netflix-analysis

July 13, 2024

Business Problem

Netflix

Netflix is one of the most popular media and video streaming platforms. They have over 8000 movies or tv shows available on their platform, as of mid-2021, they have over 200M Subscribers globally. This tabular dataset consists of listings of all the movies and tv shows available on Netflix, along with details such as - cast, directors, ratings, release year, duration, etc.

The dataset consists of a list of all the TV shows/movies available on Netflix:

- 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 or number of seasons
- Listed_in: Genre
- Description: The summary description

Objectives of the Project

- Perform EDA on the given dataset and find insights.
- Provide Useful Insights and Business recommendations that can help the business to grow.

Importing libraries

[24]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import gdown as gd
```

[25]:
```
pip install -U gdown
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages
(5.2.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
packages (from gdown) (4.12.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from gdown) (3.15.4)
Requirement already satisfied: requests[socks] in
/usr/local/lib/python3.10/dist-packages (from gdown) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from gdown) (4.66.4)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-
packages (from beautifulsoup4->gdown) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests[socks]->gdown) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2024.7.4)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in
/usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (1.7.1)
```

Loading the data

[26]:
```
! gdown 1za1rtlhwcWTIq34yfxq_Xz9ZJovFEoHO
```

```
Downloading…
From: https://drive.google.com/uc?id=1za1rtlhwcWTIq34yfxq_Xz9ZJovFEoHO
To: /content/netflix.csv
100% 3.40M/3.40M [00:00<00:00, 64.7MB/s]
```

[27]:
```
df = pd.read_csv('netflix.csv')
```

1.Basic Obervation

[28]:
```
df.head()
```

[28]:
```
  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…
```

These are the first 5 rows of the dataset.

[29]:
```
df.shape
```

[29]:
```
(8807, 12)
```

[30]:
```
df.ndim
```

[30]:
```
2
```

Netflix dataset, there are 8807 rows and 12 columns.

[31]:
```
df.tail()
```

[31]:
```
     show_id     type        title         director  \
8802   s8803    Movie       Zodiac    David Fincher
8803   s8804  TV Show  Zombie Dumb              NaN
8804   s8805    Movie   Zombieland  Ruben Fleischer
8805   s8806    Movie         Zoom     Peter Hewitt
8806   s8807    Movie       Zubaan      Mozez Singh
```

```
                                               cast        country  \
8802  Mark Ruffalo, Jake Gyllenhaal, Robert Downey J…  United States
8803                                                NaN            NaN
8804  Jesse Eisenberg, Woody Harrelson, Emma Stone, …  United States
8805  Tim Allen, Courteney Cox, Chevy Chase, Kate Ma…  United States
8806  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan…          India

             date_added  release_year rating   duration  \
8802  November 20, 2019          2007      R    158 min
8803       July 1, 2019          2018  TV-Y7  2 Seasons
8804   November 1, 2019          2009      R     88 min
8805   January 11, 2020          2006     PG     88 min
8806      March 2, 2019          2015  TV-14    111 min

                                          listed_in  \
8802                  Cult Movies, Dramas, Thrillers
8803          Kids' TV, Korean TV Shows, TV Comedies
8804                         Comedies, Horror Movies
8805              Children & Family Movies, Comedies
8806  Dramas, International Movies, Music & Musicals

                                             description
8802  A political cartoonist, a crime reporter and a…
8803  While living alone in a spooky town, a young g…
8804  Looking to survive in a world taken over by zo…
8805  Dragged from civilian life, a former superhero…
8806  A scrappy but poor boy worms his way into a ty…
```

[32]:
```
df.columns
```

[32]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

[33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8807 non-null   object
 1   type          8807 non-null   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
```

- Some columns have missing values (non-null count less than 8807).
- The data types include object (for text/string data) and int64 (for integer data).

[34]:
```
df.describe()
```

[34]:
```
       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
```

[35]:
```
df.describe(include=object)
```

[35]:
```
       show_id   type                 title       director  \
count     8807   8807                  8807           6173
unique    8807      2                  8807           4528
top         s1  Movie  Dick Johnson Is Dead  Rajiv Chilaka
freq         1   6131                     1             19

                      cast        country       date_added rating  duration  \
count                 7982           7976             8797   8803      8804
unique                7692            748             1767     17       220
top     David Attenborough  United States  January 1, 2020  TV-MA  1 Season
freq                    19           2818              109   3207      1793

                           listed_in  \
count                           8807
unique                           514
top     Dramas, International Movies
freq                             362

                                              description
count                                                8807
unique                                               8775
top     Paranormal activity at a lush, abandoned prope…
```

```
freq                                                       4
```

2.Data Cleaning

[36]:
```
df.isnull().sum().sort_values(ascending=False)
```

[36]:
```
director        2634
country          831
cast             825
date_added        10
rating             4
duration           3
show_id            0
type               0
title              0
release_year       0
listed_in          0
description        0
dtype: int64
```

[37]:
```
round(df.isnull().sum()/df.shape[0]*100,2).sort_values(ascending=False)
```

[37]:
```
director        29.91
country          9.44
cast             9.37
date_added       0.11
rating           0.05
duration         0.03
show_id          0.00
type             0.00
title            0.00
release_year     0.00
listed_in        0.00
description      0.00
dtype: float64
```

- In this dataset **director** the highest percentage of missing values at **29.91%**.
- **country** and **cast** also have notable percentages of missing values, with **9.44%** and **9.37%** .
- **date_added**, **rating**, and **duration** have very low percentages of missing values .

[38]:
```
indx = df[df['duration'].isna()]
```

[39]:
```
indx = df[df['duration'].isna()].index
df.loc[indx] = df.loc[indx].fillna(method = 'ffill' , axis = 1)
df.loc[indx ,'rating'] = 'Not Available'
df.loc[indx]
```

[39]:
```
     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…
```

[40]:
```
df[df.rating.isna()]
indices = df[df.rating.isna()].index
indices
```

[40]:
```
Index([5989, 6827, 7312, 7537], dtype='int64')
```

[41]:
```
df.loc[indices , 'rating'] = 'Not Available'
df.loc[indices]
```

[41]:
```
     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…
```

[42]:
```python
df.drop(df.loc[df['date_added'].isna()].index , axis = 0 , inplace = True)
```

[43]:
```python
df['date_added'].value_counts()
```

[43]:
```
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
```

[44]:
```python
df.isnull().sum().sort_values(ascending=False)
```

[44]:
```
director        2624
country          830
cast             825
show_id            0
type               0
title              0
date_added         0
release_year       0
rating             0
duration           0
listed_in          0
description        0
dtype: int64
```

For 'date_added' column, all values confirm to date format, So we can convert its data type from object to datetime

[45]:
```
df['date_added'] = pd.to_datetime(df['date_added'], format='%B %d, %Y',␣
↪errors='coerce')
# The 'errors=coerce' argument will handle any dates that don't match the␣
↪format by setting them to NaT (Not a Time)
df['date_added']
```

[45]:
```
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]
```

We can add the new column 'year_added' by extracting the year from 'date_added' column

[46]:
```
df['year_added'] = df['date_added'].dt.year
```

Similar way, We can add the new column 'month_added' by extracting the month from 'date_added' column

[47]:
```
df['month_added'] = df['date_added'].dt.month
```

[48]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8797 entries, 0 to 8806
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8797 non-null   object
 1   type          8797 non-null   object
 2   title         8797 non-null   object
 3   director      6173 non-null   object
 4   cast          7972 non-null   object
 5   country       7967 non-null   object
 6   date_added    8709 non-null   datetime64[ns]
 7   release_year  8797 non-null   object
 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
 12  year_added    8709 non-null   float64
 13  month_added   8709 non-null   float64
dtypes: datetime64[ns](1), float64(2), object(11)
memory usage: 1.0+ MB
```

3. Non-Graphical Analysis

```
[49]: df['type'].unique()
```

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

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

```
[51]: movies.duration.value_counts()
```

```
[51]: 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
```

```
[52]: tv_shows.duration.value_counts()
```

```
[52]: 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

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

```
[54]: movies['duration']
```

```
[54]: 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, Length: 6131, dtype: float64
```

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

```
[56]: tv_shows['duration']
```

```
[56]: 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, Length: 2666, dtype: float64
```

The oldest and the most recent movie/TV show released on the Netflix in which year?

```
[57]: df.release_year.min() , df.release_year.max()
```

```
[57]: (1925, 2021)
```

[58]:
```
df['country'].value_counts()
```

[58]:
```
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.

[59]:
```
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
```

[59]:
```
     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

[60]:
```
country_tb['country'] = country_tb['country'].str.strip()
```

[61]:
```
country_tb = country_tb.loc[country_tb['country'] != '']
```

[62]:
```
country_tb['country'].nunique()
```

[62]:
```
122
```

Netflix has movies from the total 122 countries.

[63]:
```
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)
```

[63]:
```
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]
```

[64]:
```
df['director'].value_counts()
```

[64]:
```
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.

[65]:
```
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
```

[65]:
```
     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]
```

[66]:
```
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()
```

[66]:
```
director
False    6978
Name: count, dtype: int64
```

[67]:
```
dir_tb['director'].nunique()
```

[67]:
```
4993
```

There are total 4993 unique directors in the dataset. Total movies and tv shows directed by each director

[68]:
```
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)
```

[68]:
```
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]
```

[69]:
```
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
```

[69]:
```
     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]
```

[70]:
```
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()
```

[71]:
```
genre_tb
```

[71]:
```
     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]
```

[72]:
```
cast_tb['cast'] = cast_tb['cast'].str.strip()
```

[73]:
```
cast_tb[cast_tb['cast'] == '']
```

[73]:
```
Empty DataFrame
Columns: [show_id, type, cast]
Index: []
```

[74]:
```
cast_tb.cast.nunique()
```

[74]:
```
36403
```

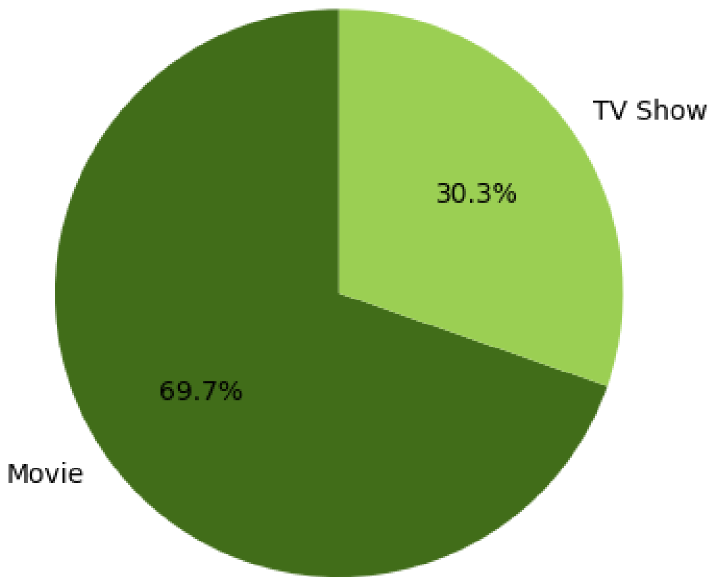
[75]:
```
x = cast_tb.groupby(['cast' , 'type'])['show_id'].count().reset_index()
x.pivot(index = 'cast' , columns = 'type' , values = 'show_id').sort_values('TV␣
↪Show' , ascending = False)
```

[75]:
```
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]
```

4. Visual Analysis - Univariate & Bivariate

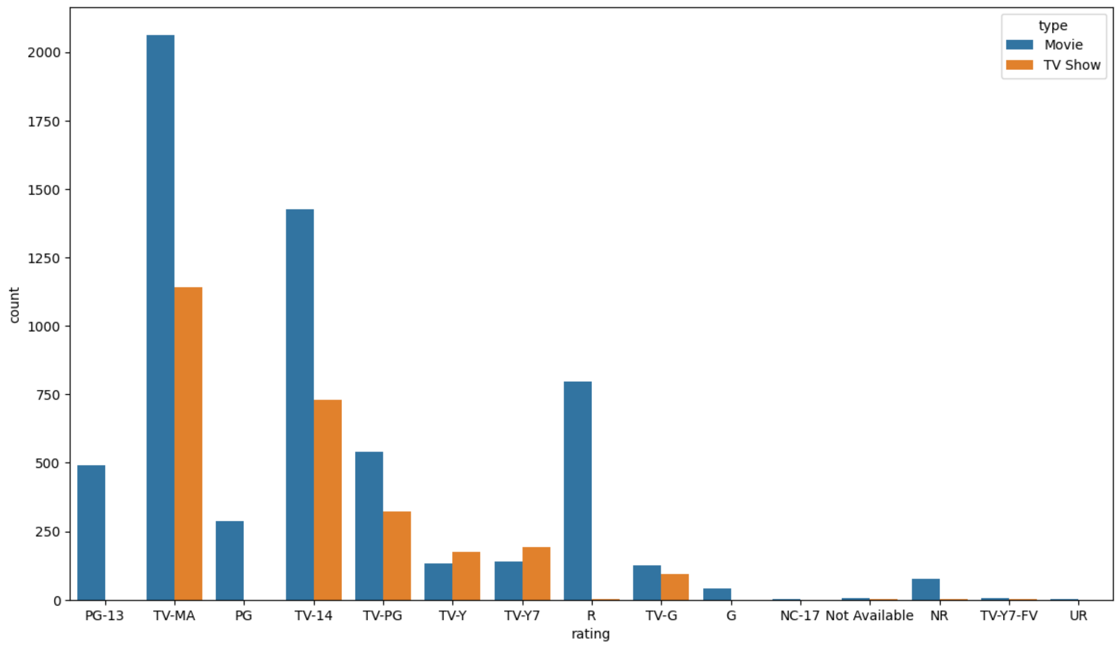
[76]:
```
movie_df = df[df['type'] == 'Movie']

tvshow_df = df[df['type'] == 'TV Show']
labels = ['Movie', 'TV Show']
sizes = [len(movie_df), len(tvshow_df)]
colors = ['#416D19', '#9BCF53']
plt.pie(sizes, labels=labels, autopct="%1.1f%%", colors=colors, startangle=90)
```

```
[76]: ([<matplotlib.patches.Wedge at 0x789bf842c8e0>,
        <matplotlib.patches.Wedge at 0x789bf842c820>],
       [Text(-0.896088761515472, -0.637985055848229, 'Movie'),
        Text(0.896088761515472, 0.6379850558482287, 'TV Show')],
       [Text(-0.4887756880993483, -0.34799184864448846, '69.7%'),
        Text(0.4887756880993483, 0.3479918486444884, '30.3%')])
```



```
[77]: plt.figure(figsize = (14,8))
      sns.countplot(x='rating',data = df,hue='type')
```

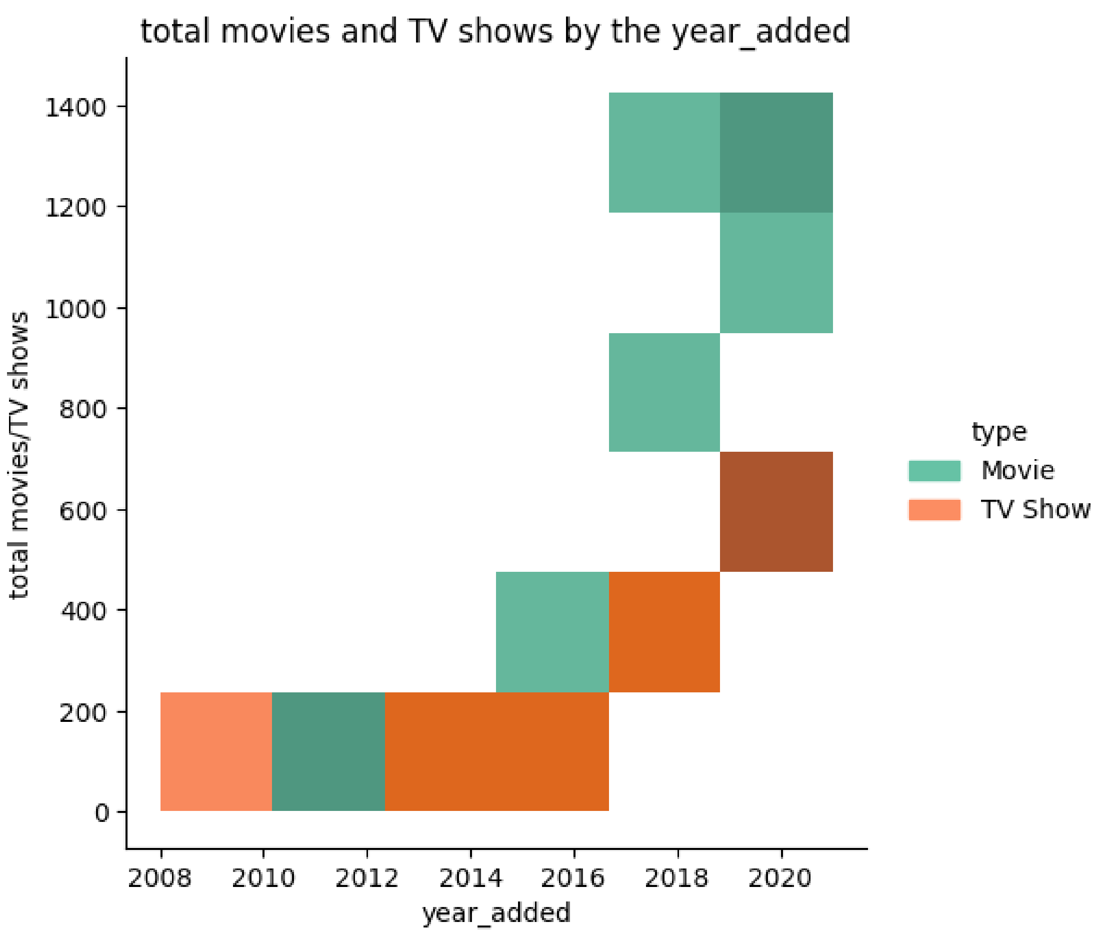
```
[77]: <Axes: xlabel='rating', ylabel='count'>
```



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

```
[79]: plt.figure(figsize = (12,6))
      sns.displot(data = d, x = 'year_added' , y = 'total movies/TV shows' ,hue =␣
       ↪'type', legend=True,palette='Set2')
      plt.title('total movies and TV shows by the year_added' , fontsize = 12)
      plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



```
[80]: 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)
```

```
[80]:               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
```

```
[81]: 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)
```

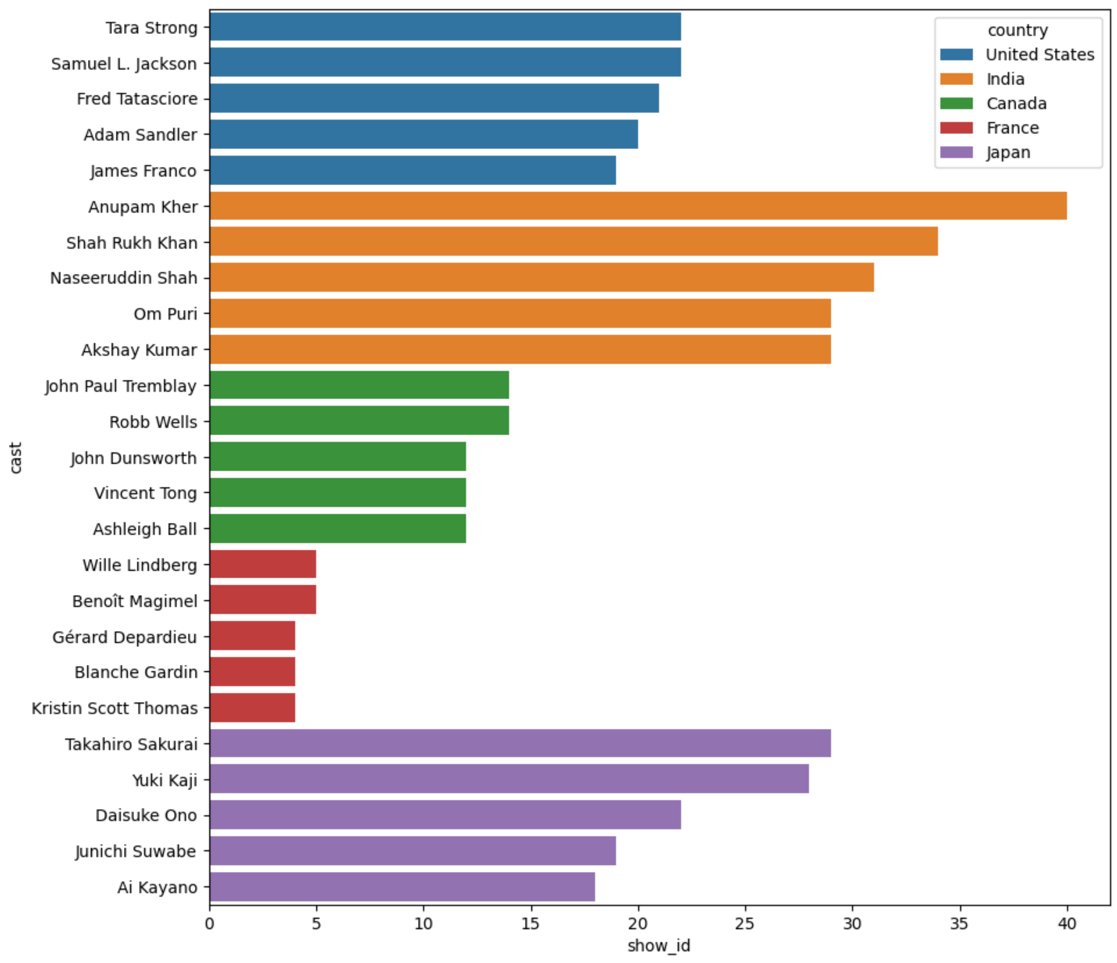
```
    top_5_actors = pd.concat( [top_5_actors , new] , ignore_index = True)
```

[82]:
```
top_5_actors
```

[82]:
```
          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         Canada   John Paul Tremblay       14
11         Canada          Robb Wells       14
12         Canada      John Dunsworth       12
13         Canada        Vincent Tong       12
14         Canada       Ashleigh Ball       12
15         France      Wille Lindberg        5
16         France      Benoît Magimel        5
17         France     Gérard Depardieu        4
18         France       Blanche Gardin        4
19         France  Kristin Scott Thomas        4
20          Japan    Takahiro Sakurai       29
21          Japan           Yuki Kaji       28
22          Japan        Daisuke Ono       22
23          Japan      Junichi Suwabe       19
24          Japan           Ai Kayano       18
```

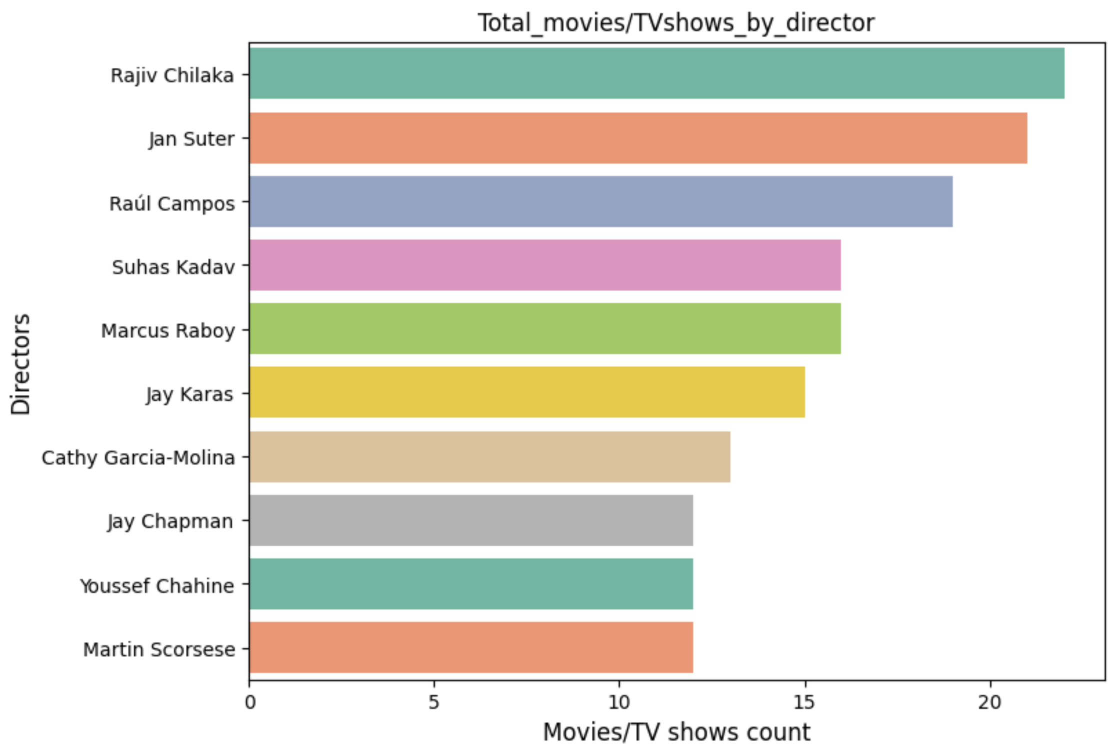
[83]:
```
plt.figure(figsize = (10,10))
sns.barplot(data = top_5_actors , y = 'cast' , x = 'show_id' , hue = 'country')
```

[83]:
```
<Axes: xlabel='show_id', ylabel='cast'>
```



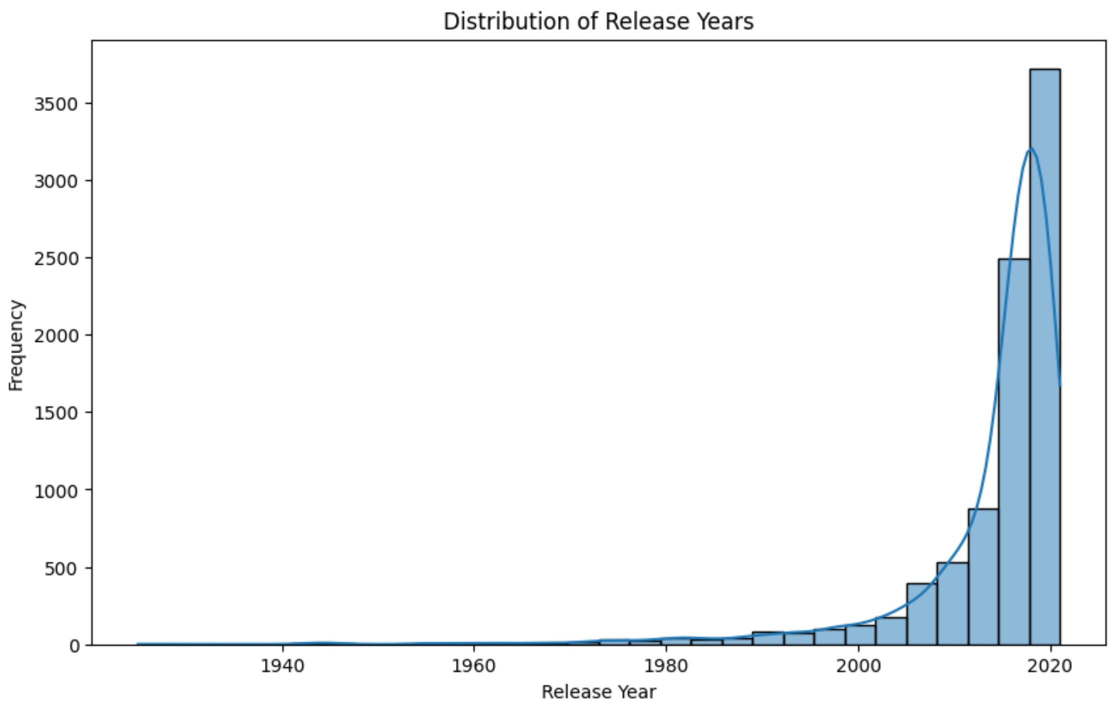
[84]:
```
top_10_dir = dir_tb.director.value_counts().head(10).index
df_new = dir_tb.loc[dir_tb['director'].isin(top_10_dir)]
```

[85]:
```
plt.figure(figsize= (8 , 6))
sns.countplot(data = df_new , y = 'director' , order = top_10_dir , orient =␣
↪'v',palette='Set2')
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()
```



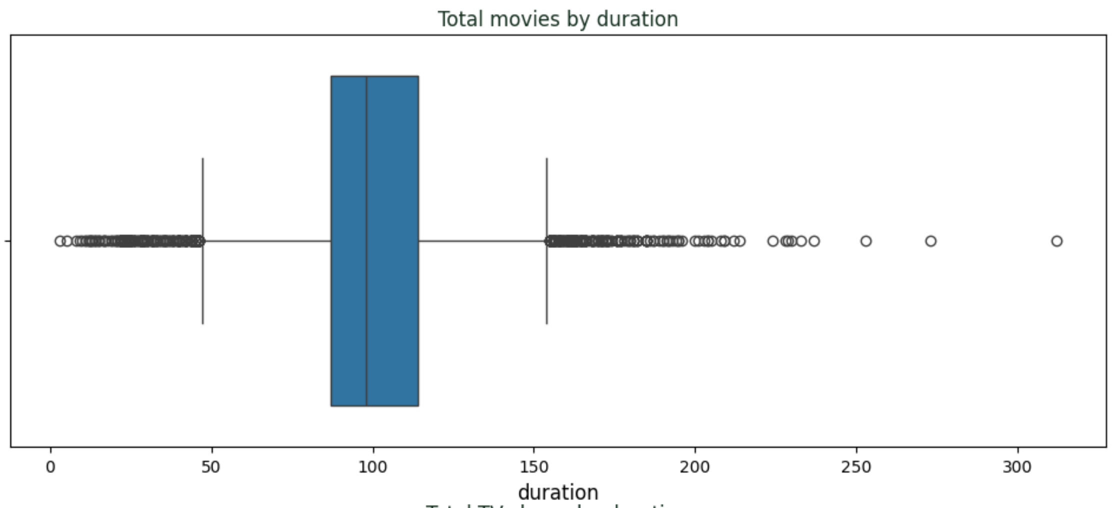
The **top 3 directors** on Netflix in terms of count of movies directed by them are - **Rajiv Chilaka, Jan Suter, Raúl Campos**

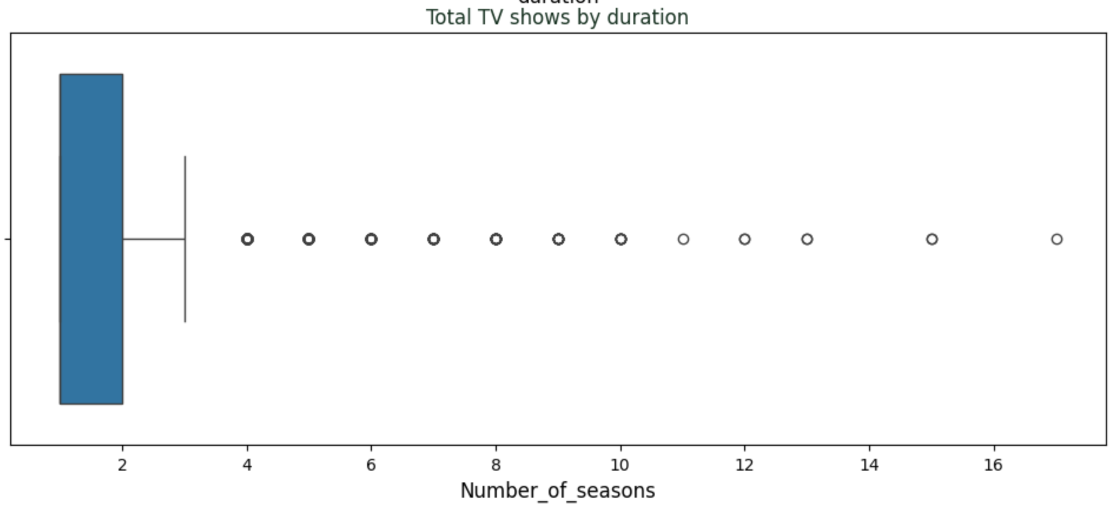
[86]:
```
plt.figure(figsize=(10, 6))
plt.title('Distribution of Release Years')
sns.histplot(df['release_year'], bins=30, kde=True)
plt.xlabel('Release Year')
plt.ylabel('Frequency')
plt.show()
```



```
[87]: fig, ax = plt.subplots(2,1, figsize=(12,10))
      sns.boxplot (data = movies , x = 'duration' ,ax =ax[0])
      ax[0].set_xlabel('duration' , fontsize = 12)
      ax[0].set_title('Total movies by duration',color='#163020')
      sns.boxplot (data = tv_shows , x = 'duration' , ax = ax[1])
      ax[1].set_xlabel('Number_of_seasons' , fontsize = 12)
      ax[1].set_title('Total TV shows by duration',color='#163020')
```

```
[87]: Text(0.5, 1.0, 'Total TV shows by duration')
```





- 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

[88]:
```
df.columns
```

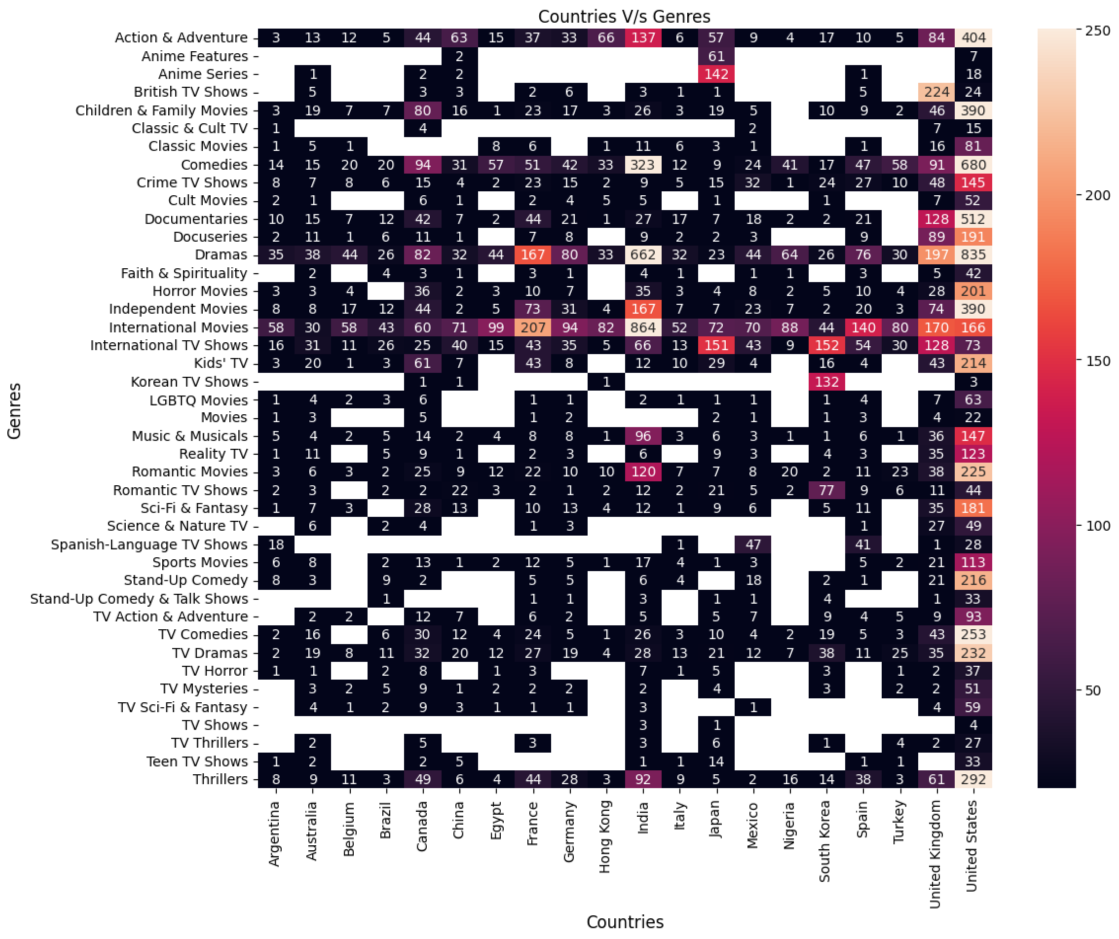
[88]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description',
       'year_added', 'month_added'],
      dtype='object')
```

[89]:
```
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)
```

[89]:

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

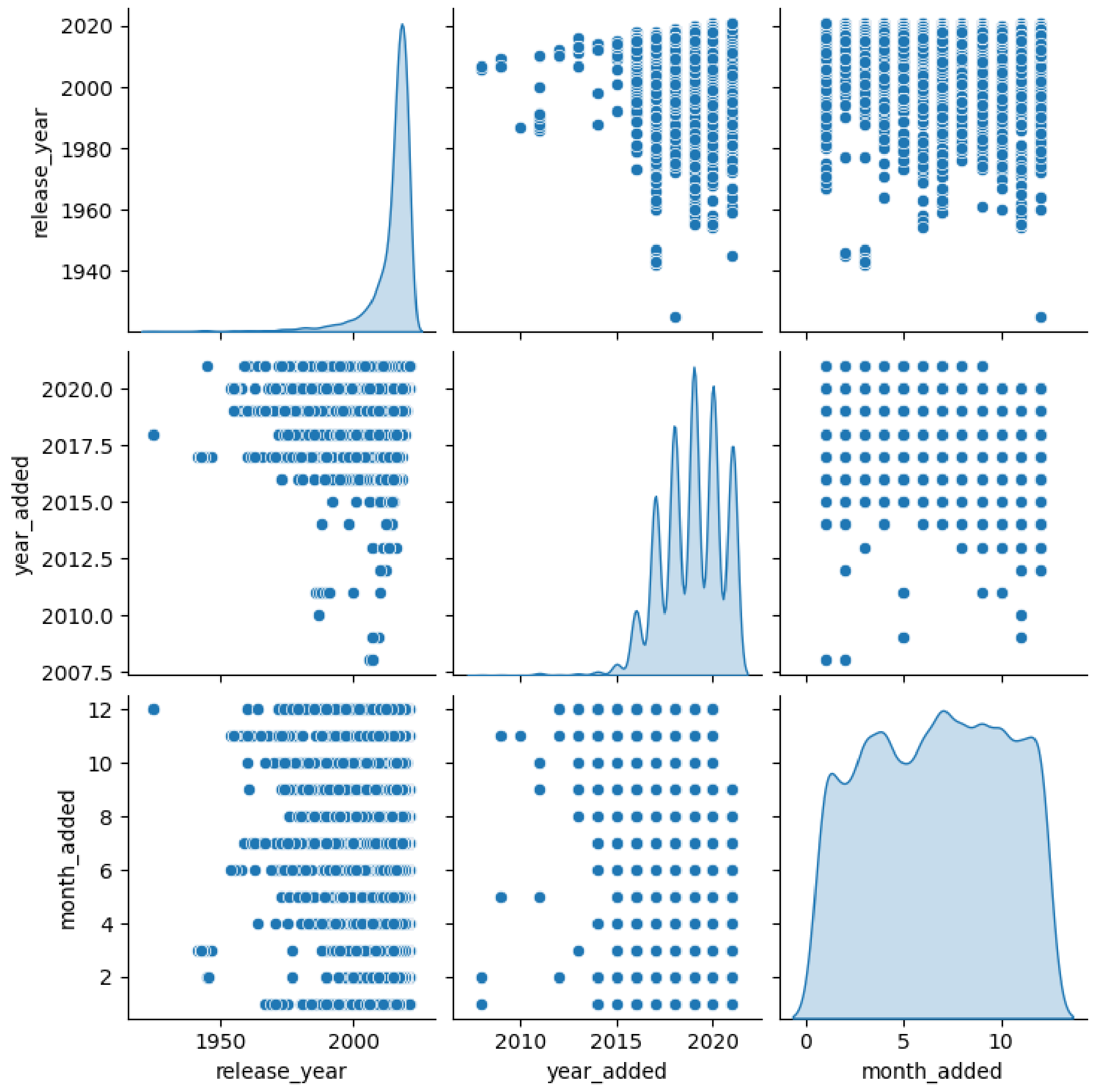


[90]:

```
sns.pairplot(df,diag_kind='kde')
```

[90]:

```
<seaborn.axisgrid.PairGrid at 0x789bf82ef0a0>
```



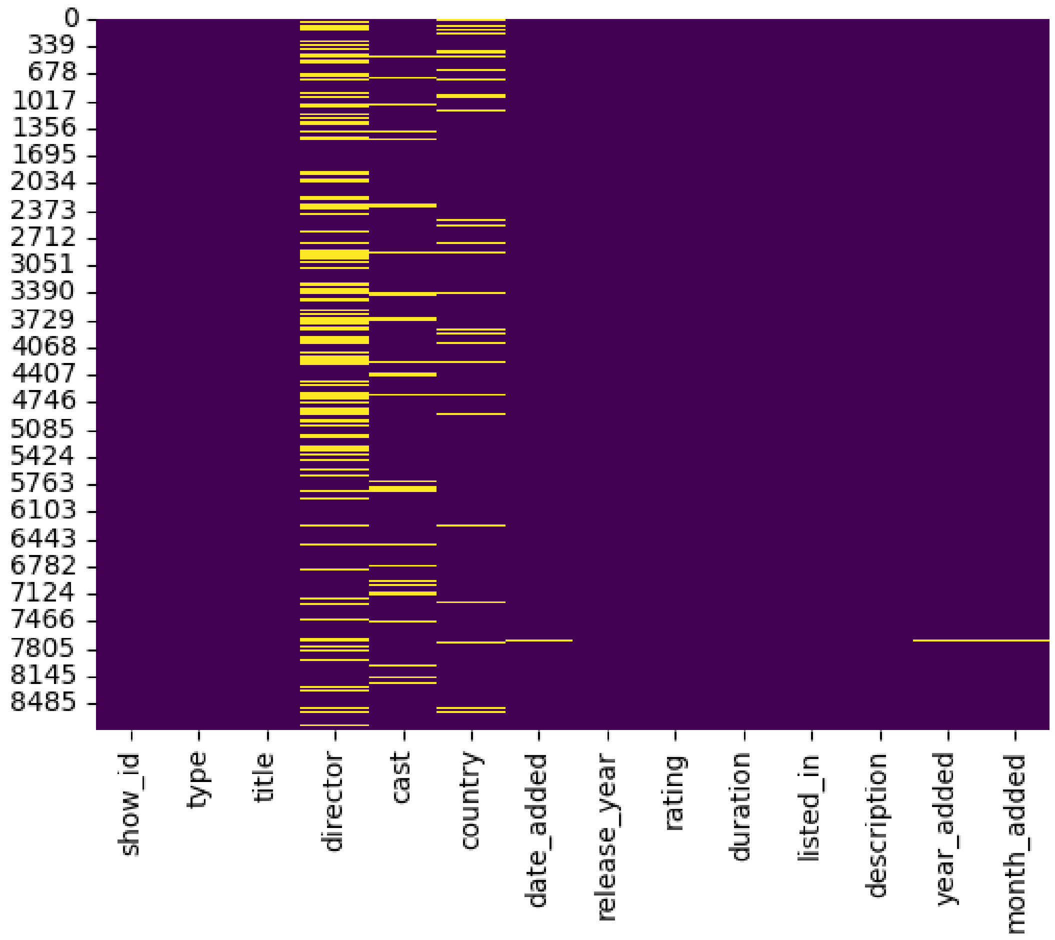
5. Missing Value & Outlier check

[91]:
```
df.isnull().sum().sort_values(ascending=False)
```

[91]:
```
director        2624
country          830
cast             825
date_added        88
year_added        88
month_added       88
show_id            0
type               0
title              0
```

```
release_year       0
rating             0
duration           0
listed_in          0
description        0
dtype: int64
```

[92]:
```
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.show()
```



[93]:
```
df.director.fillna("No Director", inplace=True)
df.cast.fillna("No Cast", inplace=True)
df.country.fillna("Country Unavailable", inplace=True)
```

[94]:
```
round(df.isnull().sum()/df.shape[0]*100,2).sort_values(ascending=False)
```

```
[94]: date_added      1.0
      year_added      1.0
      month_added     1.0
      show_id         0.0
      type            0.0
      title           0.0
      director        0.0
      cast            0.0
      country         0.0
      release_year    0.0
      rating          0.0
      duration        0.0
      listed_in       0.0
      description     0.0
      dtype: float64
```

```
[95]: df['year_added'].fillna(0, inplace=True)
      df['month_added'].fillna(0, inplace=True)
      df['date_added'].fillna(0, inplace=True)
```

```
[96]: df.isnull().sum().sort_values(ascending=False)
```

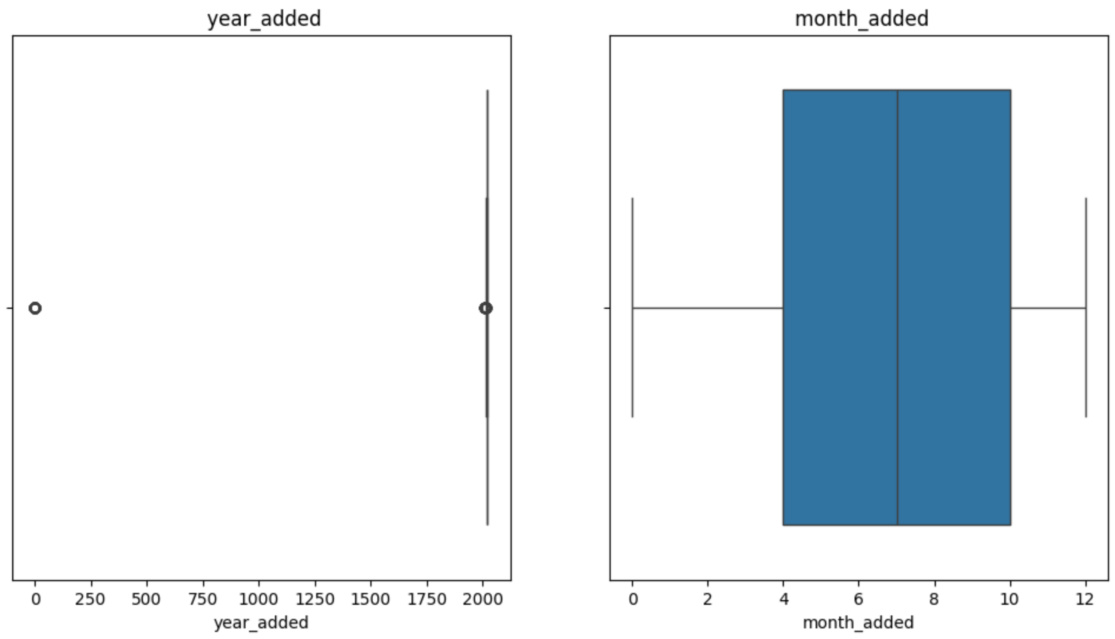
```
[96]: show_id         0
      type            0
      title           0
      director        0
      cast            0
      country         0
      date_added      0
      release_year    0
      rating          0
      duration        0
      listed_in       0
      description     0
      year_added      0
      month_added     0
      dtype: int64
```

```
[97]: df.describe()
```

```
[97]:         year_added  month_added
      count  8797.000000  8797.000000
      mean   1998.692168     6.586791
      std     200.928193     3.477842
      min       0.000000     0.000000
      25%    2018.000000     4.000000
      50%    2019.000000     7.000000
```

```
75%     2020.000000    10.000000
max     2021.000000    12.000000
```

[98]:
```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.boxplot(x=df['year_added'])
plt.title(' year_added')
plt.subplot(1, 2, 2)
sns.boxplot(x=df['month_added'])
plt.title(' month_added')

plt.show()
```



6.Insights based on Non-Graphical and Visual Analysis

- The dataset contains a mix of categorical and numerical attributes. Categorical attributes include 'type', 'country', and 'rating', providing information about the nature, production location, and audience rating of shows/movies.
- Numerical attributes include 'year_added' and 'month_added', indicating the year and month when content was added to Netflix.
- The 'duration' attribute represents the total duration in minutes or the number of seasons for TV shows.

- The distribution of 'year_added' reveals a steady increase in the number of shows/movies added to Netflix over time, suggesting continuous platform growth.
- The relationship between 'type' and 'country' can be analyzed to understand the distribution of movie and TV show content in different regions, potentially uncovering preferences or

production patterns.

- In univariate plots, a histogram of ‘duration’ can highlight the distribution of content duration, indicating whether most content is short-form or long-form.

- Bivariate plots, such as a heat map is the distribution of shows and movies across the top 20 countries with respect to different genres on Netflix. Each cell in the heatmap represents the count of content in a specific genre for a given country. The color intensity indicates the magnitude of content presence, with darker shades representing higher counts

7. **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.
- These ratings’ movies target Matured and Adult audience. Rest 20 % of the content is for kids aged below 13.
- It shows that Netflix is currently serving mostly Mature audiences or Children with parental guidance.
- 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.

8. **Recommendations**

- Netflix has to focus on TV Shows also because there are people who will like to see tv shows rather than movies.
- By approaching the top director we can plan some more movies/tv shows in order to increase the popularity
- Not only reaching top director we can also see the director with less no of movies and having high rating as there may be some financial issues or anything so inorder to get good content netflix can reach to them and netflix can produce the movie and give the director a chance.
- We have seen most no of international movies genre so need to give priority to other geners like hooro,comedy..etc
- In TV Shows we may focus on thriller genre which will be helpfull for having more no of seasons
- Most of the movies released in ott is in a year 2019 so we need to go on increasing this value in order to attract people by showing that getting subscription is usefull as netflix is releasing more movies per year
- Mainly the release in ott should focus on the festival holidays, year end and week ends which is to be mainly focussed

- Some movies can be released directly into ott which has some positive talk which may help in improving subscriptions
- Should focus on a actor who has immense following and make use of it by doing a TV Shows or web series
- Advertisement in the country which has very less movies released should be increased and attract people of that country by making their native TV Shows