Import the Libraries, Data Set and the Basic observations

```
import pandas as pd
df = pd.read_csv('/content/Netflix_dataset.csv')
df
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	liste
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documen
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	Interna TV Show Drama Mys
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crir S Interna TV Show
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docus Real
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	Interna TV S Roman Shows,
...	...	...	...	...	...	...	...	...	...	...	
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2007	R	158 min	Cult M Dra Th
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2018	TV-Y7	2 Seasons	Kic Kore Show Com
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	R	88 min	Com Horror M
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2006	PG	88 min	Child Family M Com
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2015	TV-14	111 min	Dra Interna Movies, & Mu

8807 rows × 12 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	de
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	n
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	A p
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...	T p
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	f
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 ...	ce

Next steps: Generate code with df View recommended plots New interactive sheet

```
df.shape
```

```
(8807, 12)
```

```
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
```

From the above analysis we can say that the data data has total of 12 features with lots of mixed alpha numeric data. Also we can see missing data in 5 of the total columns.

```
df.nunique()
```

```
                    0
      show_id    8807
         type       2
        title    8807
     director    4528
         cast    7692
      country     748
   date_added    1767
 release_year      74
       rating      17
     duration     220
    listed_in     514
  description    8775

dtype: int64
```

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

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

```
df.describe()
```

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

We can see that there is only single column which is having only numerical values. It infers that the idea of release year of the content range is between what timeframe. Rest all of the columns are having categorical data.

```
df.describe(include = object)
```

	show_id	type	title	director	cast	country	date_added	rating	duration	listed_in	description
count	8807	8807	8807	6173	7982	7976	8797	8803	8804	8807	8807
unique	8807	2	8807	4528	7692	748	1767	17	220	514	8775
top	s1	Movie	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	TV-MA	1 Season	Dramas, International Movies	Paranormal activity at a lush, abandoned prope...

2. **Cleaning of the Data**

Checking the overall null values in the data set.

```
df.isna().sum()
```

	0
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

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

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	desc
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	74 min	NaN	Movies	Louis C. o eternal
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	2010	84 min	NaN	Movies	Emm com Louis C
5813	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	August 15, 2016	2015	66 min	NaN	Movies	The c his t hilarious,

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

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

```
<ipython-input-20-89b4f0c8704c>:1: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in
  df.loc[ind] = df.loc[ind].fillna(method = 'ffill' , axis = 1)
<ipython-input-20-89b4f0c8704c>:1: FutureWarning: Setting an item of incompatible dtype is deprecated and will ra
  df.loc[ind] = df.loc[ind].fillna(method = 'ffill' , axis = 1)
```

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

```
df.loc[ind]
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	des
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	Not Available	74 min	Movies	Louis C (eterna
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	2010	Not Available	84 min	Movies	Emr con Louis (
5813	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	August 15, 2016	2015	Not Available	66 min	Movies	The (his hilarious

Filling the null values in rating column

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

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in
5989	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Oprah Winfrey, Ava DuVernay	NaN	January 26, 2017	2017	NaN	37 min	Movies
6827	s6828	TV Show	Gargantia on the Verdurous Planet	NaN	Kaito Ishikawa, Hisako Kanemoto, Ai Kayano, Ka...	Japan	December 1, 2016	2013	NaN	1 Season	Anime Series, International TV Shows
7312	s7313	TV Show	Little Lunch	NaN	Flynn Curry, Olivia Deeble, Madison Lu, Oisín ...	Australia	February 1, 2018	2015	NaN	1 Season	Kids' TV, TV Comedies
7537	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Leone Frisa, Paolo Vaccarino, Francesco Miglio...	Italy	March 1, 2017	2015	NaN	115 min	Dramas

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

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

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

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	liste
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documen
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	Interna TV Show Drama Mys
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crir S Interna TV Show
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docus Real
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	Interna TV S Roman Shows,
...	...	...	...	...	...	...	...	...	...	...	
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2007	R	158 min	Cult M Dra Th
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2018	TV-Y7	2 Seasons	Kic Kore Show Com
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	R	88 min	Com Horror M
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2006	PG	88 min	Chilc Family M Com
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2015	TV-14	111 min	Dra Interna Movies, & Mu

8807 rows × 12 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

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

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in
5989	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Oprah Winfrey, Ava DuVernay	NaN	January 26, 2017	2017	Not Available	37 min	Movies
6827	s6828	TV Show	Gargantia on the Verdurous Planet	NaN	Kaito Ishikawa, Hisako Kanemoto, Ai Kayano, Ka...	Japan	December 1, 2016	2013	Not Available	1 Season	Anime Series Internationa TV Shows
7312	s7313	TV Show	Little Lunch	NaN	Flynn Curry, Olivia Deeble, Madison Lu, Oisín ...	Australia	February 1, 2018	2015	Not Available	1 Season	Kids' TV, TV Comedies
7537	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Leone Frisa, Paolo Vaccarino, Francesco Miglio...	Italy	March 1, 2017	2015	Not Available	115 min	Dramas

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

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

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

rating	count
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

dtype: int64

drop the null from date_added column

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

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

```
                     count
        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
1767 rows × 1 columns

dtype: int64
```

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

```
df['date_added'] = pd.to_datetime(df['date_added'])
df['date_added']
```

```
        date_added
   0    2021-09-25
   1    2021-09-24
   2    2021-09-24
   3    2021-09-24
   4    2021-09-24
  ...          ...
8802    2019-11-20
8803    2019-07-01
8804    2019-11-01
8805    2020-01-11
8806    2019-03-02
8797 rows × 1 columns

dtype: datetime64[ns]
```

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

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

	year_added
0	2021
1	2021
2	2021
3	2021
4	2021
...	...
8802	2019
8803	2019
8804	2019
8805	2020
8806	2019

8797 rows × 1 columns

dtype: int32

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

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

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

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

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

```
                0
show_id         0
type            0
title           0
director     2624
cast          825
country       830
date_added      0
release_year    0
rating          4
duration        3
listed_in       0
description     0
year_added      0
month_added     0

dtype: int64
```

% Null values in each column

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

```
                0
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
```

We can infer that from above that after cleaning some data we still have null values in 3 columns which are greater in numbers.

For some content - country is missing. (9%), director names are missing (30%),cast is missing (9%)

3. Data Exploration and Non Graphical Analysis

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

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

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

```
movies.duration.value_counts()
```

```
            count
duration
  90 min      152
  94 min      146
  93 min      146
  97 min      146
  91 min      144
    ...       ...
 212 min        1
   8 min        1
 186 min        1
 193 min        1
 191 min        1
205 rows × 1 columns

dtype: int64
```

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

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

dtype: int64
```

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

```
# Create a copy of the movies DataFrame to avoid modifying the original
movies = movies.copy()

# Create new columns for duration based on the type of content
movies['movie_duration'] = None  # Initialize the movie_duration column
movies['tv_show_duration'] = None  # Initialize the tv_show_duration column

# Fill the movie_duration column for Movies
movies.loc[movies['type'] == 'Movie', 'movie_duration'] = movies['duration'].str[:-3].astype(float)

# Fill the tv_show_duration column for TV Shows
movies.loc[movies['type'] == 'TV Show', 'tv_show_duration'] = movies['duration'].str.extract('(\d+)')[0].astype(float

# Verify the changes
print(movies[['type', 'duration', 'movie_duration', 'tv_show_duration']].head(10))
```

```
     type duration movie_duration tv_show_duration
0   Movie   90 min           90.0             None
6   Movie   91 min           91.0             None
7   Movie  125 min          125.0             None
9   Movie  104 min          104.0             None
12  Movie  127 min          127.0             None
13  Movie   91 min           91.0             None
16  Movie   67 min           67.0             None
18  Movie   94 min           94.0             None
22  Movie  161 min          161.0             None
23  Movie   61 min           61.0             None
```

```
# Make sure you create a copy of the DataFrame if needed
tv_shows = tv_shows.copy()  # Avoid modifying the original DataFrame

# If the 'duration_in_seasons' column has a specific format, clean it
# Remove any unnecessary characters and convert to float, if needed
tv_shows.loc[:, 'duration_in_seasons'] = tv_shows['duration_in_seasons'].apply(lambda x: str(x).strip())  # Ensure it
tv_shows.loc[:, 'duration_in_seasons'] = tv_shows['duration_in_seasons'].astype(float)  # Convert to float

# Verify the changes
print(tv_shows[['title', 'duration_in_seasons']].head())
```

```
                   title duration_in_seasons
1          Blood & Water                 2.0
2              Ganglands                 1.0
3  Jailbirds New Orleans                 1.0
4           Kota Factory                 2.0
5          Midnight Mass                 1.0
```

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

```
tv_shows.duration_in_seasons
```

```
      duration_in_seasons
1                     2.0
2                     1.0
3                     1.0
4                     2.0
5                     1.0
...                   ...
8795                  2.0
8796                  2.0
8797                  3.0
8800                  1.0
8803                  2.0
2676 rows × 1 columns

dtype: object
```

```
movies.duration_in_minutes
```

```
      duration_in_minutes
0                  90 min
6                  91 min
7                 125 min
9                 104 min
12                127 min
...                   ...
8801               96 min
8802              158 min
8804               88 min
8805               88 min
8806              111 min
6131 rows × 1 columns

dtype: object
```

```
tv_shows.duration_in_seasons
```

```
      duration_in_seasons
1                     2.0
2                     1.0
3                     1.0
4                     2.0
5                     1.0
...                   ...
8795                  2.0
8796                  2.0
8797                  3.0
8800                  1.0
8803                  2.0
2676 rows × 1 columns

dtype: object
```

```
movies.duration_in_minutes
```

```
      duration_in_minutes
0                     NaN
6                     NaN
7                     NaN
9                     NaN
12                    NaN
...                   ...
8801                  NaN
8802                  NaN
8804                  NaN
8805                  NaN
8806                  NaN
6131 rows × 1 columns

dtype: float64
```

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

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

# Check for NaT values that may have resulted from invalid parsing
print("NaT values:", df['date_added'].isna().sum())

# Now create the timeperiod Series, excluding NaT values
min_date = df['date_added'].min()
max_date = df['date_added'].max()

# Ensure that min_date and max_date are not NaT
if pd.notna(min_date) and pd.notna(max_date):
    timeperiod = pd.Series((min_date.strftime('%B %Y'), max_date.strftime('%B %Y')))
    timeperiod.index = ['first', 'Most Recent']
else:
    timeperiod = pd.Series(['No valid dates', 'No valid dates'], index=['first', 'Most Recent'])
```

```
# Display the timeperiod Series
print(timeperiod)
```

```
NaT values: 98
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?

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

```
(1925, 2021)
```

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

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	lis
4250	s4251	TV Show	Pioneers: First Women Filmmakers*	NaN	NaN	NaN	2018-12-30	1925	TV-14	1 Season	TV
966	s967	Movie	Get the Grift	Pedro Antonio	Marcus Majella, Samantha Schmütz, Caito Mainie...	Brazil	2021-04-28	2021	TV-MA	95 min	Co Inter
967	s968	TV Show	Headspace Guide to Sleep	NaN	Evelyn Lewis Prieto	NaN	2021-04-28	2021	TV-G	1 Season	Doc Sc Na
968	s969	TV Show	Sexify	NaN	Aleksandra Skraba, Maria Sobocińska, Sandra Dr...	Poland	2021-04-28	2021	TV-MA	1 Season	Inter TV Sh Come
972	s973	TV Show	Fatma	NaN	Burcu Biricik, Uğur Yücel, Mehmet Yılmaz Ak, H...	Turkey	2021-04-27	2021	TV-MA	1 Season	Inter TV Sh Dra
...	...	...	...	...	...	...	...	...	...	...	
466	s467	TV Show	My Unorthodox Life	NaN	NaN	NaN	2021-07-14	2021	TV-MA	1 Season	Re
467	s468	Movie	Private Network: Who Killed Manuel Buendía?	Manuel Alcalá	Daniel Giménez Cacho	NaN	2021-07-14	2021	TV-MA	100 min	Docume Inter
468	s469	Movie	The Guide to the Perfect Family	Ricardo Trogi	Louis Morissette, Émilie Bierre, Catherine Cha...	NaN	2021-07-14	2021	TV-MA	102 min	Co I Inter
471	s472	Movie	Day of Destiny	Akay Mason, Abosi Ogba	Olumide Oworu, Denola Grey, Gbemi Akinlade, Ji...	NaN	2021-07-13	2021	TV-PG	110 min	Ch Family I Intern
8437	s8438	TV Show	The Netflix Afterparty	NaN	David Spade, London Hughes, Fortune Feimster	United States	2021-01-02	2021	TV-MA	1 Season	S Comec Sh Co

593 rows × 12 columns

Which are different ratings available on Netflix in each type of content? Check the number of content released in each type.

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

		show_id
type	rating	
Movie	66 min	1
	74 min	1
	84 min	1
	G	41
	NC-17	3
	NR	75
	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
	UR	3
TV Show	NR	5
	R	2
	TV-14	733
	TV-G	94
	TV-MA	1145
	TV-PG	323
	TV-Y	176
	TV-Y7	195
	TV-Y7-FV	1

dtype: int64

Working on the columns which are having maximum null values and the columns having comma separated multiple values for each record i.e

Country column

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

	count
country	
United States	2818
India	972
United Kingdom	419
Japan	245
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

748 rows × 1 columns

dtype: int64

We have seen 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 analyze 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 ,just to avoid the duplicacy of records in our origional table after exploding.

```
# Select relevant columns
country_tb = df[['show_id', 'type', 'country']].copy()

# Drop rows with NaN values
country_tb = country_tb.dropna()

# Split country strings and strip whitespace
country_tb['country'] = country_tb['country'].apply(lambda x: [country.strip() for country in x.split(',')])

# Explode the list of countries into separate rows
country_tb = country_tb.explode('country')

# Display the resulting DataFrame
print(country_tb)
```

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

[10019 rows x 3 columns]
```

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

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

```
123
```

Netflix has movies from the total 123 countries.

Total movies and tv shows in each country

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

```
           type  Movie  TV Show
        country
  United States 2752.0    938.0
          India  962.0     84.0
 United Kingdom  534.0    272.0
         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
123 rows × 2 columns
```

Director column

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

```
                                    count
                         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
4528 rows × 1 columns

dtype: int64
```

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

```
# Select relevant columns
dir_tb = df[['show_id', 'type', 'director']].copy()

# Drop rows with NaN values
dir_tb = dir_tb.dropna()

# Split director strings and strip whitespace
dir_tb['director'] = dir_tb['director'].apply(lambda x: [director.strip() for director in x.split(',')])

# Optional: Explode the list of directors into separate rows
dir_tb = dir_tb.explode('director')

# Display the resulting DataFrame
print(dir_tb)
```

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

[6978 rows x 3 columns]
```

```
dir_tb = 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()
```

```
          count
director
   False   6978

dtype: int64
```

```
dir_tb
```

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

6978 rows × 3 columns

Next steps: Generate code with dir_tb View recommended plots New interactive sheet

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

```
4993
```

There are total 4993 unique directors in the dataset.

Total movies and tv shows directed by each director.

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

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

4993 rows × 2 columns

'listed_in' column to understand more about genres

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

```
# Assuming genre_tb is already defined and contains the 'listed_in' column
# Split the 'listed_in' strings by commas
genre_tb['listed_in'] = genre_tb['listed_in'].apply(lambda x: x.split(','))

# Explode the list into separate rows
genre_tb = genre_tb.explode('listed_in')

# Strip whitespace from the 'listed_in' column
genre_tb['listed_in'] = genre_tb['listed_in'].str.strip()

# Optionally, reset the index if needed
genre_tb.reset_index(drop=True, inplace=True)

# Display the resulting DataFrame
print(genre_tb)
```

```
      show_id     type                listed_in
0          s1    Movie            Documentaries
1          s2  TV Show   International TV Shows
2          s2  TV Show                TV Dramas
3          s2  TV Show             TV Mysteries
4          s3  TV Show           Crime TV Shows
...       ...      ...                      ...
19318   s8806    Movie  Children & Family Movies
19319   s8806    Movie                 Comedies
19320   s8807    Movie                   Dramas
19321   s8807    Movie     International Movies
19322   s8807    Movie         Music & Musicals

[19323 rows x 3 columns]
```

```
genre_tb.listed_in.unique()
```

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

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

```
42
```

So there are 42 genres present in dataset

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

	listed_in_y
type_y	
Movie	20
TV Show	22

dtype: int64

Finally we have seen is Movies have 20 genres and TV shows have 22 genres.

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

```
                          type   Movie  TV Show
                    listed_in
         Action & Adventure      859.0      NaN
            Anime Features        71.0      NaN
              Anime Series         NaN    176.0
          British TV Shows         NaN    253.0
    Children & Family Movies     641.0      NaN
          Classic & Cult TV        NaN     28.0
            Classic Movies       116.0      NaN
                  Comedies      1674.0      NaN
            Crime TV Shows         NaN    470.0
               Cult Movies        71.0      NaN
             Documentaries       869.0      NaN
                Docuseries         NaN    395.0
                    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   1351.0
                  Kids' TV         NaN    451.0
           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    174.0
             Sports Movies       219.0      NaN
           Stand-Up Comedy       343.0      NaN
 Stand-Up Comedy & Talk Shows      NaN     56.0
       TV Action & Adventure       NaN    168.0
               TV Comedies         NaN    581.0
                 TV Dramas         NaN    763.0
                 TV Horror         NaN     75.0
              TV Mysteries         NaN     98.0
          TV Sci-Fi & Fantasy      NaN     84.0
                  TV Shows         NaN     16.0
               TV Thrillers        NaN     57.0
             Teen TV Shows         NaN     69.0
```

Thrillers	577.0	NaN

```
#Now exploring Cast column
# Select relevant columns
cast_tb = df[['show_id', 'type', 'cast']].copy()

# Drop rows with NaN values
cast_tb = cast_tb.dropna()

# Split the 'cast' strings by commas and strip whitespace
cast_tb['cast'] = cast_tb['cast'].apply(lambda x: [actor.strip() for actor in x.split(',')])

# Explode the list into separate rows
cast_tb = cast_tb.explode('cast')

# Optionally, reset the index for clarity
cast_tb.reset_index(drop=True, inplace=True)

# Display the resulting DataFrame
print(cast_tb)
```

```
      show_id     type                   cast
0          s2  TV Show            Ama Qamata
1          s2  TV Show           Khosi Ngema
2          s2  TV Show        Gail Mabalane
3          s2  TV Show       Thabang Molaba
4          s2  TV Show      Dillon Windvogel
...       ...      ...                   ...
64121   s8807    Movie      Manish Chaudhary
64122   s8807    Movie          Meghna Malik
64123   s8807    Movie         Malkeet Rauni
64124   s8807    Movie        Anita Shabdish
64125   s8807    Movie  Chittaranjan Tripathy

[64126 rows x 3 columns]
```

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

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

show_id	type	cast

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

```
36439
```

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

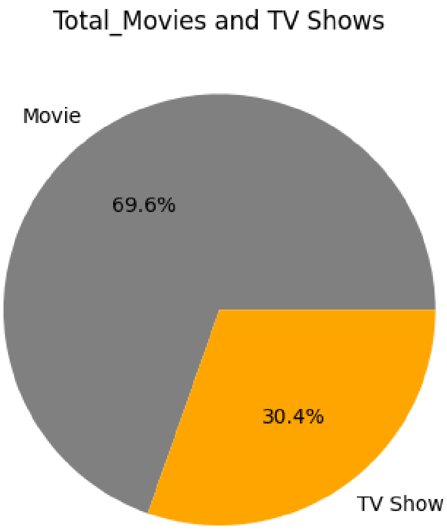
type	Movie	TV Show
cast		
Takahiro Sakurai	7.0	25.0
Yuki Kaji	10.0	19.0
Daisuke Ono	5.0	17.0
Ai Kayano	2.0	17.0
Junichi Suwabe	4.0	17.0
...	...	...
Şerif Sezer	1.0	NaN
Şevket Çoruh	1.0	NaN
Şinasi Yurtsever	3.0	NaN
Şükran Ovalı	1.0	NaN
Şopé Dìrísù	1.0	NaN

36439 rows × 2 columns

4. Visual Analysis - Univariate & Bivariate

4.1. Distribution of content across the different types

```
types = df.type.value_counts()
plt.pie(types,  labels=types.index, autopct='%1.1f%%' , colors = ['grey' , 'orange'])
plt.title('Total_Movies and TV Shows')
plt.show()
```



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

4.2 Distribution of 'date_added' column

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

```
print(df.columns)
```

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

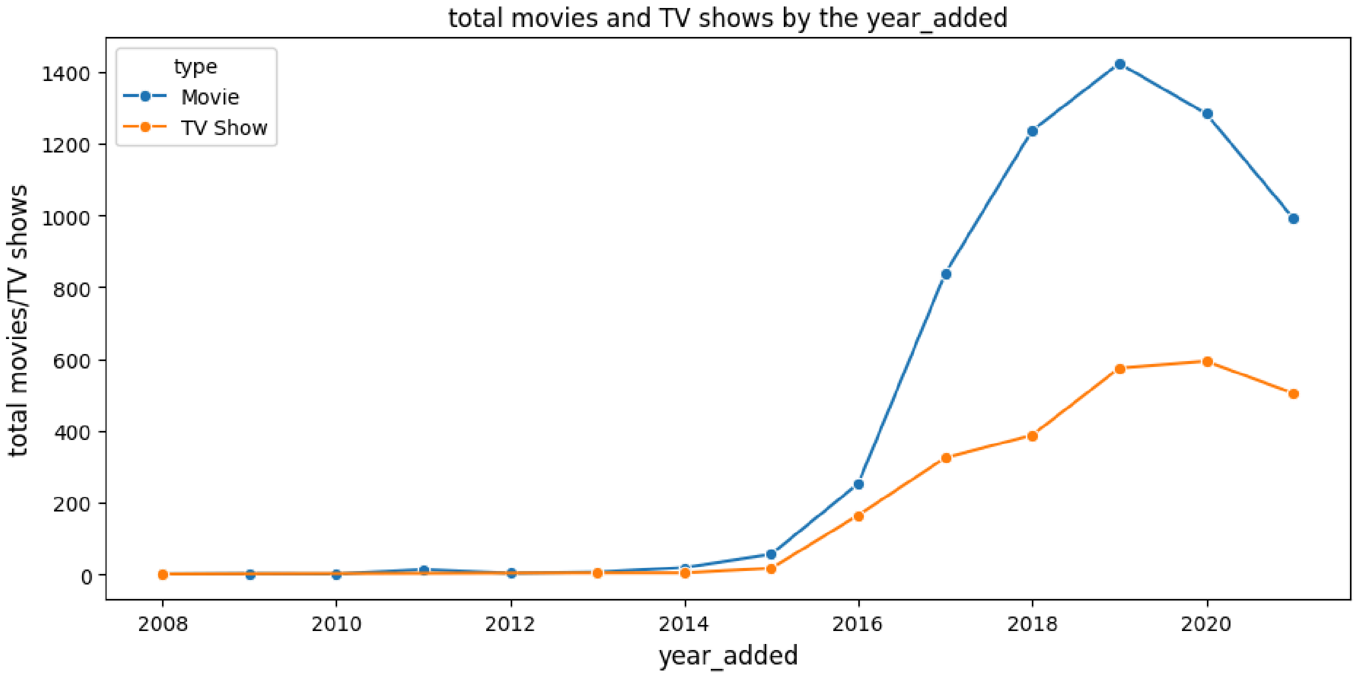
```
# Assuming 'date_added' is a datetime column
df['year_added'] = df['date_added'].dt.year
```

```
print('year_added')
```

```
year_added
```

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

```
plt.figure(figsize = (11,5))
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()
```



From the above plot:

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?

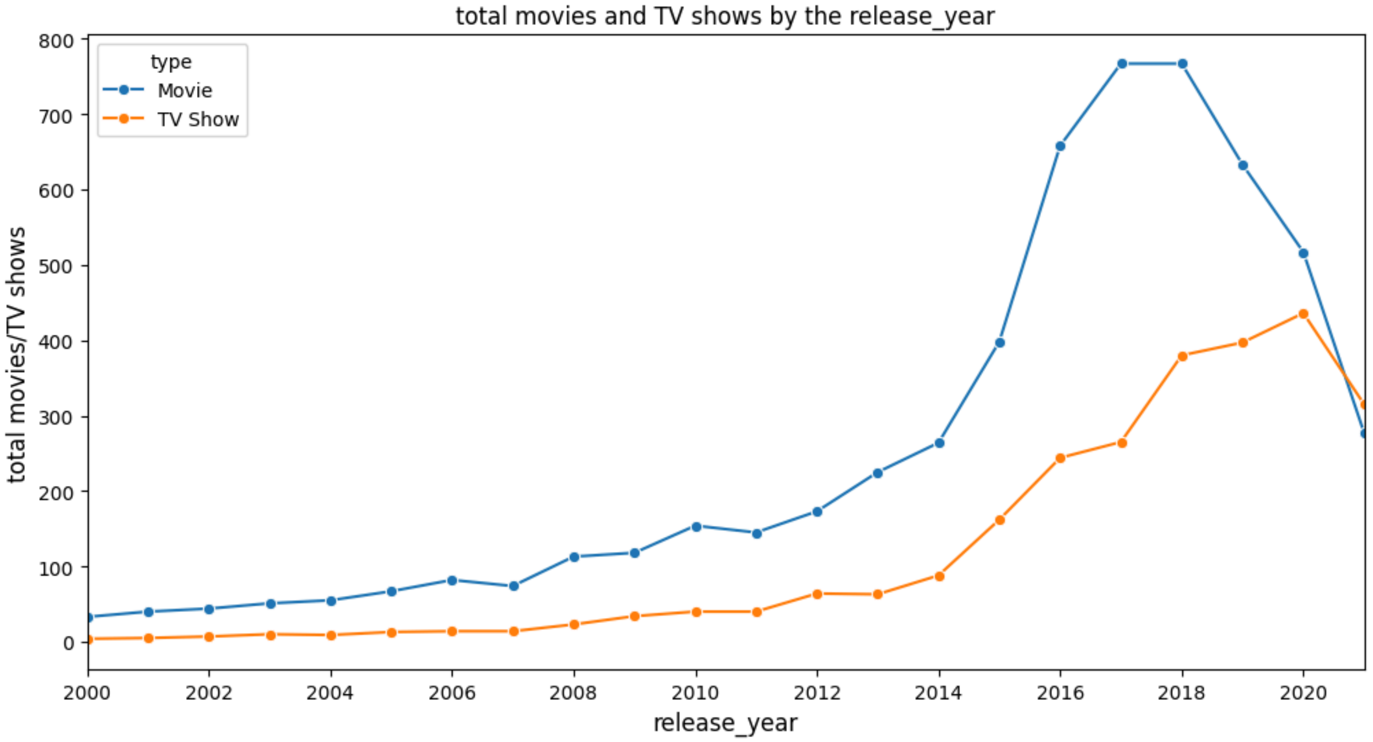
```
d = df.groupby(['type' , 'release_year'])['show_id'].count().reset_index()
d.rename({'show_id' : 'total movies/TV shows'}, axis = 1 , inplace = True)
d
```

	type	release_year	total movies/TV shows
0	Movie	1942	2
1	Movie	1943	3
2	Movie	1944	3
3	Movie	1945	3
4	Movie	1946	1
...	...	...	...
114	TV Show	2017	265
115	TV Show	2018	380
116	TV Show	2019	397
117	TV Show	2020	436
118	TV Show	2021	315

119 rows × 3 columns

Next steps: Generate code with d View recommended plots New interactive sheet

```
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()
```



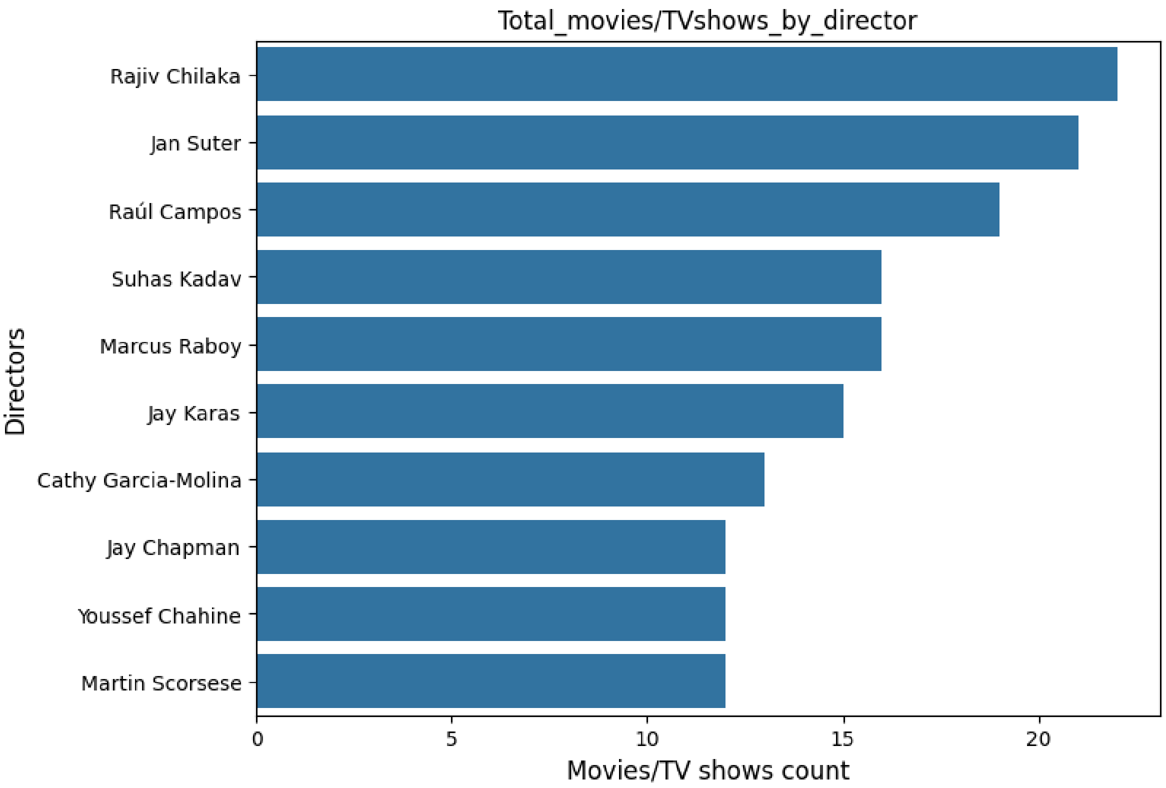
From the above observation:

1.2018 marks the highest number of movie and TV show releases. 2.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. 3.In recent years TV shows are focussed more than Movies. 4.The yearly number of releases has surged drastically from 2015.

4.4 Total movies/TV shows by each director

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

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



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

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

```
director
Rajiv Chilaka     22
Jan Suter         21
Raúl Campos       19
Suhas Kadav       16
Marcus Raboy      16
                  ..
Raymie Muzquiz     1
Stu Livingston     1
Joe Menendez       1
Eric Bross         1
Mozez Singh        1
```

```
    Name: count, Length: 4993, dtype: int64
```

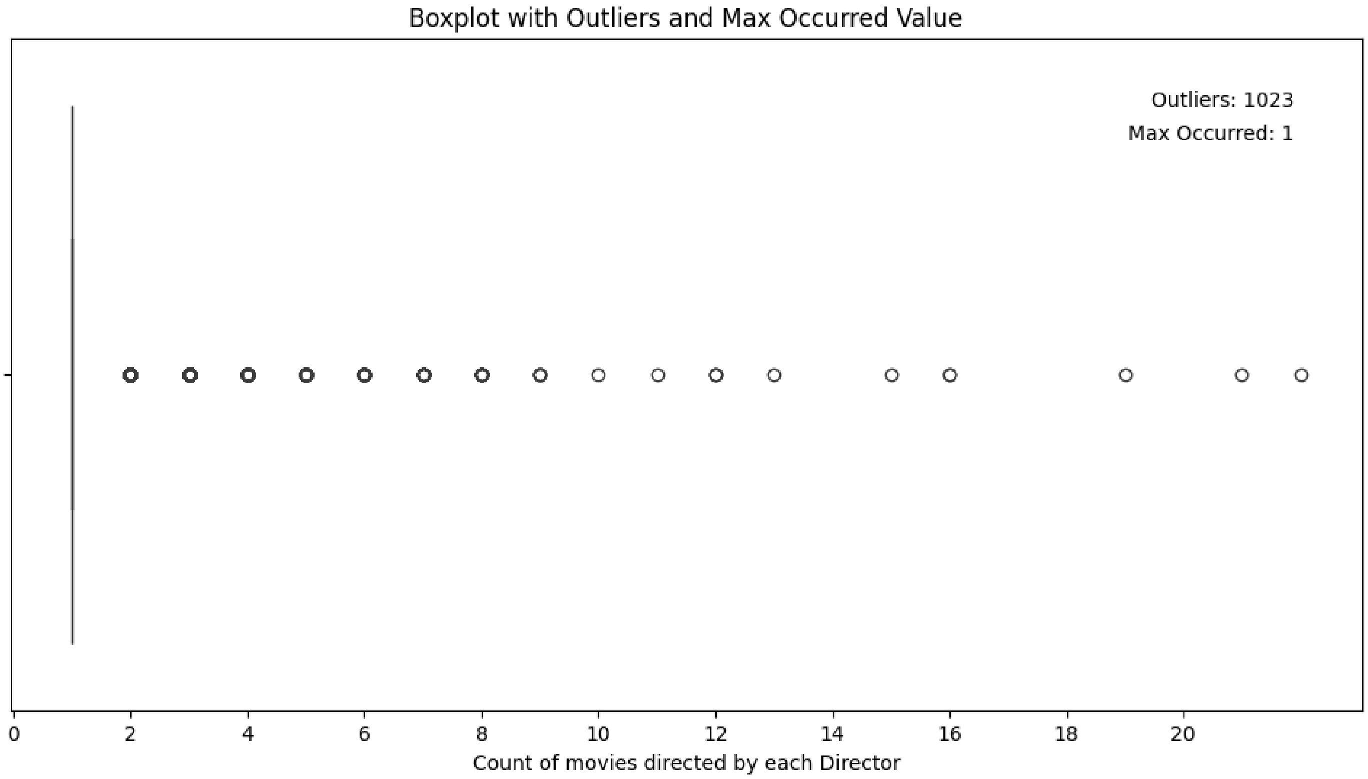
```
def calculate_outliers(data):
    # Calculate the first quartile (Q1)
    q1 = np.percentile(data, 25)

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

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

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

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

    return outliers


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

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

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

    return max_occurred_value


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

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

```
max_occurred_value
```

```
1
```

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

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

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


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

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



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

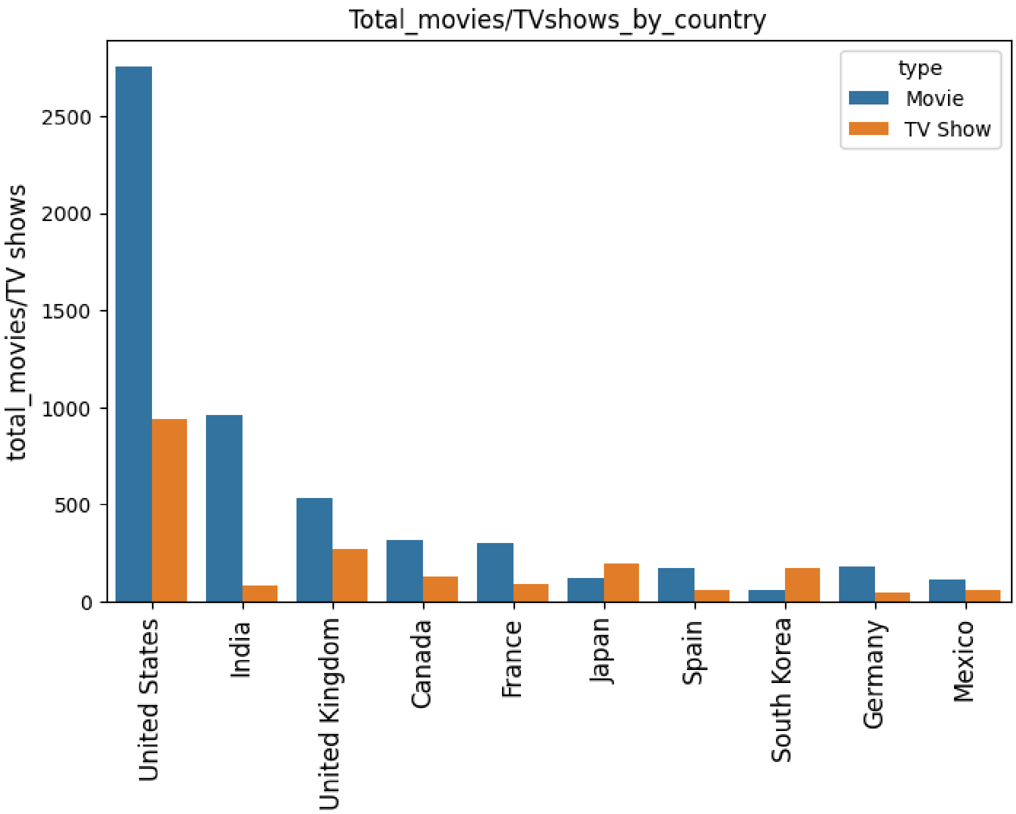
4.5 Total movies/TV shows by each country

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

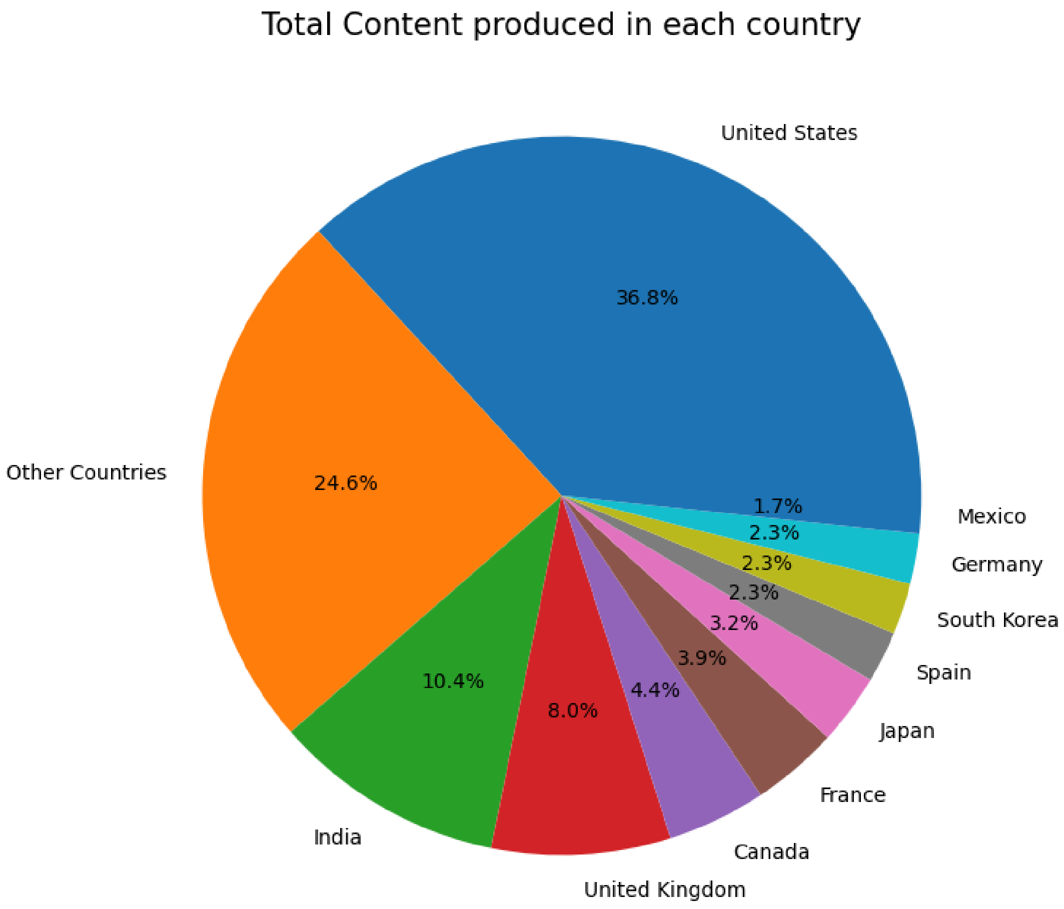
```
x = df_new.groupby(['country' , 'type'])['show_id'].count().reset_index()
x.pivot(index = 'country' , columns = 'type' , values = 'show_id').sort_values('Movie',ascending = False)
```

type	Movie	TV Show
country		
United States	2752	938
India	962	84
United Kingdom	534	272
Canada	319	126
France	303	90
Germany	182	44
Spain	171	61
Japan	119	199
Mexico	111	58
South Korea	61	170

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



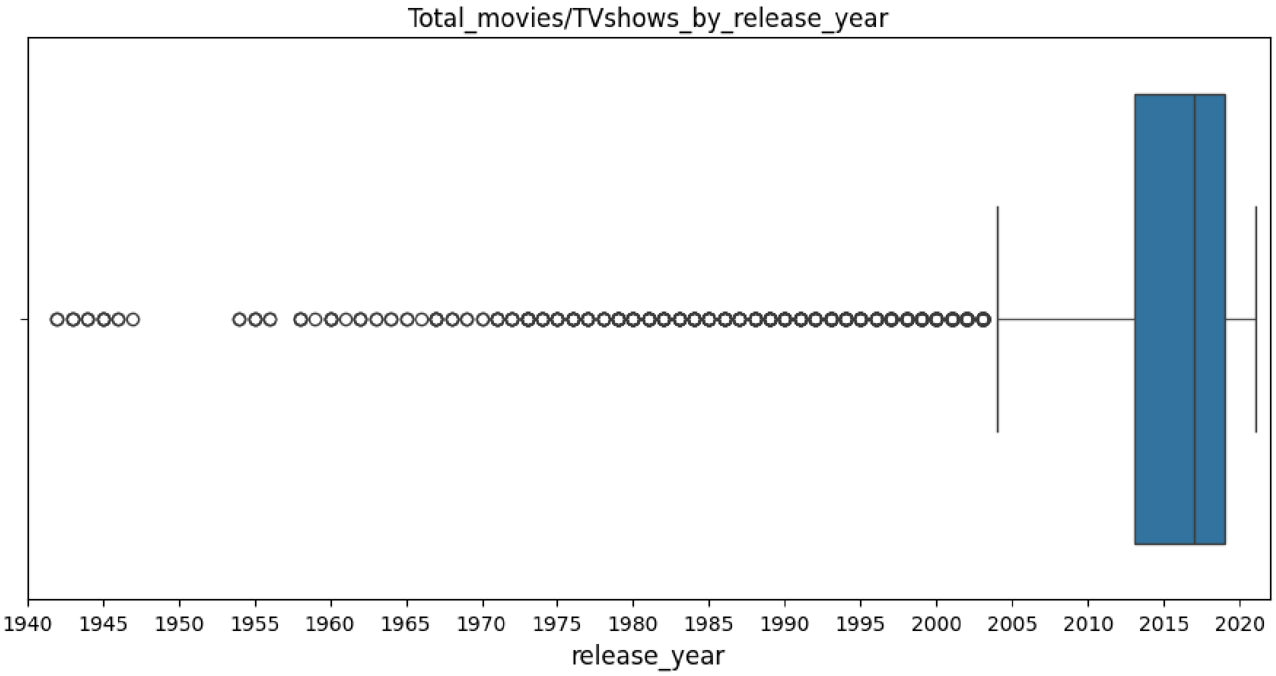
```
top_10_country = country_tb.country.value_counts().head(10).index
country_tb['cat'] = country_tb['country'].apply(lambda x : x if x in top_10_country else 'Other Countries' )


x = country_tb.cat.value_counts()

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



Observation from above pie chart: 1.United States is the HIGHEST contributor country on Netflix, followed by India and United Kingdom. 2.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

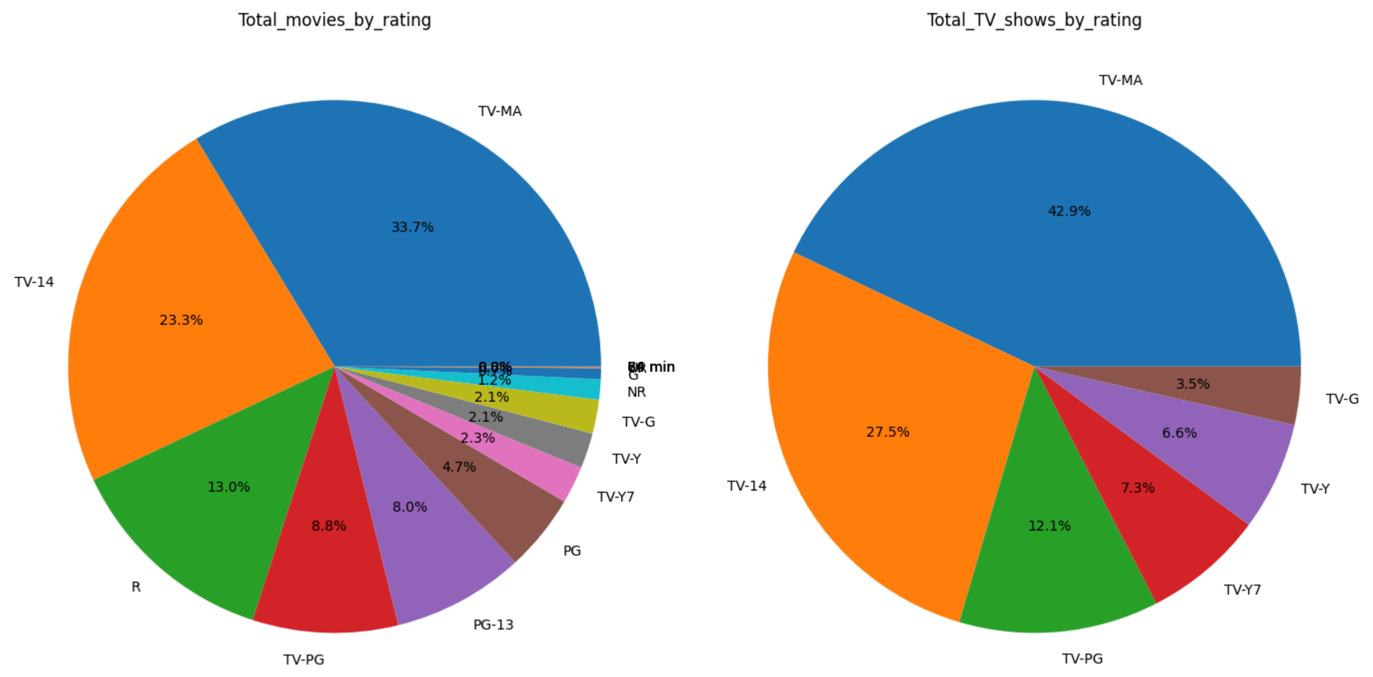
```
plt.figure(figsize= (11,5))
sns.boxplot(data = df , x = 'release_year')
plt.xlabel('release_year' , fontsize = 12)
plt.title('Total_movies/TVshows_by_release_year')
plt.xticks(np.arange(1940 , 2021 , 5))
plt.xlim((1940 , 2022))
plt.show()
```



1.Netflix have major content which is released in the year range 2000-2021 2.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

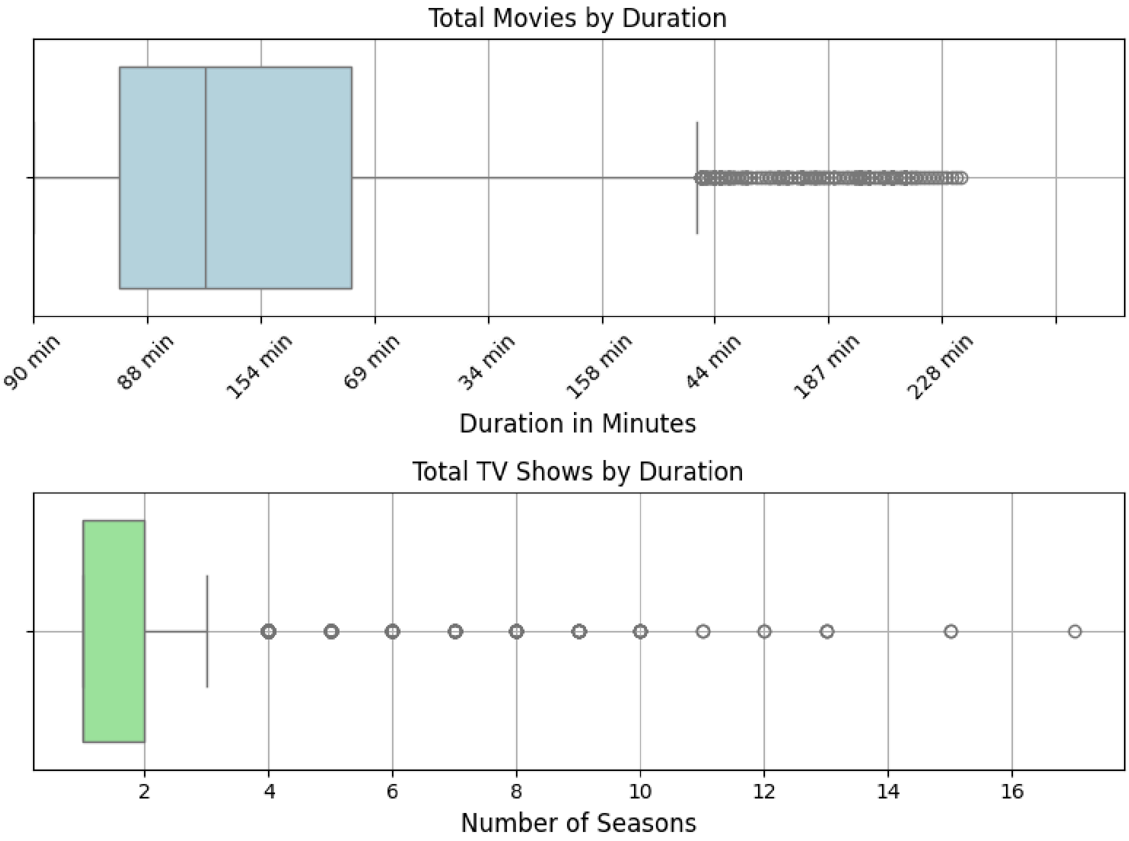
```
m = movies.loc[~movies.rating.isin(['Not Available' , 'NC-17' , 'TV-Y7-FV'])]
m = m.rating.value_counts()
t = tv_shows.loc[~tv_shows.rating.isin(['Not Available' , 'R' , 'NR', 'TV-Y7-FV'])]
t = t.rating.value_counts()


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

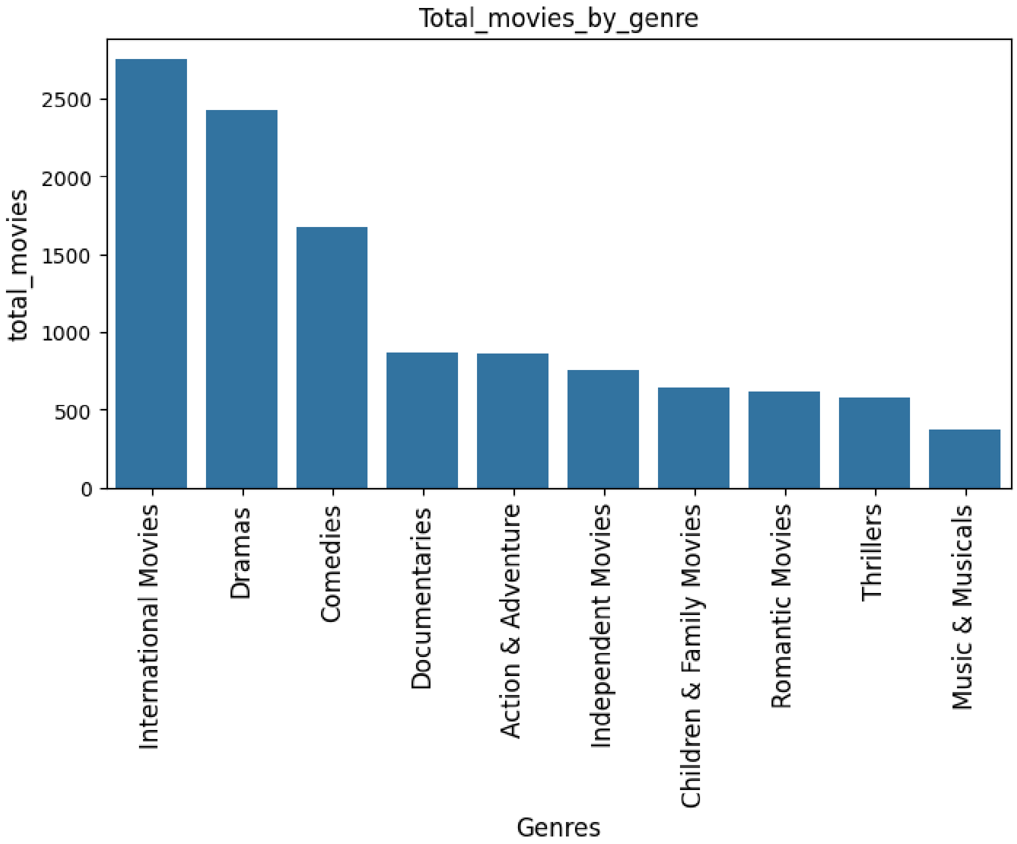
```
# Assuming `movies` and `tv_shows` are pandas DataFrames

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

# Box plot for movies duration with x-axis improvements
sns.boxplot(data=movies, x='duration_in_minutes', ax=ax[0], color='lightblue')
ax[0].set_xlabel('Duration in Minutes', fontsize=12)
ax[0].set_title('Total Movies by Duration')

# Limit the number of ticks to 10 and rotate for readability
ax[0].xaxis.set_major_locator(plt.MaxNLocator(10))
ax[0].tick_params(axis='x', rotation=45)

# Optionally, limit the x-axis range if there are extreme outliers
ax[0].set_xlim(0, 240)  # You can adjust this based on your data

# Add gridlines for better readability
ax[0].grid(True)

# Box plot for TV shows duration
sns.boxplot(data=tv_shows, x='duration_in_seasons', ax=ax[1], color='lightgreen')
ax[1].set_xlabel('Number of Seasons', fontsize=12)
ax[1].set_title('Total TV Shows by Duration')

# Add gridlines for better readability
ax[1].grid(True)

# Adjust layout to avoid overlap
plt.tight_layout()

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



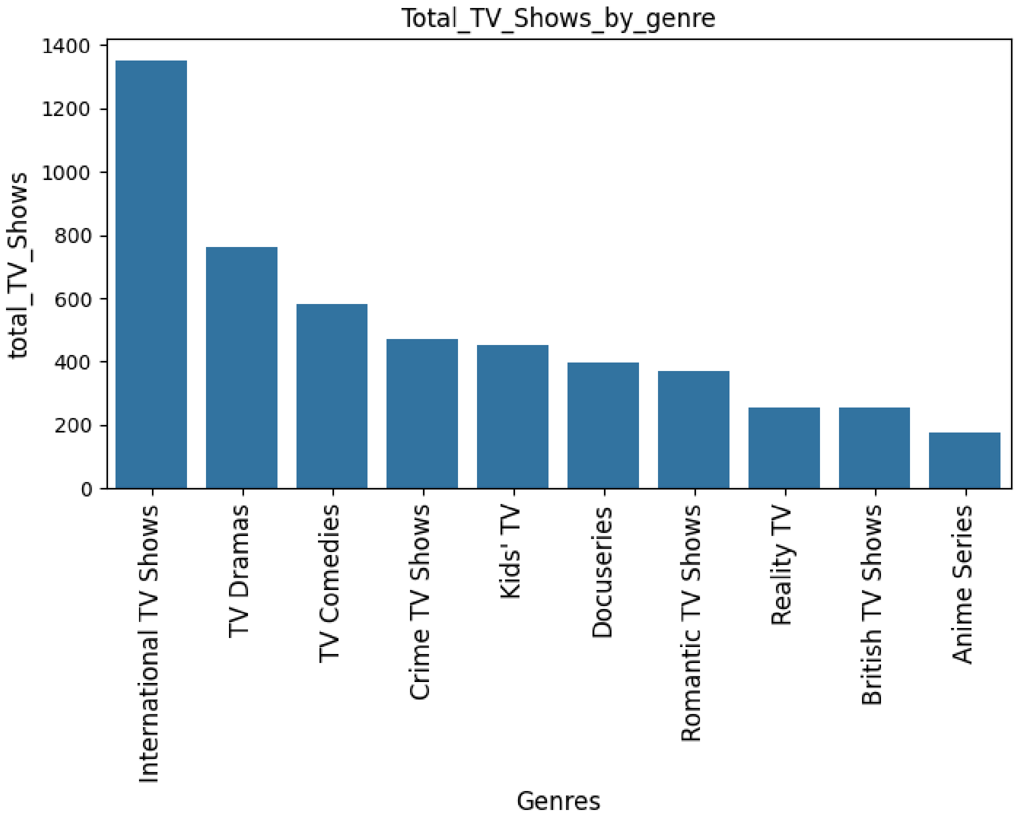
Movie Duration: 50 mins - 150 mins is the range excluding potential outliers (values lying outside the whiskers of boxplot) TV Show Duration: 1-3 seasons is the range for TV shows excluding potential outliers

4.9 Total movies/TV shows in each Genre

```
# Lets check the count for top 10 genres in Movies and TV_shows


top_10_movie_genres = genre_tb[genre_tb['type'] == 'Movie'].listed_in.value_counts().head(10).index
df_movie = genre_tb.loc[genre_tb['listed_in'].isin(top_10_movie_genres)]


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


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



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



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

5. Bivariate Analysis

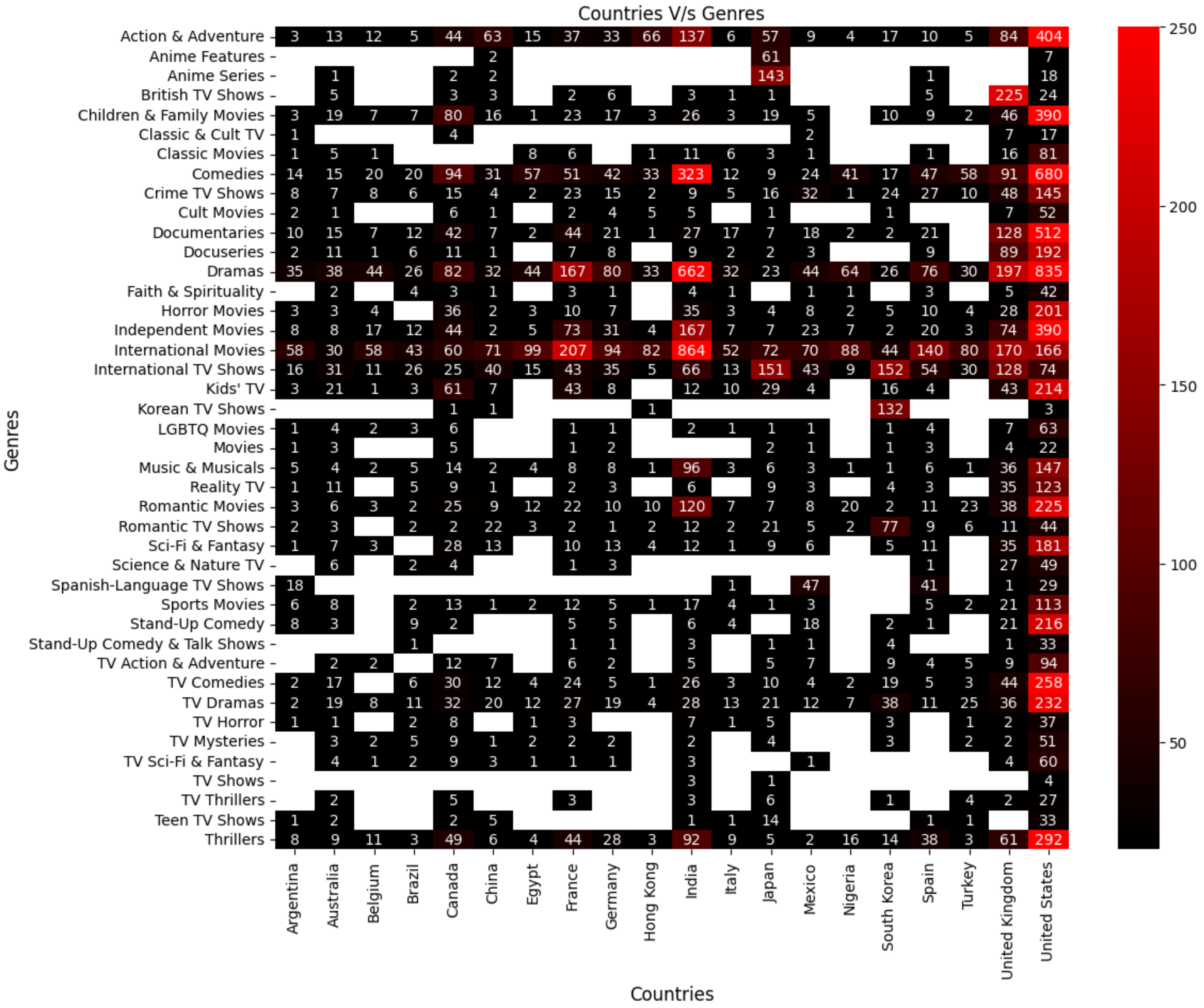
5.1 Lets check popular genres in top 20 countries

```
top_20_country = country_tb.country.value_counts().head(20).index
top_20_country = country_tb.loc[country_tb['country'].isin(top_20_country)]
```

```
x = top_20_country.merge(genre_tb , on = 'show_id').drop_duplicates()
country_genre = x.groupby([ 'country' , 'listed_in'])['show_id'].count().sort_values(ascending = False).reset_index()
country_genre = country_genre.pivot(index = 'listed_in' , columns = 'country' , values = 'show_id')
```

```
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap

plt.figure(figsize = (12,10))
red_black_cmap = LinearSegmentedColormap.from_list("RedBlack", ["black", "red"])
sns.heatmap(data = country_genre , annot = True , fmt=".0f" , vmin = 20 , vmax = 250 , cmap= red_black_cmap)
plt.xlabel('Countries' , fontsize = 12)
plt.ylabel('Genres' , fontsize = 12)
plt.title('Countries V/s Genres' , fontsize = 12)
```

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



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

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

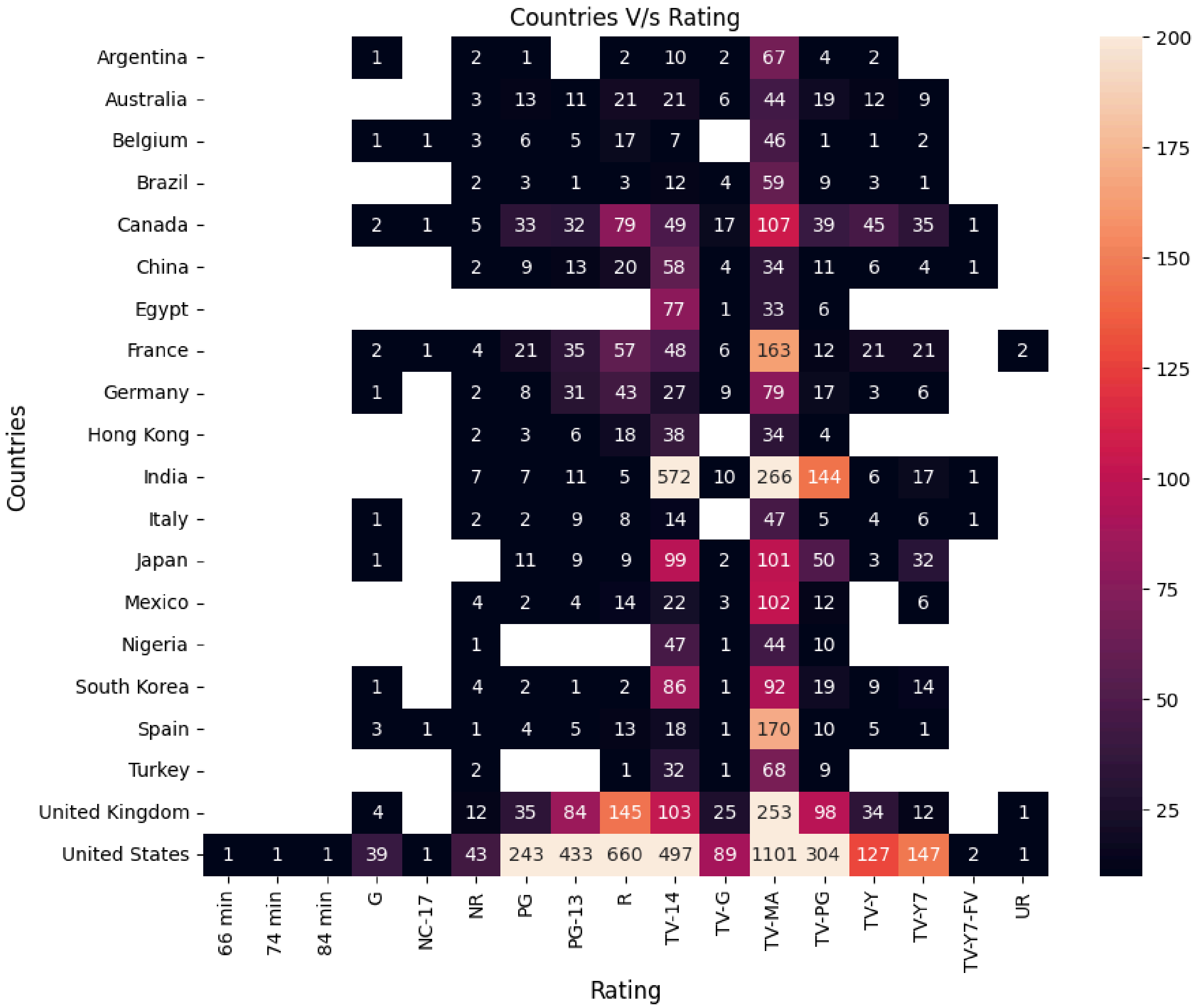
United States and UK have a good mix of almost all genres.

Maximum International movies are produced in India.

```
#5.2 Country-wise Rating of Content
```

```
x = top_20_country.merge(df , on = 'show_id').groupby(['country_x' , 'rating'])['show_id'].count().reset_index()
country_rating = x.pivot(index = ['country_x'] , columns = 'rating' , values = 'show_id')
plt.figure(figsize = (10,8))
sns.heatmap(data = country_rating , annot = True , fmt=".0f"  , vmin = 10 , vmax=200)
plt.ylabel('Countries' , fontsize = 12)
plt.xlabel('Rating' , fontsize = 12)
plt.title('Countries V/s Rating' , fontsize = 12)
```

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



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

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

	country	cast	show_id
49485	United States	Tara Strong	22
48410	United States	Samuel L. Jackson	22
40532	United States	Fred Tatasciore	21
35797	United States	Adam Sandler	20
41743	United States	James Franco	19

```
country_list = ['India'  , 'United Kingdom' , 'Canada' , 'France' , 'Japan']
top_5_actors = x.loc[x['country'].isin(['United States'])].sort_values('show_id' , ascending = False).head(5)
for i in country_list:
    new = x.loc[x['country'].isin([i])].sort_values('show_id' , ascending = False).head(5)
    top_5_actors = pd.concat( [top_5_actors , new] , ignore_index = True)
```

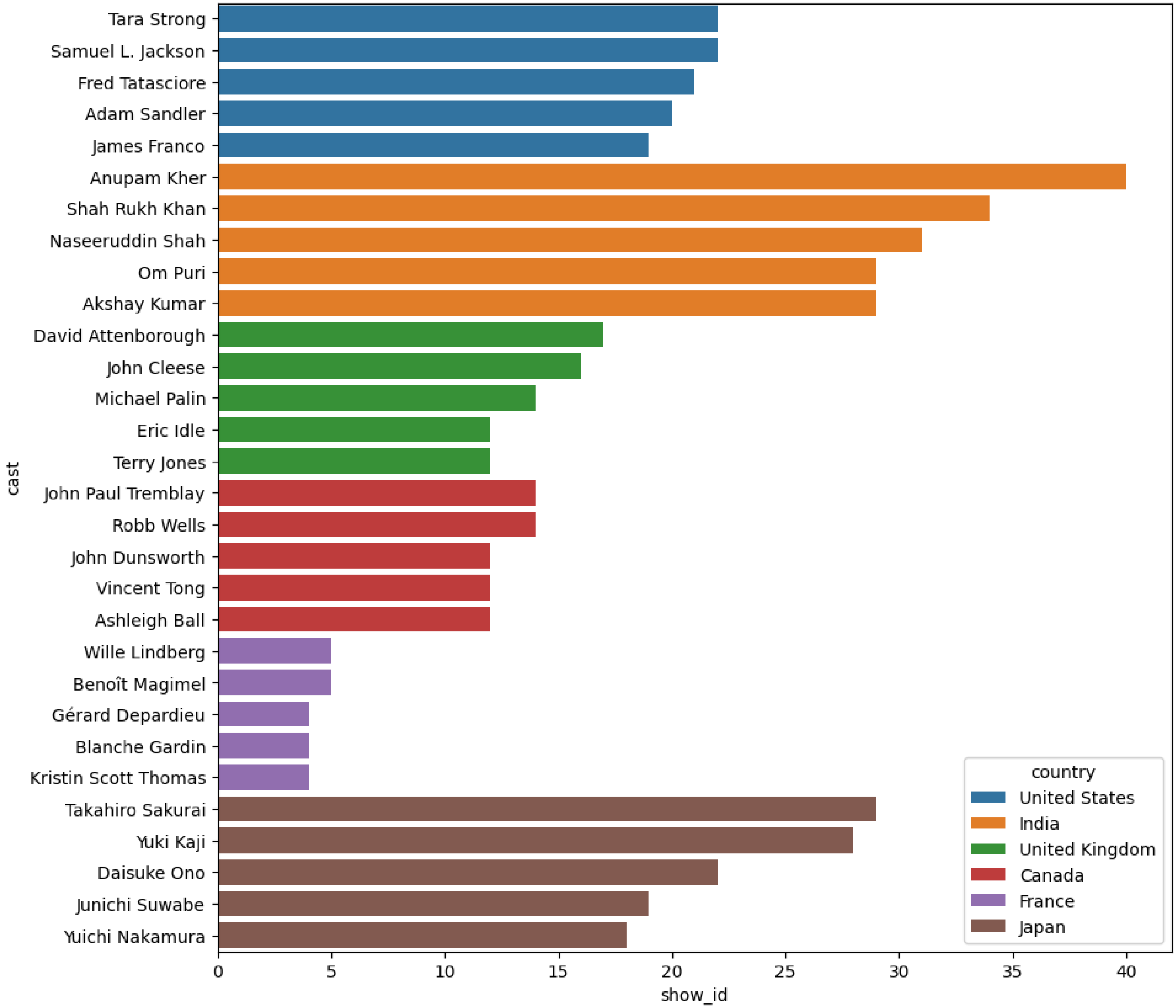
```
# top 5 actors in top countries and their movies/tv shows count
top_5_actors
```

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

Next steps: Generate code with top_5_actors | View recommended plots | New interactive sheet

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

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



```
#5.4 Top 5 directors by Genre
genre_list = [ 'Children & Family Movies', 'Comedies','Dramas', 'International Movies', 'Documentaries' ,
              'International TV Shows', 'Sci-Fi & Fantasy', 'Thrillers', 'Horror Movies']

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

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

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

top_5_dir
```