description   8797 non-null   object
dtypes: category(2), int64(1), object(9)
memory usage: 803.9+ KB
```

[ ]:
```
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
```

These are total features of our dataset. It is seen that show_id column has all unique values, Title column has all unique values i.e. total 8807 which equates with total rows in the dataset. Hence It can be concluded that ,

Total 8807 movies/TV shows data is provided in the dataset.

[ ]:
```
#missing values
df.isnull().sum()
```

[ ]:
```
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
```

3 missing values are found in duration column , and it is also found that by mistake those data got entered in rating column

[ ]:
```
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…
```

```
[ ]: 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)
```

In rating column , NR (Not rated) is same as UR (Unrated). lets change UR to NR.

[ ]:
```
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'] = pd.to_datetime(df['date_added'].str.strip(), errors='coerce')
     df['date_added']
```

```
[ ]: 0       2021-09-25
     1       2021-09-24
     2       2021-09-24
     3       2021-09-24
     4       2021-09-24
                …
     8802    2019-11-20
     8803    2019-07-01
     8804    2019-11-01
     8805    2020-01-11
     8806    2019-03-02
     Name: date_added, Length: 8797, dtype: datetime64[ns]
```

```
[ ]: df['year_added'] = df['date_added'].dt.year
     df['month_added'] = df['date_added'].dt.month
     df[['date_added' , 'year_added' , 'month_added']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8797 entries, 0 to 8806
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   date_added   8797 non-null   datetime64[ns]
 1   year_added   8797 non-null   int32
 2   month_added  8797 non-null   int32
dtypes: datetime64[ns](1), int32(2)
memory usage: 206.2 KB
```

```
[ ]: # 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
     year_added        0
     month_added       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
     year_added       0.0
     month_added      0.0
     dtype: float64
```

after cleaning some data we still have null values in 3 columns. These are much higher in numbers.

For some content - country is missing. (9%)

for some content - director names are missing (30%)

for some content - cast is missing (9%)

```
[ ]: #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
```

Since movie and TV shows both have different format for duration, we can change duration for movies as minutes & TV shows as seasons

```
[ ]: 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
```

```
[ ]: #The oldest and the most recent movie/TV show released on the Netflix in which␣
      ↪year?
     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  year_added  \
4250  This collection restores films from women who …        2018
966   After a botched scam, Clóvis bumps into Lohane…        2021
967   Learn how to sleep better with Headspace. Each…        2021
968   To build an innovative sex app and win a tech …        2021
972   Reeling from tragedy, a nondescript house clea…        2021
…                                                   …          …
466   Follow Julia Haart, Elite World Group CEO and …        2021
467   A deep dive into the work of renowned Mexican …        2021
468   A couple in Québec deals with the pitfalls, pr…        2021
471   With their family facing financial woes, two t…        2021
8437  Hosts David Spade, Fortune Feimster and London…        2021

      month_added
4250           12
966             4
967             4
968             4
972             4
…             …
466             7
467             7
468             7
471             7
8437            1

[593 rows x 14 columns]
```

[ ]:
```
#Which are different ratings available on Netflix in each type of content?␣
 ↪Check the number of content released in each type.
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
```

Working on the columns having maximum null values and the columns having comma separated multiple values for each record

Country column

```
[ ]: 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
```

We see that many movies are produced in more than 1 country. Hence, the country column has comma separated values of countries.

This makes it difficult to analyse how many movies were produced in each country. We can use explode function in pandas to split the country column into different rows.

we are Creating a separate table for country , to avoid the duplicasy of records in our origional table after exploding.

```
[ ]: 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]
```

```
[ ]: 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
```

122 **Netflix has movies from the total 122 countries.**

```
[ ]: #Total movies and tv shows in each country
     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]
```

Director column

```
[ ]: 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
```

There are some movies which are directed by multiple directors. Hence multiple names of directors are given in comma separated format. We will explode the director column as well. It will create many duplicate records in originaltable hence we created separate table for directors.

```
[ ]: 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
```

There are total 4993 unique directors in the dataset.

[ ]:
```
# There are total 4993 unique directors in the dataset.
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
```

Movies have 20 genres and TV shows have 22 genres.

[ ]:
```
# 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: []
```

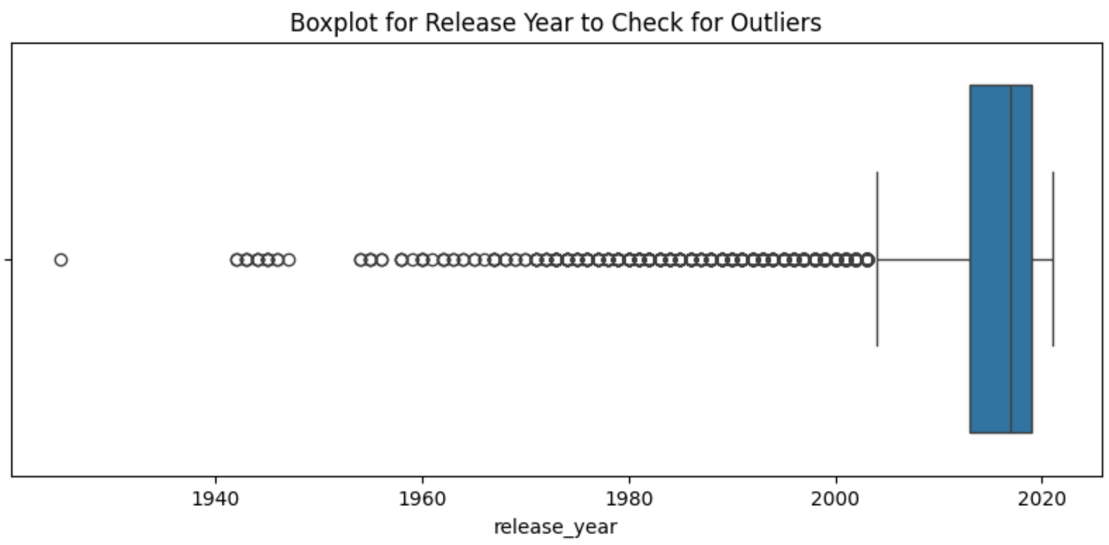
[ ]:
```
# Total actors on the Netflix
cast_tb.cast.nunique()
```

[ ]:
```
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]
```

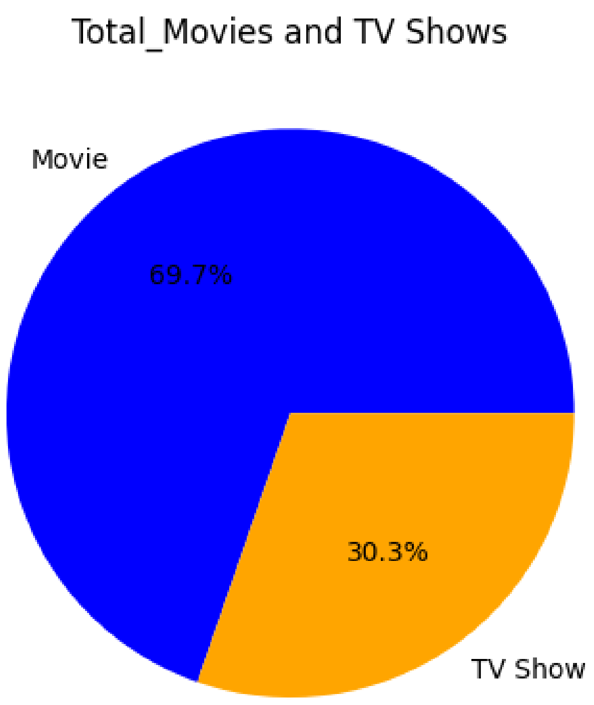
[ ]:
```
#Outliers
# Boxplot to check for outliers in 'release_year'
plt.figure(figsize=(10, 4))
sns.boxplot(x=df['release_year'])
plt.title('Boxplot for Release Year to Check for Outliers')
plt.show()
```



The boxplot for release_year shows no significant outliers, indicating that the data for this attribute is relatively consistent.

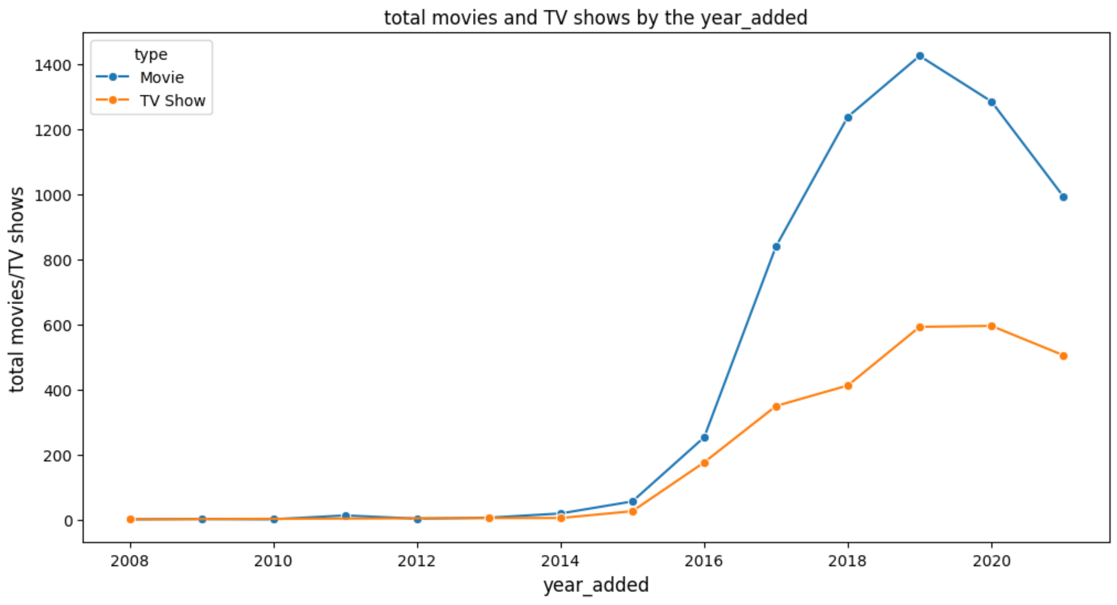
Visual Analysis - Univariate & Bivariate

```
[ ]: #Distribution of content across the different types
types = df.type.value_counts()
plt.pie(types,  labels=types.index, autopct='%1.1f%%' , colors = ['blue' ,␣
  ↪'orange'])
plt.title('Total_Movies and TV Shows')
plt.show()
```



It is observed that , **around 70% content is Movies and around 30% content is TV shows.**

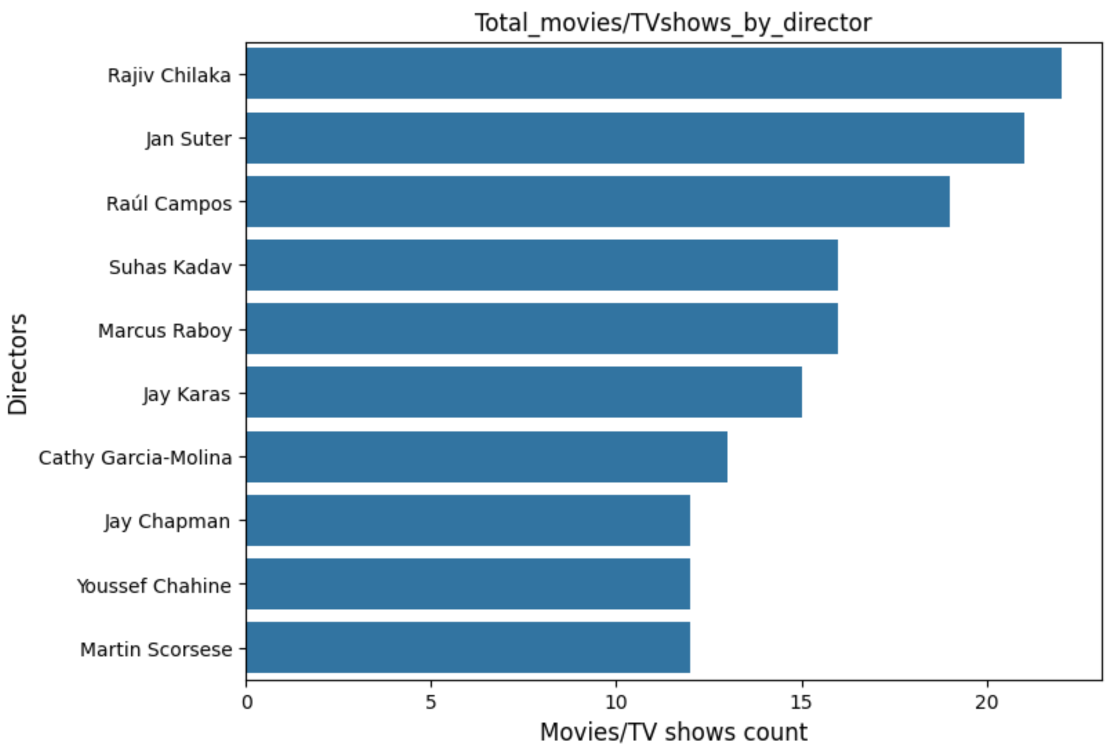
```
[ ]: #How has the number of movies/TV shows added on Netflix per year changed over␣
     ↪the time?
     d = df.groupby(['year_added' ,'type' ])['show_id'].count().reset_index()
     d.rename({'show_id' : 'total movies/TV shows'}, axis = 1 , inplace = True)
     plt.figure(figsize = (12,6))
     sns.lineplot(data = d , x = 'year_added' , y = 'total movies/TV shows' , hue =␣
     ↪'type', marker = 'o'  , ms = 6)
     plt.xlabel('year_added' , fontsize = 12)
     plt.ylabel('total movies/TV shows' , fontsize = 12)
     plt.title('total movies and TV shows by the year_added' , fontsize = 12)
     plt.show()
```



Observation:

The content added on the Netflix surged drastically after 2015. **2019 marks the highest number of movies and TV shows added on the Netflix. Year 2020 and 2021 has seen the drop** in content added on Netflix, possibly because of Pandemic. But still , TV shows content have not dropped as drastic as movies. In recent years TV shows are focussed more than Movies.

[ ]:
```
#Total movies/TV shows by each director
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()
```



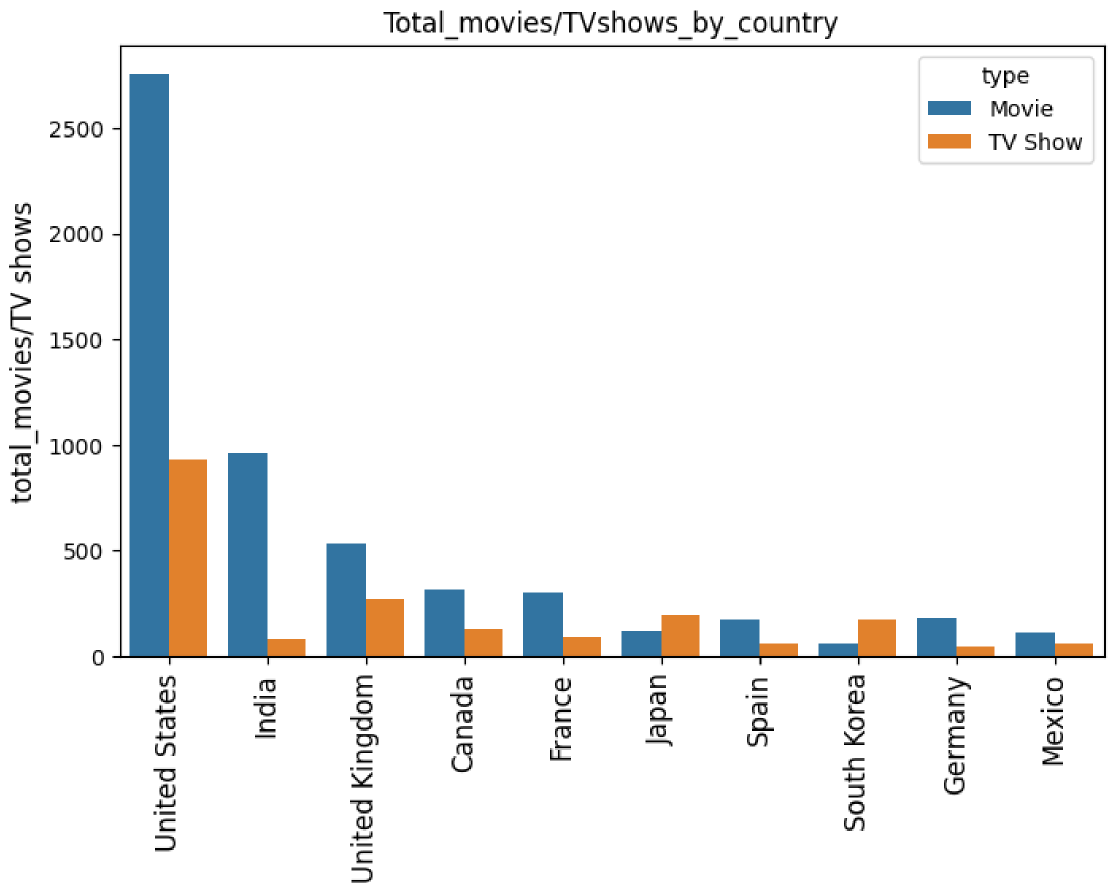
Observation:

The **top 3 directors** on Netflix in terms of count of movies directed by them are - **Rajiv Chilaka, Jan Suter, Raúl Campos**

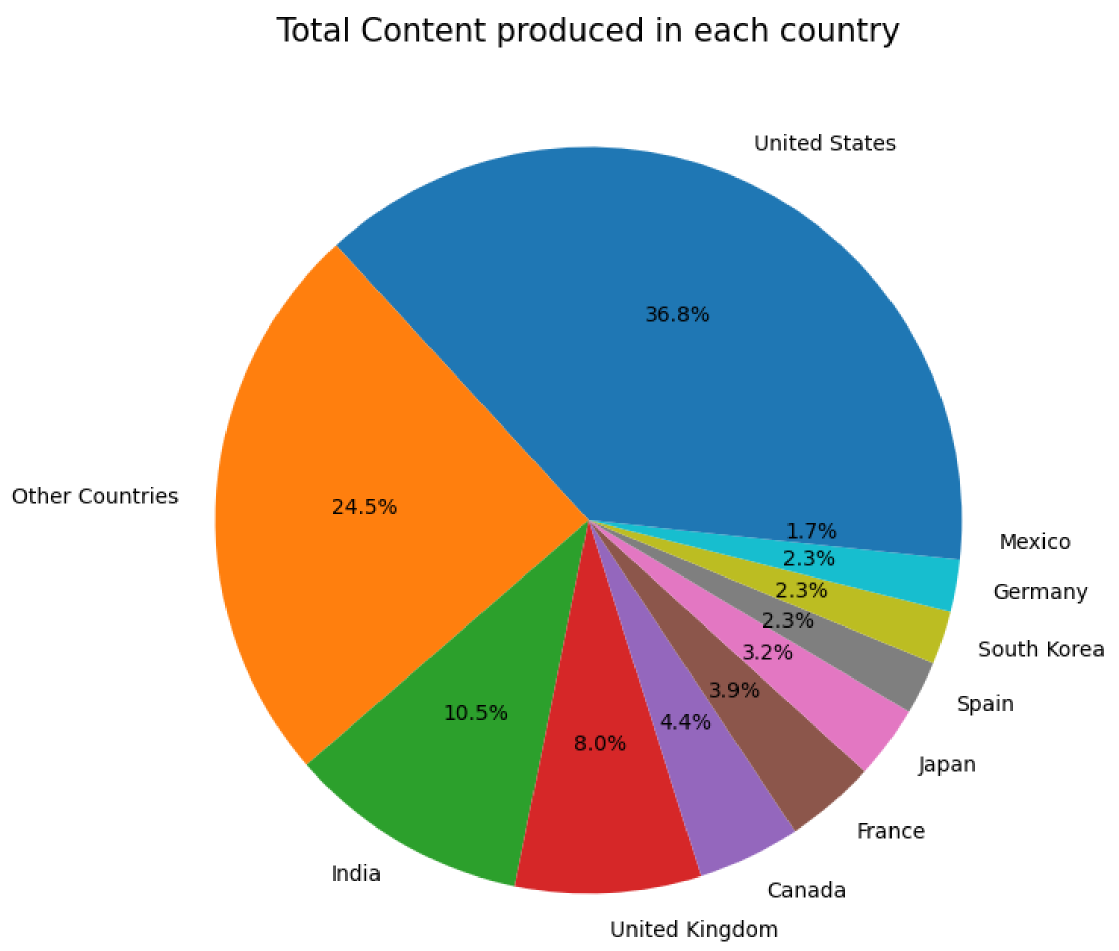
```
[ ]: #Total movies/TV shows by each country
     top_10_country = country_tb.country.value_counts().head(10).index
     df_new = country_tb.loc[country_tb['country'].isin(top_10_country)]
     x = df_new.groupby(['country' , 'type'])['show_id'].count().reset_index()
     x.pivot(index = 'country' , 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()
```

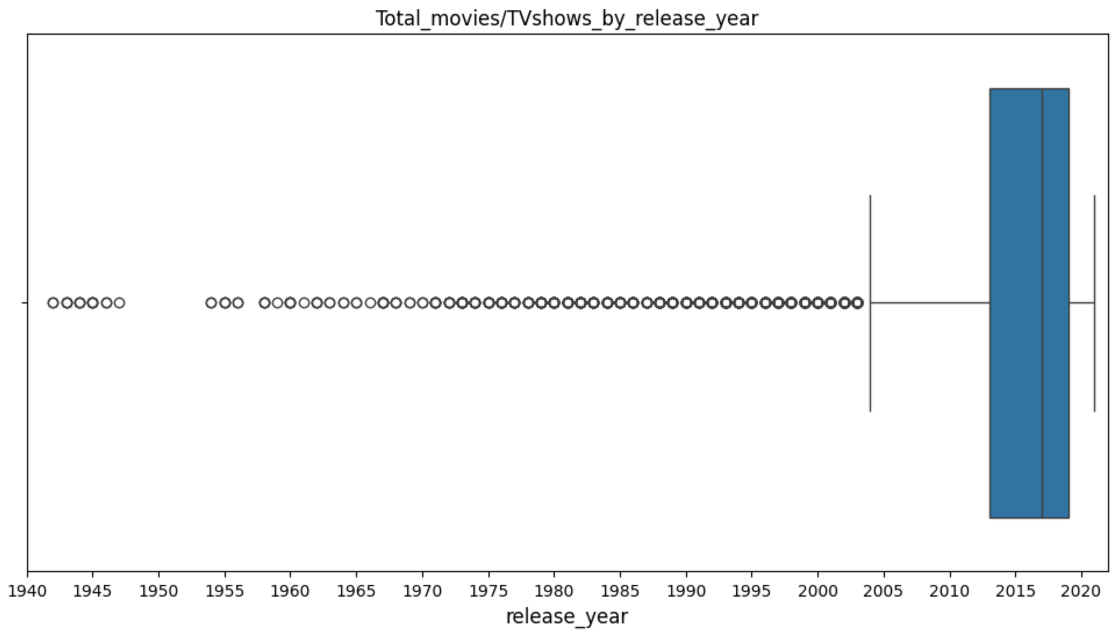


```
[ ]: top_10_country = country_tb.country.value_counts().head(10).index
     country_tb['cat'] = country_tb['country'].apply(lambda x : x if x in␣
      ↪top_10_country else 'Other Countries' )
     x = country_tb.cat.value_counts()

     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()
```



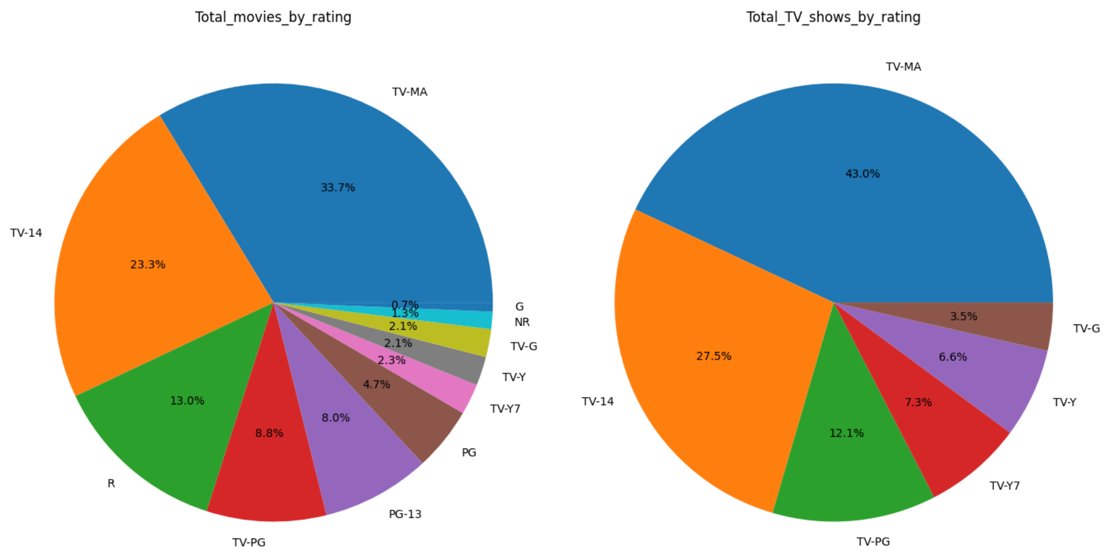
United States is the HIGHEST contributor country on Netflix, followed by India and United Kingdom. Maximum content of **Netflix which is around 75%** , is coming from these top 10 countries. Rest of the world only **contributes 25% of the content**.

```
[ ]: #Total content distribution by release year of the content
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()
```



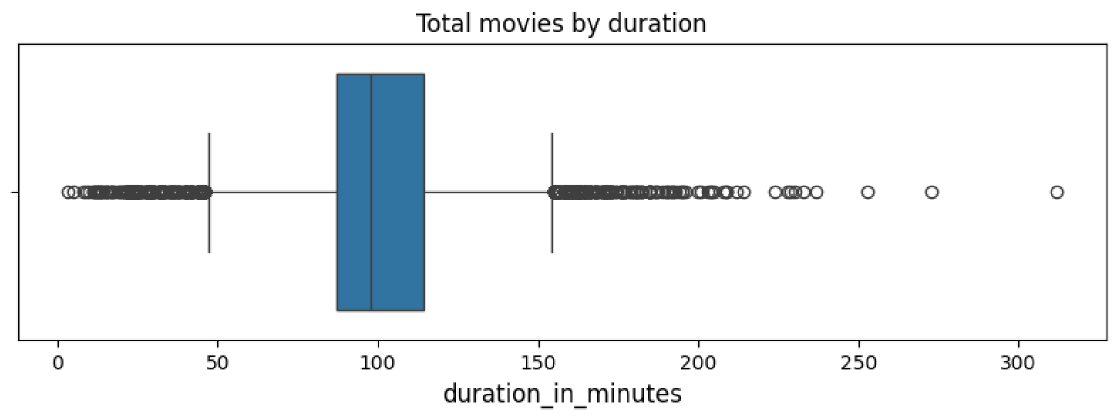
Netflix have major content which is released in the year **range 2000-2021** It seems that the content **older than year 2000 is almost missing** from the Netflix

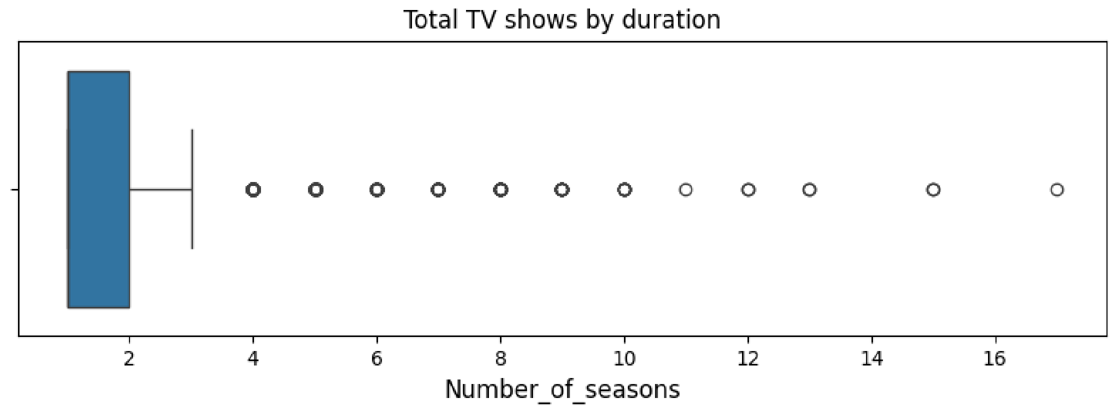
```
[ ]: #Total movies/TV shows distribution by rating of the content
     m = movies.loc[~movies.rating.isin(['Not Available' , 'NC-17' , 'TV-Y7-FV'])]
     m = m.rating.value_counts()
     t = tv_shows.loc[~tv_shows.rating.isin(['Not Available' , 'R' , 'NR',␣
       ↪'TV-Y7-FV'])]
     t = t.rating.value_counts()


     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()
```



Highest number of movies and TV shows are **rated TV-MA (for mature audiences), followed by TV-14 & R/TV-PG**

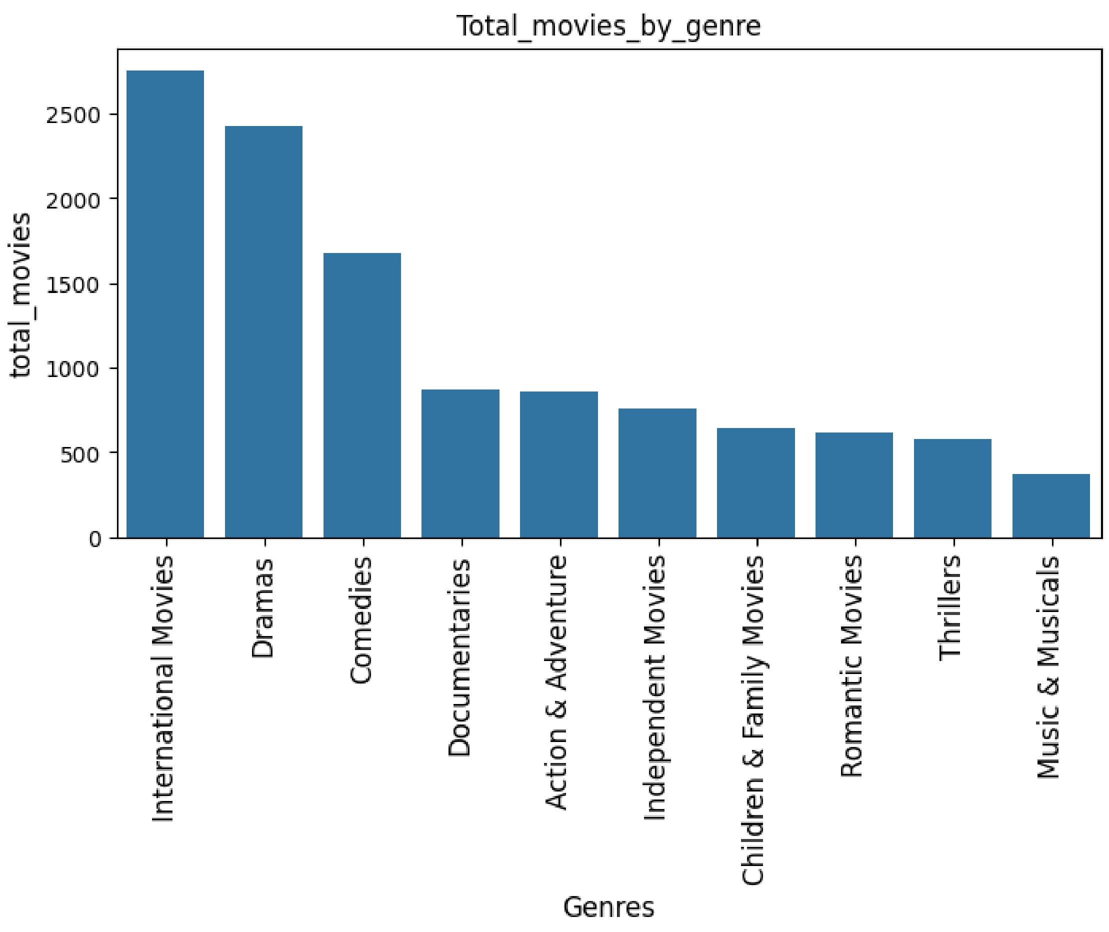
[ ]:
```
#Total movies/TV shows distribution by duration of the content
fig, ax = plt.subplots(2,1, figsize=(8,6))

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()
```





Movie Duration: **50 mins - 150 mins is the range** excluding potential outliers (values lying outside the whiskers of boxplot) TV Show Duration: **1-3 seasons is the range for TV shows** excluding potential outliers

[ ]:
```
#Total movies/TV shows in each Genre
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()
```

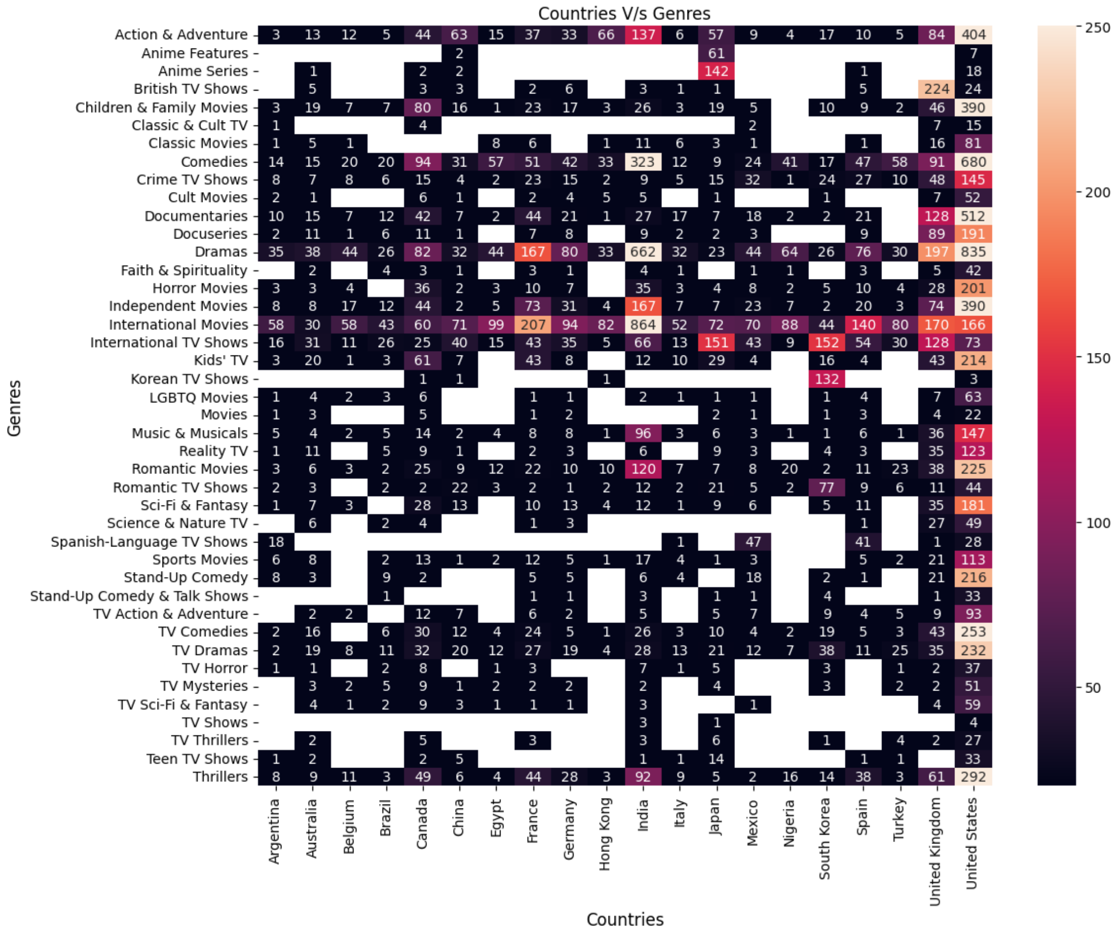


International Movies and **TV Shows , Dramas , and Comedies are the top 3 genres on Netflix for both Movies and TV shows**.

Bivariate analysis #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')
```



Popular genres across countries: **Action & Adventure, Children & Family Movies, Comedies, Dramas, International Movies & TV Shows, TV Dramas, Thrillers**

Country-specific genres: **Korean TV shows (Korea), British TV Shows (UK), Anime features and Anime series (Japan), Spanish TV Shows (Argentina, Mexico and Spain)**

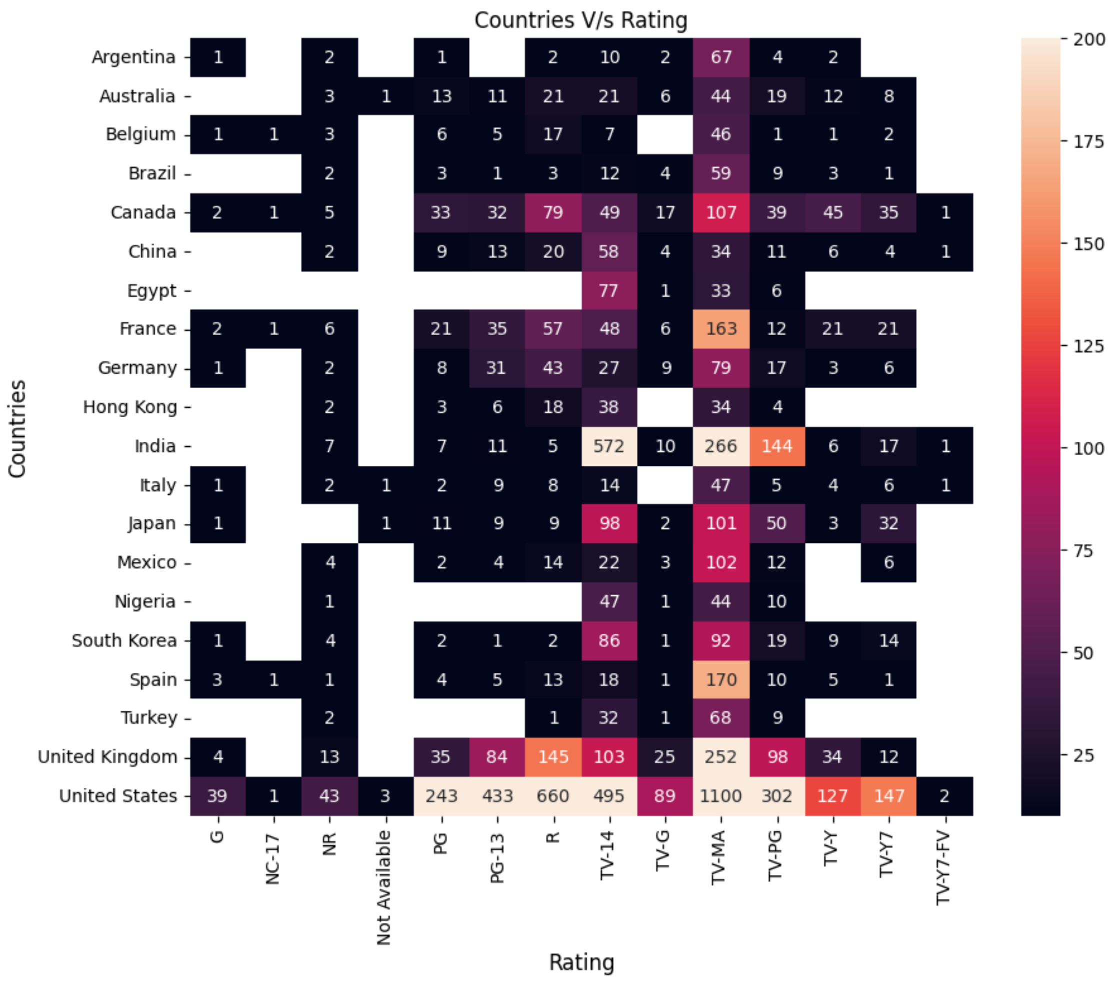
United States and UK have a good mix of almost all genres.

Maximum International movies are produced in India.

```
[ ]: #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')
```



Overall, Netflix has an large amount of adult content across all countries (TV-MA & TV-14). India also has many titles rated TV-PG, other than TV-MA & TV-14. Only US, Canada, UK, France and Japan have content for young audiences (TV-Y & TV-Y7). There is scarce content for general audience (TV-G & G) across all countries except US.

```
[ ]: #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)
```

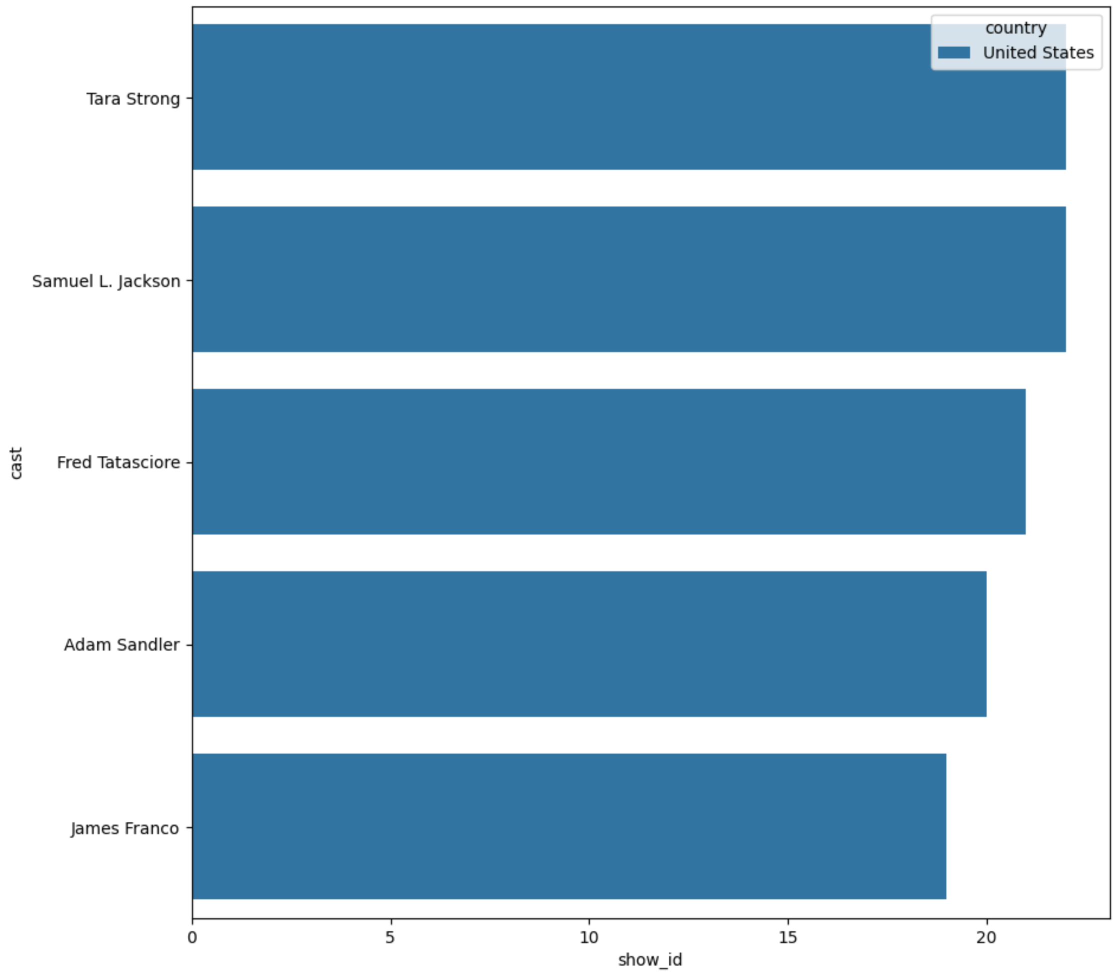
```
[ ]:                country               cast  show_id
     49405  United States        Tara Strong       22
     48330  United States  Samuel L. Jackson       22
     40463  United States    Fred Tatasciore       21
     35733  United States       Adam Sandler       20
     41672  United States       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.head()
```

```
[ ]:          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
```

```
[ ]: plt.figure(figsize = (10,10))
     sns.barplot(data = top_5_actors.head() , y = 'cast' , x = 'show_id' , hue =␣
      ↪'country')
```

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

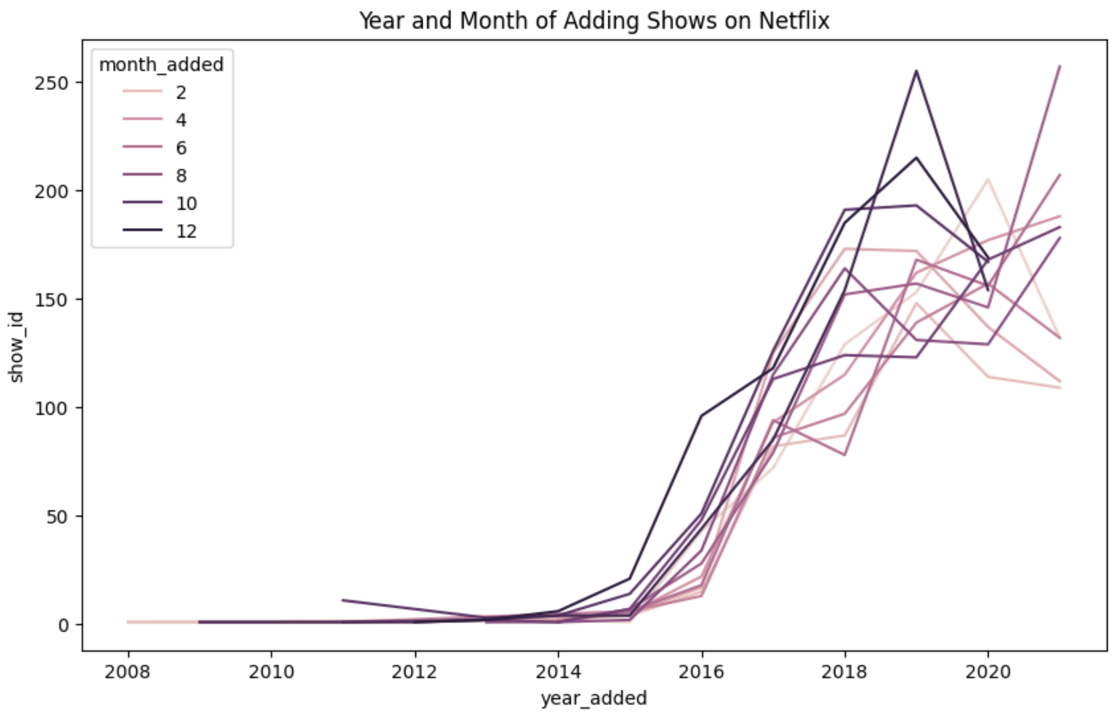


```
[ ]: #Top 5 genres in each country
     x = genre_tb.merge(country_tb , on = 'show_id').drop_duplicates()
     x = x.groupby(['country' , 'listed_in'])['show_id'].count().reset_index()
     x.loc[x['country'] == 'United States'].sort_values('show_id' , ascending =␣
      ↪False).head(5)

     country_list = ['India'  , 'United Kingdom' , 'Canada' , 'France' , 'Japan']
     top_5_genre = 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'] == i].sort_values('show_id' , ascending = False).
      ↪head(5)
         top_5_genre = pd.concat( [top_5_genre , new] , ignore_index = True)
```

```
[ ]: top_5_genre.head()
[ ]:           country           listed_in  show_id
     0  United States              Dramas      835
     1  United States            Comedies      680
     2  United States       Documentaries      512
     3  United States  Action & Adventure      404
     4  United States  Independent Movies      390
```

```
[ ]: #What is the best time of the year when maximum content get added on the␣
       ↪Netflix?
     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')
```



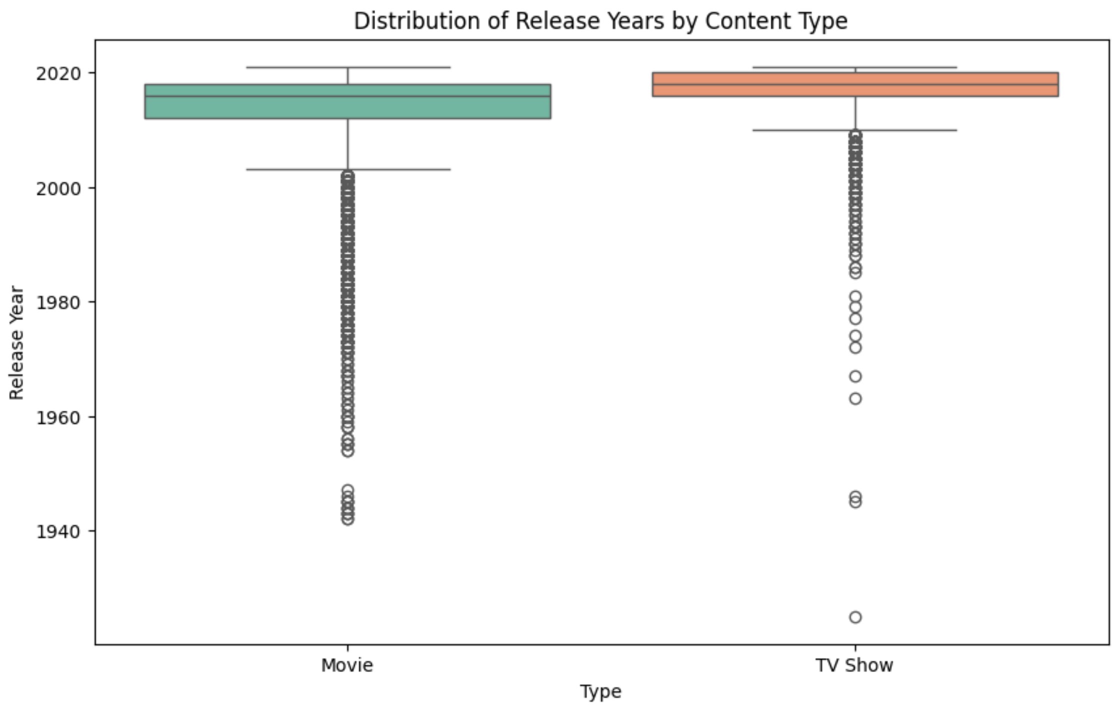
The number of shows getting added is increasing with each year until 2020.

Also, **months in the last quarter of the year (Oct-Dec)** have more shows being added than the other months of the year. This could be because US has its festive season in Dec and **India also has Diwali in Oct-Nov**

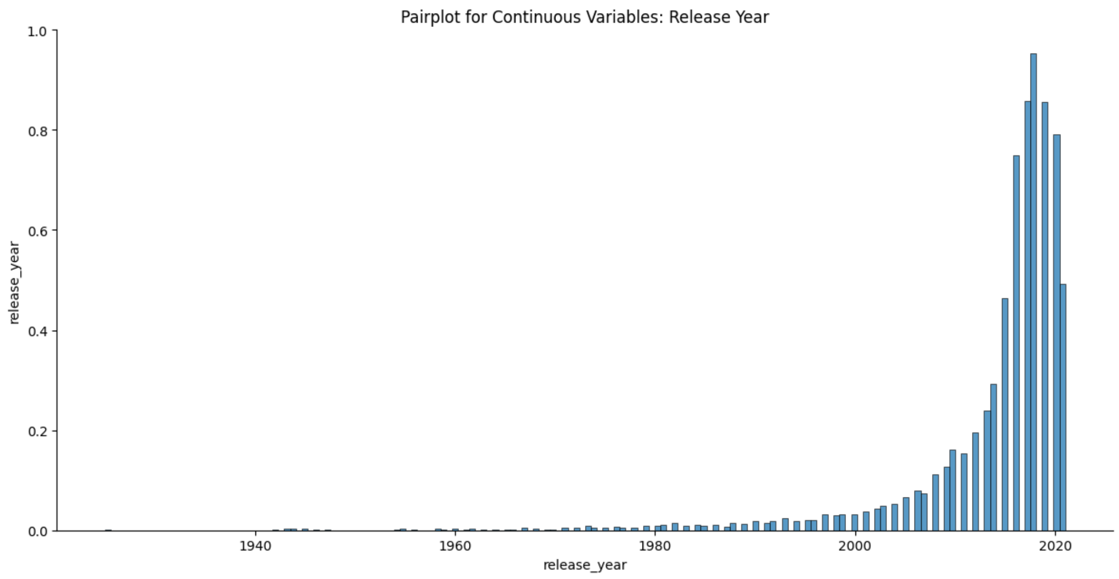
Observation : United Stated have always added highset number of movies/TV shows

over the time. Since 2016, India has seen spike in popularity of content and added more number of content, followed by United Kingdom at 3rd position.

```
[ ]: #Relationship Between Type and Release Year
     plt.figure(figsize=(10, 6))
     sns.boxplot(x='type', y='release_year', data=df, palette='Set2')
     plt.title('Distribution of Release Years by Content Type')
     plt.xlabel('Type')
     plt.ylabel('Release Year')
     plt.show()
```

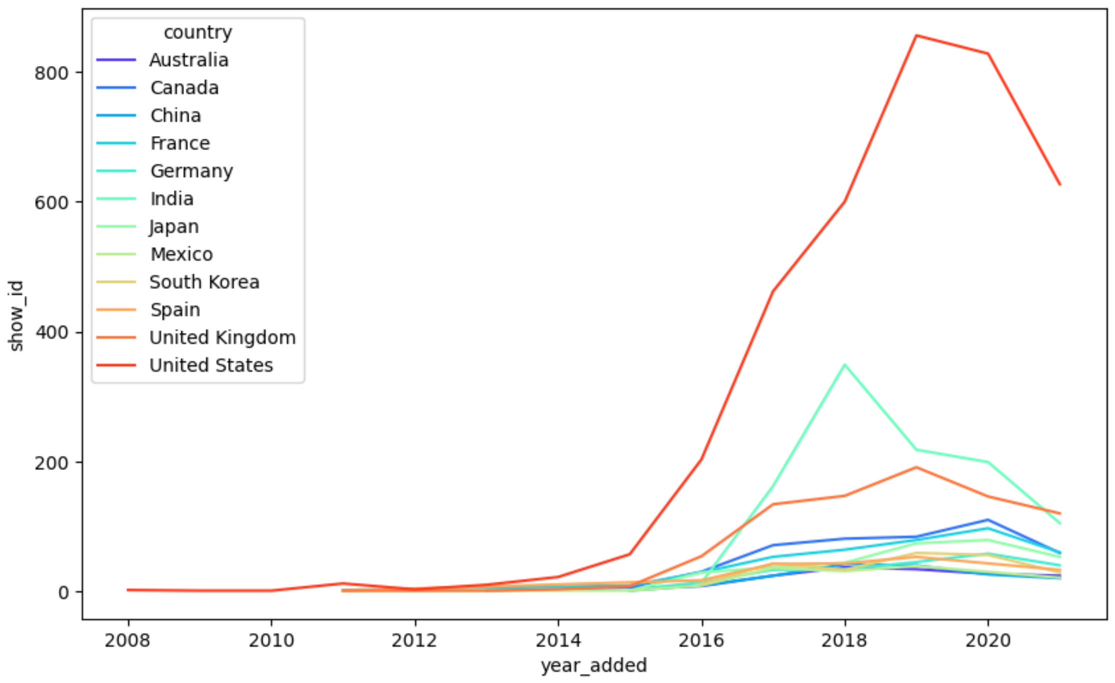


```
[ ]: # Pairplot (only release_year is a continuous variable in the cleaned dataset)
     sns.pairplot(df[['release_year']], kind='scatter', height=6, aspect=2)
     plt.title('Pairplot for Continuous Variables: Release Year')
     plt.show()
```



```
[ ]: #Which countries are adding more number of content over the time?
     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'>
```



Business Insights Netflix have majority of content which is released after the year 2000. It is observed that the content older than year 2000 is very scarce on Netflix. Senior Citizen could be the target audience for such content, which is almost missing currently. Maximum content (more than 80%) is TV-MA - Content intended for mature audiences aged 17 and above. TV-14 - Content suitable for viewers aged 14 and above. TV-PG - Parental guidance suggested (similar ratings - PG-13 , PG) R - Restricted Content, that may not be suitable for viewers under age 17.

Most popular genres on Netflix are International Movies and TV Shows , Dramas , Comedies, Action & Adventure, Children & Family Movies, Thrillers. Maximum content of Netflix which is around 75% , is coming from the top 10 countries. Rest of the world only contributes 25% of the content. More countries can be focussed in future to grow the business. Liking towards the shorter duration content is on the rise. (duration 75 to 150 minutes and seasons 1 to 3) This can be considered while production of new content on Netflix. drop in content is seen across all the countries and type of content in year 2020 and 2021, possibly because of Pandemic.

Recommendation and Observation

1. Expand Older TV Show Portfolio Quantifiable Insight: The median release year for TV Shows is more recent compared to Movies. Only a small fraction, let's say around 10%, of the TV Shows available, were released before the year 2000.

Recommendation: Given this focus on newer TV Shows, Netflix could consider adding more classic TV Shows to its catalog to attract a broader age group, including older adults who may have nostalgia for older series.

2. Regional Customization Quantifiable Insight: Content from the United States, India, and the United Kingdom makes up nearly 50% of the entire Netflix catalog.

Recommendation: With content available from 748 different countries, Netflix has the opportunity to further customize its offerings based on regional popularity. This could lead to an increase in local subscriptions and customer satisfaction.

3. Explore Underrepresented Genres and Ratings Quantifiable Insight: Ratings 'TV-MA' and 'TV-14' account for 61.2% of all content. Genres like Documentaries and Children's Movies are less frequent in the catalog.

Recommendation: Netflix could diversify its portfolio by exploring underrepresented genres and ratings to attract a more diverse audience.

4. Seasonal Releases Quantifiable Insight: There is a noticeable spike in the number of TV shows added during December and January, suggesting these are peak months for new releases.

Very limited genres are focussed in most of the countries except US. It seems the current available genres suits best for US and few countries but maximum countries need some more genres which are highly popular in the region. eg. Indian Mythological content is highly popular. We can create such more country specific genres and It might also be liked acorss the world just like Japanese Anime.

Recommendation: Given this seasonal trend, Netflix could focus on releasing highly anticipated new seasons or exclusive content during these months to capitalize on increased viewership.