Netflix Business Case Study

1. Defining Problem Statement and Analyzing Basic Metrics

Netflix Dataset Overview Netflix, a prominent media and video streaming platform with a global subscriber base exceeding 200 million, boasts a collection of over 8,000 movies and TV shows as of mid-2021. The dataset in question comprises comprehensive details on each title, encompassing information such as cast, directors, ratings, release year, duration, and more.

Key Dataset Attributes:

- Show_id: Unique identifier for each Movie/TV Show
- Type: Categorization as a Movie or TV Show
- Title: Name of the Movie/TV Show
- Director: Directorial credits for the Movie
- Cast: Cast members involved in the production
- Country: Country of origin for the Movie/TV Show
- Date_added: Addition date on Netflix
- Release_year: Original release year of the content
- Rating: TV rating assigned to the content
- Duration: Total viewing time in minutes or number of seasons
- Listed_in: Genre classification
- Description: Concise summary of the content

Business problem: Examine the data and derive valuable insights to assist Netflix in strategic decision-making regarding content creation and business expansion across diverse regions.

Project Objectives:

1. Conduct Exploratory Data Analysis (EDA) to gain a comprehensive understanding of the dataset.
2. Extract actionable insights to aid decision-making for content production and business expansion.
3. Provide strategic recommendations for business growth without introducing new information.

By leveraging the existing dataset, the goal is to uncover patterns, preferences, and trends that can inform Netflix's content strategy, enabling them to tailor their offerings to specific audiences and optimize business outcomes across different countries.

2. Observations on the shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), Missing Value Detection,Statistical Summary

Importing the libraries we need

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

Loading Netflix Dataset

```
df = pd.read_csv('netflix.csv')
```

```
df.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rat
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-

```
df.shape
```

```
(8807, 12)
```

The dataset consists of 8807 rows and 12 columns, where each row represents information about a movie or TV show. The columns include 'show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added', 'release_year', 'rating', 'duration', 'listed_in', and 'description'.

```
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
```

```
df.nunique()
```

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

The dataset contains a mix of categorical and numerical columns. Notably, the 'show_id' column has all unique values, and the 'title' column also has all unique values, equal to the total number of rows. This indicates that each entry in the dataset is distinct, comprising a total of 8807 movies and TV shows

```
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

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

	show_id	type	title	director	cast	country	date_added	rating	dura
count	8807	8807	8807	6173	7982	7976	8797	8803	
unique	8807	2	8807	4528	7692	748	1767	17	
top	s1	Movie	Dick Johnson	Rajiv	David	United	January 1,	TV-MA	1 Se

The 'describe' function provides insights into categorical columns, including counts, unique values, top values, and frequency. For instance, the 'rating' column contains various content ratings, and 'Not Available' is used for entries with missing ratings.

Data Cleaning

In the data cleaning phase, we address missing values in the dataset:

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

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

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

	show_id	type	title	director	cast	country	date_added	release_year	rating
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	74 min

Three missing values are identified in the 'duration' column. Additionally, it is observed that, by mistake, these missing duration entries have been erroneously entered into the 'rating' column.

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

	show_id	type	title	director	cast	country	date_added	release_year	ratir
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	N Availab

Addressing the absence of information in the 'rating' column, we aim to fill the null values to enhance the completeness of our dataset.

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

	show_id	type	title	director	cast	country	date_added	release_yea
5989	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Oprah Winfrey, Ava DuVernay	NaN	January 26, 2017	20
6827	s6828	TV Show	Gargantia on the Verdurous	NaN	Kaito Ishikawa, Hisako Kanemoto,	Japan	December 1, 2016	20

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

```
Int64Index([5989, 6827, 7312, 7537], dtype='int64')
```

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

	show_id	type	title	director	cast	country	date_added	release_yea
5989	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Oprah Winfrey, Ava DuVernay	NaN	January 26, 2017	20
6827	s6828	TV Show	Gargantia on the Verdurous	NaN	Kaito Ishikawa, Hisako Kanemoto,	Japan	December 1, 2016	20

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

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

In the 'rating' column, it has been observed that 'NR' (Not Rated) is equivalent to 'UR' (Unrated). To ensure consistency, we opt to update occurrences of 'UR' to 'NR' in the 'rating' column.

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

```
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: rating, dtype: int64
```

Removed null values from the 'date_added' column.

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

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

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

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

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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   int64         
 2   month_added  8797 non-null   int64         
dtypes: datetime64[ns](1), int64(2)
memory usage: 274.9 KB
```

Since all entries in the 'date_added' column adhere to the date format, we're converting its data type from a general object to a datetime format. Following this, we're enhancing our dataset by introducing two new columns: 'year_added' (derived from extracting the year from 'date_added') and 'month_added' (derived from extracting the month from 'date_added'). This allows for more detailed insights into the temporal aspects of our data.

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

```
show_id           0
type              0
title             0
director       2624
cast            825
country         830
```

```
date_added         0
release_year       0
rating             0
duration           0
listed_in          0
description        0
year_added         0
month_added        0
dtype: int64
```

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

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

The percentage of null values in certain columns remains notable even after our data cleaning efforts. Specifically:

- Approximately 9% of entries lack information on the country of origin.
- The director's names are missing for about 30% of the entries.
- Similarly, cast information is absent for 9% of the entries. These substantial gaps in data prompt the need for further consideration and potential strategies to handle or impute these missing values for a more comprehensive dataset.

3. Data Exploration and Non Graphical Analysis

Non-Graphical Analysis: Value counts and unique attributes

```
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()
```

```
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: duration, Length: 205, dtype: int64
```

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

```
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: duration, dtype: int64
```

```
movies['duration'] = movies['duration'].str[:-3]
movies['duration'] = movies['duration'].astype('float')
tv_shows['duration'] = tv_shows.duration.str[:-7].apply(lambda x : x.strip())
tv_shows['duration'] = tv_shows['duration'].astype('float')
tv_shows.rename({'duration': 'duration_in_seasons'} ,axis = 1 , inplace = True)
movies.rename({'duration': 'duration_in_minutes'} ,axis = 1 , inplace = True)
tv_shows.duration_in_seasons
```

```
<ipython-input-56-f8ebf267ef00>: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-cc
  movies['duration'] = movies['duration'].str[:-3]
<ipython-input-56-f8ebf267ef00>: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-cc
  movies['duration'] = movies['duration'].astype('float')
<ipython-input-56-f8ebf267ef00>: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-cc
  tv_shows['duration'] = tv_shows.duration.str[:-7].apply(lambda x : x.strip())
<ipython-input-56-f8ebf267ef00>:4: 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-cc
  tv_shows['duration'] = tv_shows['duration'].astype('float')
<ipython-input-56-f8ebf267ef00>:5: 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-cc
  tv_shows.rename({'duration': 'duration_in_seasons'} ,axis = 1 , inplace = True)
<ipython-input-56-f8ebf267ef00>:6: 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-cc
  movies.rename({'duration': 'duration_in_minutes'} ,axis = 1 , inplace = True)
1       2.0
2       1.0
3       1.0
4       2.0
5       1.0
       ...
8795    2.0
8796    2.0
8797    3.0
8800    1.0
8803    2.0
Name: duration_in_seasons, Length: 2666, dtype: float64
```

```
movies.duration_in_minutes
```

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

Given the distinct formats for duration in movies (in minutes) and TV shows (in seasons), we can standardize the representation. To explore the timeline of movie additions on Netflix, we aim to determine when the first movie was added and identify the most recent movie addition based on the provided dataset duration.

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

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

What is the release year of the oldest and most recent movie or TV show available on Netflix?

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

```
(1925, 2021)
```

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

	show_id	type	title	director	cast	country	date_added	release_yea
4250	s4251	TV Show	Pioneers: First Women Filmmakers*	NaN	NaN	NaN	2018-12-30	192
966	s967	Movie	Get the Grift	Pedro Antonio	Marcus Majella, Samantha Schmütz, Caito Mainie...	Brazil	2021-04-28	202
967	s968	TV Show	Headspace Guide to Sleep	NaN	Evelyn Lewis Prieto	NaN	2021-04-28	202
968	s969	TV Show	Sexify	NaN	Aleksandra Skraba, Maria Sobocińska, Sandra Dr...	Poland	2021-04-28	202
972	s973	TV Show	Fatma	NaN	Burcu Biricik, Uğur Yücel, Mehmet Yılmaz Ak, H...	Turkey	2021-04-27	202

What are the various ratings assigned to content on Netflix for each type (movies and TV shows)? Additionally, explore the number of releases for each content type.

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

```
type     rating
Movie    G                 41
         NC-17              3
         NR                78
         Not Available      5
         PG               287
         PG-13            490
         R                797
         TV-14           1427
         TV-G             126
         TV-MA           2062
```

```
               TV-PG             540
               TV-Y              131
               TV-Y7             139
               TV-Y7-FV            5
      TV Show  NR                  4
               Not Available       2
               R                   2
               TV-14             730
               TV-G               94
               TV-MA            1143
               TV-PG             321
               TV-Y              175
               TV-Y7             194
               TV-Y7-FV            1
      Name: show_id, dtype: int64
```

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

```
      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: country, Length: 748, dtype: int64
```

We notice that many movies are made in multiple countries, and this information is stored in the 'country' column as comma-separated values. To make it easier to analyze how many movies were produced in each country, we can use the "explode" function in pandas. This function allows us to split the 'country' column into different rows.

As a result, we are creating a separate table specifically for countries to prevent the duplication of records in our original table after using the explode function.

```
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-63-88f820136e36>: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_
  country_tb.dropna(inplace = True)
<ipython-input-63-88f820136e36>: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_
  country_tb['country'] = country_tb['country'].apply(lambda x : x.split(','))
```

	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 × 3 columns

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

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

```
country_tb = country_tb.loc[country_tb['country'] != '']
country_tb['country'].nunique()
```

```
122
```

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

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

122 rows × 2 columns

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

```
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: director, Length: 4528, dtype: int64
```

Netflix offers movies from a total of 122 countries. We aim to determine the count of movies and TV shows available in each country. Additionally, due to some movies having multiple directors, the 'director' column contains comma-separated names. To address this, we plan to use the explode function on the 'director' column. However, this could generate duplicate records in the original table, leading us to create a distinct table dedicated to directors.

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

```
<ipython-input-68-8de37009c172>: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_
  dir_tb.dropna(inplace = True)
<ipython-input-68-8de37009c172>: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_
  dir_tb['director'] = dir_tb['director'].apply(lambda x : x.split(','))
```

	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 × 3 columns

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

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

```
dir_tb.director.apply(lambda x : True if len(x) == 0 else False).value_counts()
```

```
False    6978
Name: director, 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

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

```
4993
```

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

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

4993 rows × 2 columns

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

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

```
<ipython-input-76-e08a3d46054c>: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_
  genre_tb['listed_in'] = genre_tb['listed_in'].apply(lambda x : x.split(','))
```

	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 × 3 columns

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

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

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

```
42
```

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

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

The dataset comprises a total of 4,993 unique directors. To delve deeper into the directorial landscape, we explore the count of movies and TV shows directed by each director. Utilizing the 'listed_in' column, we gain insights into the diverse genres present in the dataset. In total, there are 42 genres, with movies spanning 20 genres and TV shows covering 22 genres.

```
x = genre_tb.groupby(['listed_in' , 'type'])['show_id'].count().reset_index()
x.pivot(index = 'listed_in' , columns = 'type' , values = 'show_id').sort_index()
```

Documentaries	869.0	NaN
Docuseries	NaN	394.0
Dramas	2427.0	NaN
Faith & Spirituality	65.0	NaN
Horror Movies	357.0	NaN
Independent Movies	756.0	NaN
International Movies	2752.0	NaN
International TV Shows	NaN	1350.0
Kids' TV	NaN	449.0
Korean TV Shows	NaN	151.0
LGBTQ Movies	102.0	NaN
Movies	57.0	NaN
Music & Musicals	375.0	NaN
Reality TV	NaN	255.0
Romantic Movies	616.0	NaN
Romantic TV Shows	NaN	370.0
Sci-Fi & Fantasy	243.0	NaN
Science & Nature TV	NaN	92.0
Spanish-Language TV Shows	NaN	173.0
Sports Movies	219.0	NaN
Stand-Up Comedy	343.0	NaN
Stand-Up Comedy & Talk Shows	NaN	56.0
TV Action & Adventure	NaN	167.0
TV Comedies	NaN	574.0
TV Dramas	NaN	762.0
TV Horror	NaN	75.0
TV Mysteries	NaN	98.0
TV Sci-Fi & Fantasy	NaN	83.0
TV Shows	NaN	16.0
TV Thrillers	NaN	57.0
Teen TV Shows	NaN	69.0
Thrillers	577.0	NaN

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

```
<ipython-input-81-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_
  cast_tb.dropna(inplace = True)
<ipython-input-81-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_
  cast_tb['cast'] = cast_tb['cast'].apply(lambda x : x.split(','))
```

	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 × 3 columns

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

	show_id	type	cast

```
cast_tb.cast.nunique()
```

```
36403
```

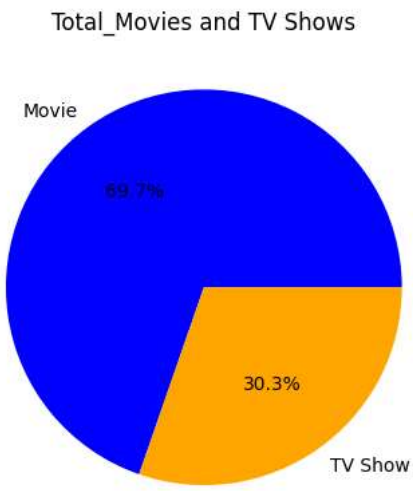
```
x = cast_tb.groupby(['cast' , 'type'])['show_id'].count().reset_index()
x.pivot(index = 'cast' , columns = 'type' , values = 'show_id').sort_values('TV Show' , ascending = False)
```

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

36403 rows × 2 columns

4. Visual Analysis - Univariate, Bivariate after pre-processing of the data

```
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()
```



Observation: Around 70% of content is Movies, and around 30% is TV shows.

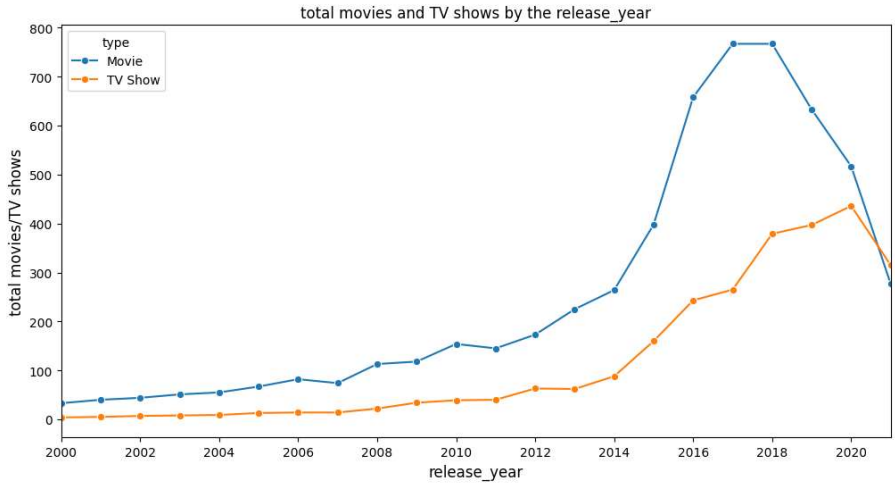
```
d = df.groupby(['type' , 'release_year'])['show_id'].count().reset_index()
d.rename({'show_id' : 'total movies/TV shows'}, axis = 1 , inplace = True)
d
```

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

119 rows × 3 columns

Line Plot

```
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()
```

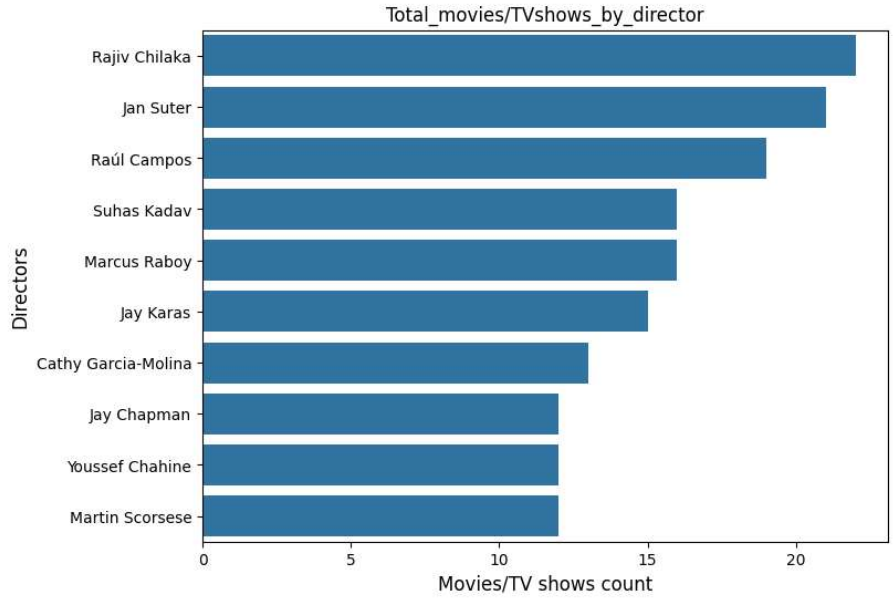


Observations:

Content added on Netflix surged drastically after 2015. 2019 marks the highest number of movies and TV shows added. A drop in content added in 2020 and 2021, possibly due to the pandemic. TV shows have not dropped as drastically as movies, indicating a focus on TV shows in recent years.

Count Plot

```
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()
```



Overall, this code segment visualizes the total count of movies/TV shows directed by the top 10 directors in the dataset dir_tb using a vertical countplot.

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

```
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: director, Length: 4993, dtype: int64
```

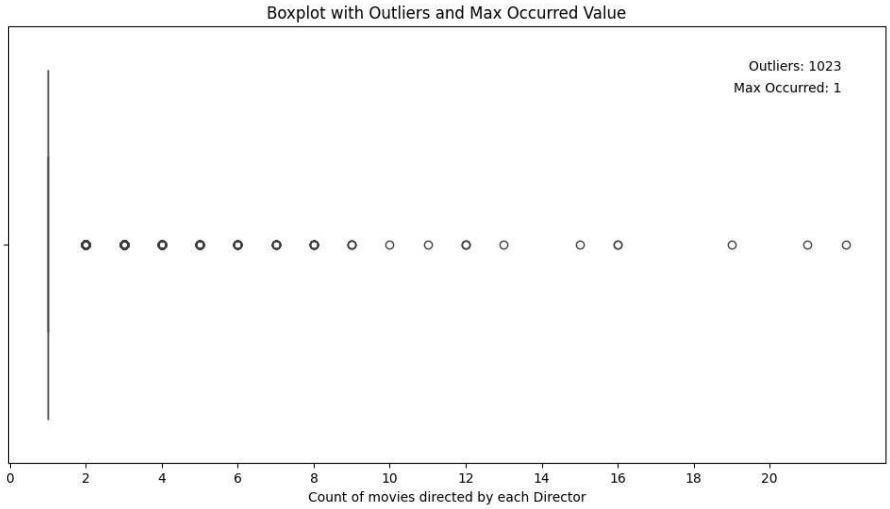
```
def calculate_outliers(data):
    q1 = np.percentile(data, 25)
    q3 = np.percentile(data, 75)

    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    outliers = [value for value in data if value < lower_bound or value > upper_bound]

    return outliers


def calculate_max_occurred_value(data):
    unique_values, value_counts = np.unique(data, return_counts=True)
    max_count_index = np.argmax(value_counts)
    max_occurred_value = unique_values[max_count_index]

    return max_occurred_value
outliers = calculate_outliers(x)
max_occurred_value = calculate_max_occurred_value(x)
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')
outliers = calculate_outliers(x)
max_occurred_value = calculate_max_occurred_value(x)
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")
plt.show()
```



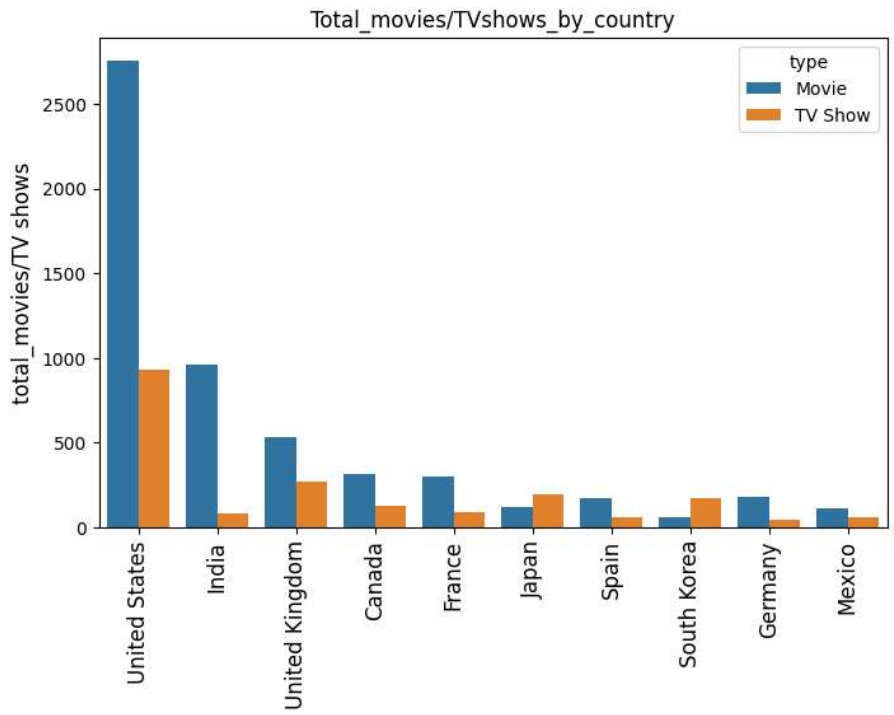

```
top_10_country = country_tb.country.value_counts().head(10).index
df_new = country_tb.loc[country_tb['country'].isin(top_10_country)]
```

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

type	Movie	TV Show
country		
United States	2752	932
India	962	84
United Kingdom	534	271
Canada	319	126
France	303	90
Germany	182	44
Spain	171	61
Japan	119	198
Mexico	111	58
South Korea	61	170

Count Plot

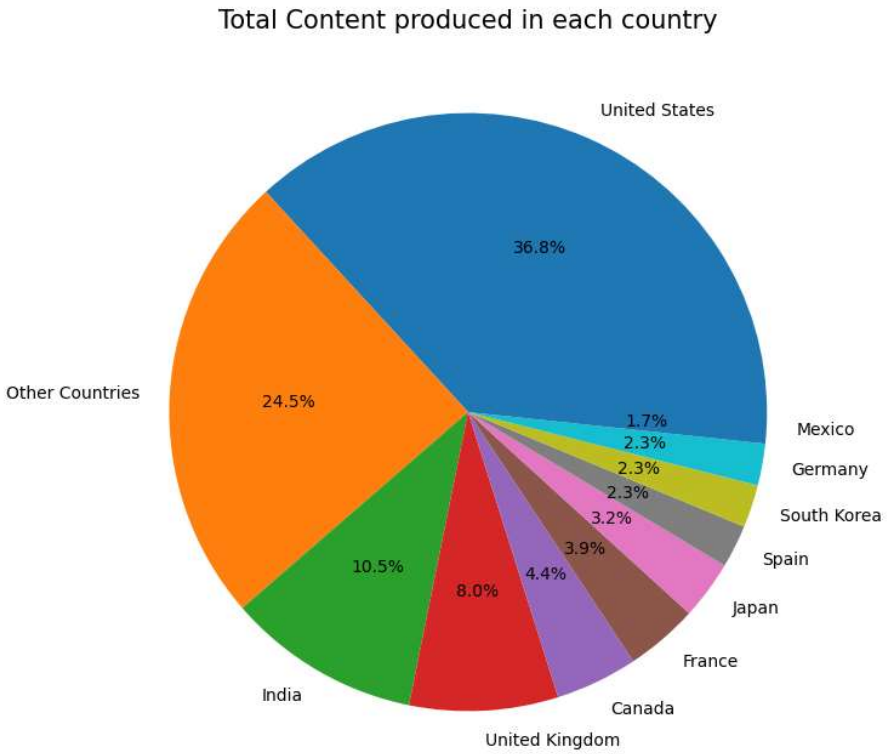
```
plt.figure(figsize= (8,5))
sns.countplot(data = df_new , x = 'country' , order = top_10_country , hue = 'type')
plt.xticks(rotation = 90 , fontsize = 12)
plt.ylabel('total_movies/TV shows' , fontsize = 12)
plt.xlabel('')
plt.title('Total_movies/TVshows_by_country')
plt.show()
```



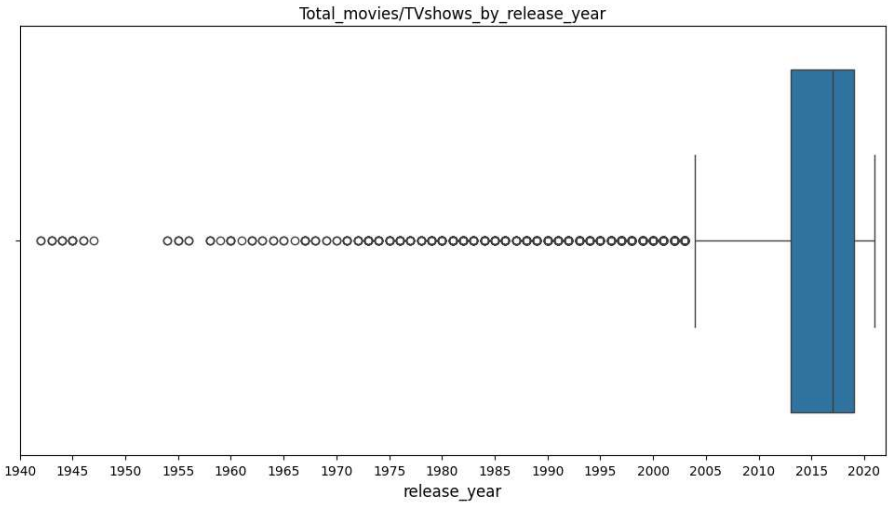
Pie Chart

```
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()
```

```
<ipython-input-97-2803d6277dcc>: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_
  country_tb['cat'] = country_tb['country'].apply(lambda x : x if x in top_10_country el
```

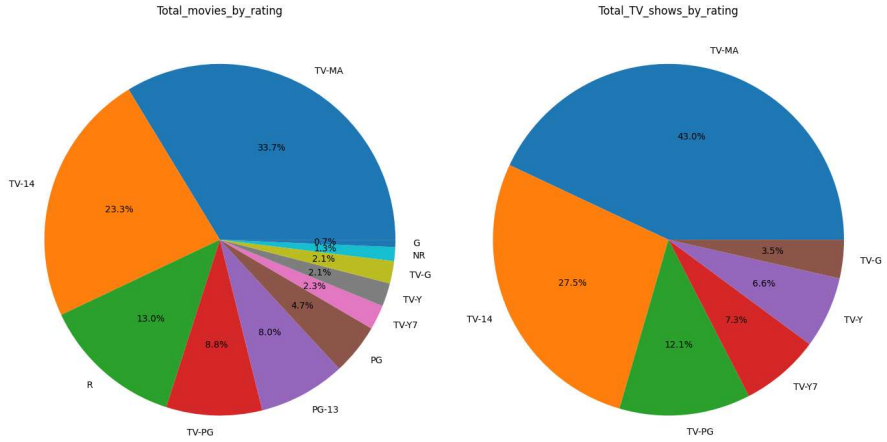



```
plt.figure(figsize= (12,6))
sns.boxplot(data = df , x = 'release_year')
plt.xlabel('release_year' , fontsize = 12)
plt.title('Total_movies/TVshows_by_release_year')
plt.xticks(np.arange(1940 , 2021 , 5))
plt.xlim((1940 , 2022))
plt.show()
```

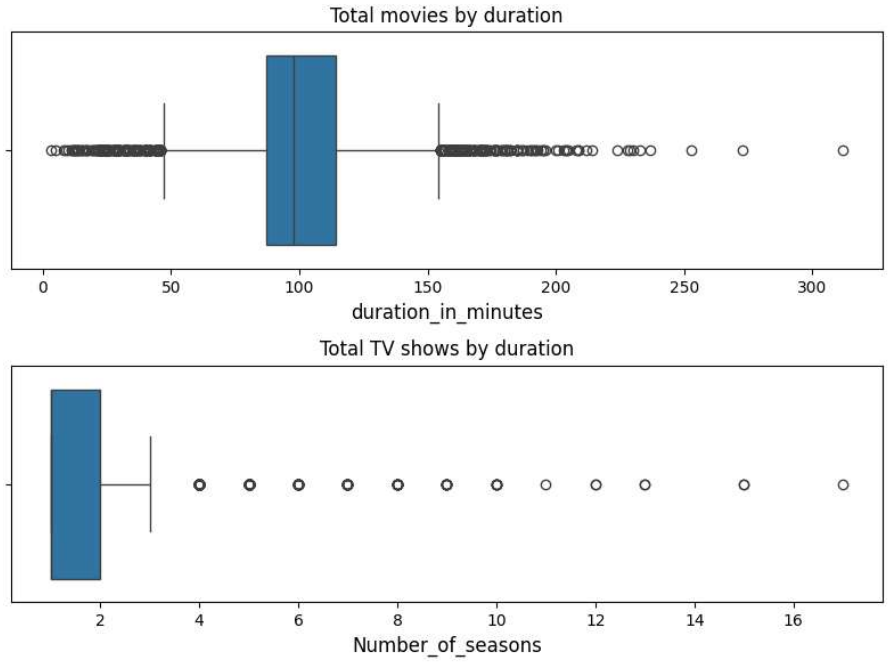


Pie Chart

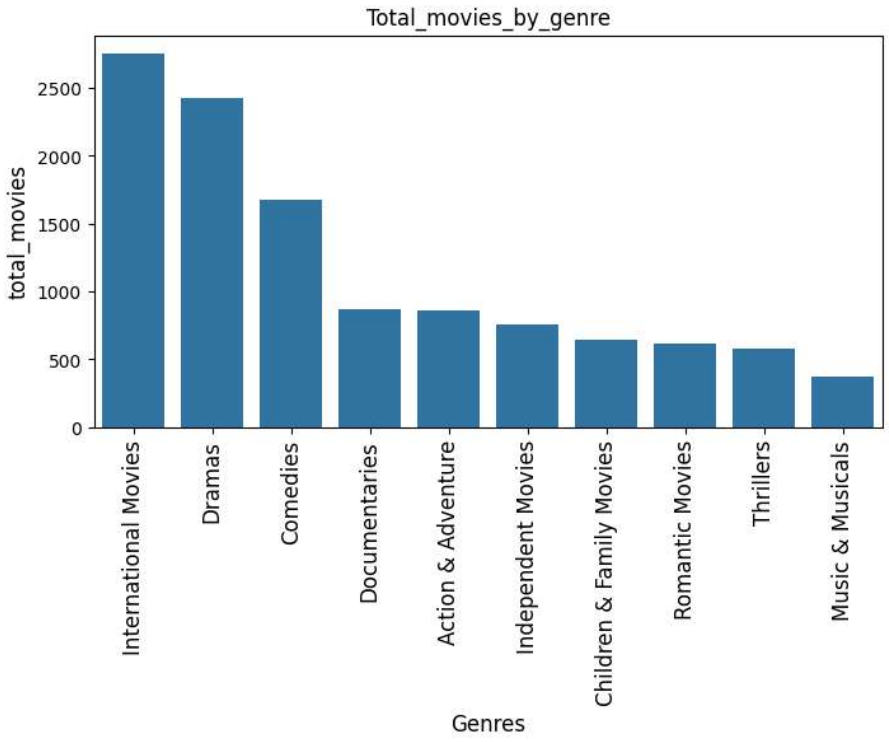
```
m = movies.loc[~movies.rating.isin(['Not Available' , 'NC-17' , 'TV-Y7-FV'])]
m = m.rating.value_counts()
t = tv_shows.loc[~tv_shows.rating.isin(['Not Available' , 'R' , 'NR', 'TV-Y7-FV'])]
t = t.rating.value_counts()


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()
```



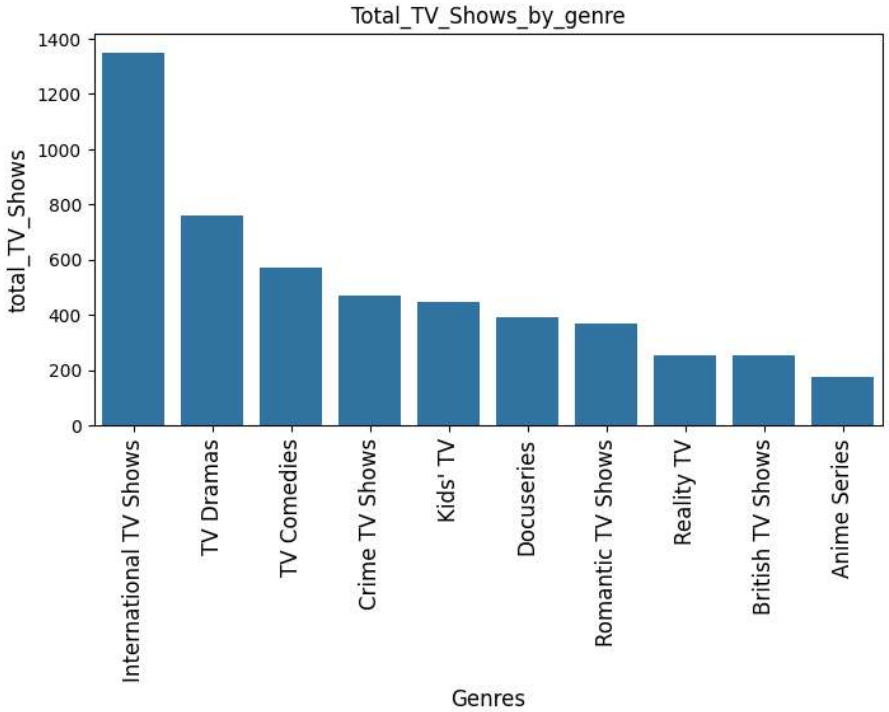

```
fig, ax = plt.subplots(2,1, figsize=(8,6))

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()
```




```
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()
```



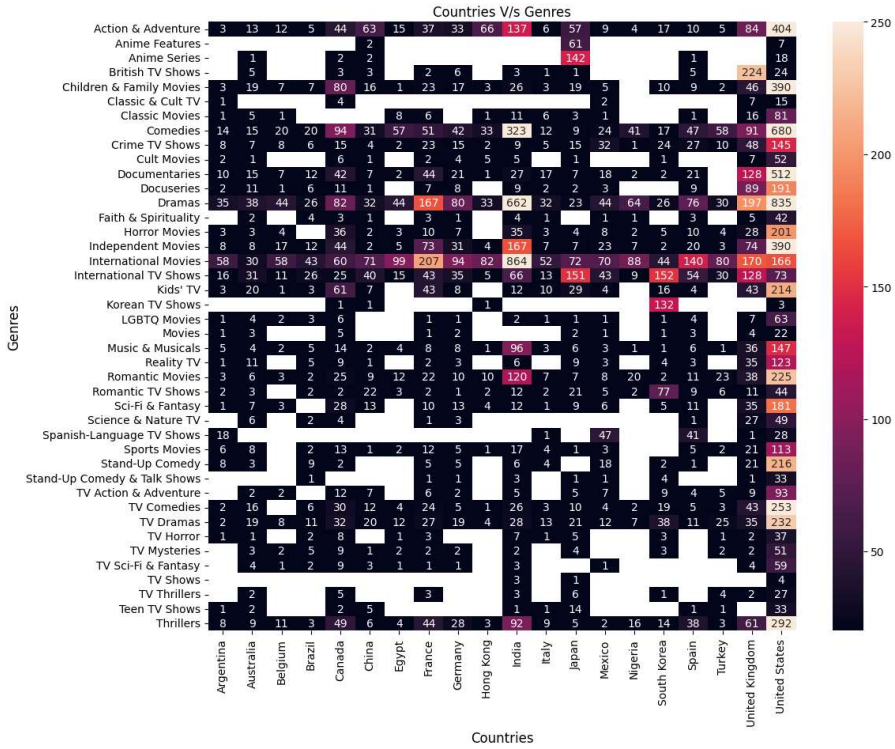

```
top_20_country = country_tb.country.value_counts().head(20).index
top_20_country = country_tb.loc[country_tb['country'].isin(top_20_country)]
```

```
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')
```

Heat Map

```
plt.figure(figsize = (12,10))
sns.heatmap(data = country_genre , annot = True , fmt=".0f" , vmin = 20 , vmax = 250 )
plt.xlabel('Countries' , fontsize = 12)
plt.ylabel('Genres' , fontsize = 12)
plt.title('Countries V/s Genres' , fontsize = 12)
```

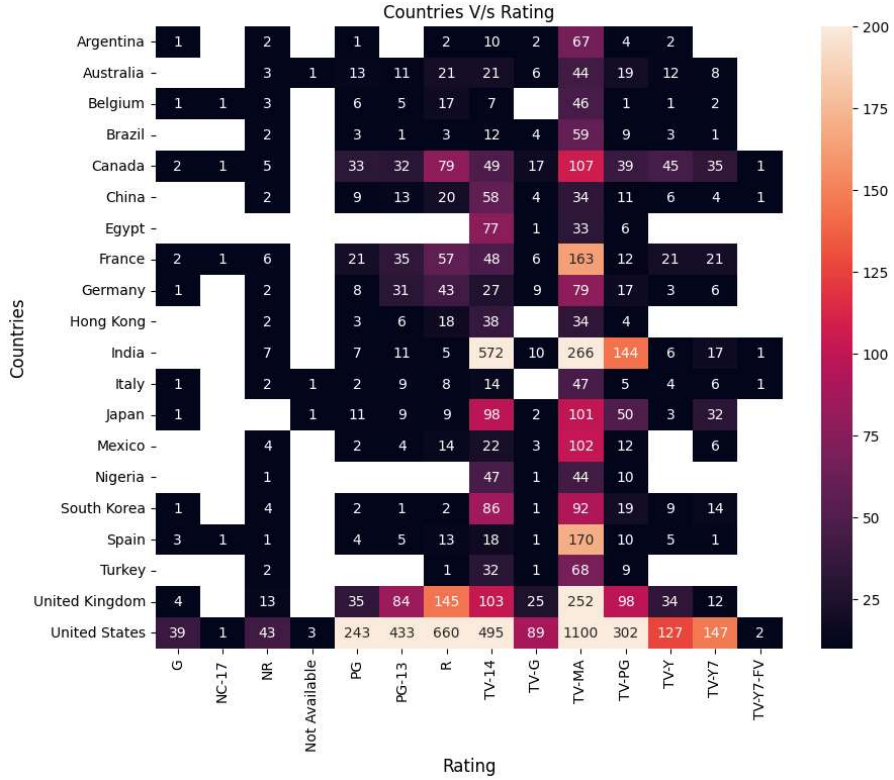
```
Text(0.5, 1.0, 'Countries V/s Genres')
```



Heat Map

```
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')
```




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

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

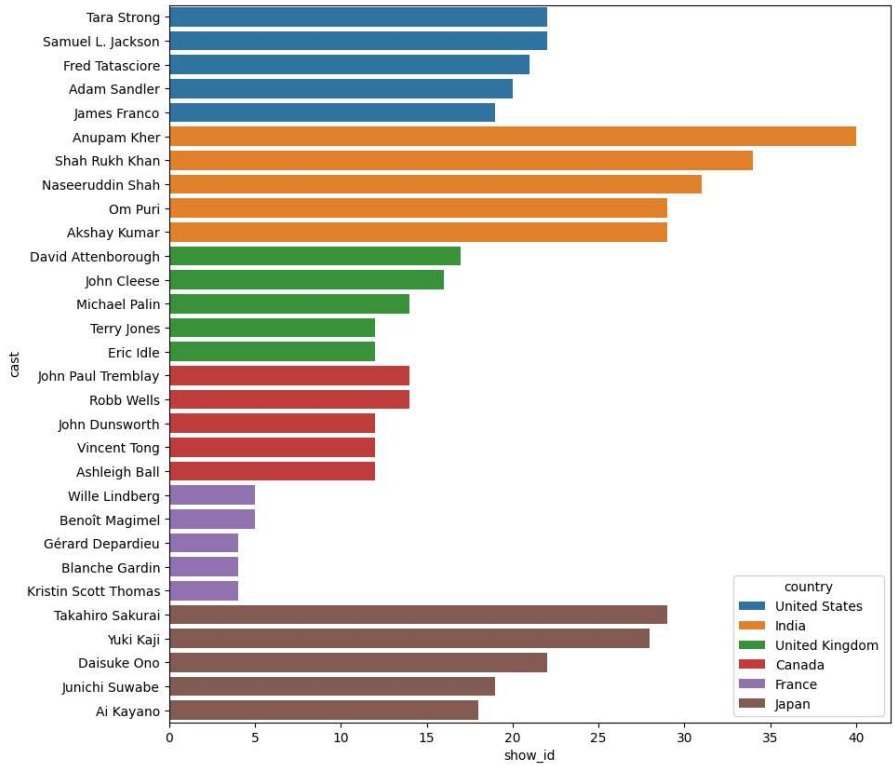
```
country_list = ['India'  , 'United Kingdom' , 'Canada' , 'France' , 'Japan']
top_5_actors = x.loc[x['country'].isin(['United States'])].sort_values('show_id' , ascending = False).head(5)
for i in country_list:
    new = x.loc[x['country'].isin([i])].sort_values('show_id' , ascending = False).head(5)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

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

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

Bar Plot

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

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




```
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
```

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

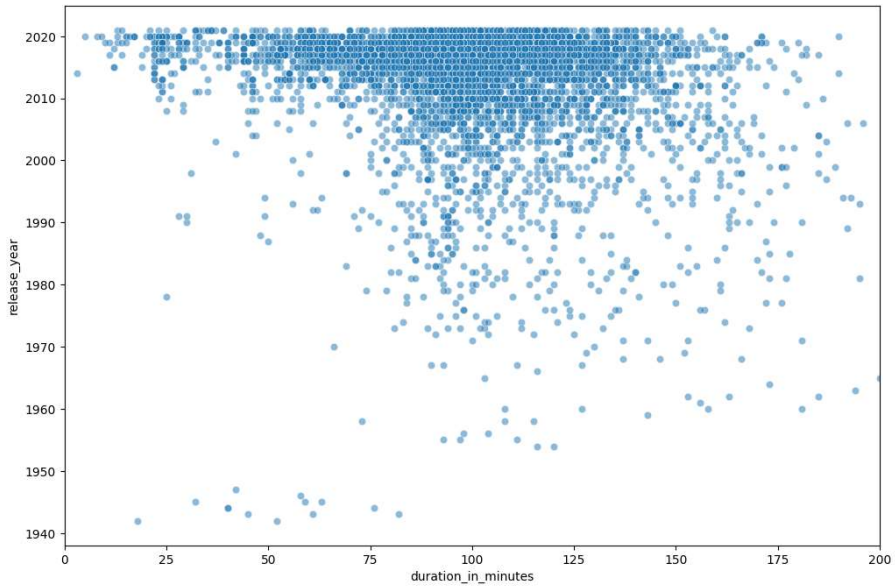
```
x = genre_tb.merge(country_tb , on = 'show_id').drop_duplicates()
x = x.groupby(['country' , 'listed_in'])['show_id'].count().reset_index()
x.loc[x['country'] == 'United States'].sort_values('show_id' , ascending = False).head(5)

country_list = ['India'  , 'United Kingdom' , 'Canada' , 'France' , 'Japan']
top_5_genre = x.loc[x['country'].isin(['United States'])].sort_values('show_id' , ascending = False).head(5)

for i in country_list:
    new = x.loc[x['country'] == i].sort_values('show_id' , ascending = False).head(5)
    top_5_genre = pd.concat( [top_5_genre , new] , ignore_index = True)
top_5_genre
```

	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

Scatter Plot

```
plt.figure(figsize = (12,8))
sns.scatterplot(x="duration_in_minutes", y="release_year", data=movies, alpha=0.5)
plt.xlim((0,200))
```

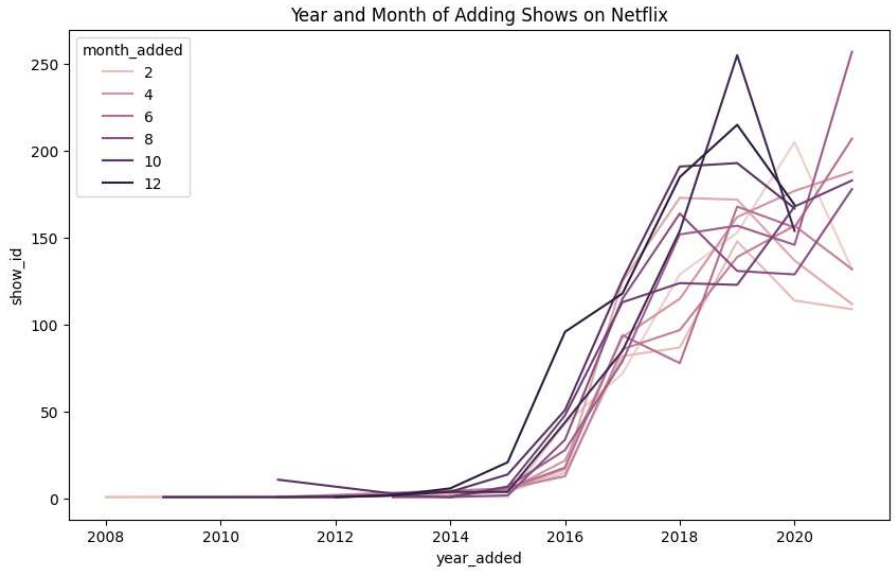
```
(0.0, 200.0)
```



Line Plot

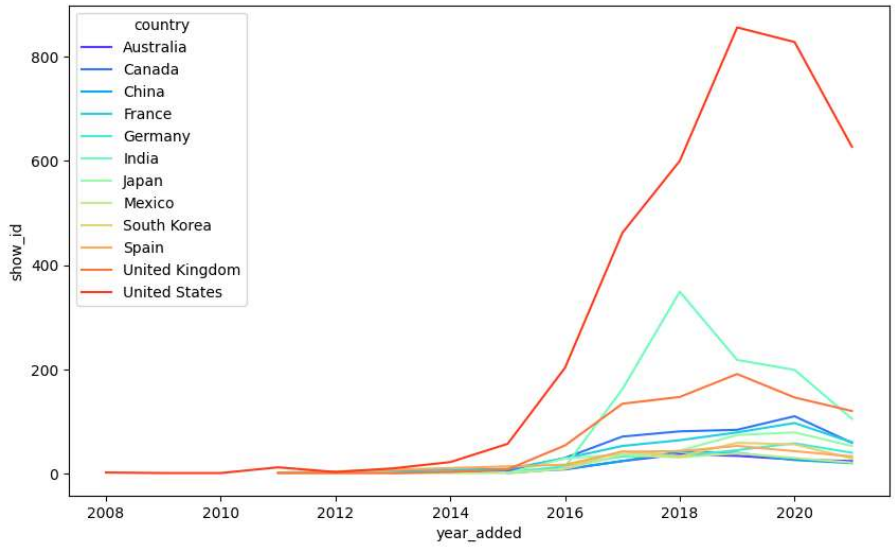
```
month_year = df.groupby(['year_added' , 'month_added'])['show_id'].count().reset_index()
plt.figure(figsize = (10,6))
sns.lineplot(data=month_year, x = 'year_added', y = 'show_id', hue='month_added')
plt.title('Year and Month of Adding Shows on Netflix')
```

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




```
country_list = country_tb.country.value_counts().head(12).index
top_12_country = country_tb.loc[country_tb['country'].isin(country_list)]
country_year = top_12_country.merge(df , on = 'show_id')[['show_id','country_x' ,'type_x' , 'year_added' ]]
country_year.columns = ['show_id', 'country', 'type', 'year_added']
country_year = country_year.groupby(['country' , 'year_added'])['show_id'].count().reset_index()
plt.figure(figsize = (10,6))
sns.lineplot(data = country_year , x = 'year_added' , y = 'show_id' , hue = 'country' , palette ='rainbow' )
```

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




```
country_year.head()
```