Problem Statement:

Analyze the data to generate insights for Netflix to decide on show/movie production and business expansion in different countries.

Description:

Netflix is one of the most popular media and video streaming platforms. They have over 8000 movies or tv shows available on theirplatform, as of mid-2021, they have over 222M Subscribers globally. This tabular dataset consists of listings of all the movies and tv shows availableon Netflix, along with details such as - cast, directors, ratings, release year, duration, etc.

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

Show_id: Unique ID for every Movie / Tv Show Type: Identifier - A
Movie or TV Show
Title: Title of the Movie / Tv Show Director: Director of the
Movie
Cast: Actors involved in the movie/show
Country: Country where the movie/show was produced Date_added: Date it
was added on Netflix Release_year: Actual Release year of the movie/show
Rating: TV Rating of the movie/show
Duration: Total Duration - in minutes or number of seasons Listed_in: Genre
Description: The summary description

Objective:

Provide Useful Insights and Business recommendations that can help the business grow.

1. Importing Libraries , Loading the data and Basic Observations

```
import numpy as np import
pandas as pd
import matplotlib.pyplot as plt import
seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('Documents/Documents/Python_Scalar/netflix.csv') df.head()
```

```
  show_id     type                 title         director  \
0      s1    Movie  Dick Johnson Is Dead  Kirsten Johnson
1      s2  TV Show         Blood & Water              NaN
```

```
  2 s3 s4  TV Show    Ganglands Julien Leclercq Jailbirds
  3        TV Show                      New Orleans      NaN
  4    s5  TV Show                    Kota Factory       NaN
```

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

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

```
                                           listed_in  \
0                                      Documentaries
1    International TV Shows, TV Dramas, TV Mysteries
2  Crime TV Shows, International TV Shows, TV Act...
3                             Docuseries, Reality TV
4  International TV Shows, Romantic TV Shows, TV ...
```

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

df.shape

```
(8807, 12)
```

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
```

```
These are the total features of our dataset. It is seen that the
show_id column has all unique values,
The title column has all unique values, i.e., 8807, which equates with
the total rows in the dataset.
A total of 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
```

```
It gives idea of release year of the content ranges between what
timeframe. Rest all the columns are having categorical data.
```

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

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

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

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

                                              description
count                                                8807
unique                                               8775
top     Paranormal activity at a lush, abandoned prope...
freq                                                    4

Overall null values in each column of the dataset -
```

Data Cleaning

```
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()]
```

Empty DataFrame

```
Columns: [show_id, type, title, director, cast, country, date_added,
release_year, rating, duration, listed_in, description]
Index: []
```

```
ind = df[df['duration'].isna()].index
df.loc[ind] = df.loc[ind].fillna(method = 'ffill' , axis = 1)
# replaced the wrong entries done in the rating column
df.loc[ind ,'rating'] = 'Not Available'
```

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

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

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

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

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

Fill the null values in rating column

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

```
Index([], dtype='int64')
```

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

```
Empty DataFrame
Columns: [show_id, type, title, director, cast, country, date_added,
release_year, rating, duration, listed_in, description]
Index: []
```

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

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

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

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

```
rating
TV-MA            3207
TV-14            2160
TV-PG             863
R                 799
PG-13             490
TV-Y7             334
TV-Y              307
PG                287
TV-G              220
NR                 83
G                  41
Not Available       7
TV-Y7-FV            6
NC-17               3
Name: count, dtype: int64
```

dropped the null from 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: count, Length: 1767, dtype: int64
```

```
df['date_added'] = df['date_added'].astype(str) # Convert the column 
to string
```

```
df['date_added'] = df['date_added'].str.strip()  # Remove
leading/trailing whitespaces

# Replace NaN values with an empty string, then strip whitespaces
df['date_added'] = df['date_added'].fillna('').astype(str).str.strip()

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

df['date_added']
```

```
0      2021-09-25
1      2021-09-24
2      2021-09-24
3      2021-09-24
4      2021-09-24
          ...
8802   2019-11-20
8803   2019-07-01
8804   2019-11-01
8805   2020-01-11
8806   2019-03-02
Name: date_added, Length: 8807, 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

#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'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype

 0   date added   8797 non-null   datetime64[ns]
 1   year_added   8797 non-null   float64
 2   month added  8797 non-null   float64
dtypes: datetime64[ns](1), float64(2)
memory usage: 206.5 KB
```

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

```
country           831
date_added         10
release_year        0
rating              0
duration            0
listed_in           0
description         0
dtype: int64
```

```
# % Null values in each column
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
```

3. Non Graphical Analysis and Data Exploration

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

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

```
movies = df.loc[df['type'] == 'Movie']
tv_shows = df.loc[df['type'] == 'TV Show']
movies.duration.value_counts()
```

```
duration
90 min     152
94 min     146
93 min     146
97 min     146
91 min     144
          ...
212 min      1
8 min        1
186 min      1
193 min      1
191 min      1
Name: count, Length: 205, dtype: int64
```

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

```
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
Name: count, dtype: int64
```

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

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

```
1       2.0
2       1.0
3       1.0
4       2.0
5       1.0
       ...
8795    2.0
8796    2.0
8797    3.0
8800    1.0
8803    2.0
Name: duration_in_seasons, Length: 2676, 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
```

```
# when was first movie added on netflix and when is the most recent
movie added on netflix as per data i.e. 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
```

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

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

```
(1925, 2021)
```

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

```
     show_id     type                                        title  \
4250   s4251  TV Show            Pioneers: First Women Filmmakers*
966     s967    Movie                                Get the Grift
967     s968  TV Show                     Headspace Guide to Sleep
968     s969  TV Show                                       Sexify
972     s973  TV Show                                        Fatma
...      ...      ...                                          ...
466     s467  TV Show                          My Unorthodox Life
467     s468    Movie  Private Network: Who Killed Manuel Buendía?
468     s469    Movie               The Guide to the Perfect Family
471     s472    Movie                              Day of Destiny
8437   s8438  TV Show                        The Netflix Afterparty

                     director  \
4250                      NaN
966            Pedro Antonio
967                       NaN
968                       NaN
972                       NaN
```

```
...                        ...
466                        NaN
467              Manuel Alcalá
468              Ricardo Trogi
471    Akay Mason, Abosi Ogba
8437                       NaN

                                                     cast        country  
\
4250                                                  NaN            NaN  

966   Marcus Majella, Samantha Schmütz, Caito Mainie...         Brazil  

967                                  Evelyn Lewis Prieto            NaN  

968    Aleksandra Skraba, Maria Sobocińska, Sandra Dr...         Poland  

972   Burcu Biricik, Uğur Yücel, Mehmet Yılmaz Ak, H...         Turkey  

...                                                   ...            ...  

466                                                   NaN            NaN  

467                                 Daniel Giménez Cacho            NaN  

468   Louis Morissette, Émilie Bierre, Catherine Cha...            NaN  

471   Olumide Oworu, Denola Grey, Gbemi Akinlade, Ji...            NaN  

8437       David Spade, London Hughes, Fortune Feimster  United States  


             date_added release_year rating  duration  \
4250  December 30, 2018         1925  TV-14  1 Season   
966      April 28, 2021         2021  TV-MA    95 min   
967      April 28, 2021         2021   TV-G  1 Season   
968      April 28, 2021         2021  TV-MA  1 Season   
972      April 27, 2021         2021  TV-MA  1 Season   
...                 ...          ...    ...       ...   
466       July 14, 2021         2021  TV-MA  1 Season   
467       July 14, 2021         2021  TV-MA   100 min   
468       July 14, 2021         2021  TV-MA   102 min   
471       July 13, 2021         2021  TV-PG   110 min   
8437    January 2, 2021         2021  TV-MA  1 Season   

                                           listed_in  \
4250                                        TV Shows   
966                     Comedies, International Movies   
967                    Docuseries, Science & Nature TV   
968      International TV Shows, TV Comedies, TV Dramas   
972     International TV Shows, TV Dramas, TV Thrillers   
```

```
...                                                      ...
466                                               Reality TV
467                    Documentaries, International Movies
468                 Comedies, Dramas, International Movies
471   Children & Family Movies, Dramas, Internationa...
8437          Stand-Up Comedy & Talk Shows, TV Comedies

                                             description
4250 This collection restores films from women who ...
966 After a botched scam, Clóvis bumps into Lohane...
967 Learn how to sleep better with Headspace. Each...
968 To build an innovative sex app and win a tech ...
972   Reeling from tragedy, a nondescript house clea...
...                                                  ...
466   Follow Julia Haart, Elite World Group CEO and ...
467   A deep dive into the work of renowned Mexican ...
468   A couple in Québec deals with the pitfalls, pr...
471   With their family facing financial woes, two t...
8437  Hosts David Spade, Fortune Feimster and London...

[593 rows x 12 columns]
```

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

Total movies and tv shows in each country

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

```
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
Name: count, Length: 748, dtype: int64
```

```
#Creating a separate table for country , to avoid the duplicasy of
records in our origional table after exploding
country_tb = df[['show_id' , 'type' , 'country']]
country_tb.dropna(inplace = True)
country_tb['country'] = country_tb['country'].apply(lambda x :
x.split(','))
```

```
country_tb = country_tb.explode('country')
country_tb
```

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

[10019 rows x 3 columns]
```

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

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

```
# 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 x 2 columns]
```

```
# Grouping by country and type, then counting the top number of
show_ids for movie and tv shows
x = country_tb.groupby(['country', 'type'])
['show_id'].count().reset_index()
pivoted_data = x.pivot(index='country', columns='type',
values='show_id').fillna(0)
filtered_data = pivoted_data[pivoted_data['Movie'] >=
10].sort_values('Movie', ascending=False)
filtered_data
```

```
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
Germany                 182.0     44.0
Spain                   171.0     61.0
Japan                   119.0    199.0
China                   114.0     48.0
Mexico                  111.0     58.0
Egypt                   102.0     15.0
Hong Kong               100.0      5.0
Australia                94.0     66.0
Nigeria                  94.0      9.0
Indonesia                86.0      4.0
Turkey                   83.0     30.0
Philippines              80.0      3.0
Belgium                  78.0     12.0
Italy                    75.0     25.0
Argentina                71.0     20.0
Brazil                   66.0     31.0
South Korea              61.0    170.0
South Africa             51.0     11.0
Thailand                 46.0     24.0
Netherlands              42.0      8.0
United Arab Emirates     36.0      1.0
Denmark                  34.0     14.0
Ireland                  32.0     14.0
Poland                   32.0      9.0
Sweden                   31.0     11.0
New Zealand              25.0      8.0
Lebanon                  24.0      7.0
Chile                    24.0      5.0
Norway                   21.0      9.0
Colombia                 20.0     32.0
Pakistan                 20.0      4.0
Taiwan                   19.0     70.0
Israel                   19.0     11.0
```

```
Switzerland             18.0      1.0
Malaysia                18.0      8.0
Singapore               18.0     23.0
Czech Republic          16.0      6.0
Romania                 14.0      0.0
Uruguay                 13.0      1.0
Russia                  11.0     16.0
Austria                 11.0      1.0
Qatar                   10.0      0.0
Peru                    10.0      0.0
Luxembourg              10.0      2.0
Hungary                 10.0      1.0
Bulgaria                10.0      0.0
```

```python
# Netflix has movies from the total c untries
country_tb = country_tb.loc[country_tb['country'] != '']
country_tb['country'].nunique()
```

```
122
```

```python
# Director Columns Details
df['director'].value_counts()
```

```
director
Rajiv Chilaka                     19
Raúl Campos, Jan Suter            18
Marcus Raboy                      16
Suhas Kadav                       16
Jay Karas                         14
                                  ..
Raymie Muzquiz, Stu Livingston     1
Joe Menendez                       1
Eric Bross                         1
Will Eisenberg                     1
Mozez Singh                        1
Name: count, Length: 4528, dtype: int64
```

```python
# There are some movies which are directed by multiple directors.
Creating a table will help to explore more

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

```
    show_id     type                          director
0        s1    Movie                 [Kirsten Johnson]
2        s3  TV Show                 [Julien Leclercq]
5        s6  TV Show                   [Mike Flanagan]
6        s7    Movie  [Robert Cullen,  José Luis Ucha]
7        s8    Movie                    [Haile Gerima]
```

```
...      ...      ...                                ...
8801   s8802    Movie                  [Majid Al Ansari]
8802   s8803    Movie                    [David Fincher]
8804   s8805    Movie                  [Ruben Fleischer]
8805   s8806    Movie                     [Peter Hewitt]
8806   s8807    Movie                      [Mozez Singh]

[6173 rows x 3 columns]
```

```
dir_tb = dir_tb.explode('director')
dir_tb['director'] = dir_tb['director'].str.strip()
dir_tb.director.apply(lambda x : True if len(x) == 0 else
False).value_counts()
```

```
director
False    6978
Name: count, dtype: int64
```

```
dir_tb['director'].nunique() #The total unique directors in the
dataset.
```

```
4993
```

```
# Total movies and tv shows directed by each director
df['director'].apply(type).value_counts()
```

```
director
<class 'str'>      6173
<class 'float'>    2634
Name: count, dtype: int64
```

```
# First, replace NaN values with empty strings to avoid issues during
splitting
df['director'] = df['director'].fillna('')

# Split the 'director' column and explode it into separate rows
df =
df.assign(director=df['director'].str.split(',')).explode('director')

x = df.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
                     188.0    2446.0
Rajiv Chilaka         22.0       NaN
Raúl Campos           18.0       NaN
 Jan Suter            18.0       NaN
Suhas Kadav           16.0       NaN
...                    ...       ...
```

```
Vanessa Roth           NaN      1.0
Vijay Roche            NaN      1.0
Vijay S. Bhanushali    NaN      1.0
Wouter Bouvijn         NaN      1.0
Yasuhiro Irie          NaN      1.0

[5121 rows x 2 columns]
```

```
# 'listed_in' column to understand more about genres
genre_tb = df[['show_id' , 'type', 'listed_in']]
genre_tb['listed_in'] = genre_tb['listed_in'].apply(lambda x :
x.split(','))
genre_tb = genre_tb.explode('listed_in')
genre_tb['listed_in'] = genre_tb['listed_in'].str.strip()

genre_tb
```

```
     show_id     type                listed_in
0         s1    Movie            Documentaries
1         s2  TV Show    International TV Shows
1         s2  TV Show                TV Dramas
1         s2  TV Show             TV Mysteries
2         s3  TV Show           Crime TV Shows
...      ...      ...                       ...
8805   s8806    Movie  Children & Family Movies
8805   s8806    Movie                 Comedies
8806   s8807    Movie                   Dramas
8806   s8807    Movie     International Movies
8806   s8807    Movie         Music & Musicals

[20914 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()  # total genres present in dataset
```

```
42
```

```
# 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              939.0      NaN
Anime Features                  109.0      NaN
Anime Series                      NaN    176.0
British TV Shows                  NaN    255.0
Children & Family Movies        782.0      NaN
Classic & Cult TV                 NaN     28.0
Classic Movies                  127.0      NaN
Comedies                       1846.0      NaN
Crime TV Shows                    NaN    481.0
Cult Movies                      77.0      NaN
Documentaries                  1053.0      NaN
Docuseries                        NaN    415.0
Dramas                         2587.0      NaN
Faith & Spirituality             71.0      NaN
Horror Movies                   399.0      NaN
Independent Movies              852.0      NaN
International Movies           3001.0      NaN
International TV Shows            NaN   1392.0
Kids' TV                          NaN    453.0
Korean TV Shows                   NaN    155.0
LGBTQ Movies                    113.0      NaN
Movies                           67.0      NaN
Music & Musicals                418.0      NaN
Reality TV                        NaN    255.0
Romantic Movies                 640.0      NaN
Romantic TV Shows                 NaN    373.0
Sci-Fi & Fantasy                289.0      NaN
Science & Nature TV               NaN     92.0
Spanish-Language TV Shows         NaN    178.0
Sports Movies                   253.0      NaN
Stand-Up Comedy                 386.0      NaN
Stand-Up Comedy & Talk Shows      NaN     60.0
TV Action & Adventure             NaN    169.0
TV Comedies                       NaN    593.0
TV Dramas                         NaN    793.0
TV Horror                         NaN     77.0
TV Mysteries                      NaN    102.0
TV Sci-Fi & Fantasy               NaN     84.0
TV Shows                                NaN     34.0
TV Thrillers                            NaN     63.0
Teen TV Shows                           NaN     69.0
Thrillers                             608.0      NaN
```

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

```
     show_id     type                     cast
1         s2  TV Show               Ama Qamata
1         s2  TV Show              Khosi Ngema
1         s2  TV Show            Gail Mabalane
1         s2  TV Show           Thabang Molaba
1         s2  TV Show         Dillon Windvogel
...      ...      ...                      ...
8806   s8807    Movie        Manish Chaudhary
8806   s8807    Movie            Meghna Malik
8806   s8807    Movie           Malkeet Rauni
8806   s8807    Movie          Anita Shabdish
8806   s8807    Movie   Chittaranjan Tripathy

[69852 rows x 3 columns]
```

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

```
# checking empty strings
```

```
cast_tb[cast_tb['cast'] == '']
```

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

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

```
39296
```

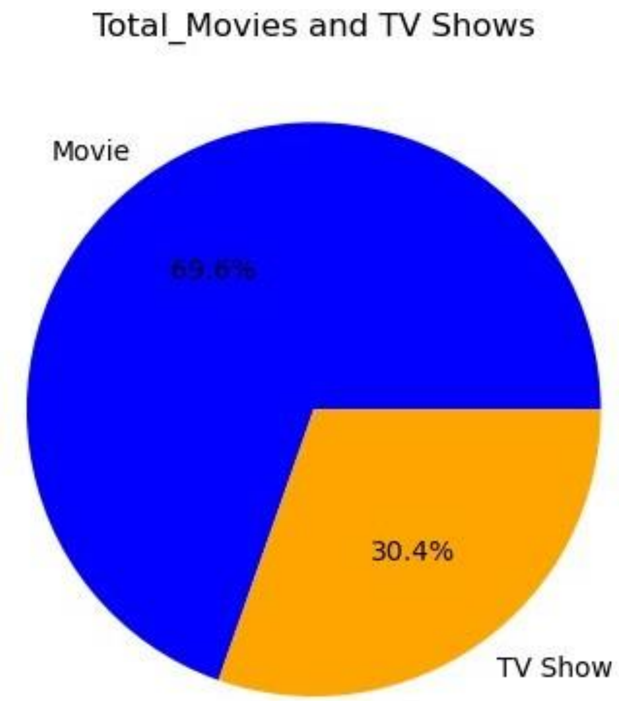
```
# Total movies/TV shows by each actor
x = cast_tb.groupby(['cast' , 'type'])
['show_id'].count().reset_index()
x.pivot(index = 'cast' , columns = 'type' , values =
'show_id').sort_values('TV Show' , ascending = False)
```

```
type                   Movie  TV Show
cast
 Takahiro Sakurai        9.0     25.0
 Yuki Kaji              11.0     18.0
 Junichi Suwabe       6.0     18.0
 Ai Kayano            2.0     17.0
 Daisuke Ono          6.0     15.0
...                   ...      ...
Çağlar Çorumlu        1.0      NaN
Çetin Tekindor        1.0      NaN
İbrahim Büyükak       1.0      NaN
Şahin Irmak           1.0      NaN
ŞọpẹÍDìrísù         1.0      NaN

[39296 rows x 2 columns]
```

4. Visual Analysis - Univariate & Bivariate

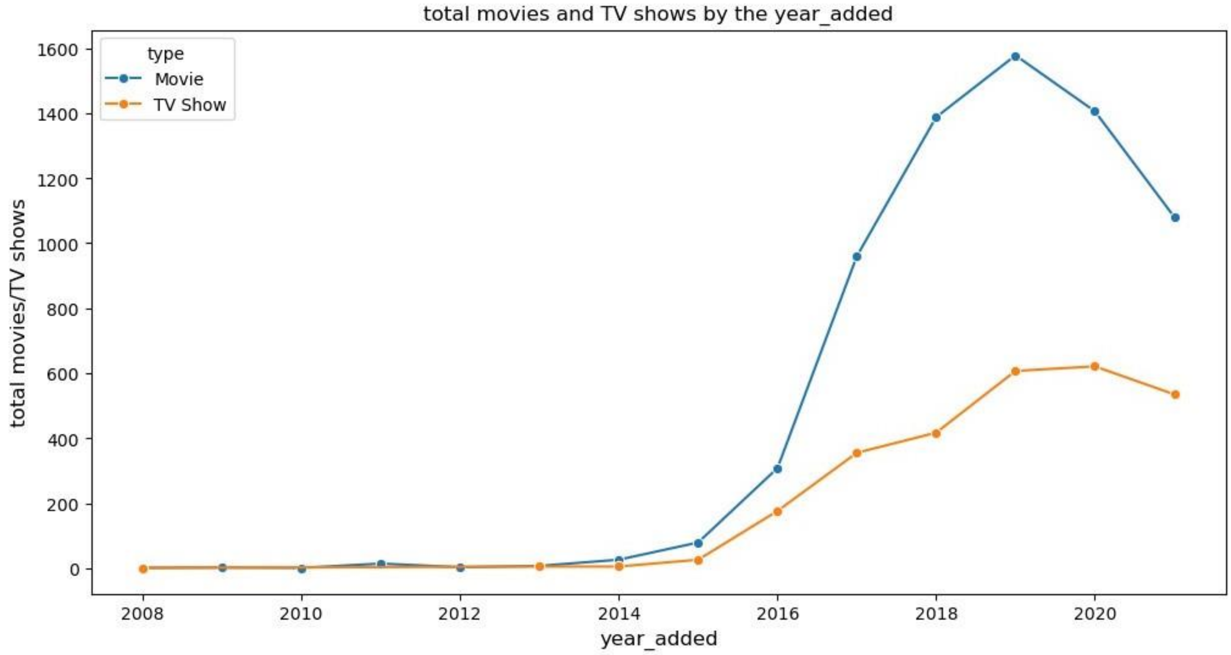
4.1. Distribution of content across the different types

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



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

```
d = df.groupby(['year_added' ,'type' ])
['show_id'].count().reset_index()
d.rename({'show_id' : 'total movies/TV shows'}, axis = 1 , inplace =
True)
plt.figure(figsize = (12,6))
sns.lineplot(data = d , x = 'year_added' , y = 'total movies/TV shows'
, hue = 'type', marker = 'o' , ms = 6)
plt.xlabel('year_added' , fontsize = 12)
plt.ylabel('total movies/TV shows' , fontsize = 12)
plt.title('total movies and TV shows by the year_added' , fontsize =
12)
plt.show()
```

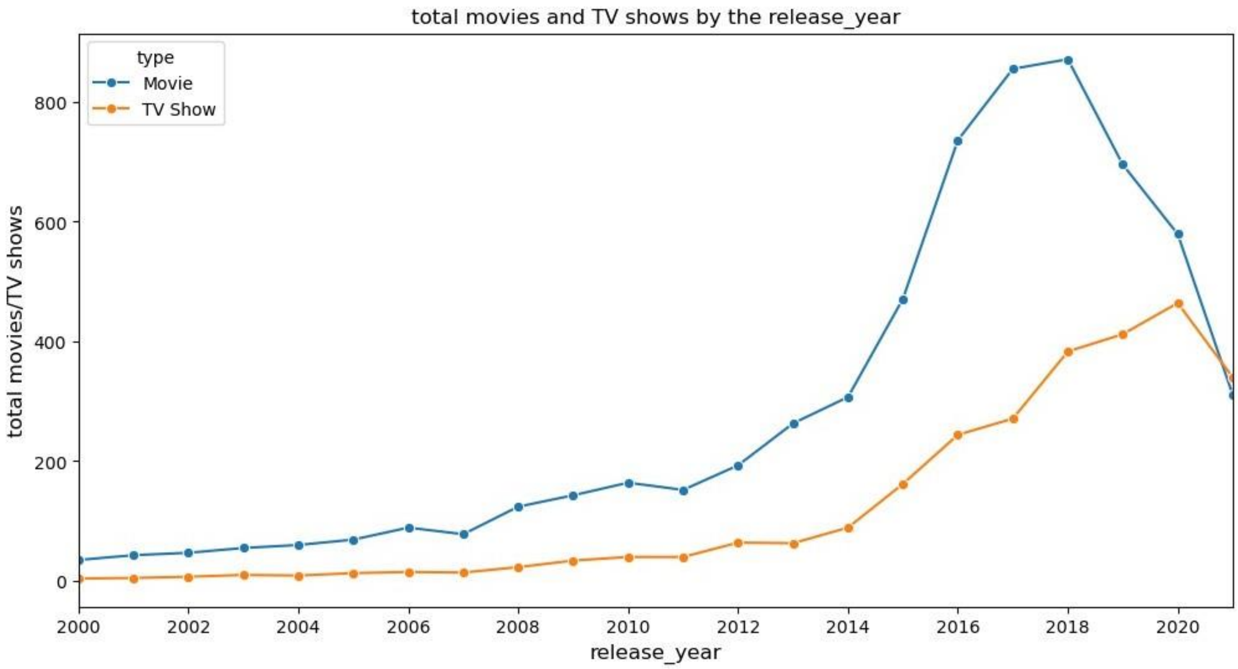


```
Observation:

2019 marks the highest number of movie and TV show releases.
Since 2020, A drop in movies is seen
Rise in TV shows is observed clearly from 2019 and slight decrease
after 2020
Both Movies and TV shows were almost at same pace till 2014 and from
2014 there was a surge in movies and which increased drastically
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.

# How has the number of movies released per year changed over the last
20-30 years?
d = df.groupby(['type' , 'release_year'])
```

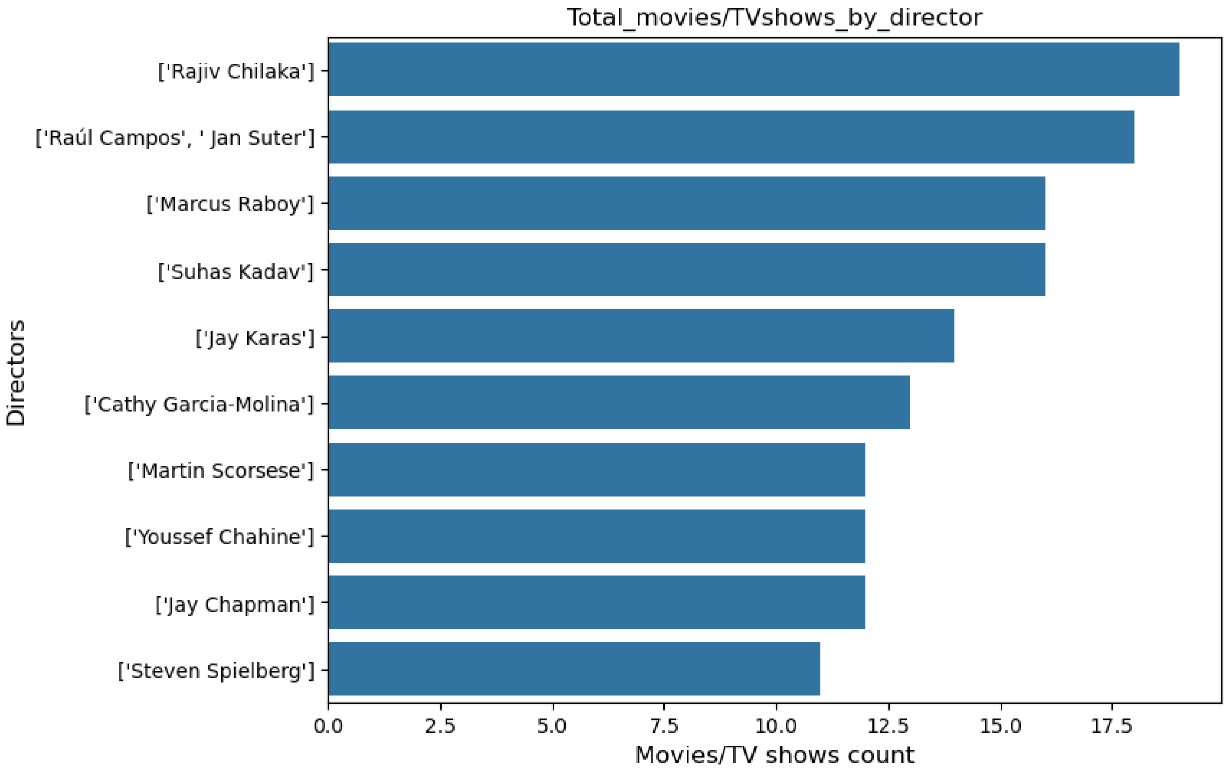
```
['show_id'].count().reset_index()
d.rename({'show_id' : 'total movies/TV shows'}, axis = 1 , inplace =
True)
d
```

```
        type  release_year  total movies/TV shows
0      Movie          1942                      2
1      Movie          1943                      4
2      Movie          1944                      7
3      Movie          1945                      4
4      Movie          1946                      1
..       ...           ...                    ...
114  TV Show          2017                    271
115  TV Show          2018                    383
116  TV Show          2019                    412
117  TV Show          2020                    464
118  TV Show          2021                    340

[119 rows x 3 columns]
```

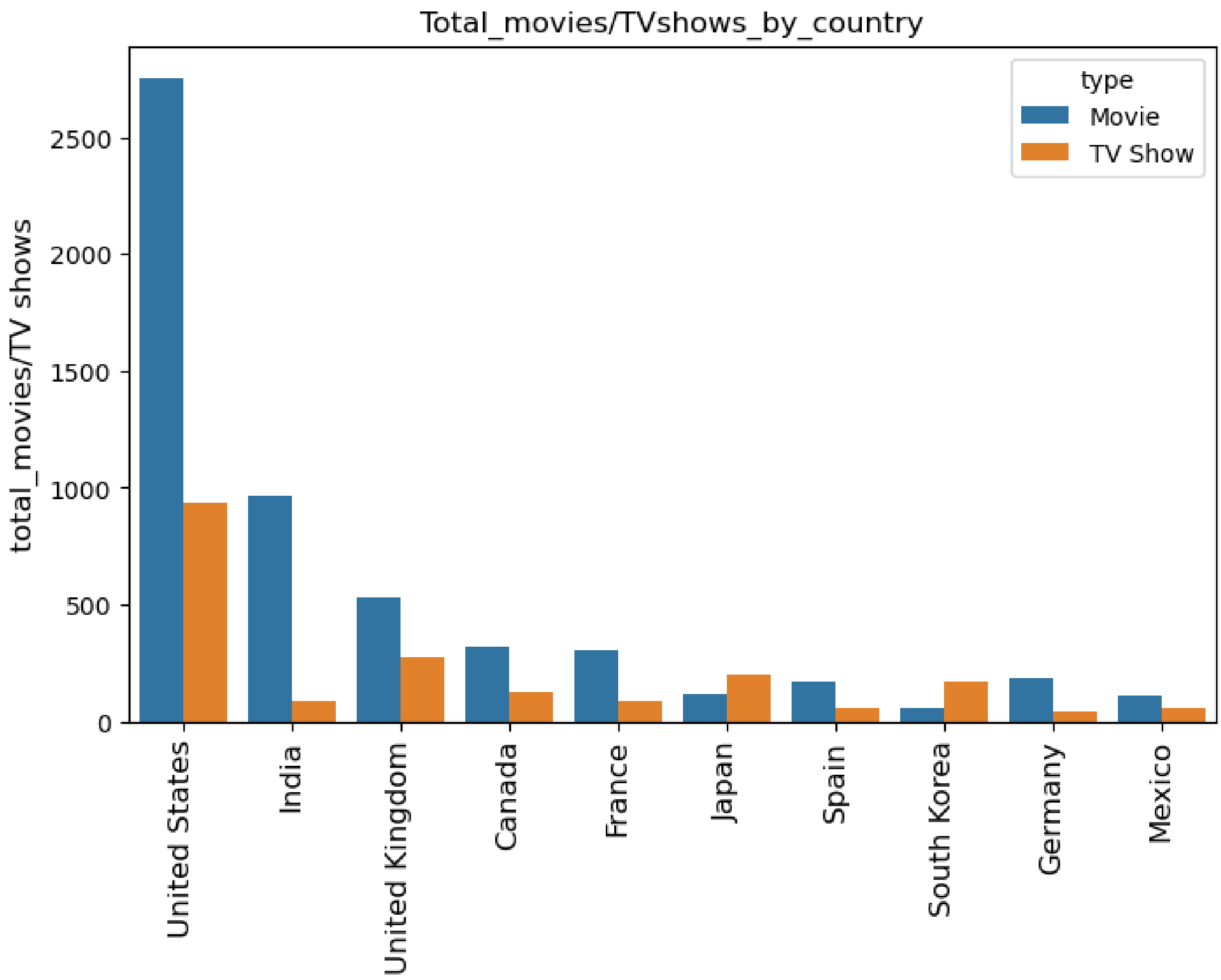
```
plt.figure(figsize = (12,6))
sns.lineplot(data = d , x = 'release_year' , y = 'total movies/TV
shows' , hue = 'type' , marker = 'o' , ms = 6 )
plt.xlabel('release_year' , fontsize = 12)
plt.ylabel('total movies/TV shows' , fontsize = 12)
plt.title('total movies and TV shows by the release_year' ,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



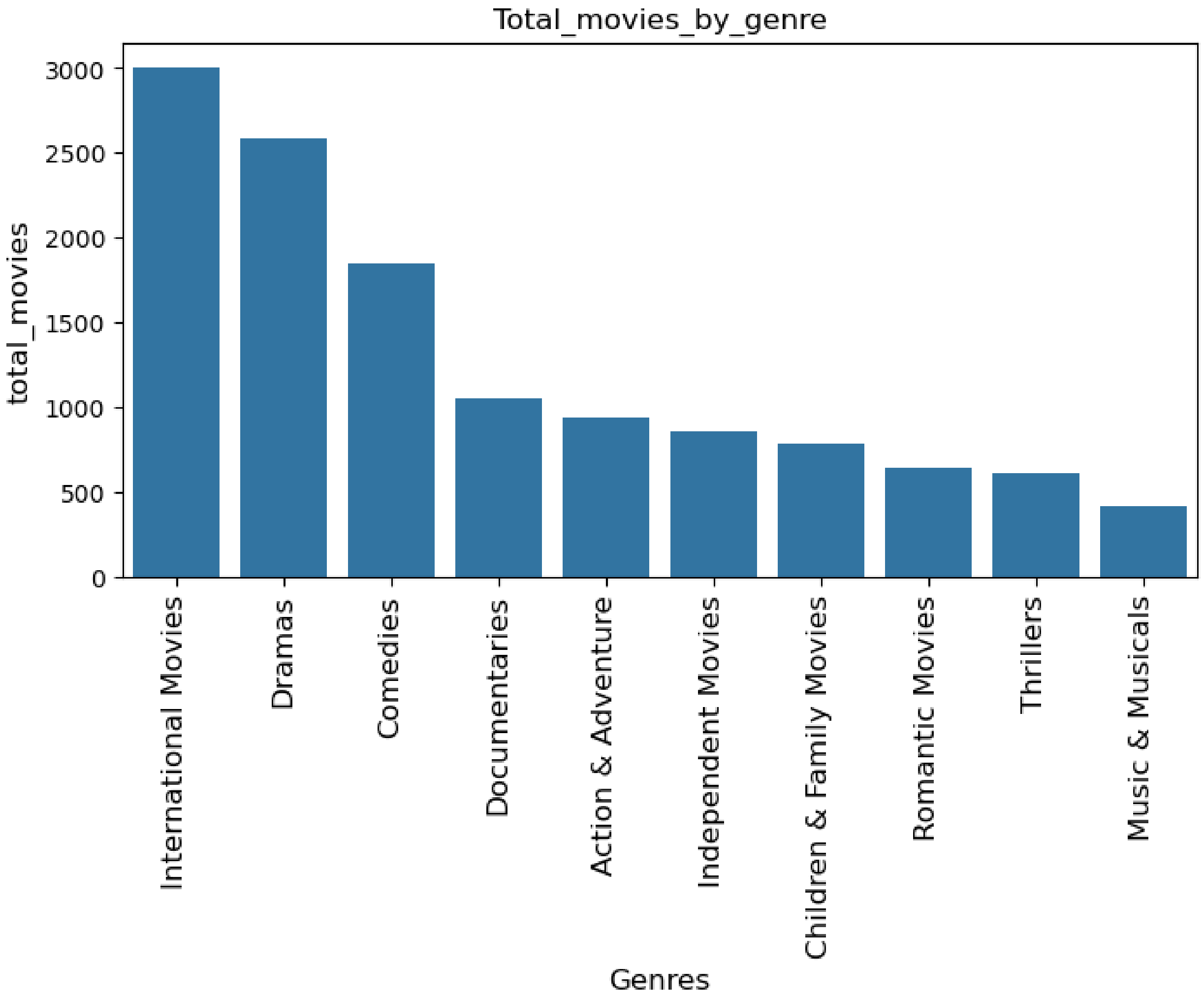
```
Observation:

2018 marks the highest number of movies.and 2019 marks highest for TV
show releases
Since 2018, A drop in movies is seen and rise in TV shows is observed
clearly, and TV shows surpasses the movies count in mid 2020.
The yearly number of releases has surged drastically after mid-2014.

# Total movies/TV shows by each director
top_10_dir = dir_tb.director.value_counts().head(10).index
df_new = dir_tb.loc[dir_tb['director'].isin(top_10_dir)]
plt.figure(figsize= (8 , 6))
sns.countplot(data = df_new , y = 'director' , order = top_10_dir ,
orient = 'v')
plt.xlabel('total_movies/TV shows' , fontsize = 12)
plt.xlabel('Movies/TV shows count')
plt.ylabel('Directors' , fontsize = 12)
plt.title('Total_movies/TVshows_by_director')
plt.show()
```



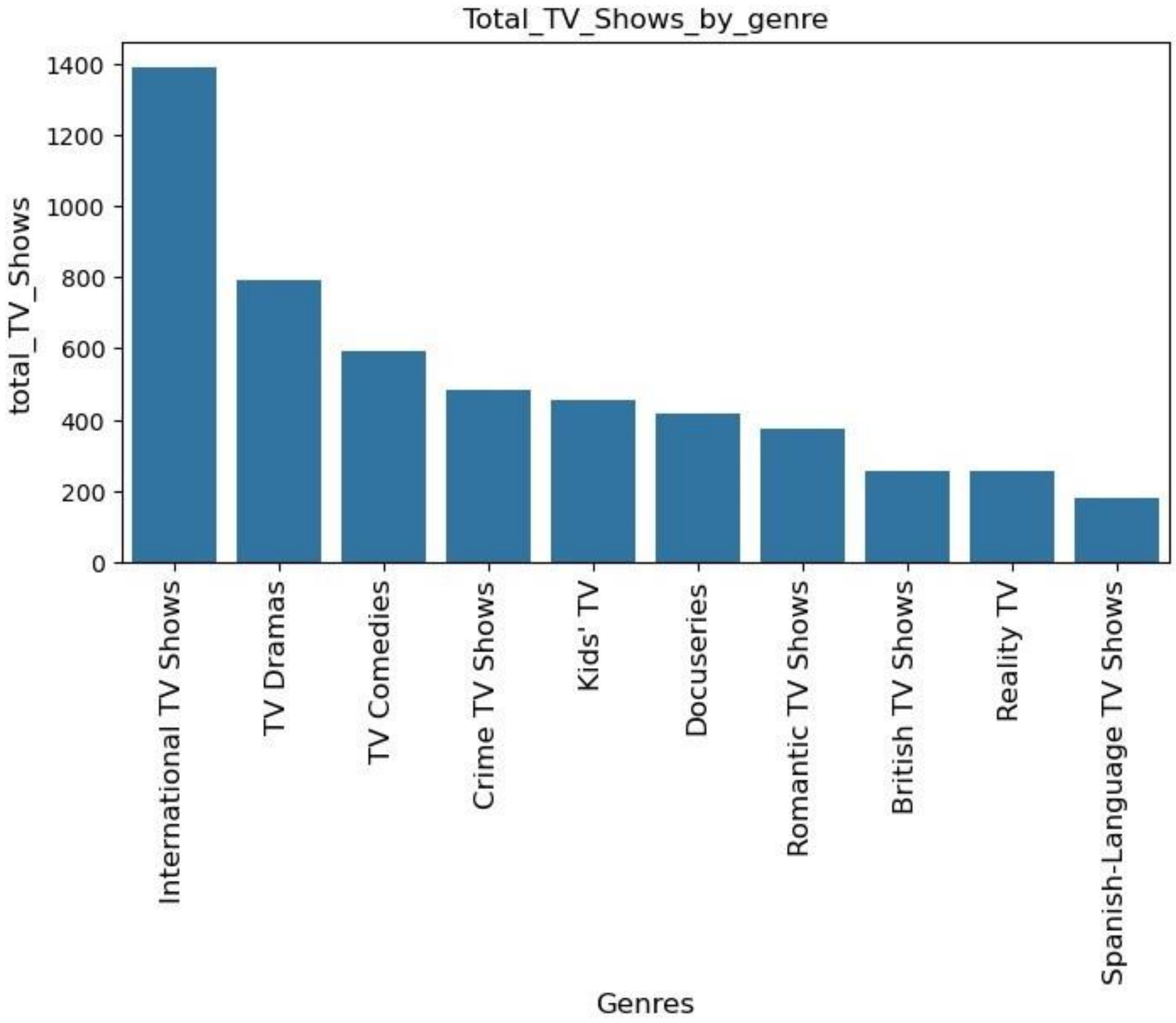
```
# Total movies/TV shows by each country
top_10_country = country_tb.country.value_counts().head(10).index
df_new = country_tb.loc[country_tb['country'].isin(top_10_country)]
plt.figure(figsize= (8,5))
sns.countplot(data = df_new , x = 'country' , order = top_10_country ,
hue = 'type')
plt.xticks(rotation = 90 , fontsize = 12)
plt.ylabel('total_movies/TV shows' , fontsize = 12)
plt.xlabel('')
plt.title('Total_movies/TVshows_by_country')
plt.show()
```



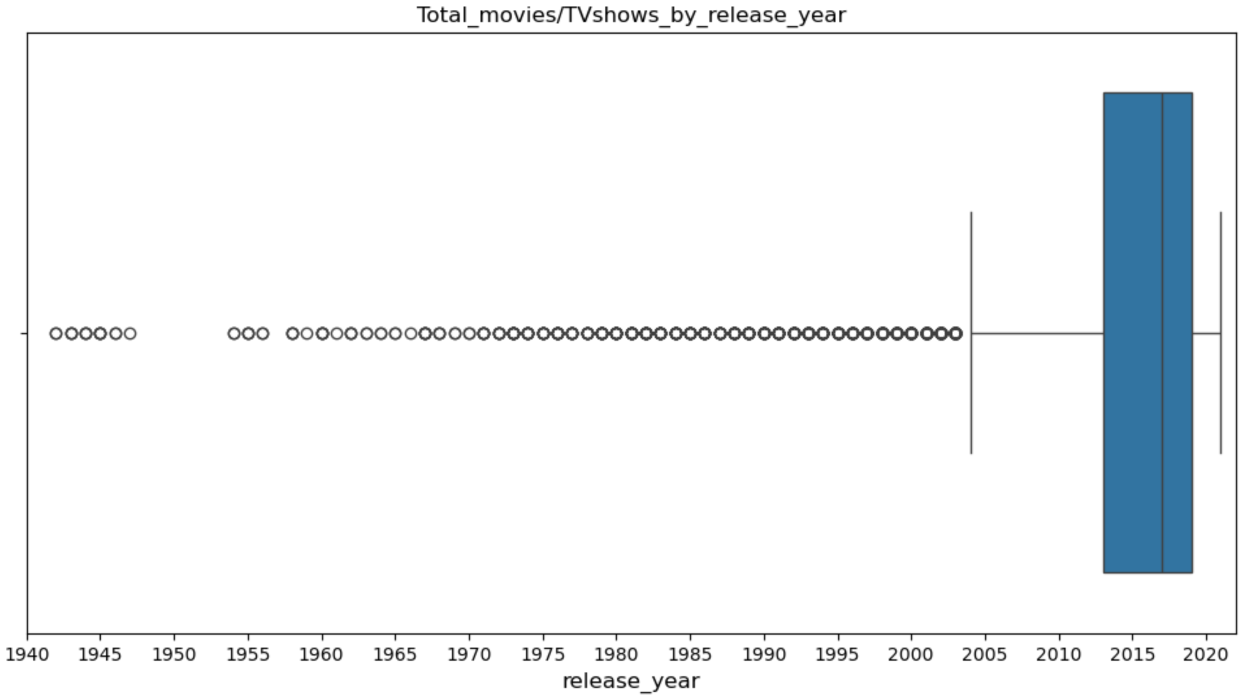
```
# Total movies/TV shows in each Genre
top_10_movie_genres = genre_tb[genre_tb['type'] ==
'Movie'].listed_in.value_counts().head(10).index
df_movie =
genre_tb.loc[genre_tb['listed_in'].isin(top_10_movie_genres)]
plt.figure(figsize= (8,4))
sns.countplot(data = df_movie , x = 'listed_in' , order =
top_10_movie_genres)
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()
```



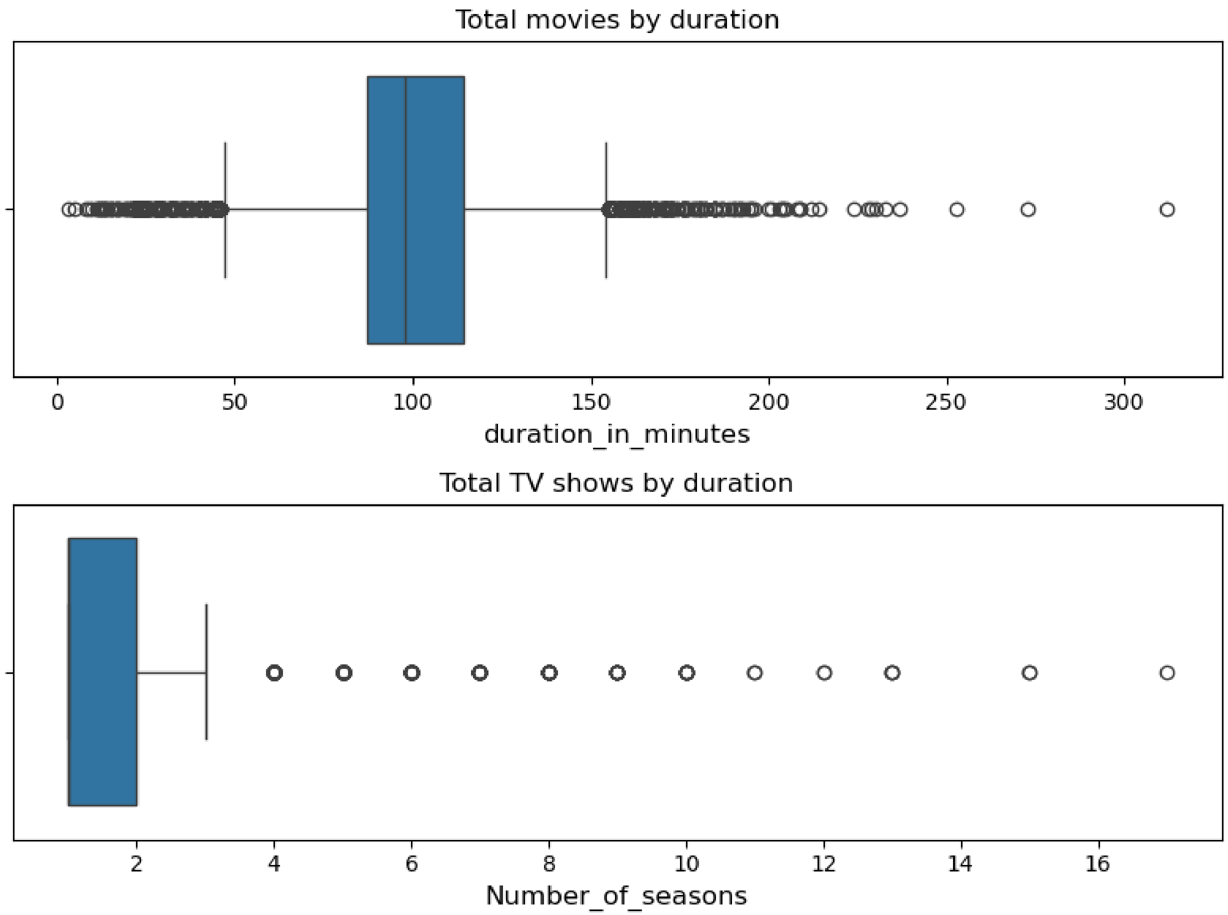
```
top_10_TV_genres = genre_tb[genre_tb['type'] == 'TV
Show'].listed_in.value_counts().head(10).index
df_tv = genre_tb.loc[genre_tb['listed_in'].isin(top_10_TV_genres)]
plt.figure(figsize= (8,4))
sns.countplot(data = df_tv , x = 'listed_in' , order =
top_10_TV_genres)
plt.xticks(rotation = 90 , fontsize = 12)
plt.ylabel('total_TV_Shows' , fontsize = 12)
plt.xlabel('Genres' , fontsize = 12)
plt.title('Total_TV_Shows_by_genre')
plt.show()
```



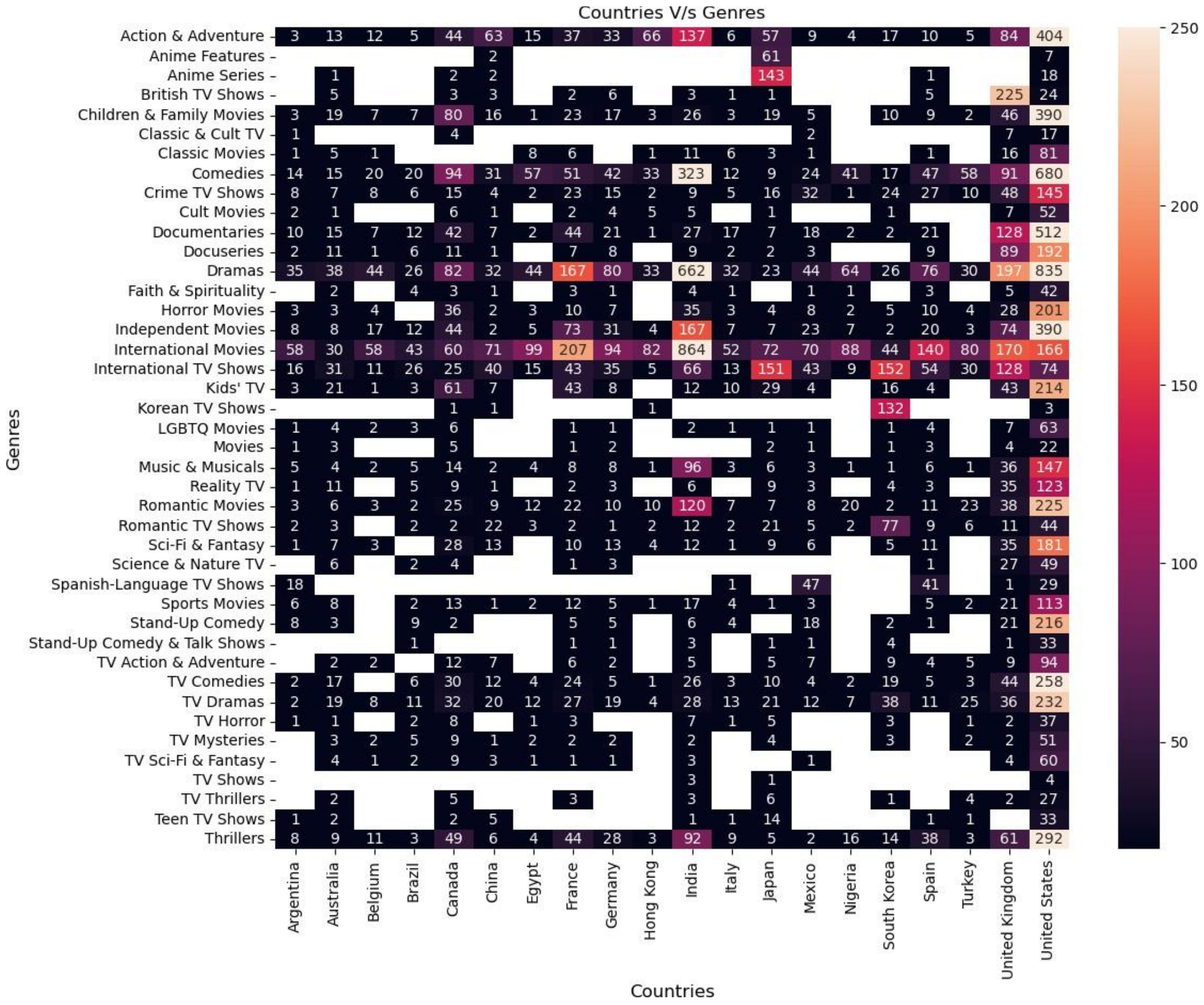
```
# Movies and TV shows from 1940 till 2020

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()
```



```
# Total Movies and TV shows duration
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()
```



```
5. Bivariate Analysis

# popular genres in top 20 countries

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

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

plt.figure(figsize = (12,10))

sns.heatmap(data = country_genre , annot = True , fmt=".0f" , vmin = 20
, vmax = 250 )

plt.xlabel('Countries' , fontsize = 12) plt.ylabel('Genres' , fontsize
= 12) plt.title('Countries V/s Genres' , fontsize = 12)
```



```
# 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(['India'])].sort_values('show_id' , ascending
= False).head(10)
```

```
      country              cast  show_id
14597   India       Anupam Kher       36
16275   India           Om Puri       26
18327   India    Shah Rukh Khan       25
14875   India       Boman Irani       25
16320   India      Paresh Rawal       25
17901   India      Akshay Kumar       23
16130   India  Naseeruddin Shah       20
15627   India    Kareena Kapoor       20
17912   India  Amitabh Bachchan       20
14734   India            Asrani       17
```

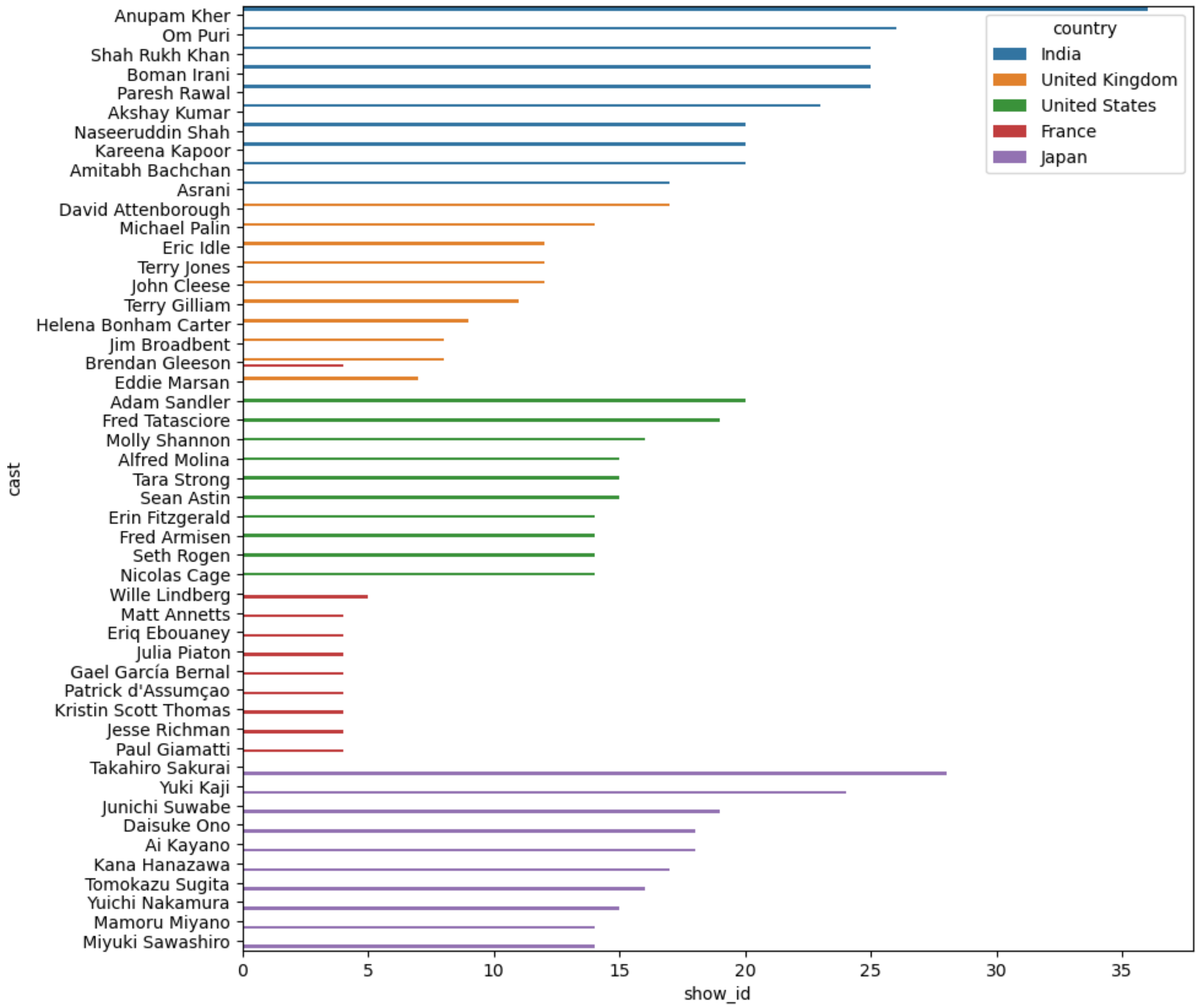
```
country_list = ['India' , 'United Kingdom' , 'United States',
'France' , 'Japan']
top_10_actors =
x.loc[x['country'].isin(['India'])].sort_values('show_id' , ascending
= False).head(10)
for i in country_list:
    new = x.loc[x['country'].isin([i])].sort_values('show_id' ,
ascending = False).head(10)
    top_10_actors = pd.concat( [top_10_actors , new] , ignore_index =
True)

top_10_actors
```

```
           country                cast  show_id
0            India         Anupam Kher       36
1            India             Om Puri       26
2            India      Shah Rukh Khan       25
3            India         Boman Irani       25
4            India        Paresh Rawal       25
5            India        Akshay Kumar       23
6            India    Naseeruddin Shah       20
7            India      Kareena Kapoor       20
8            India    Amitabh Bachchan       20
9            India              Asrani       17
10           India         Anupam Kher       36
11           India             Om Puri       26
12           India      Shah Rukh Khan       25
13           India         Boman Irani       25
14           India        Paresh Rawal       25
15           India        Akshay Kumar       23
16           India    Naseeruddin Shah       20
17           India      Kareena Kapoor       20
18           India    Amitabh Bachchan       20
19           India              Asrani       17
20  United Kingdom   David Attenborough       17
21  United Kingdom        Michael Palin       14
22  United Kingdom            Eric Idle       12
23  United Kingdom          Terry Jones       12
24  United Kingdom          John Cleese       12
25  United Kingdom        Terry Gilliam       11
26  United Kingdom  Helena Bonham Carter        9
27  United Kingdom        Jim Broadbent        8
28  United Kingdom      Brendan Gleeson        8
29  United Kingdom         Eddie Marsan        7
30   United States         Adam Sandler       20
31   United States      Fred Tatasciore       19
32   United States        Molly Shannon       16
33   United States        Alfred Molina       15
34   United States          Tara Strong       15
35   United States           Sean Astin       15
```

```
36   United States        Erin Fitzgerald       14
37   United States           Fred Armisen       14
38   United States             Seth Rogen       14
39   United States           Nicolas Cage       14
40          France         Wille Lindberg        5
41          France           Matt Annetts        4
42          France          Eriq Ebouaney        4
43          France           Julia Piaton        4
44          France     Gael García Bernal        4
45          France     Patrick d'Assumçao        4
46          France   Kristin Scott Thomas        4
47          France        Brendan Gleeson        4
48          France          Jesse Richman        4
49          France          Paul Giamatti        4
50           Japan       Takahiro Sakurai       28
51           Japan              Yuki Kaji       24
52           Japan         Junichi Suwabe       19
53           Japan            Daisuke Ono       18
54           Japan              Ai Kayano       18
55           Japan          Kana Hanazawa       17
56           Japan        Tomokazu Sugita       16
57           Japan        Yuichi Nakamura       15
58           Japan          Mamoru Miyano       14
59           Japan       Miyuki Sawashiro       14
```

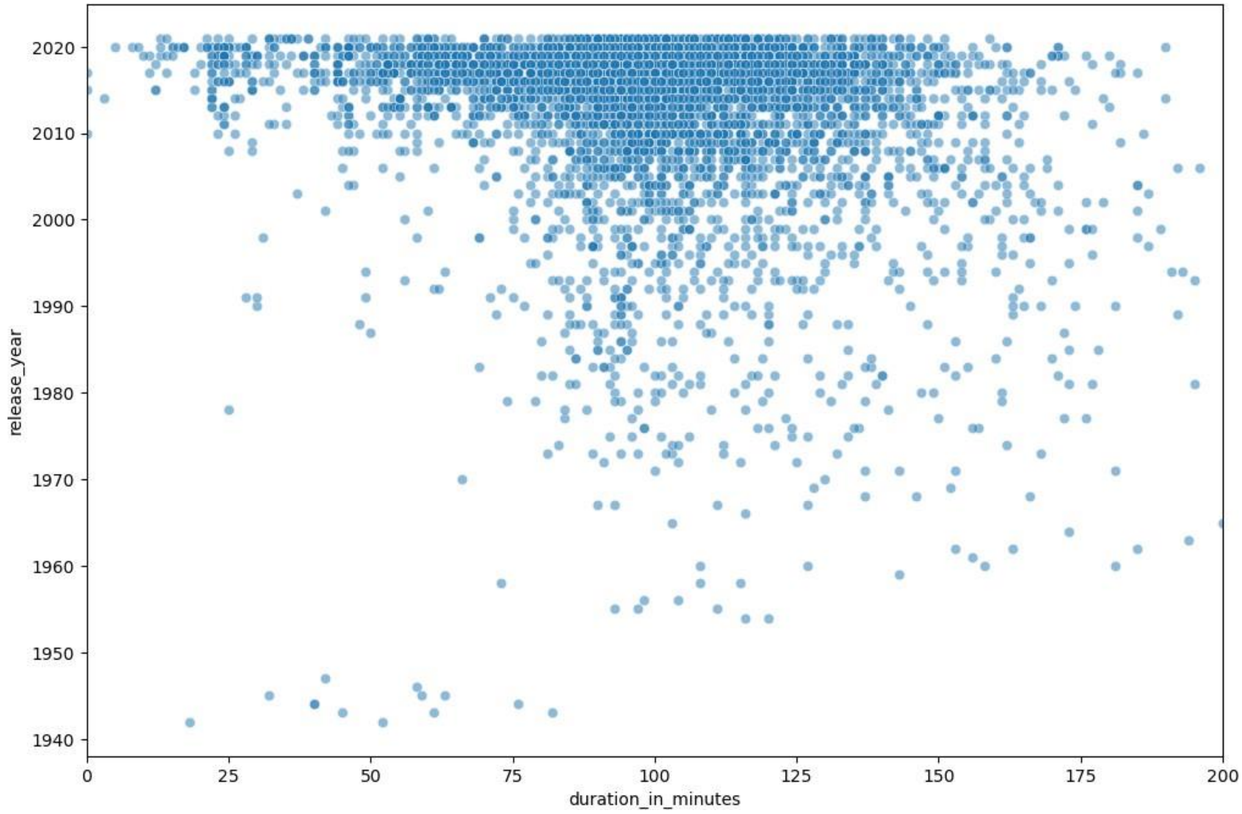
```
plt.figure(figsize = (10,10))
sns.barplot(data = top_10_actors , y = 'cast' , x = 'show_id' , hue =
'country')
```

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



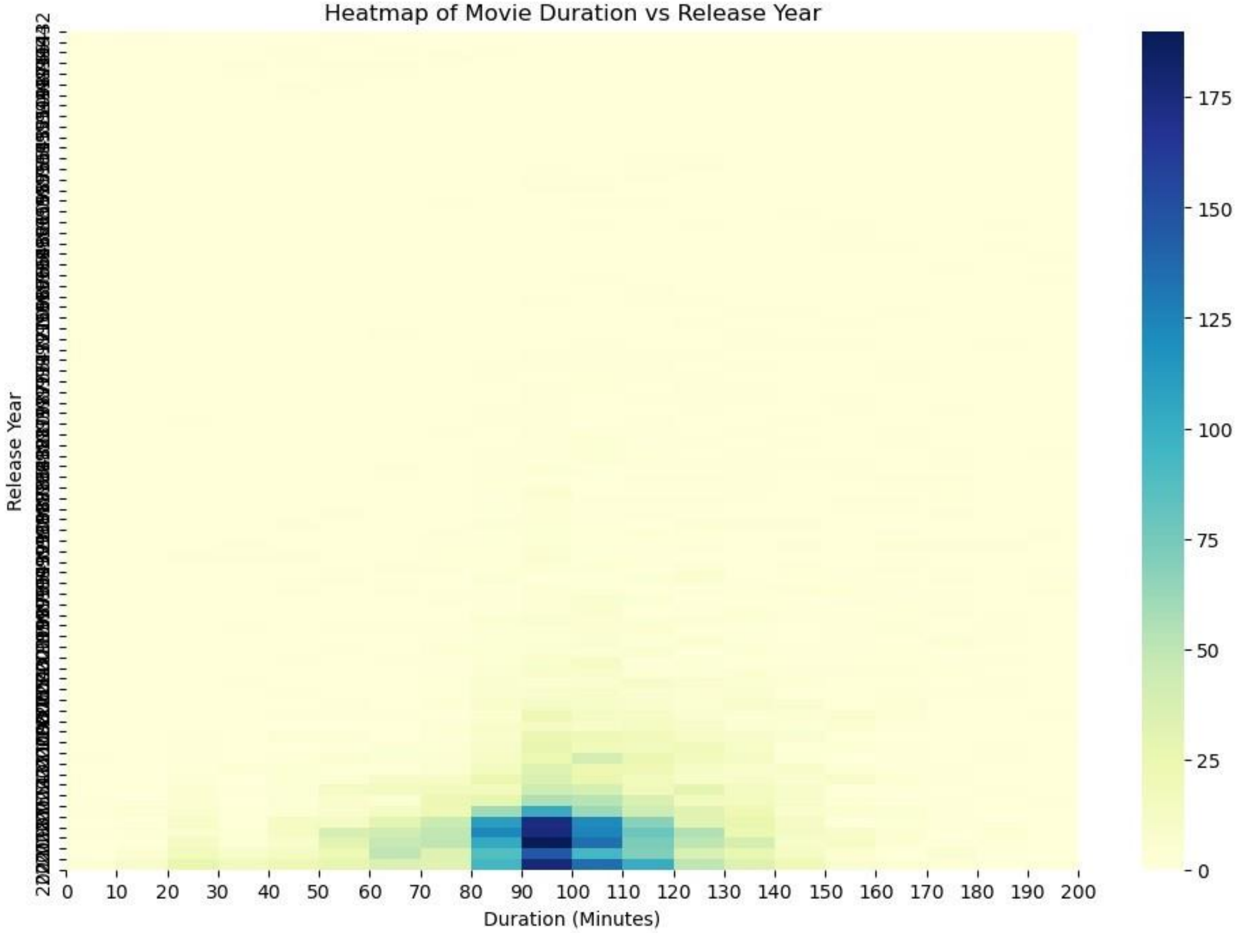
```
# Variation in duration of movies by Release year
print(movies.columns)

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

# Replace NaN with 0 in the 'duration' column and then extract the
numbers
movies['duration'] = movies['duration'].fillna('0')
movies['duration_in_minutes'] = movies['duration'].str.extract('(\
d+)').astype(int)

# Drop rows with NaN values in 'duration' and then extract the numbers
movies = movies.dropna(subset=['duration'])
```

```
movies['duration_in_minutes'] = movies['duration'].str.extract('(\
d+)').astype(int)

# Plot scatter plot with duration and release year
plt.figure(figsize=(12,8))
sns.scatterplot(x=movies['duration_in_minutes'],
y=movies['release_year'], alpha=0.5)
plt.xlim((0, 200))  # Adjust x-axis range if needed
plt.show()
```



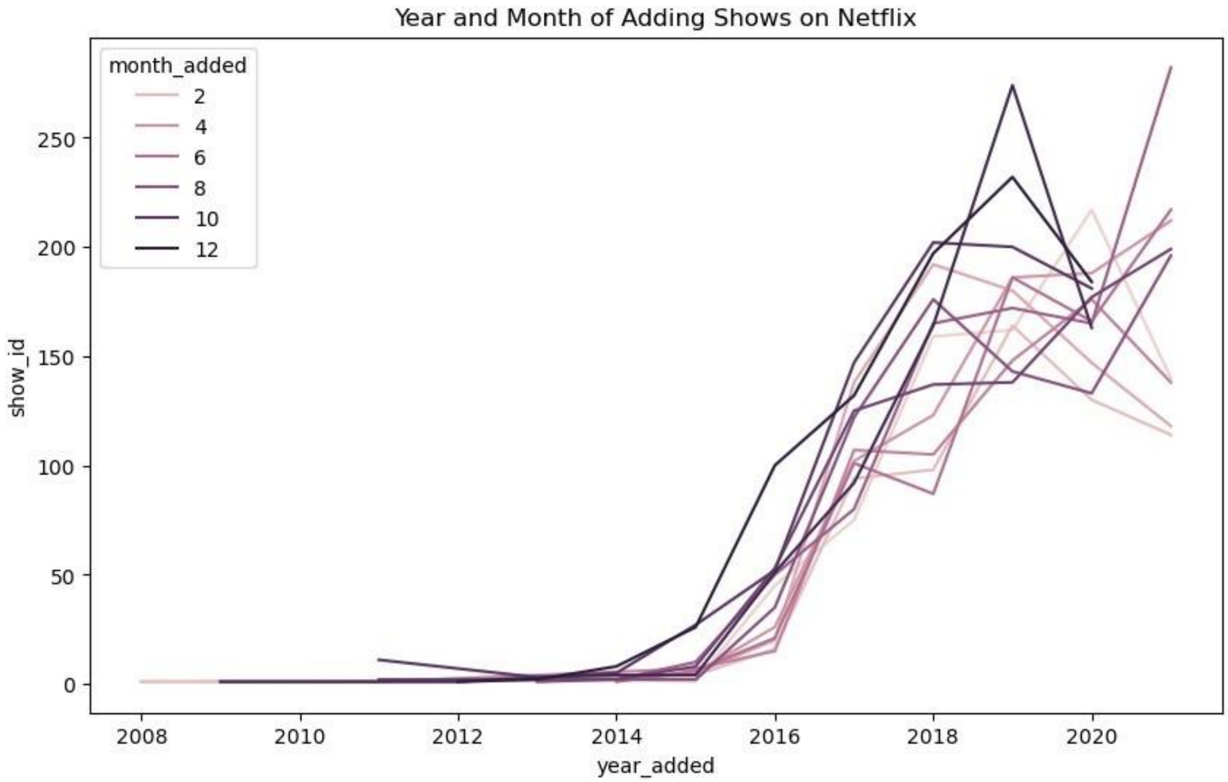
```
# Variation in duration of movies by Release year using Heatmaps
duration_bins = np.arange(0, 201, 10) # Bins for duration from 0 to
200, in steps of 10
release_year_bins = np.arange(movies['release_year'].min(),
movies['release_year'].max() + 1, 1)
heatmap_data, xedges, yedges =
np.histogram2d(movies['duration_in_minutes'], movies['release_year'],
bins=[duration_bins, release_year_bins])
plt.figure(figsize=(12,8))
sns.heatmap(heatmap_data.T, cmap="YlGnBu", xticklabels=10,
yticklabels=10, cbar=True)
```

```
plt.xlabel('Duration (Minutes)')
plt.ylabel('Release Year')


plt.xticks(np.arange(len(duration_bins)), duration_bins)
plt.yticks(np.arange(len(release_year_bins)), release_year_bins)

plt.title('Heatmap of Movie Duration vs Release Year')
plt.show()
```



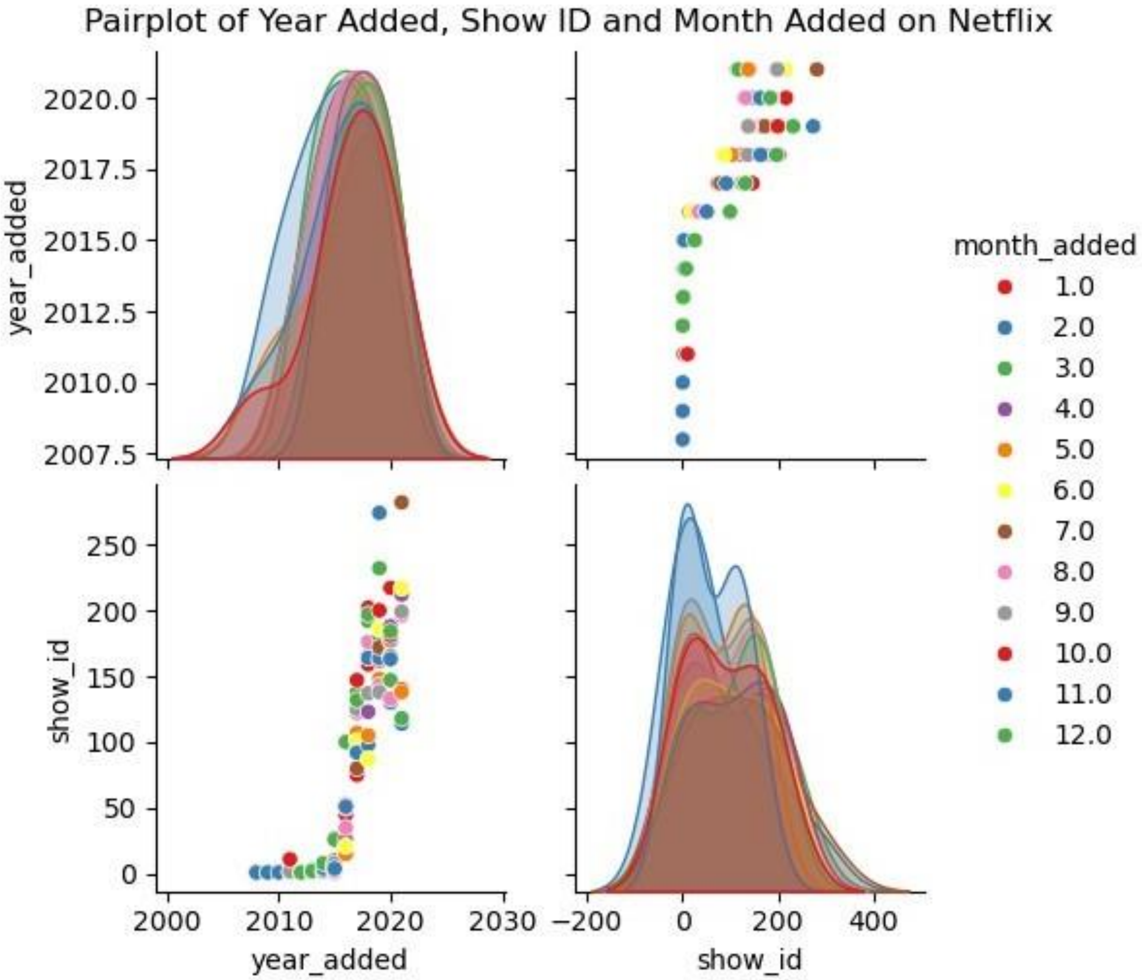
```
# What is the best time of the year on the Netflix?
month_year = df.groupby(['year_added' , 'month_added'])
['show_id'].count().reset_index()
plt.figure(figsize = (10,6))
sns.lineplot(data=month_year, x = 'year_added', y = 'show_id',
hue='month_added')
plt.title('Year and Month of Adding Shows on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

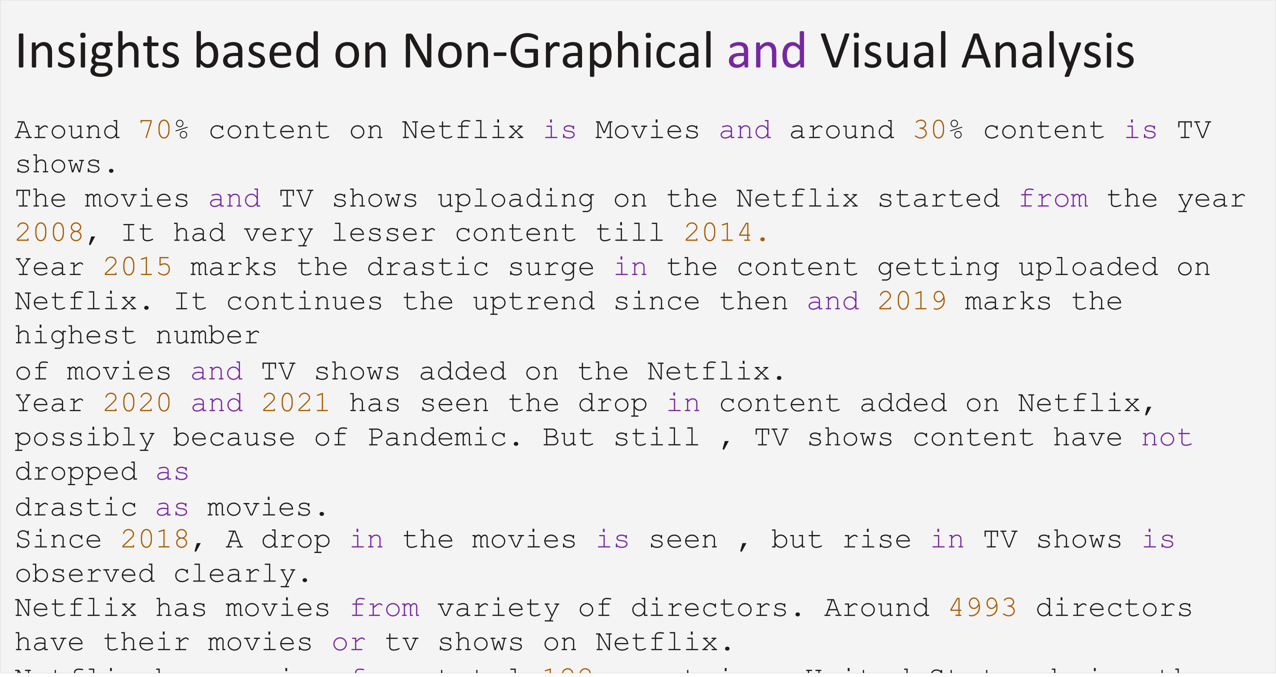
```
Text(0.5, 1.0, 'Year and Month of Adding Shows on Netflix')
```



```
# What is the best time of the year on the Netflix?

month_year['month_added'] =
month_year['month_added'].astype('category') # Convert month_added to
a categorical variable (if not already)
plt.figure(figsize=(10,6)) # Generate pairplot
sns.pairplot(data=month_year, vars=['year_added', 'show_id'],
hue='month_added', palette='Set1')
plt.suptitle('Pairplot of Year Added, Show ID and Month Added on
Netflix', y=1.02) # Display the plot
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```





Insights based on Non-Graphical and Visual Analysis

```
Around 70% content on Netflix is Movies and around 30% content is TV
shows.
The movies and TV shows uploading on the Netflix started from the year
2008, It had very lesser content till 2014.
Year 2015 marks the drastic surge in the content getting uploaded on
Netflix. It continues the uptrend since then and 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.
Since 2018, A drop in the movies is seen , but rise in TV shows is
observed clearly.
Netflix has movies from variety of directors. Around 4993 directors
have their movies or tv shows on Netflix.
```

highset contributor with almost 37% of all the content.
The release year for shows is concentrated in the range 2005-2021.
various ratings of content is avaialble on netfilx, for the various viewers categories like kids, adults , families.
Highest number of movies and TV shows are rated TV-MA (for mature audiences).
Content in most of the ratings is available in lesser quanitity except in US.
Ratings like TV-Y7 , TV-Y7 FV , PG ,TV-G , G , TV-Y , TV-PG are very less avaialble in all countries except US.
Mostly country specific popular genres are observed in each country.
Indian Actors have been acted in maximum movies on netflix. Top 10 actors are in India based on quantity of movies.

Business Insights

Netflix have majority of content which is released after the year 2000. It is observed that the content older than year 2000 is very scarce on Netflix. Senior Citizen could be the target audience for such content, which is almost missing currently.
Maximum content (more than 80%) is
TV-MA - Content intended for mature audiences aged 17 and above.
TV-14 - Content suitable for viewers aged 14 and above.
TV-PG - Parental guidance suggested (similar ratings - PG-13 , PG)
R - Restricted Content, that may not be suitable for viewers under age 17.
These ratings' movies target Matured and Adult audience. Rest 20 % of the content is for kids aged below 13. It shows that Netflix is currently serving mostly Mature audiences or Children with parental guidance.

Most popular genres on Netflix are International Movies and TV Shows , Dramas , Comedies, Action & Adventure, Children & Family Movies, Thrillers.
Maximum content of Netflix which is around 75% , is coming from the top 10 countries. Rest of the world only contributes 25% of the content.
More countries can be focussed in future to grow the business.
drop in content is seen across all the countries and type of content in year 2020 and 2021, possibly because of Pandemic.

Recommendations:

Country specific insights - The content need to be targetting the demographic of any country.
Netflix can produce higher number of content in the perticular rating as per demographic of the country.
Eg. The country like India , which is highly populous , has maximum content available only in three rating TV-MA, TV-14 , TV-PG.
It is unlikely to serve below 14 age and above 35 year age group

```
Maximum countries need some more genres which are highly popular in
the region. eg. Indian Mythological content is highly popular.
We can create such more country specific genres and It might also
be liked acorss the world just like Japanese Anime.
```