BusinessCaseStudy_Netflix

June 27, 2024

1 Business Problem

Analyze the data and generate insights that could help Netflix in deciding which type of shows/movies to produce and how they can grow the business in different countries.

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

2 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.

3 1. Importing Libraries , loading the Netflix dataset and Basic Observations

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
!wget "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/
  ↪original/netflix.csv" -O netflix.csv
df = pd.read_csv('netflix.csv')
df.head()
```

```
--2024-06-26 04:54:56--  https://d2beiqkhq929f0.cloudfront.net/public_assets/ass
ets/000/000/940/original/netflix.csv
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)…
18.239.15.127, 18.239.15.217, 18.239.15.11, …
Connecting to d2beiqkhq929f0.cloudfront.net
(d2beiqkhq929f0.cloudfront.net)|18.239.15.127|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 3399671 (3.2M) [text/plain]
Saving to: ‘netflix.csv’

netflix.csv         100%[===================>]   3.24M  3.56MB/s    in 0.9s

2024-06-26 04:54:58 (3.56 MB/s) - ‘netflix.csv’ saved [3399671/3399671]
```

[1]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi…	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act…
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K…	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV …

```
                                                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 top 5 rows of the dataset shown above. The actual size of the dataset is given below. Total 8807 rows and 12 columns.

[2]:
```
df.shape
```

[2]:
```
(8807, 12)
```

[3]:
```
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
```

[4]:
```
df.nunique()
```

[4]:
```
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 this dataset. We can observe that show_id column has all unique values, title column has all unique values i.e. total 8807 which matches with total rows in the dataset. Hence we can initially concluded that , total 8807 movies/TV shows data is provided in the dataset.

[5]:
```
df.describe()
```

[5]:
```
       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
```

[6]:
```
df.describe(include = object)
```

[6]:
```
       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
```

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.

4 2. Data Cleaning

Overall null values in each column of the dataset -

[7]:
```
df.isna().sum()
```

[7]:
```
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

[8]:
```
df[df['duration'].isna()]
```

[8]:
```
     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…
```

[9]:
```
ind = df[df['duration'].isna()].index
```

[11]:
```
df.loc[ind] = df.loc[ind].fillna(method = 'ffill' , axis = 1)
```

[12]:
```
df.loc[ind ,'rating'] = 'Not Available' # replace the incorrect entries done in␣
 ↪the rating column
```

[13]:
```
df.loc[ind]
```

[13]:
```
     show_id   type                                title    director  \
5541   s5542  Movie                       Louis C.K. 2017  Louis C.K.
5794   s5795  Movie                   Louis C.K.: Hilarious  Louis C.K.
5813   s5814  Movie  Louis C.K.: Live at the Comedy Store  Louis C.K.

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

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

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

- Fill the null values in rating column

[14]:
```
df[df.rating.isna()]
```

[14]:
```
     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…
```

[15]:
```
indices = df[df.rating.isna()].index
indices
```

[15]:
```
Index([5989, 6827, 7312, 7537], dtype='int64')
```

[16]:
```
df.loc[indices , 'rating'] = 'Not Available'
```

[17]:
```
df.loc[indices]
```

[17]:
```
     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…
```

[18]:
```
df.rating.unique()
```

[18]:
```
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.

[19]:
```
df.loc[df['rating'] == 'UR' , 'rating'] = 'NR'
df.rating.value_counts()
```

[19]:
```
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
```

- Dropping the null from date_added column

[28]:
```
df.drop(df.loc[df['date_added'].isna()].index , axis = 0 , inplace = True)
```

[29]:
```
df['date_added'].value_counts()
```

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

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

[23]:
```
df['date_added'] = df['date_added'].str.strip() # striping te column of leading␣
 ↪and trailing white spaces, since such records are availble.
df['date_added'] = pd.to_datetime(df['date_added'])
df['date_added']
```

[23]:
```
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: 8807, dtype: datetime64[ns]
```

We can add the new column 'year_added' by extracting the year from 'date_added' column. Also new column 'month_added' by extracting the month from 'date_added' column

[24]:
```
df['year_added'] = df['date_added'].dt.year
```

[25]:
```
df['month_added'] = df['date_added'].dt.month
```

[26]:
```
df[['date_added' , 'year_added' , 'month_added']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 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   float64
 2   month_added  8797 non-null   float64
dtypes: datetime64[ns](1), float64(2)
memory usage: 206.5 KB
```

[30]:
```
# total null values in each column
df.isna().sum()
```

[30]:
```
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
```

Getting the percentage of null values in each column

[31]:
```
round((df.isna().sum()/ df.shape[0])*100)
```

[31]:
```
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
```

5 Observations after cleaning the data

We can see that, after cleaning some data we still have null values in 3 columns. These are much higher in numbers. For some records: - director names are missing (30%) - cast is missing (9%) - country is missing. (9%)

6 3. Data Exploration and Non Graphical Analysis

[32]:
```
# Two types of content present in dataset - either Movie or TV Show
df['type'].unique()
```

[32]:
```
array(['Movie', 'TV Show'], dtype=object)
```

[33]:
```
movies  = df.loc[df['type'] == 'Movie']
tv_shows = df.loc[df['type'] == 'TV Show']
```

[34]:
```
movies.duration.value_counts()
```

[34]:
```
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
```

[35]:
```
tv_shows.duration.value_counts()
```

[35]:
```
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

[37]:
```
movies['duration'] = movies['duration'].str[:-3]
movies['duration'] = movies['duration'].astype('float')
```

```
<ipython-input-37-53ddd658a0dd>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies['duration'] = movies['duration'].str[:-3]
<ipython-input-37-53ddd658a0dd>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies['duration'] = movies['duration'].astype('float')
```

[39]:
```
tv_shows['duration'] = tv_shows.duration.str[:-7].apply(lambda x : x.strip())
tv_shows['duration'] = tv_shows['duration'].astype('float')
```

```
<ipython-input-39-8490b01afea6>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  tv_shows['duration'] = tv_shows.duration.str[:-7].apply(lambda x : x.strip())
<ipython-input-39-8490b01afea6>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  tv_shows['duration'] = tv_shows['duration'].astype('float')
```

[40]:
```
tv_shows.rename({'duration': 'duration_in_seasons'} ,axis = 1 , inplace = True)
movies.rename({'duration': 'duration_in_minutes'} ,axis = 1 , inplace = True)
```

```
<ipython-input-40-6fba49e9528a>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  tv_shows.rename({'duration': 'duration_in_seasons'} ,axis = 1 , inplace =
True)
<ipython-input-40-6fba49e9528a>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies.rename({'duration': 'duration_in_minutes'} ,axis = 1 , inplace = True)
```

[41]:
```
movies.duration_in_minutes
```

[41]:
```
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
```

[42]:
```
tv_shows.duration_in_seasons
```

[42]:
```
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
```

When was first movie added on netflix and when is the most recent movie added on netflix as per data i.e. dataset duration

[43]:
```
timeperiod = pd.Series((df['date_added'].min().strftime('%B %Y') ,␣
↪df['date_added'].max().strftime('%B %Y')))
timeperiod.index = ['first' , 'Most Recent']
timeperiod
```

[43]:
```
first            January 2008
Most Recent    September 2021
dtype: object
```

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

[44]:
```
df.release_year.min() , df.release_year.max()
```

[44]:
```
(1925, 2021)
```

[45]:
```
df.loc[(df.release_year == df.release_year.min()) | (df.release_year == df.
↪release_year.max())].sort_values('release_year')
```

[45]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration
4250	s4251	TV Show	Pioneers: First Women Filmmakers*	NaN	NaN	NaN	2018-12-30	1925	TV-14	1 Season
966	s967	Movie	Get the Grift	Pedro Antonio	Marcus Majella, Samantha Schmütz, Caito Mainie…	Brazil	2021-04-28	2021	TV-MA	95 min
967	s968	TV Show	Headspace Guide to Sleep	NaN	Evelyn Lewis Prieto	NaN	2021-04-28	2021	TV-G	1 Season
968	s969	TV Show	Sexify	NaN	Aleksandra Skraba, Maria Sobocińska, Sandra Dr…	Poland	2021-04-28	2021	TV-MA	1 Season
972	s973	TV Show	Fatma	NaN	Burcu Biricik, Uğur Yücel, Mehmet Yılmaz Ak, H…	Turkey	2021-04-27	2021	TV-MA	1 Season
…	…	…	…	…	…	…	…	…	…	…
466	s467	TV Show	My Unorthodox Life	NaN	NaN	NaN	2021-07-14	2021	TV-MA	1 Season
467	s468	Movie	Private Network: Who Killed Manuel Buendía?	Manuel Alcalá	Daniel Giménez Cacho	NaN				
468	s469	Movie	The Guide to the Perfect Family	Ricardo Trogi	Louis Morissette, Émilie Bierre, Catherine Cha…	NaN				
471	s472	Movie	Day of Destiny	Akay Mason, Abosi Ogba	Olumide Oworu, Denola Grey, Gbemi Akinlade, Ji…	NaN				
8437	s8438	TV Show	The Netflix Afterparty	NaN	David Spade, London Hughes, Fortune Feimster	United States				

```
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 …      2018.0
966   After a botched scam, Clóvis bumps into Lohane…      2021.0
967   Learn how to sleep better with Headspace. Each…      2021.0
968   To build an innovative sex app and win a tech …      2021.0
972   Reeling from tragedy, a nondescript house clea…      2021.0
…                                                    …         …
466   Follow Julia Haart, Elite World Group CEO and …      2021.0
467   A deep dive into the work of renowned Mexican …      2021.0
468   A couple in Québec deals with the pitfalls, pr…      2021.0
471   With their family facing financial woes, two t…      2021.0
8437  Hosts David Spade, Fortune Feimster and London…      2021.0

      month_added
4250         12.0
966           4.0
967           4.0
968           4.0
972           4.0
…            …
466           7.0
467           7.0
468           7.0
471           7.0
8437          1.0

[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.

[46]:
```
df.groupby(['type' , 'rating'])['show_id'].count()
```

[46]:
```
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
```

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

- Country column

[47]:
```
df['country'].value_counts()
```

[47]:
```
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 can observe that many movies are produced in more than 1 country. Hence, the country column has comma separated nested values of countries.

Going to unnest these values. 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 duplication of records in our origional table after exploding.

[48]:
```
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
```

```
<ipython-input-48-88f820136e36>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  country_tb.dropna(inplace = True)
<ipython-input-48-88f820136e36>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  country_tb['country'] = country_tb['country'].apply(lambda x : x.split(','))
```

[48]:

	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]

[49]:
```
# some duplicate values are found, which have unnecessary spaces. some empty␣
 ↪strings found
country_tb['country'] = country_tb['country'].str.strip()
```

[50]:
```
country_tb.loc[country_tb['country'] == '']
```

[50]:

	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	

[51]:
```
country_tb = country_tb.loc[country_tb['country'] != '']
```

[52]:
```
country_tb['country'].nunique()
```

[52]: 122

Netflix has movies from the total 122 countries.

Total movies and tv shows in each country

[53]:
```
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)
```

[53]:

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

[54]:
```
df['director'].value_counts()
```

[54]:
```
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 nested in the director column with comma. We will explode the director column as well. It will create many duplicate records in original table hence we created separate table for directors.

[55]:
```
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
```

```
<ipython-input-55-8de37009c172>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dir_tb.dropna(inplace = True)
<ipython-input-55-8de37009c172>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dir_tb['director'] = dir_tb['director'].apply(lambda x : x.split(','))
```

[55]:

	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]
```

[56]:
```
dir_tb = dir_tb.explode('director')
```

[57]:
```
dir_tb['director'] = dir_tb['director'].str.strip()
```

[58]:
```
# checking if empty stirngs are there in director column
dir_tb.director.apply(lambda x : True if len(x) == 0 else False).value_counts()
```

[58]:
```
director
False    6978
Name: count, dtype: int64
```

[59]:
```
dir_tb
```

[59]:
```
     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]
```

[60]:
```
dir_tb['director'].nunique()
```

[60]:
```
4993
```

There are total 4993 unique directors in the dataset.

Total movies and tv shows directed by each director

[61]:
```
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)
```

[61]:
```
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]
```

- Exploring the 'listed_in' column to understand more about genres

[62]:
```
genre_tb = df[['show_id' , 'type', 'listed_in']]
```

[63]:
```
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()
```

```
<ipython-input-63-95f42dd5f79d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  genre_tb['listed_in'] = genre_tb['listed_in'].apply(lambda x : x.split(','))
```

[64]:
```
genre_tb
```

[64]:
```
     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]
```

[65]:
```
genre_tb.listed_in.unique()
```

[65]:
```
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)
```

[66]:
```
genre_tb.listed_in.nunique()
```

[66]:
```
42
```

Total 42 genres present in dataset

[67]:
```
df.merge(genre_tb , on = 'show_id' ).groupby(['type_y'])['listed_in_y'].
  ↪nunique()
```

[67]:
```
type_y
Movie      20
TV Show    22
Name: listed_in_y, dtype: int64
```

Movies have 20 genres and TV shows have 22 genres.

[68]:
```
# 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()
```

[68]:

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
```

7 Exploring cast column to unnest the nested data

[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
```

```
<ipython-input-69-af27dcdfd024>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  cast_tb.dropna(inplace = True)
<ipython-input-69-af27dcdfd024>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cast_tb['cast'] = cast_tb['cast'].apply(lambda x : x.split(','))
```

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

[71]:
```
# checking empty strings
cast_tb[cast_tb['cast'] == '']
```

[71]:
```
Empty DataFrame
Columns: [show_id, type, cast]
Index: []
```

[72]:
```
# Total actors on the Netflix
cast_tb.cast.nunique()
```

[72]:
```
36403
```

[73]:
```
# 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)
```

[73]:

type	Movie	TV Show
cast		
Takahiro Sakurai	7.0	25.0
Yuki Kaji	10.0	19.0

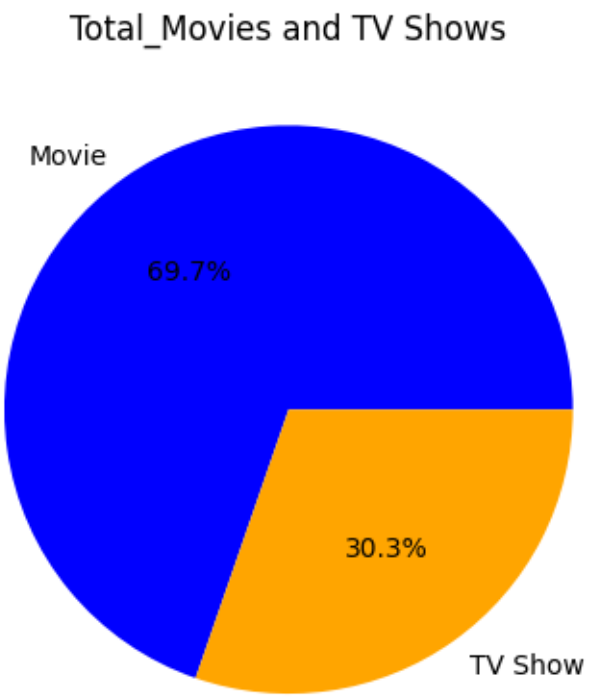
```
Junichi Suwabe      4.0     17.0
Daisuke Ono         5.0     17.0
Ai Kayano           2.0     17.0
…                   …       …
Şerif Sezer         1.0      NaN
Şevket Çoruh        1.0      NaN
Şinasi Yurtsever    3.0      NaN
Şükran Ovalı        1.0      NaN
Ṣọpẹ́ Dìrísù        1.0      NaN

[36403 rows x 2 columns]
```

8 4. Graphical Analysis (Plots)

- 4.1. Distribution of content across the different types

[74]:
```
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()
```



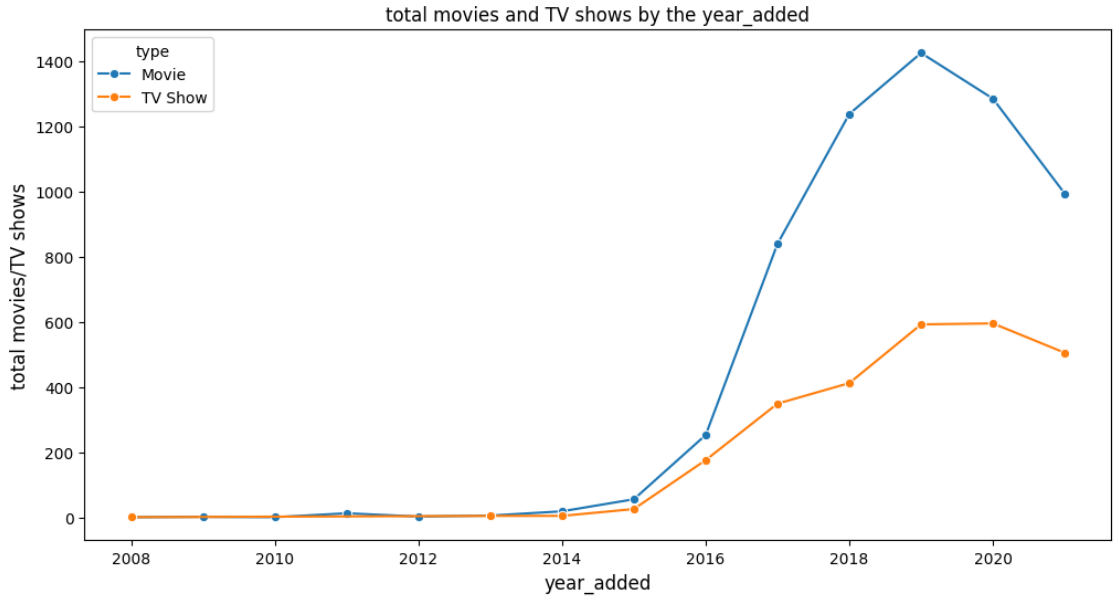
Observing that , around 70% content is Movies and around 30% content is TV shows.

- 4.2 Distribution of 'date_added' column

How has the number of movies/TV shows added on Netflix per year changed over the time?

[75]:
```
d = df.groupby(['year_added' ,'type' ])['show_id'].count().reset_index()
d.rename({'show_id' : 'total movies/TV shows'}, axis = 1 , inplace = True)
```

[76]:
```
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.

- 4.3 Distribution of 'Release_year' column

How has the number of movies released per year changed over the last 20-30 years?

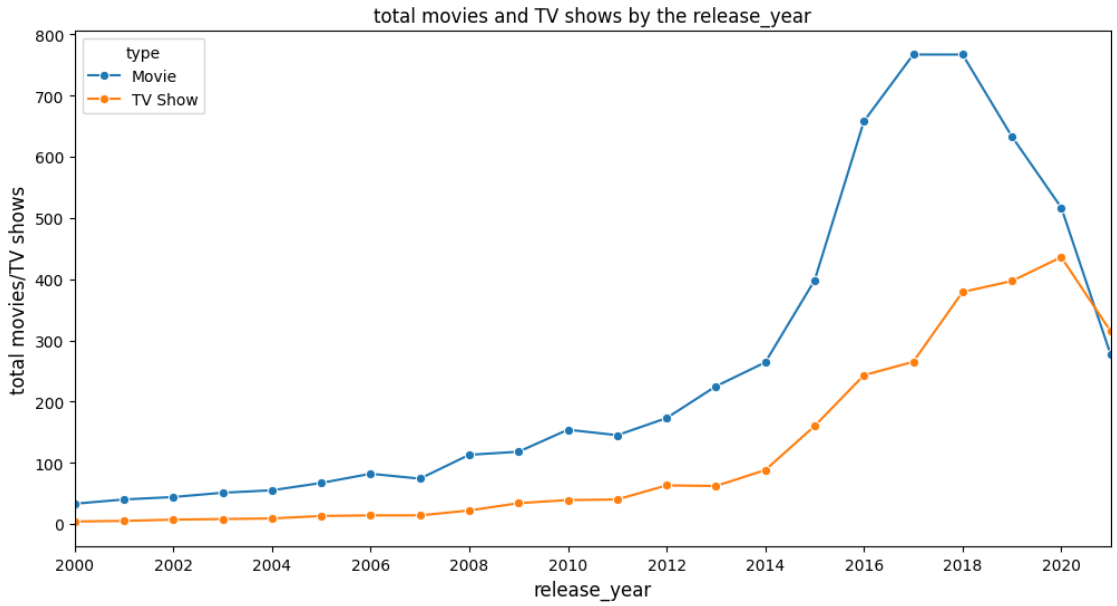
[77]:
```
d = df.groupby(['type' , 'release_year'])['show_id'].count().reset_index()
d.rename({'show_id' : 'total movies/TV shows'}, axis = 1 , inplace = True)
d
```

[77]:

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

[119 rows x 3 columns]

[78]:
```
plt.figure(figsize = (12,6))
sns.lineplot(data = d , x = 'release_year' , y = 'total movies/TV shows' , hue␣
 ↪= 'type' , marker = 'o'  , ms = 6 )
plt.xlabel('release_year' , fontsize = 12)
plt.ylabel('total movies/TV shows' , fontsize = 12)
plt.title('total movies and TV shows by the release_year' , fontsize = 12)
plt.xlim( left = 2000 , right = 2021)
plt.xticks(np.arange(2000 , 2021 , 2))
plt.show()
```



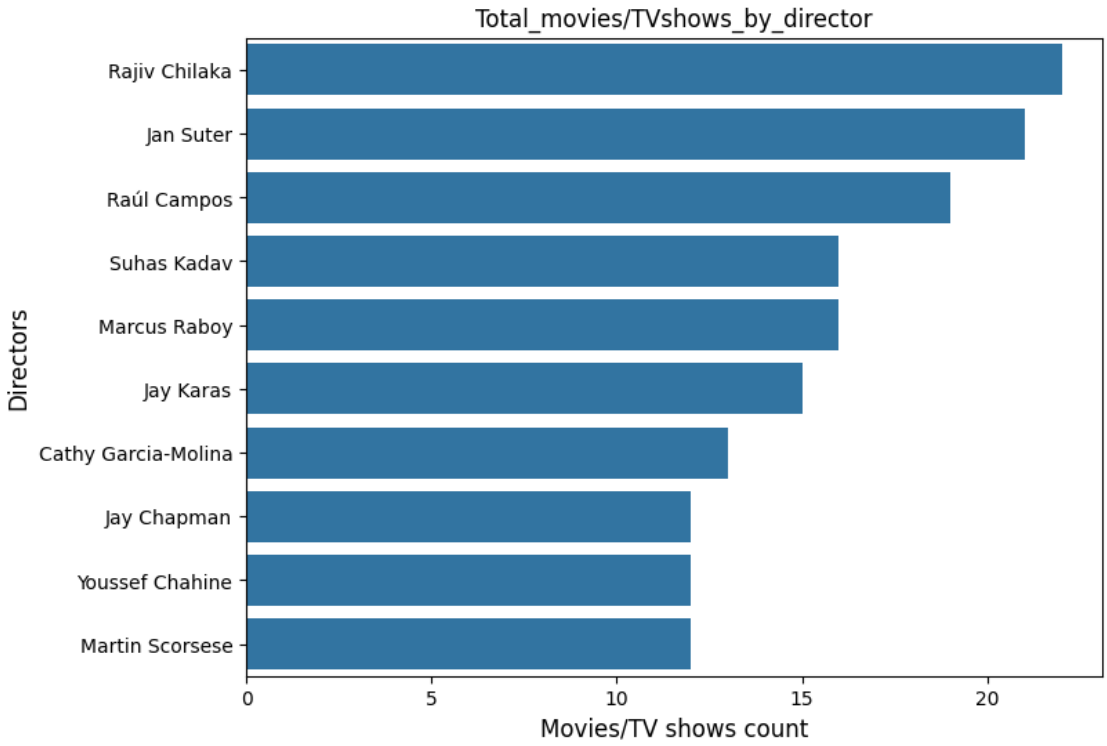
Observation: * 2018 marks the highest number of movie and TV show releases. * Since 2018, A drop in movies is seen and rise in TV shows is observed clearly, and TV shows surpasses the movies

count in mid 2020. * In recent years TV shows are focussed more than Movies. * The yearly number of releases has surged drastically from 2015.

- 4.4 Total movies/TV shows by each director

[79]:
```
# total Movies directed by top 10 directors
top_10_dir = dir_tb.director.value_counts().head(10).index
df_new = dir_tb.loc[dir_tb['director'].isin(top_10_dir)]
```

[80]:
```
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

- 4.4 Checking Outliers for number of movies directed by each director

[81]:
```
x = dir_tb.director.value_counts()
x
```

[81]:
```
director
Rajiv Chilaka     22
Jan Suter         21
Raúl Campos       19
Suhas Kadav       16
Marcus Raboy      16
                  ..
Raymie Muzquiz     1
Stu Livingston     1
Joe Menendez       1
Eric Bross         1
Mozez Singh        1
Name: count, Length: 4993, dtype: int64
```

[82]:
```
def calculate_outliers(data):
    # Calculate the first quartile (Q1)
    q1 = np.percentile(data, 25)

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

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

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

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

    return outliers


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

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

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

```
    return max_occurred_value
```

[83]:
```
outliers = calculate_outliers(x)  # calling outlier calculation method
max_occurred_value = calculate_max_occurred_value(x)  # calling method to find␣
 ↪the maximum-occurred value
set(outliers)
```

[83]:
```
{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 19, 21, 22}
```

[84]:
```
max_occurred_value
```

[84]:
```
1
```

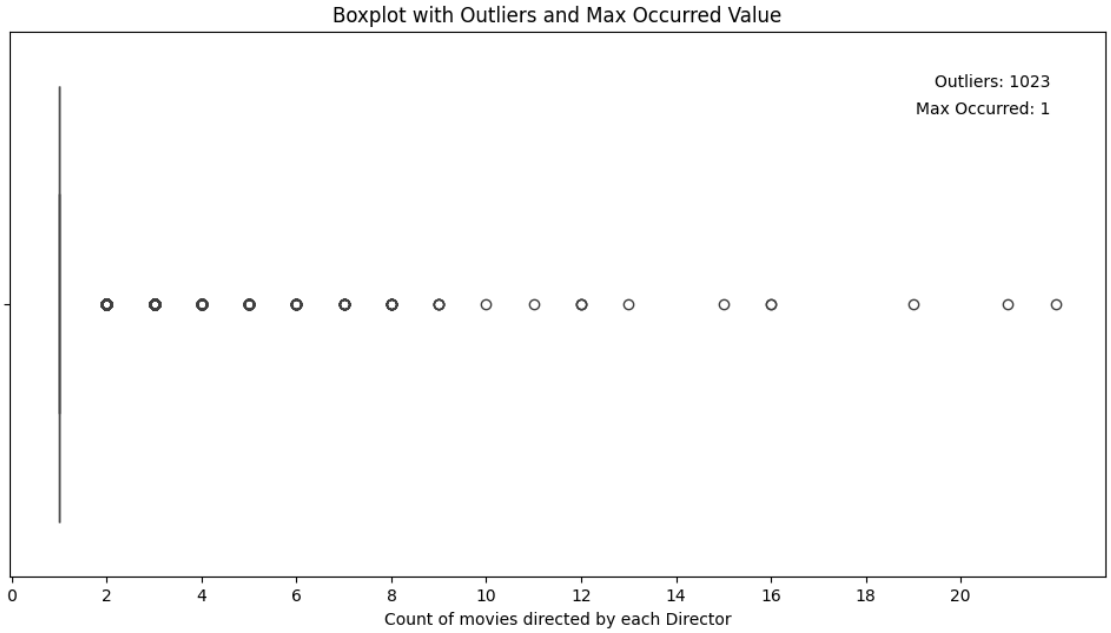
[85]:
```
plt.figure(figsize = (12,6))
sns.boxplot(data=x, showfliers=True, whis=1.5 , orient = 'h')

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

# Annotate the plot
plt.text(0.95, 0.9, f"Outliers: {len(outliers)}", transform=plt.gca().
 ↪transAxes, ha='right')
plt.text(0.95, 0.85, f"Max Occurred: {max_occurred_value}", transform=plt.gca().
 ↪transAxes, ha='right')


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

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



It is Observed that maximum occured value is 1, which means maximum directors on the Netflix have directed 1 movie/Tv show. There are few directors who have directed more than 1 movies/tv shows and they are outliers.

- 4.5 Total movies/TV shows by each country

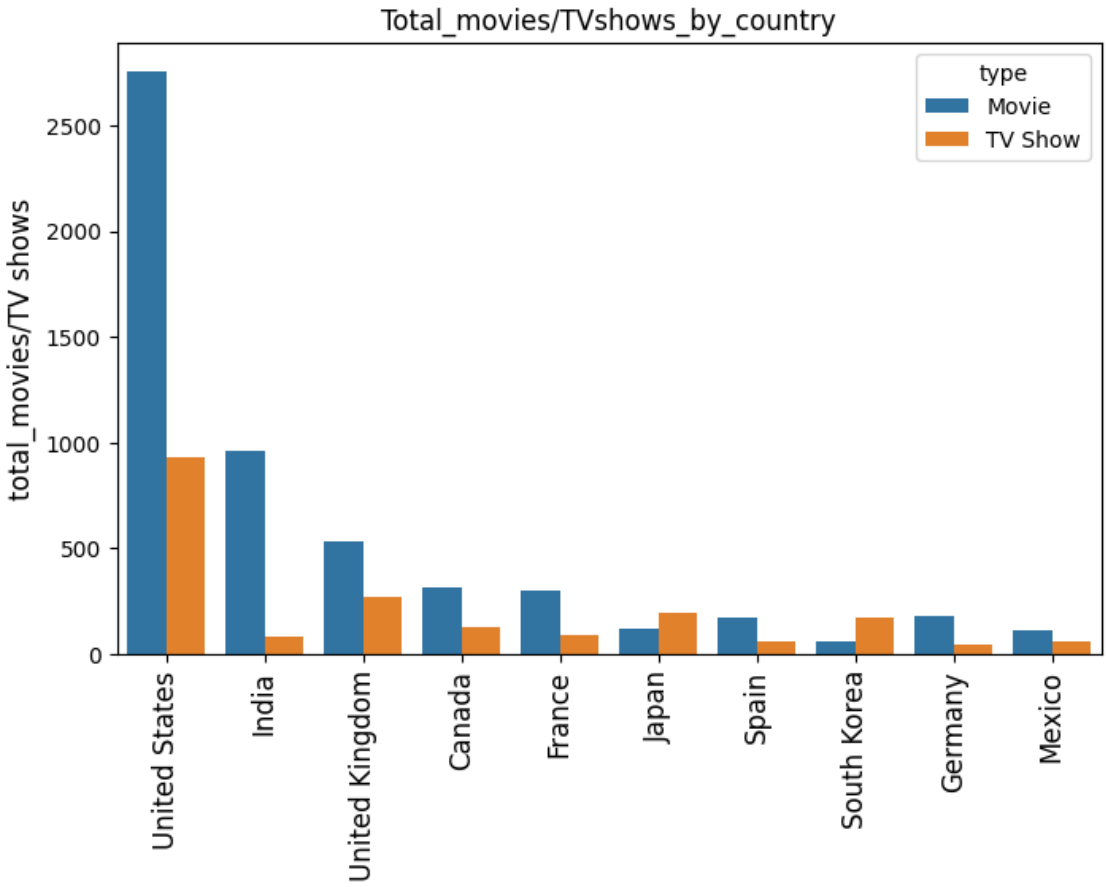
[86]:
```
# Lets check for top 10 countries
top_10_country = country_tb.country.value_counts().head(10).index
df_new = country_tb.loc[country_tb['country'].isin(top_10_country)]
```

[87]:
```
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)
```

[87]:

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

[88]:
```
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()
```



[89]:
```
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' )
```

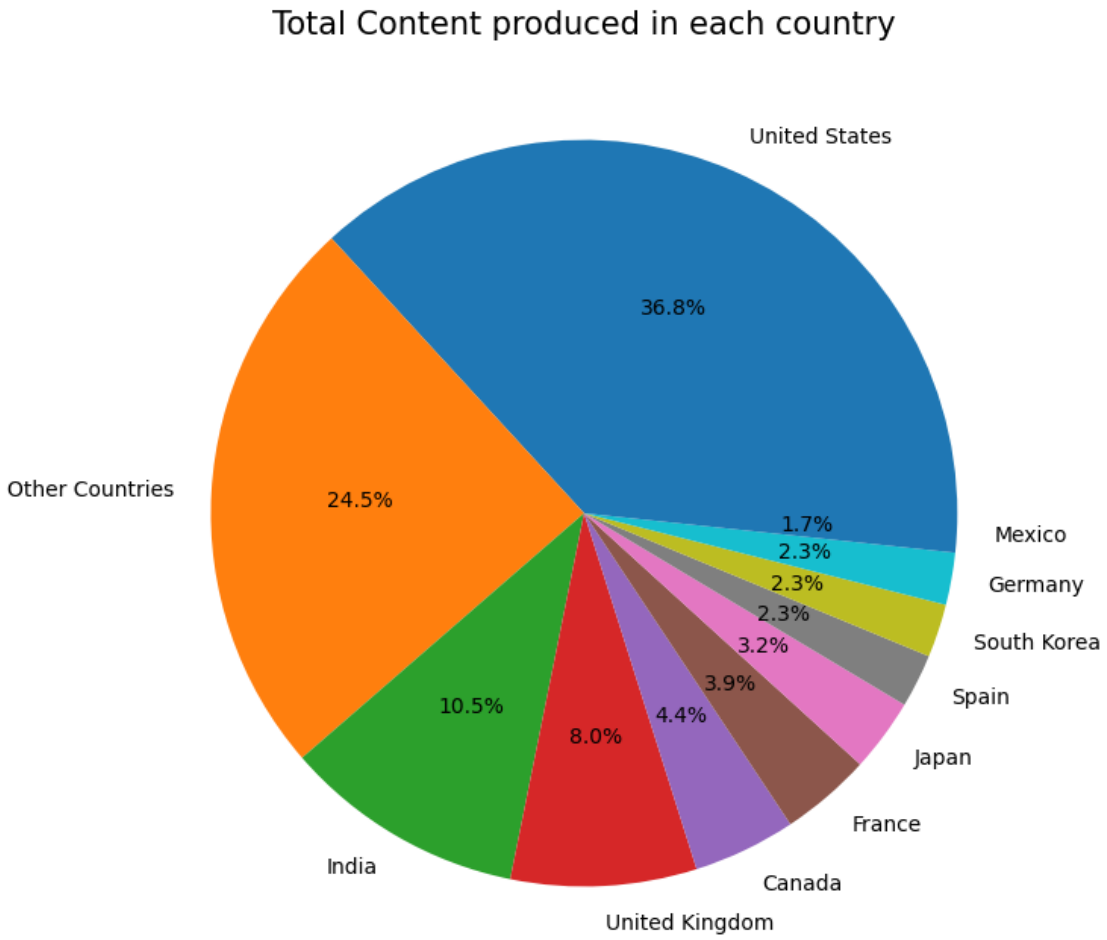
```
<ipython-input-89-8e32e79d11da>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

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

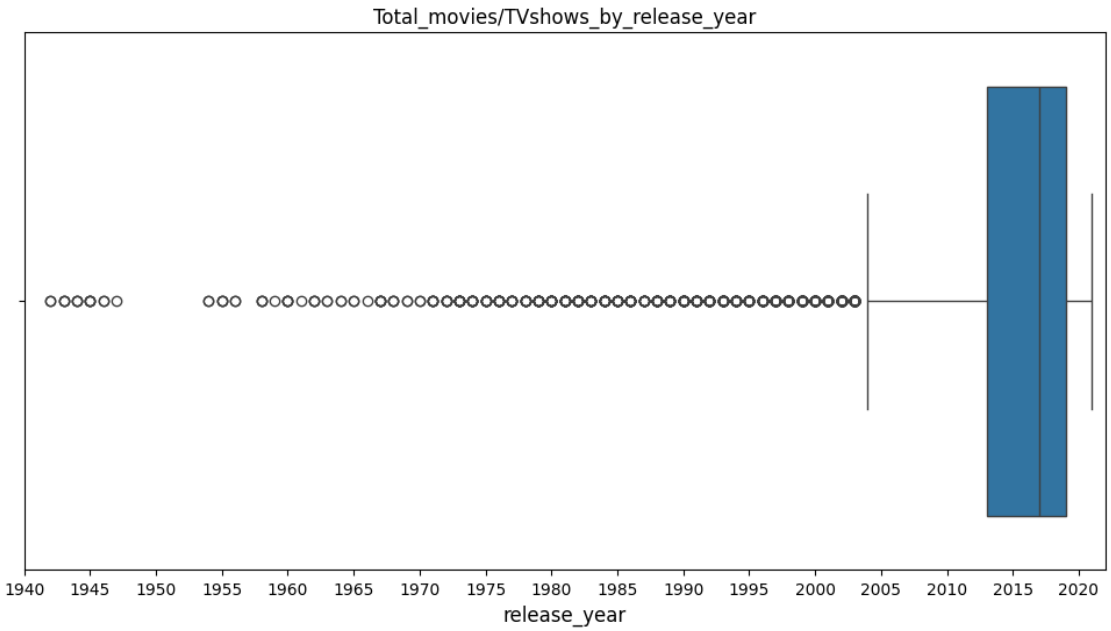
[90]:
```
x = country_tb.cat.value_counts()

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()
```



- Observation:
  - 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.
- 4.6 Total content distribution by release year of the content

[91]:
```
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()
```



Observations: * 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.

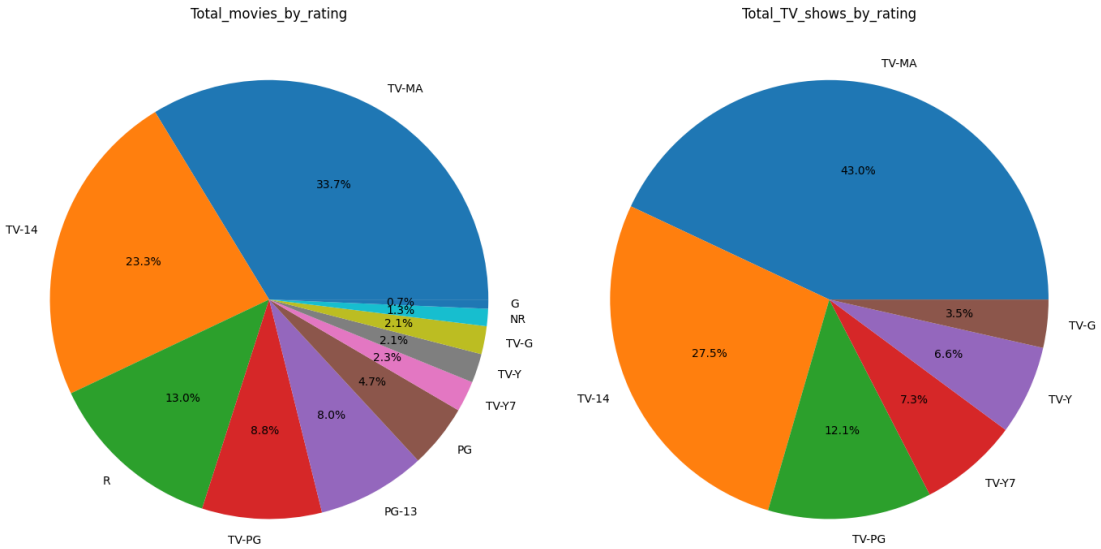
- 4.7 Total movies/TV shows distribution by rating of the content

[93]:
```
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

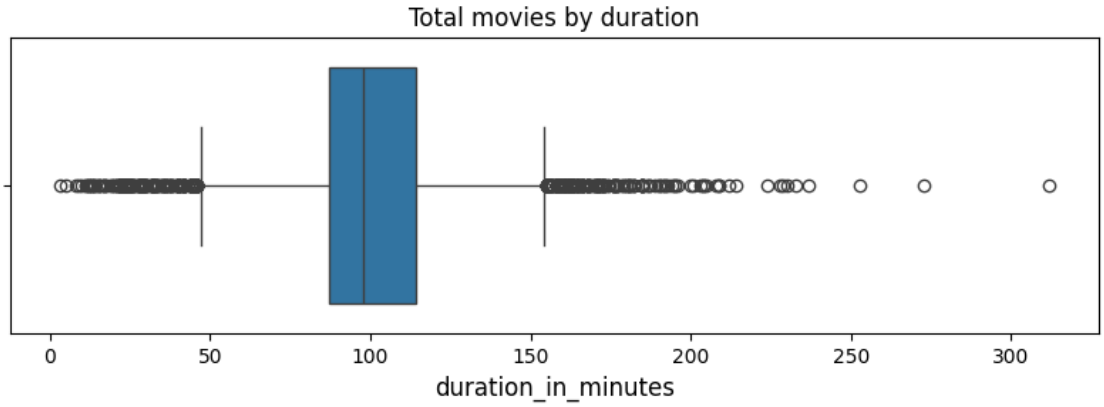
- 4.8 Total movies/TV shows distribution by duration of the content

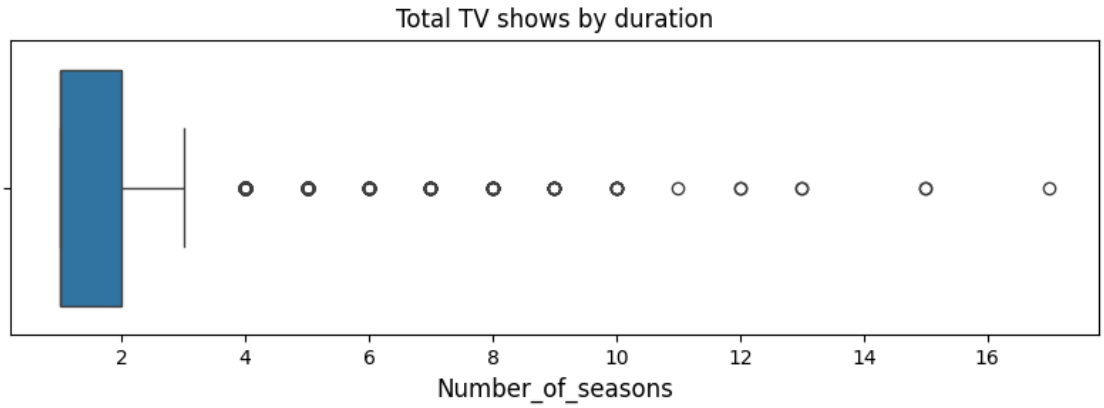
[94]:
```
fig, ax = plt.subplots(2,1, figsize=(8,6))

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()
```





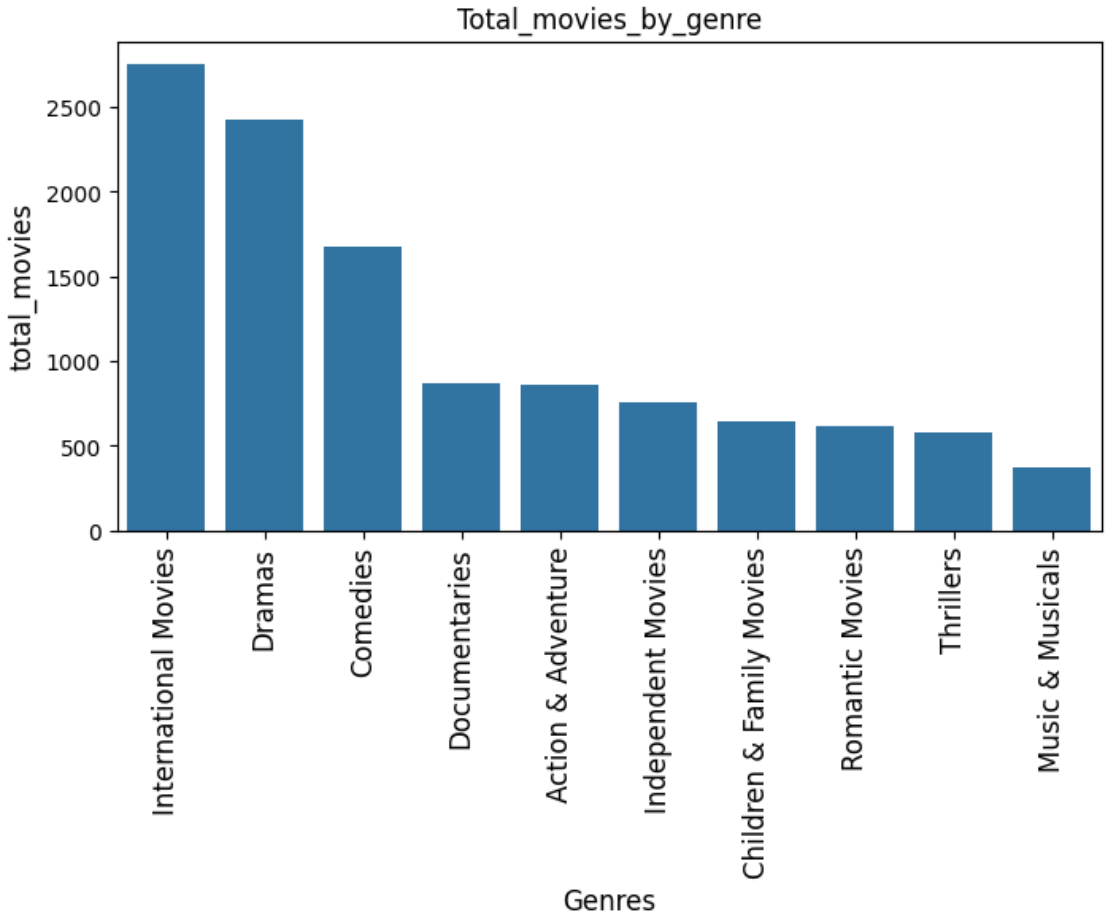
- Movie Duration: 50 mins - 150 mins is the range excluding potential outliers (values lying outside the whiskers of boxplot)
- TV Show Duration: 1-3 seasons is the range for TV shows excluding potential outliers
- 4.9 Total movies/TV shows in each Genre

[96]:
```
# Lets check the count for top 10 genres in Movies and TV_shows
top_10_movie_genres = genre_tb[genre_tb['type'] == 'Movie'].listed_in.
 ↪value_counts().head(10).index
df_movie = genre_tb.loc[genre_tb['listed_in'].isin(top_10_movie_genres)]
```

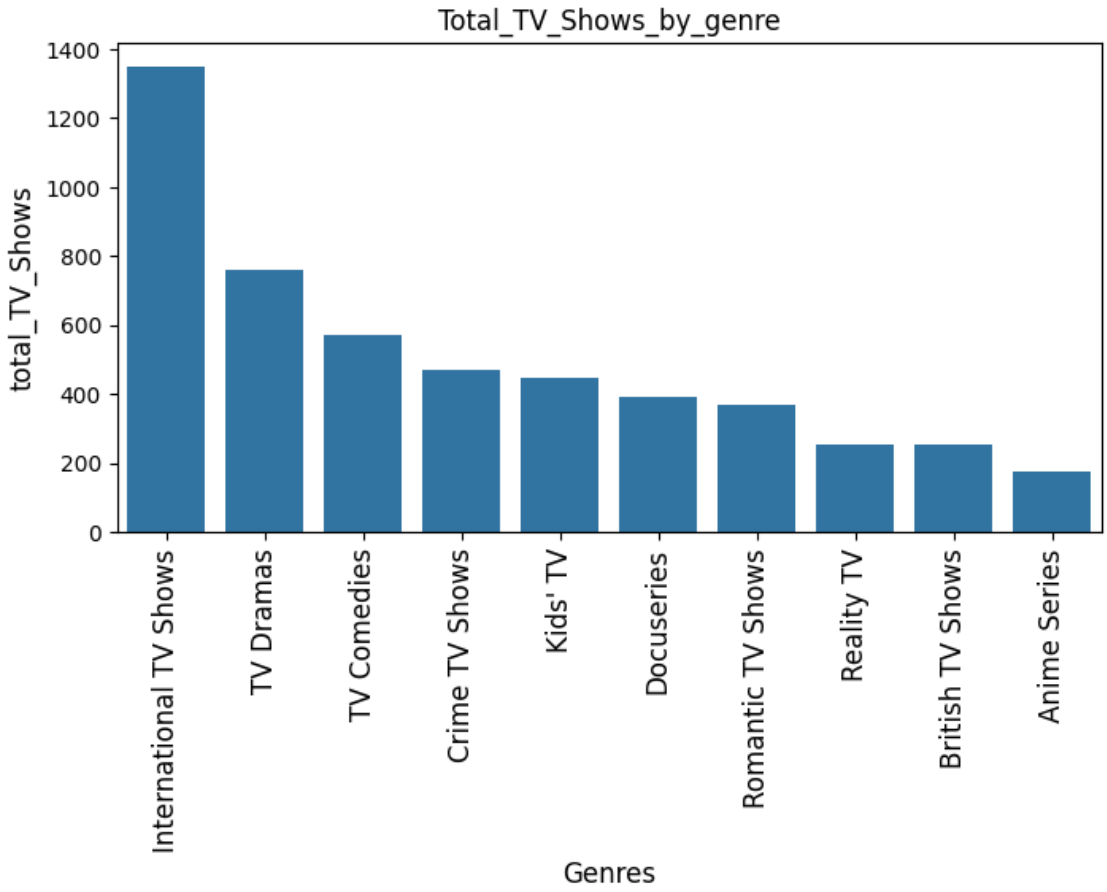
[97]:
```
top_10_TV_genres = genre_tb[genre_tb['type'] == 'TV Show'].listed_in.
 ↪value_counts().head(10).index
df_tv = genre_tb.loc[genre_tb['listed_in'].isin(top_10_TV_genres)]
```

[98]:
```
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()
```



[99]:
```
plt.figure(figsize= (8,4))
sns.countplot(data = df_tv , x = 'listed_in' , order = top_10_TV_genres)
plt.xticks(rotation = 90 , fontsize = 12)
plt.ylabel('total_TV_Shows' , fontsize = 12)
plt.xlabel('Genres' , fontsize = 12)
plt.title('Total_TV_Shows_by_genre')
plt.show()
```



- International Movies and TV Shows , Dramas , and Comedies are the top 3 genres on Netflix for both Movies and TV shows.
- 4.10 Top 5 genres in each country

[101]:
```
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)
```

[102]:
```
top_5_genre
```

[102]:

	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
5	India	International Movies	864
6	India	Dramas	662
7	India	Comedies	323
8	India	Independent Movies	167
9	India	Action & Adventure	137
10	United Kingdom	British TV Shows	224
11	United Kingdom	Dramas	197
12	United Kingdom	International Movies	170
13	United Kingdom	International TV Shows	128
14	United Kingdom	Documentaries	128
15	Canada	Comedies	94
16	Canada	Dramas	82
17	Canada	Children & Family Movies	80
18	Canada	Kids' TV	61
19	Canada	International Movies	60
20	France	International Movies	207
21	France	Dramas	167
22	France	Independent Movies	73
23	France	Comedies	51
24	France	Thrillers	44
25	Japan	International TV Shows	151
26	Japan	Anime Series	142
27	Japan	International Movies	72
28	Japan	Anime Features	61
29	Japan	Action & Adventure	57

- 4.11 Top 5 directors by Genre

[103]:
```
genre_list = [ 'Children & Family Movies', 'Comedies','Dramas', 'International␣
 ↪Movies', 'Documentaries' ,
             'International TV Shows', 'Sci-Fi & Fantasy', 'Thrillers',␣
 ↪'Horror Movies']

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

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

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

top_5_dir
```

[103]:

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

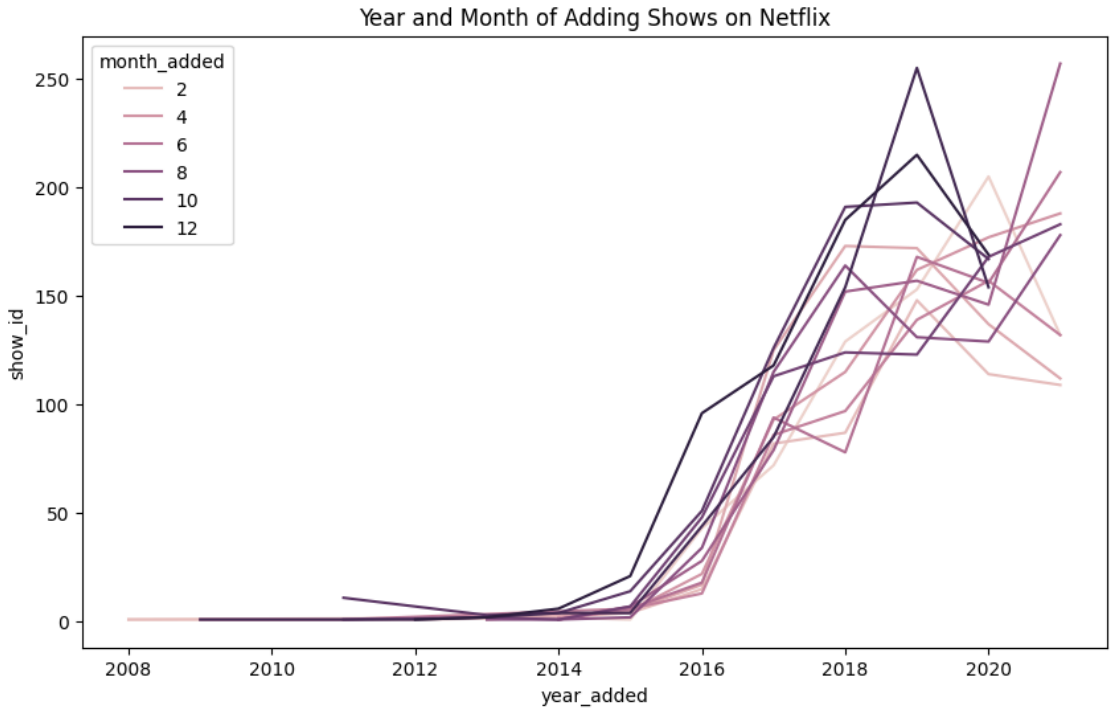
```
10635          Sci-Fi & Fantasy       Barry Sonnenfeld        3
11974                  Thrillers    Rathindran R Prasad        4
11698                  Thrillers          David Fincher        4
11612                  Thrillers         Anurag Kashyap        3
11636                  Thrillers          Brad Anderson        3
11754                  Thrillers         Gregory Hoblit        3
6280             Horror Movies            Rocky Soraya        6
6260             Horror Movies               Poj Arnon        5
6267             Horror Movies    Rathindran R Prasad        4
6191             Horror Movies            Leigh Janiak        3
6052             Horror Movies  Banjong Pisanthanakun        3
```

- 4.12 What is the best time of the year when maximum content get added on the Netflix?

[105]:
```
month_year = df.groupby(['year_added' , 'month_added'])['show_id'].count().
↪reset_index()
```

[106]:
```
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')
```

[106]:
```
Text(0.5, 1.0, 'Year and Month of Adding Shows on Netflix')
```



9 Insights based on Exploration and Graphical Analysis

• Approximately 70% of Netflix's content consists of movies, while around 30% consists of TV shows. • The addition of movies and TV shows to Netflix began in 2008, with very limited content until 2014. • The year 2015 marked a significant surge in content uploads on Netflix, continuing an upward trend that peaked in 2019 with the highest number of movies and TV shows added. The years 2020 and 2021 saw a decline in content, likely due to the pandemic, though the drop in TV shows was not as pronounced as that of movies. • Since 2018, there has been a noticeable decline in movies but a clear rise in TV shows, with TV shows surpassing the number of movies by mid-2020, indicating their growing popularity in recent years. • Netflix features movies from a wide range of directors, with approximately 4,993 directors having their movies or TV shows available on the platform. • Netflix offers movies from a total of 122 countries, with the United States being the largest contributor, accounting for almost 37% of all content. • The release years for shows are concentrated between 2005 and 2021. • The duration of most movies ranges from 50 to 150 minutes, excluding potential outliers. • TV shows typically span 1 to 3 seasons, excluding potential outliers. • Netflix offers a variety of content ratings to cater to different viewer categories such as kids, adults, and families. The highest number of movies and TV shows are rated TV-MA (for mature audiences). • Most ratings are available in limited quantities outside the US. Ratings like TV-Y7, TV-Y7 FV, PG, TV-G, G, TV-Y, and TV-PG are scarce in all countries except the US. • The top three genres on Netflix for both movies and TV shows are International Movies and TV Shows, Dramas, and Comedies. • Each country has its own popular genres, but the United States has a diverse mix of almost all genres. For example, Korean TV shows are popular in Korea, British TV shows in the UK, and Anime features and series in Japan. • Indian actors feature in the most movies on Netflix, with the top five actors by quantity of movies being based in India. • Shorter-duration movies have gained popularity over the last 10 years.

10 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. * drop in content is seen across all the countries and type of content in year 2020 and 2021, possibly because of Pandemic.

#Recommendations

- Most countries, except the US, have a very limited focus on genres. The currently available genres seem to cater primarily to the US and a few other countries, leaving many regions without genres that are highly popular locally. For example, Indian mythological content is extremely popular. By creating more country-specific genres, we might find global appeal similar to Japanese anime.

• Country-specific insights: Content should be tailored to the demographics of each country. Netflix can increase the quantity of content in specific ratings that match the demographics of a country.

For example: In a populous country like India, most content is available in just three ratings: TV-MA, TV-14, and TV-PG. This approach may not adequately serve audiences below 14 years old and above 35 years old.

- Country Japan have only 3 rating of content largely served - TV-MA, TV-14 , TV-PG. Japan have high population of age above 60, and this can be served by increasing the content suitable for this age group.
- Netflix is currently catering primarily to mature audiences and children with parental guidance. There is potential to expand its offerings to include other audience groups such as families, senior citizens, and children of various ages.