Business Case: Netflix - Data Exploration and Visualisation

About NETFLIX

Netflix is one of the most popular media and video streaming platforms. They have over 10000 movies or tv shows available on their platform, as of mid-2021, they have over 222M Subscribers globally. This tabular dataset consists of listings of all the movies and tv shows available on Netflix, along with details such as - cast, directors, ratings, release year, duration, etc.

Business Problem

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

Dataset

Link: Dataset_link

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

Hints

1. The exploration should have a goal. As you explore the data, keep in mind that you want to answer which type of shows to produce and how to grow the business.
2. Ensure each recommendation is backed by data. The company is looking for data-driven insights, not personal opinions or anecdotes.

Business Case: Netflix - Data Exploration and Visualisation

3. Assume that you are presenting your findings to business executives who have only a basic understanding of data science. Avoid unnecessary technical jargon.
4. Start by exploring a few questions: What type of content is available in different countries?
    1. How has the number of movies released per year changed over the last 20-30 years?
    2. Comparison of tv shows vs. movies.
    3. What is the best time to launch a TV show?
    4. Analysis of actors/directors of different types of shows/movies.
    5. Does Netflix has more focus on TV Shows than movies in recent years
    6. Understanding what content is available in different countries

Evaluation Criteria (100 Points):

1. Defining Problem Statement and Analysing basic metrics (10 Points)

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 (10 Points)

3. Non-Graphical Analysis: Value counts and unique attributes (10 Points)

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

Note: Pre-processing involves unnesting of the data in columns like Actor, Director, Country

4.1 For continuous variable(s): Distplot, countplot, histogram for univariate analysis (10 Points)

4.2 For categorical variable(s): Boxplot (10 Points)

4.3 For correlation: Heatmaps, Pairplots (10 Points)

5. Missing Value & Outlier check (Treatment optional) (10 Points)

6. Insights based on Non-Graphical and Visual Analysis (10 Points)

6.1 Comments on the range of attributes

6.2 Comments on the distribution of the variables and relationship between them

6.3 Comments for each univariate and bivariate plot

Business Case: Netflix - Data Exploration and Visualisation

7. Business Insights (10 Points) - Should include patterns observed in the data along with what you can infer from it

8. Recommendations (10 Points) - Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand

Please find the link for my case study.

https://colab.research.google.com/drive/1P0REtMNEeh0RUngY3twcEPskybZCc9Pq?usp=sharing

Business Problem: To analyse and understand Netflix data which generates insights that could help Netflix decide which type of shows/movies to produce more and attract and acquire more subscribers in order to increase the business.

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

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

```
plt.figure(figsize=(15,15))
plt.axis("off")
img = plt.imread("Netfliximage.jpeg")
plt.imshow(img)
plt.show()
```


```
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	P(
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV

Checking Netflix data sanity

```
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.shape
```

```
(8807, 12)
```

```
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 = "all")
```

	show_id	type	title	director	cast	country	date_added	release_yea
count	8807	8807	8807	6173	7982	7976	8797	8807.00000
unique	8807	2	8807	4528	7692	748	1767	Na
top	s1	Movie	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	Na
freq	1	6131	1	19	19	2818	109	Na
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2014.18019
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8.81931
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1925.00000
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2013.00000
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2017.00000
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2019.00000
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2021.00000

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

	count	unique	top	freq
show_id	8807	8807	s1	1
type	8807	2	Movie	6131
title	8807	8807	Dick Johnson Is Dead	1
director	6173	4528	Rajiv Chilaka	19
cast	7982	7692	David Attenborough	19
country	7976	748	United States	2818
date_added	8797	1767	January 1, 2020	109
rating	8803	17	TV-MA	3207
duration	8804	220	1 Season	1793
listed_in	8807	514	Dramas, International Movies	362
description	8807	8775	Paranormal activity at a lush, abandoned prope...	4

```
df.size
```

```
105684
```

```
df.columns
```

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

```
df.dtypes
```

```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration        object
listed_in       object
description     object
dtype: object
```

BASIC METRIC

Time period of data frame

```
start_year = df["release_year"].min()
start_year
```

```
1925
```

```
end_year = df["release_year"].max()
end_year
```

```
2021
```

the data is between the years 1925 and 2021

Top 5 countries in descending order of movies produced

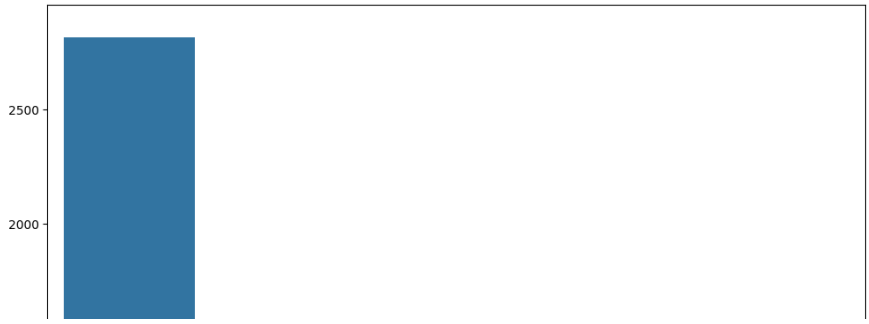
```
df["country"].value_counts()[:5]
```

```
United States     2818
India              972
United Kingdom     419
Japan              245
South Korea        199
Name: country, dtype: int64
```

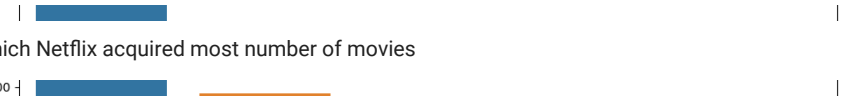
```
y_values = df["country"].value_counts()[:5]
y_values.index
```

```
Index(['United States', 'India', 'United Kingdom', 'Japan', 'South Korea'], dtype='object')
```

```
#plotting top 5 countries
plt.figure(figsize = (12, 10))
sns.barplot(x = y_values.index, y = y_values)
plt.xticks(rotation = 90)
plt.show()
```




```
# Inference : US has most number of movies and tv shows on Netflix followed by India, which is 65% less compared to US.
```




```
df.dtypes
```

```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration        object
listed_in       object
description     object
dtype: object
```

```
df["date_added"] = pd.to_datetime(df["date_added"])
df.dtypes
```

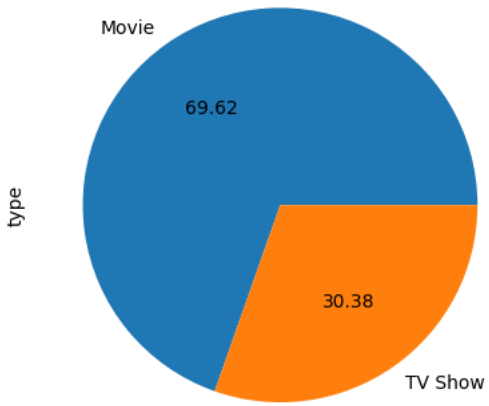
```
show_id                 object
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
dtype: object
```

```
def year_only(x):
  return x.year

df["Year_NF"] = df["date_added"].apply(year_only)
df["Year_NF"].value_counts()
```

```
2019.0    2016
2020.0    1879
2018.0    1649
2021.0    1498
2017.0    1188
2016.0     429
2015.0      82
2014.0      24
2011.0      13
2013.0      11
2012.0       3
2009.0       2
2008.0       2
2010.0       1
Name: Year_NF, dtype: int64
```

```
#INFERENCE: Neflix acquired most number of movies in 2019.
```

```
df["type"].value_counts().plot(kind = "pie", autopct = "%.2f")
plt.show()
```



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(10 Points)

```
df.shape #shape of the dataframe
```

```
(8807, 13)
```

```
df.dtypes #data-types of the columns
```

```
show_id                 object
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
Year_NF                float64
dtype: object
```

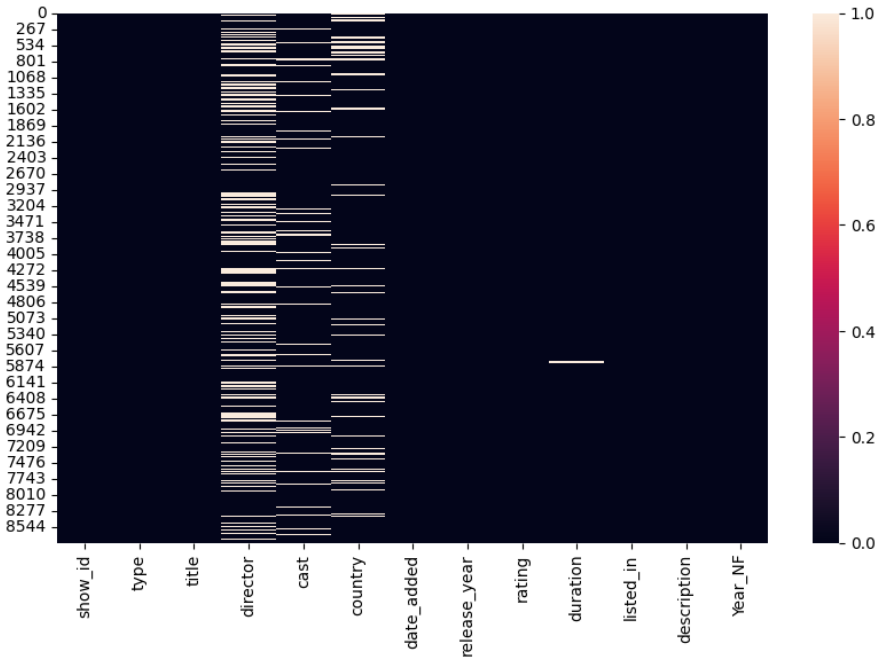
```
#type is a categorical column
#rating is a categorical column
df["type"] = df["type"].astype("category")
df["rating"] = df["rating"].astype("category")
df.dtypes
```

```
show_id                 object
type                  category
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                category
duration                object
listed_in               object
description             object
Year_NF                float64
dtype: object
```

```
df.isna().sum().sort_values(ascending = False) #null value detection
```

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

```
plt.figure(figsize=(10, 6))
sns.heatmap(df.isna())
plt.show()
```




```
#From the above heat map, it can be inferred that director, cast and country has maximum number of null values.
```

```
df.describe(include = "all")
```

```
<ipython-input-31-05881adaf55a>:1: FutureWarning: Treating datetime data as categorical
  df.describe(include = "all")
```

	show_id	type	title	director	cast	country	date_added	release_yea
count	8807	8807	8807	6173	7982	7976	8797	8807.00000
unique	8807	2	8807	4528	7692	748	1714	Na
top	s1	Movie	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	United States	2020-01-01 00:00:00	Na
freq	1	6131	1	19	19	2818	110	Na
last	NaN	NaN	NaN	NaN	NaN	NaN	2021-09-25	Na

3. Non-Graphical Analysis: Value counts and unique attributes (10 Points)

```
df.head(3)
```

	show_id	type	title	director	cast	country	date_added	release_year	rat
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	2020	P(
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	TV
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	2021-09-24	2021	TV

```
#type
df["type"].unique()
```

```
['Movie', 'TV Show']
Categories (2, object): ['Movie', 'TV Show']
```

```
df["type"].value_counts()
```

```
Movie      6131
TV Show    2676
Name: type, dtype: int64
```

```
#Column type has two unique values i.e Movie and TV Show
```

```
#director
df["director"].unique()
```

```
array(['Kirsten Johnson', nan, 'Julien Leclercq', ..., 'Majid Al Ansari',
       'Peter Hewitt', 'Mozez Singh'], dtype=object)
```

```
df["director"].nunique()
```

```
4528
```

```
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
```

```
#"director" column has 4528 unique directors and Rajiv Chilaka has highest credit to his name
```

```
#rating
df["rating"].unique()
```

```
['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', ..., '66 min', 'NR', NaN, 'TV-Y7-FV', 'UR']
Length: 18
Categories (17, object): ['66 min', '74 min', '84 min', 'G', ..., 'TV-Y', 'TV-Y7', 'TV-Y7-FV', 'UR']
```

```
df["rating"].nunique()
```

```
17
```

```
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            80
G             41
TV-Y7-FV       6
UR             3
NC-17          3
74 min         1
84 min         1
66 min         1
Name: rating, dtype: int64
```

```
#release year
df["release_year"].unique()
```

```
array([2020, 2021, 1993, 2018, 1996, 1998, 1997, 2010, 2013, 2017, 1975,
       1978, 1983, 1987, 2012, 2001, 2014, 2002, 2003, 2004, 2011, 2008,
       2009, 2007, 2005, 2006, 1994, 2015, 2019, 2016, 1982, 1989, 1990,
       1991, 1999, 1986, 1992, 1984, 1980, 1961, 2000, 1995, 1985, 1976,
       1959, 1988, 1981, 1972, 1964, 1945, 1954, 1979, 1958, 1956, 1963,
       1970, 1973, 1925, 1974, 1960, 1966, 1971, 1962, 1969, 1977, 1967,
       1968, 1965, 1946, 1942, 1955, 1944, 1947, 1943])
```

```
df["release_year"].nunique()
```

```
74
```

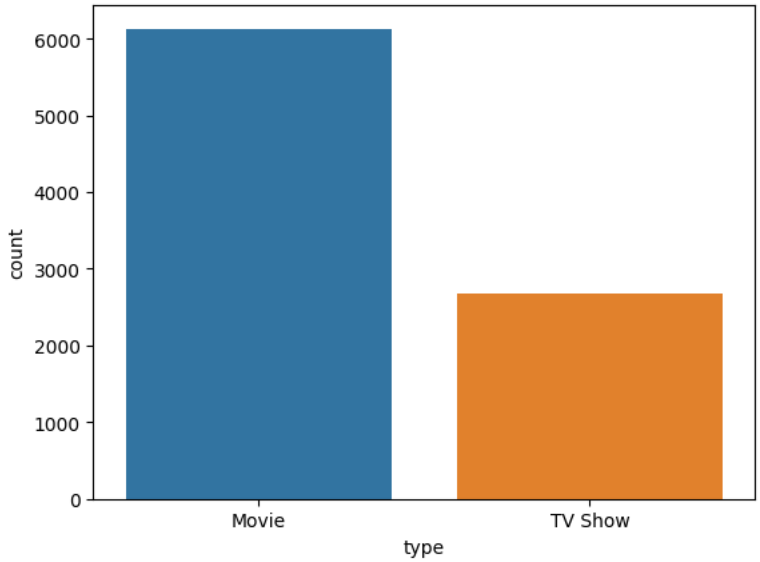
```
df["release_year"].value_counts()
```

```
2018    1147
2017    1032
2019    1030
2020     953
2016     902
        ...
1959       1
1925       1
1961       1
1947       1
1966       1
Name: release_year, Length: 74, dtype: int64
```

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

```
sns.countplot(x=df["type"]) #Univariate analysis for "type" --> type has no null values, hence imputation is not required
```

```
<Axes: xlabel='type', ylabel='count'>
```



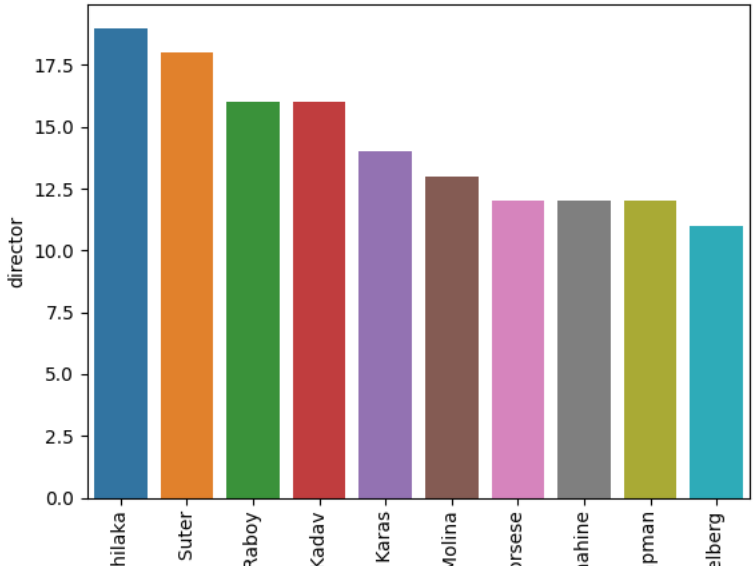

```
df["director"].fillna("Anonymous") #director column has null values so simple imputation is done by using Anonymous word.
```

```
0       Kirsten Johnson
1             Anonymous
2       Julien Leclercq
3             Anonymous
4             Anonymous
              ...
8802      David Fincher
8803          Anonymous
8804    Ruben Fleischer
8805       Peter Hewitt
8806        Mozez Singh
Name: director, Length: 8807, dtype: object
```

```
t_dir = df["director"].value_counts().reset_index().iloc[:10]
t_dir
```

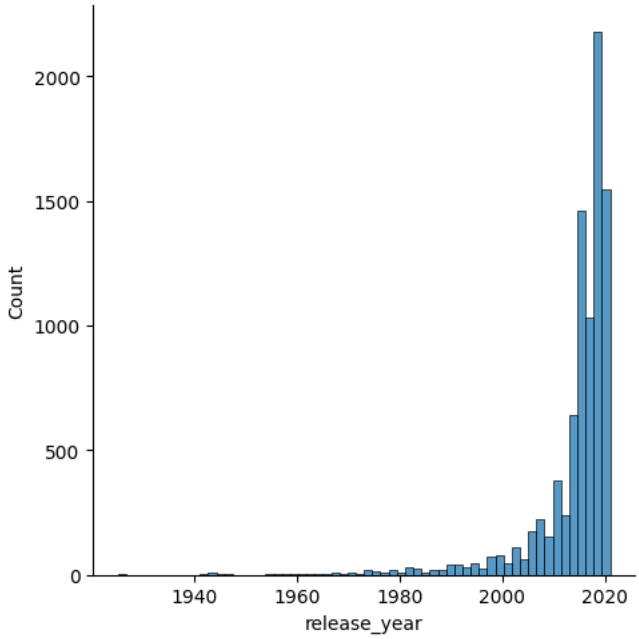
	index	director
0	Rajiv Chilaka	19
1	Raúl Campos, Jan Suter	18
2	Marcus Raboy	16
3	Suhas Kadav	16
4	Jay Karas	14
5	Cathy Garcia-Molina	13
6	Martin Scorsese	12
7	Youssef Chahine	12
8	Jay Chapman	12
9	Steven Spielberg	11

```
sns.barplot(x="index", y="director", data=t_dir) #univariate analysis on directors
plt.xticks(rotation=90)
plt.show()
```

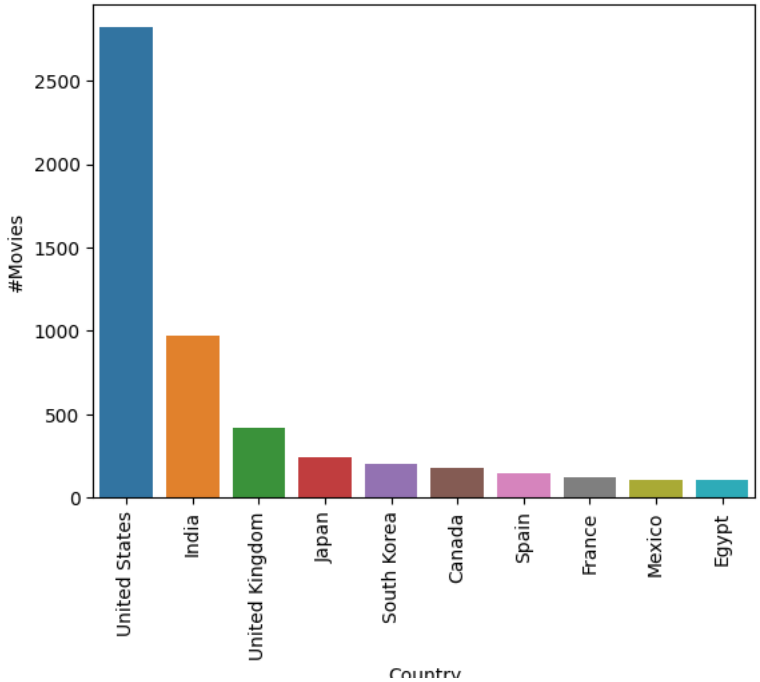



```
plt.figure(figsize=(24,12)) #Univariate analysis on release year
sns.displot(x="release_year", data=df, bins=60)
plt.show()
```

```
<Figure size 2400x1200 with 0 Axes>
```




```
#Univariate analysis of top 10 countries
def str_split(x):
  return str(x).split(",")

def str_remove(x):
  x = str(x).replace("[", "")
  x = str(x).replace("]", "")
  return x

t1 = df
t1["country"].apply(str_split)
t1.explode("country")
t1["country"].apply(str_remove)
t_country = t1["country"].value_counts().iloc[:10].reset_index()
sns.barplot(x="index", y="country", data=t_country)
plt.xticks(rotation=90)
plt.xlabel("Country")
plt.ylabel("#Movies")
plt.show()
```




```
#Bivariate Analysis


#Country and the average movie duration


def time(x):
  return float(str(x).replace("min",""))
df_movies = df[df["type"] == "Movie"]
df_movies["duration"] = df_movies["duration"].apply(time)
df_movies.head(1)
```

```
<ipython-input-54-ffba333eca98>: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
  df_movies["duration"] = df_movies["duration"].apply(time)
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	d
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	2020	PG-13	

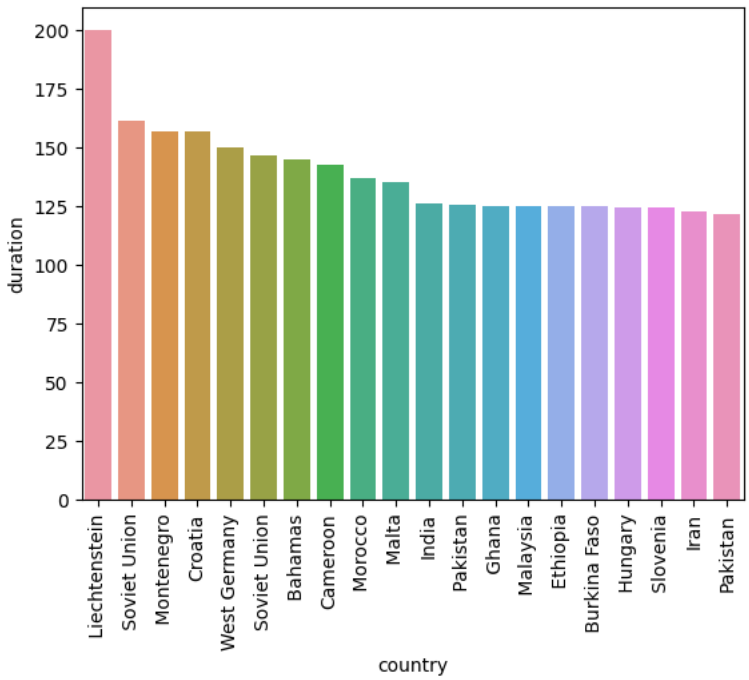
```
def str_split(x):
  return str(x).split(",")

def str_remove(x):
  x = x.replace("[", "")
  x = x.replace("]", "")
  return x

df_movies["country"] = df_movies["country"].apply(str_split)
df_movies = df_movies.explode("country")
df_movies["country"] = df_movies["country"].apply(str_remove)
df_movies.head(2)
```

```
<ipython-input-55-8132fe291b34>:9: 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
  df_movies["country"] = df_movies["country"].apply(str_split)
```

	show_id	type	title	director	cast	country	date_added	release_year	rat
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	2020	PG
			My Little	Robert					

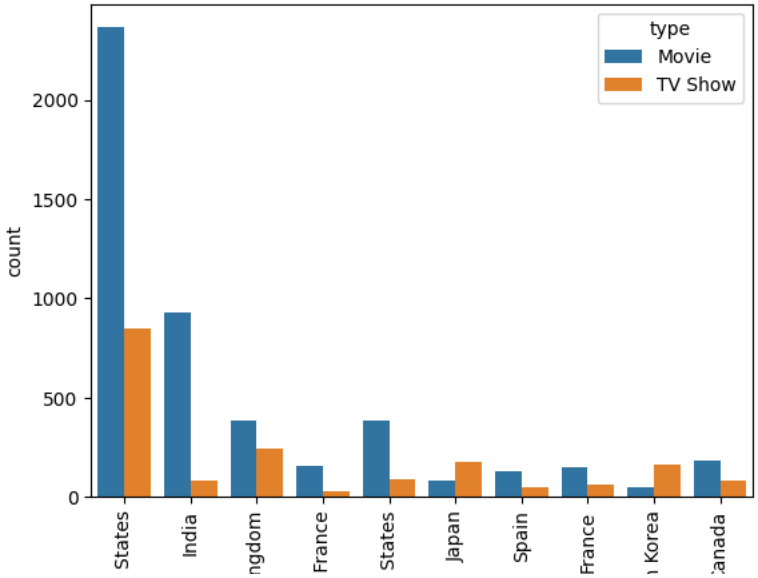
```
avg_t_mov = df_movies.groupby("country")["duration"].mean().sort_values(ascending=False).iloc[:20].reset_index()
```

```
sns.barplot(x="country", y="duration", data=avg_t_mov)
plt.xticks(rotation=90)
plt.show()
```



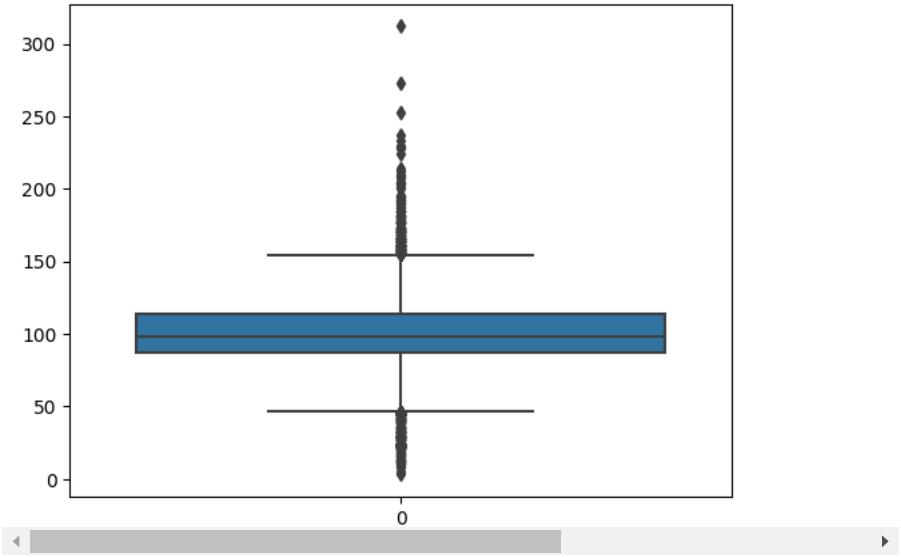

```
#Countries and Type --> Bivariate Analysis


temp_df = df
temp_df = temp_df.dropna(subset="country")

def str_split(x):
  return str(x).split(",")

def str_remove(x):
  x=x.replace("[","")
  x=x.replace("]","")
  return x

temp_df["country"] = temp_df["country"].apply(str_split)
temp_df = temp_df.explode("country")
temp_df["country"] = temp_df["country"].apply(str_remove)

arr = temp_df["country"].value_counts().iloc[:10].index
temp3 = temp_df[temp_df["country"].isin(arr)]
sns.countplot(x="country", data=temp3, hue="type")
plt.xticks(rotation=90)
plt.show()
```

```
<ipython-input-59-d46daedd2c4d>:12: 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
  temp_df["country"] = temp_df["country"].apply(str_split)
```




```
t2 = df[df["type"] == "Movie"] # Univariate boxplot on movies duration
def time(x):
  return float(str(x).replace("min", ""))

t2["duration"] = t2["duration"].apply(time)
sns.boxplot(t2["duration"])
plt.show()
```

```
<ipython-input-60-97581424537c>:5: 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
  t2["duration"] = t2["duration"].apply(time)
```




```
# Multi-variate Analysis
```

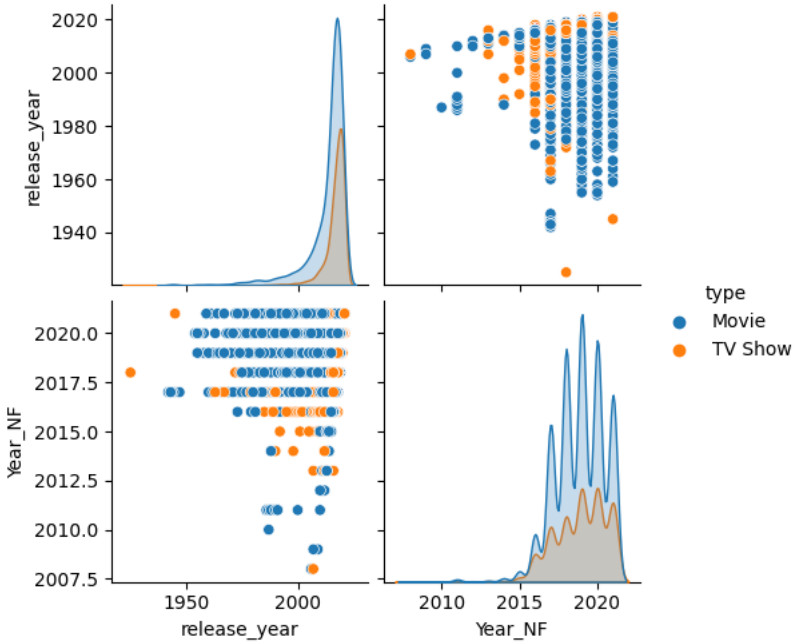
```
df.dtypes
```

```
show_id                 object
type                  category
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
```

```
release_year             int64
rating                category
duration                 object
listed_in                object
description              object
Year_NF                 float64
dtype: object
```

```
sns.pairplot(df, hue="type")
```

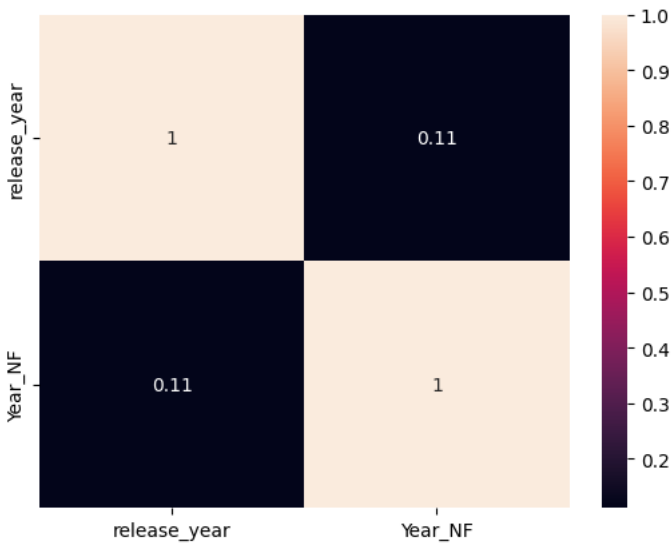
```
<seaborn.axisgrid.PairGrid at 0x7bc971690b80>
```




```
#Conclusion: From above pairplot, countplot and displot we can conclude that there are more number of movies produced as compared to tv
# shows except in year 2021 where tv shows outnumbered the number of movies
```

```
sns.heatmap(df.corr(), annot=True)
```

```
<ipython-input-65-6dc1c4c1753e>:1: FutureWarning: The default value of numeric_only in
  sns.heatmap(df.corr(), annot=True)
<Axes: >
```



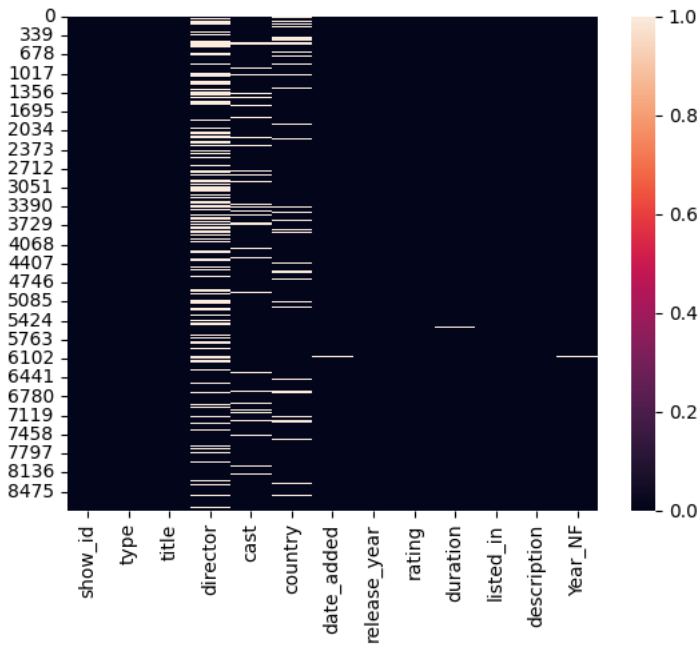

```
#Heatmap is giving correlation between attributes and since the numerical columns are very less we get a limited view and cannot infer more f
```

Missing Value & Outlier Check

```
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
Year_NF           10
dtype: int64
```

```
sns.heatmap(df.isna())
plt.show()
```




```
#From the above heat map, it can be inferred that director, cast and country has maximum number of null values.
```

```
t3 = df
```

```
def fun(x):
  return str(x).split(",")

def remove(x):
  x = str(x).replace("[", "")
  x = str(x).replace("]", "")
  return x

t3["country"] = t3["country"].apply(fun)
t3["country"] = t3["country"].apply(remove)
```

```
def imp(df):
  a = df["director"].value_counts().reset_index()
  df["top_director"] = a[a["director"] == a["director"].max()]
```

6. Insights based on Non-Graphical and visual Analysis

```
#6.1 Comments on the range of attributes
```

```
df["show_id"].nunique()
```

```
8807
```

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

```
['Movie', 'TV Show']
Categories (2, object): ['Movie', 'TV Show']
```

```
df["title"].nunique()
```

```
8807
```

```
df["director"].nunique()
```

```
4528
```

```
df["cast"].unique()
```

```
array([nan,
       'Ama Qamata, Khosi Ngema, Gail Mabalane, Thabang Molaba, Dillon Windvogel, Natasha Thahane, Arno Greeff, Xolile Tshabalala,
Getmore Sithole, Cindy Mahlangu, Ryle De Morny, Greteli Fincham, Sello Maake Ka-Ncube, Odwa Gwanya, Mekaila Mathys, Sandi Schultz,
Duane Williams, Shamilla Miller, Patrick Mofokeng',
       'Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabiha Akkari, Sofia Lesaffre, Salim Kechiouche, Noureddine Farihi, Geert Van
Rampelberg, Bakary Diombera',
       ...,
       'Jesse Eisenberg, Woody Harrelson, Emma Stone, Abigail Breslin, Amber Heard, Bill Murray, Derek Graf',
       'Tim Allen, Courteney Cox, Chevy Chase, Kate Mara, Ryan Newman, Michael Cassidy, Spencer Breslin, Rip Torn, Kevin Zegers',
       'Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanana, Manish Chaudhary, Meghna Malik, Malkeet Rauni, Anita Shabdish, Chittaranjan
Tripathy'],
      dtype=object)
```

```
df["country"].nunique()
```

```
749
```

```
df["date_added"].unique()
```

```
array(['2021-09-25T00:00:00.000000000', '2021-09-24T00:00:00.000000000',
       '2021-09-23T00:00:00.000000000', ...,
       '2018-12-06T00:00:00.000000000', '2016-03-09T00:00:00.000000000',
       '2020-01-11T00:00:00.000000000'], dtype='datetime64[ns]')
```

```
df["release_year"].unique()
```

```
array([2020, 2021, 1993, 2018, 1996, 1998, 1997, 2010, 2013, 2017, 1975,
       1978, 1983, 1987, 2012, 2001, 2014, 2002, 2003, 2004, 2011, 2008,
       2009, 2007, 2005, 2006, 1994, 2015, 2019, 2016, 1982, 1989, 1990,
       1991, 1999, 1986, 1992, 1984, 1980, 1961, 2000, 1995, 1985, 1976,
       1959, 1988, 1981, 1972, 1964, 1945, 1954, 1979, 1958, 1956, 1963,
       1970, 1973, 1925, 1974, 1960, 1966, 1971, 1962, 1969, 1977, 1967,
       1968, 1965, 1946, 1942, 1955, 1944, 1947, 1943])
```

```
df["rating"].unique()
```

```
['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', ..., '66 min', 'NR', NaN, 'TV-Y7-FV', 'UR']
Length: 18
Categories (17, object): ['66 min', '74 min', '84 min', 'G', ..., 'TV-Y', 'TV-Y7', 'TV-Y7-FV', 'UR']
```

```
df["rating"].nunique()
```

```
17
```

```
df["duration"].unique()
```

```
array(['90 min', '2 Seasons', '1 Season', '91 min', '125 min',
       '9 Seasons', '104 min', '127 min', '4 Seasons', '67 min', '94 min',
       '5 Seasons', '161 min', '61 min', '166 min', '147 min', '103 min',
       '97 min', '106 min', '111 min', '3 Seasons', '110 min', '105 min',
       '96 min', '124 min', '116 min', '98 min', '23 min', '115 min',
       '122 min', '99 min', '88 min', '100 min', '6 Seasons', '102 min',
       '93 min', '95 min', '85 min', '83 min', '113 min', '13 min',
       '182 min', '48 min', '145 min', '87 min', '92 min', '80 min',
       '117 min', '128 min', '119 min', '143 min', '114 min', '118 min',
```

```
       '108 min', '63 min', '121 min', '142 min', '154 min', '120 min',
       '82 min', '109 min', '101 min', '86 min', '229 min', '76 min',
       '89 min', '156 min', '112 min', '107 min', '129 min', '135 min',
       '136 min', '165 min', '150 min', '133 min', '70 min', '84 min',
       '140 min', '78 min', '7 Seasons', '64 min', '59 min', '139 min',
       '69 min', '148 min', '189 min', '141 min', '130 min', '138 min',
       '81 min', '132 min', '10 Seasons', '123 min', '65 min', '68 min',
       '66 min', '62 min', '74 min', '131 min', '39 min', '46 min',
       '38 min', '8 Seasons', '17 Seasons', '126 min', '155 min',
       '159 min', '137 min', '12 min', '273 min', '36 min', '34 min',
       '77 min', '60 min', '49 min', '58 min', '72 min', '204 min',
       '212 min', '25 min', '73 min', '29 min', '47 min', '32 min',
       '35 min', '71 min', '149 min', '33 min', '15 min', '54 min',
       '224 min', '162 min', '37 min', '75 min', '79 min', '55 min',
       '158 min', '164 min', '173 min', '181 min', '185 min', '21 min',
       '24 min', '51 min', '151 min', '42 min', '22 min', '134 min',
       '177 min', '13 Seasons', '52 min', '14 min', '53 min', '8 min',
       '57 min', '28 min', '50 min', '9 min', '26 min', '45 min',
       '171 min', '27 min', '44 min', '146 min', '20 min', '157 min',
       '17 min', '203 min', '41 min', '30 min', '194 min', '15 Seasons',
       '233 min', '237 min', '230 min', '195 min', '253 min', '152 min',
       '190 min', '160 min', '208 min', '180 min', '144 min', '5 min',
       '174 min', '170 min', '192 min', '209 min', '187 min', '172 min',
       '16 min', '186 min', '11 min', '193 min', '176 min', '56 min',
       '169 min', '40 min', '10 min', '3 min', '168 min', '312 min',
       '153 min', '214 min', '31 min', '163 min', '19 min', '12 Seasons',
       nan, '179 min', '11 Seasons', '43 min', '200 min', '196 min',
       '167 min', '178 min', '228 min', '18 min', '205 min', '201 min',
       '191 min'], dtype=object)
```

```
df["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 ...
                                ...
8802                       Cult Movies, Dramas, Thrillers
8803               Kids' TV, Korean TV Shows, TV Comedies
8804                               Comedies, Horror Movies
8805                   Children & Family Movies, Comedies
8806       Dramas, International Movies, Music & Musicals
Name: listed_in, Length: 8807, dtype: object
```

```
df["description"].unique()
```

```
array(['As her father nears the end of his life, filmmaker Kirsten Johnson stages his death in inventive and comical ways to help them
both face the inevitable.',
       'After crossing paths at a party, a Cape Town teen sets out to prove whether a private-school swimming star is her sister who
was abducted at birth.',
       'To protect his family from a powerful drug lord, skilled thief Mehdi and his expert team of robbers are pulled into a violent
and deadly turf war.',
       ...,
       'Looking to survive in a world taken over by zombies, a dorky college student teams with an urban roughneck and a pair of
grifter sisters.',
       'Dragged from civilian life, a former superhero must train a new crop of youthful saviors when the military preps for an attack
by a familiar villain.',
       "A scrappy but poor boy worms his way into a tycoon's dysfunctional family, while facing his fear of music and the truth about
his past."],
      dtype=object)
```

7. Business Insights - Should include patterns obeserved in the data along with what you can infer from it

```
temp4 = df
```

```
def str_split(x):
  return str(x).split(', ')

def str_remove(x):
  x = x.replace('[','')
  x = x.replace(']','')
  return x
```

```
temp4['listed_in'] =  temp4['listed_in'].apply(str_split)
temp4 = temp4.explode('listed_in')
temp4['listed_in'] = temp4['listed_in'].apply(str_remove)
temp4.head(2)
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	Year_NF
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	'United States'	2021-09-25	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...	2021.0
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane,	'South Africa'	2021-09-24	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...	2021.0

```
temp4.groupby("listed_in")["title"].nunique().sort_values(ascending = False)
```

```
listed_in
International Movies            2752
Dramas                          2427
Comedies                        1674
International TV Shows          1351
Documentaries                    869
Action & Adventure               859
TV Dramas                        763
Independent Movies               756
Children & Family Movies         641
Romantic Movies                  616
TV Comedies                      581
Thrillers                        577
Crime TV Shows                   470
Kids' TV                         451
Docuseries                       395
Music & Musicals                 375
Romantic TV Shows                370
Horror Movies                    357
Stand-Up Comedy                  343
Reality TV                       255
British TV Shows                 253
Sci-Fi & Fantasy                 243
Sports Movies                    219
Anime Series                     176
Spanish-Language TV Shows        174
TV Action & Adventure            168
Korean TV Shows                  151
Classic Movies                   116
LGBTQ Movies                     102
TV Mysteries                      98
Science & Nature TV               92
TV Sci-Fi & Fantasy               84
TV Horror                         75
Anime Features                    71
Cult Movies                       71
Teen TV Shows                     69
Faith & Spirituality              65
TV Thrillers                      57
Movies                            57
Stand-Up Comedy & Talk Shows      56
Classic & Cult TV                 28
TV Shows                          16
Name: title, dtype: int64
```

Duaration based analysis

```
import pandas as pd
import numpy as np
import seaborn as sns
df = pd.read_csv("netflix.csv")

def time(x):
  return float(str(x).replace(' min',''))


df_movies = df[df['type'] == 'Movie']
```

```
df_movies['duration'] = df_movies['duration'].apply(time)
df_movies.head(1)
```

```
<ipython-input-105-25d33dc29b59>:11: 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-c
  df_movies['duration'] = df_movies['duration'].apply(time)
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90.0	Documentaries	As her father nears the end of his life, filmm...

```
def str_split(x):
  return str(x).split(', ')

def str_remove(x):
  x = x.replace('[','')
  x = x.replace(']','')
  x.replace(r"['\"]", '')
  return x

df_movies['country'] =  df_movies['country'].apply(str_split)
df_movies = df_movies.explode('country')
df_movies['country'] = df_movies['country'].apply(str_remove)
df_movies.head(2)
```

```
<ipython-input-106-f54d744287a3>:10: 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-c
  df_movies['country'] =  df_movies['country'].apply(str_split)
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90.0	Documentaries	As her father nears the end of his life, filmm...
6	s7	Movie	My Little Pony: A New Generation	Robert Cullen, José Luis Ucha	Vanessa Hudgens, Kimiko Glenn, James Marsden, ...	nan	September 24, 2021	2021	PG	91.0	Children & Family Movies	Equestria's divided. But a bright-eyed hero be...

```
df_movies.groupby('country')['duration'].mean().sort_values(ascending = False)
```

```
country
Liechtenstein      200.000000
Montenegro         157.000000
Soviet Union       156.666667
Bahamas            145.000000
Cameroon           143.000000
                        ...
Guatemala           68.000000
Uganda              68.000000
Kazakhstan          67.000000
United Kingdom,     62.000000
Syria               52.000000
Name: duration, Length: 123, dtype: float64
```

```
df_movies.groupby('country')['duration'].mean().sort_values(ascending = False).loc['India']
```

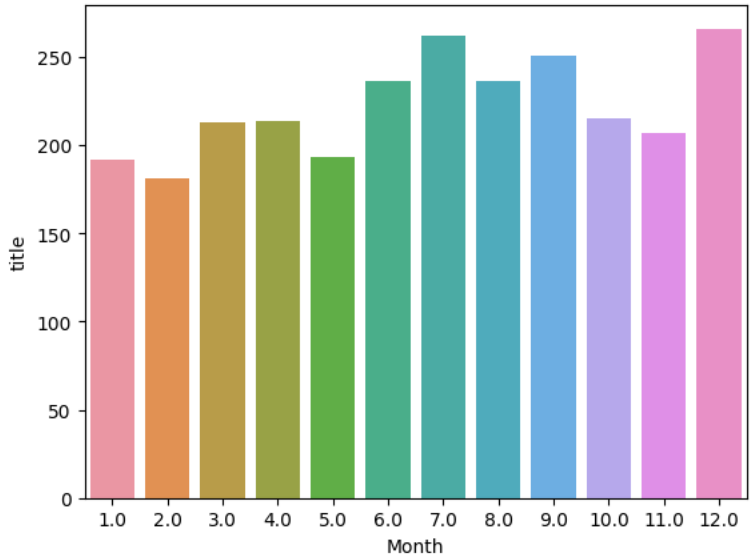
```
125.91268191268192
```

INFERENCE: Average duration of movies for all countries is given above and for Indian audience the preferred duration is around 125 minutes. Netflix can acquire movies in India with 125 minutes duration or edit movies to reduce time to around 125 minutes.

RIGHT TIME TO LAUNCH TV SHOWS AND MOVIES

```
def month(x):
  return x.month

df['Month'] = df['date_added'].apply(month)
new = df.groupby(['type','Month'])['title'].count().reset_index()
new_movies = new[new['type'] == 'Movie']
new_tvshows = new[new['type'] == 'TV Show']

sns.barplot(x = 'Month',y = 'title',data =new_tvshows)
```

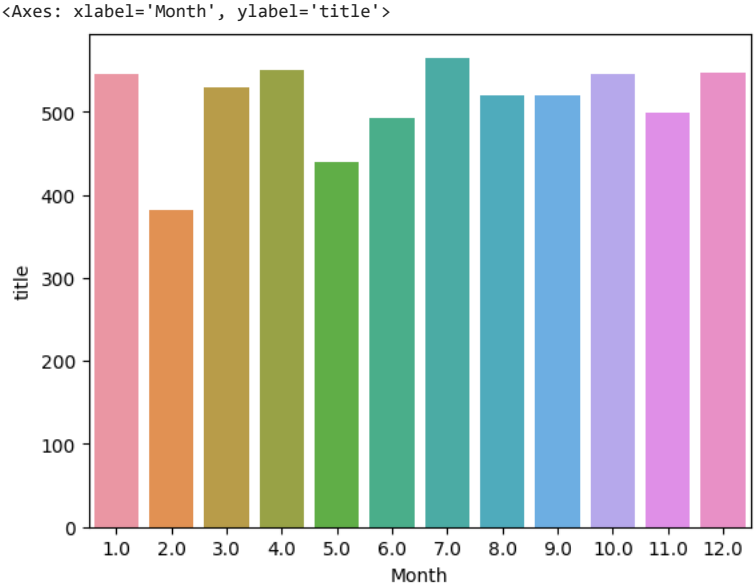
```
<Axes: xlabel='Month', ylabel='title'>
```



Inference: The best time to release a TV show is July, Sept and Dec.

```
sns.barplot(x="Month", y="title", data=new_movies)
```

```
<Axes: xlabel='Month', ylabel='title'>
```



Inference: There is no particular trend for movies but Netflix can avoid acquiring movies during Feb, May and June

Analysis of actors/directors of different types of shows/movies

```
temp5 = df[df["type"] == "Movie"]
temp5 = temp5.dropna(subset='cast')
```

```
def str_split(x):
  return str(x).split(', ')

def str_remove(x):
  x = x.replace("[",'')
  x = x.replace(']','')
  return x

temp5['cast'] =  temp5['cast'].apply(str_split)
temp5 = temp5.explode('cast')
temp5['cast'] = temp5['cast'].apply(str_remove)
temp5.head(2)
```

	show_id	type	title	director	cast	country	date_added	release_year	rat
6	s7	Movie	My Little Pony: A New Generation	Robert Cullen, José Luis Ucha	Vanessa Hudgens	NaN	September 24, 2021	2021	
6	s7	Movie	My Little Pony: A New Generation	Robert Cullen, José Luis Ucha	Kimiko Glenn	NaN	September 24, 2021	2021	

```
temp5.groupby('cast')['title'].count().sort_values(ascending = False)
```

```
cast
Anupam Kher         42
Shah Rukh Khan      35
Naseeruddin Shah    32
Om Puri             30
Akshay Kumar        30
                    ..
Jacob Buster         1
Jacob Blair          1
Jacob Bertrand       1
Jacob Batalon        1
Ṣọpẹ́ Dìrísù         1
Name: title, Length: 25951, dtype: int64
```

Anupam Kher is one of the best actors followed by Shah Rukh Khan

```
temp5 = df[df["type"] == "TV Show"]
temp5 = temp5.dropna(subset='cast')

def str_split(x):
  return str(x).split(', ')

def str_remove(x):
  x = x.replace("[",'')
  x = x.replace(']','')
  return x
temp5['cast'] =  temp5['cast'].apply(str_split)
temp5 = temp5.explode('cast')
temp5['cast'] = temp5['cast'].apply(str_remove)
temp5.groupby('cast')['title'].count().sort_values(ascending = False)
```

```
cast
Takahiro Sakurai         25
Yuki Kaji                19
Junichi Suwabe           17
Daisuke Ono              17
Ai Kayano                17
                         ..
Ivy Yin                   1
Iván Pellicer             1
Iván Álvarez de Araya     1
Iza Moreira               1
Şükrü Özyıldız            1
Name: title, Length: 14863, dtype: int64
```

```
#Inference: Takahiro Sakurai is the best among the actors in TV Show.
```

```
temp5 = df[df["type"] == "Movie"]
temp5 = temp5.dropna(subset='director')

def str_split(x):
  return str(x).split(', ')

def str_remove(x):
  x = x.replace("[",'')
  x = x.replace(']','')
  return x

temp5['director'] =  temp5['director'].apply(str_split)
temp5 = temp5.explode('director')
temp5['director'] = temp5['director'].apply(str_remove)
temp5.groupby('director')['title'].nunique().sort_values(ascending = False)
```

```
director
Rajiv Chilaka            22
Jan Suter                21
Raúl Campos              19
Suhas Kadav              16
Marcus Raboy             15
                         ..
José Miguel Contreras     1
José Ortuño               1
Bob Odenkirk              1
Jovanka Vuckovic          1
Avinash Walzade           1
Name: title, Length: 4777, dtype: int64
```

```
#Inference: Rajiv Chilaka is the most director followed by Jan Suter in Movies
```

```
temp5 = df[df["type"] == "TV Show"]
temp5 = temp5.dropna(subset='director')

def str_split(x):
  return str(x).split(', ')

def str_remove(x):
  x = x.replace("[",'')
  x = x.replace(']','')
  return x

temp5['director'] =  temp5['director'].apply(str_split)
temp5 = temp5.explode('director')
temp5['director'] = temp5['director'].apply(str_remove)
temp5.groupby('director')['title'].nunique().sort_values(ascending = False)
```

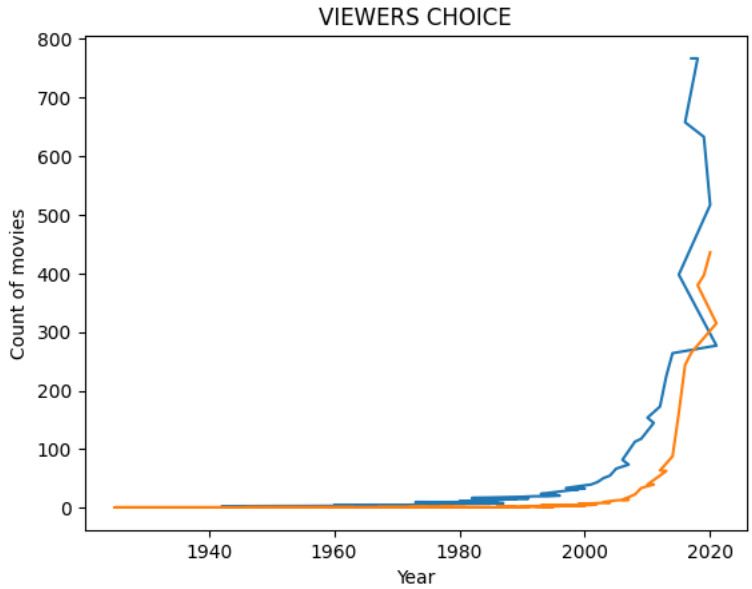
```
director
Ken Burns              3
Alastair Fothergill    3
Stan Lathan            2
Jung-ah Im             2
Joe Berlinger          2
                      ..
Hong Won-ki            1
Hiroyuki Seshita       1
Hikaru Toda            1
Hernán Guerschuny      1
Ziad Doueiri           1
Name: title, Length: 299, dtype: int64
```

```
#Inference: Ken Burns is the most famous director in TV Shows
```

```
#Does Netflix has more focus on TV Shows than movies in recent years
```

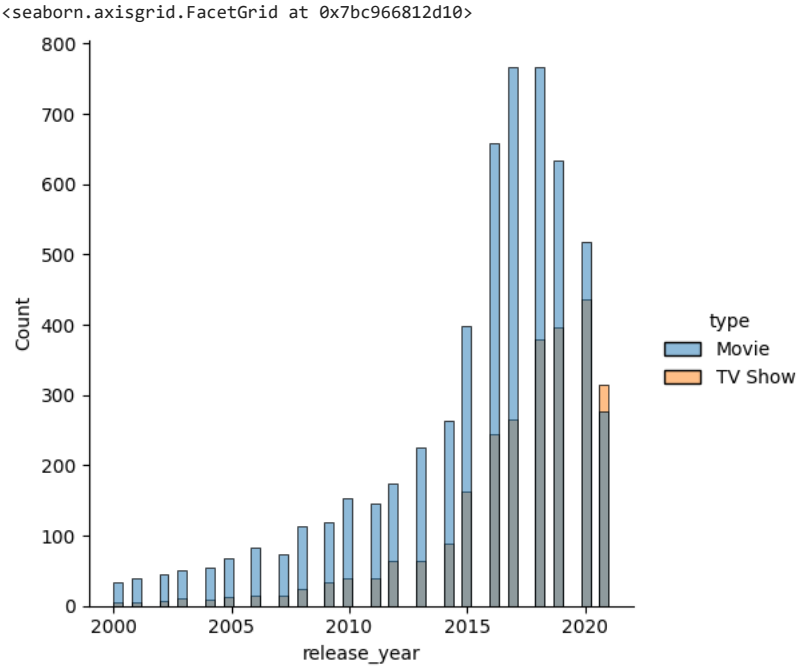
```
df_movies = df[df['type'] == 'Movie']
nf_df_tvshows = df[df['type'] == 'TV Show']
y = df_movies['release_year'].value_counts()
y2 = nf_df_tvshows['release_year'].value_counts()
plt.title('VIEWERS CHOICE')
plt.plot(y)
plt.plot(y2)
plt.xlabel('Year')
plt.ylabel('Count of movies')
```

```
Text(0, 0.5, 'Count of movies')
```




```
#After year 2000


nf_df_2000 = df[df['release_year'] >= 2000]
sns.displot(data = nf_df_2000,x = 'release_year',hue = 'type')
```

```
<seaborn.axisgrid.FacetGrid at 0x7bc966812d10>
```



Inference: The counts of both movies and TV shows have increased post 2000, and the number of movies has consistently been higher than that of TV shows. However, most probably due to the impact of COVID, the number of movies has decreased; and in 2021, the count of TV shows surpassed that of movies.

Understanding what content is available in different countries

```
temp6 = df
temp6 = temp6.dropna(subset = 'country')
def str_split(x):
  return str(x).split(', ')

def str_remove(x):
  x = x.replace('[','')
```

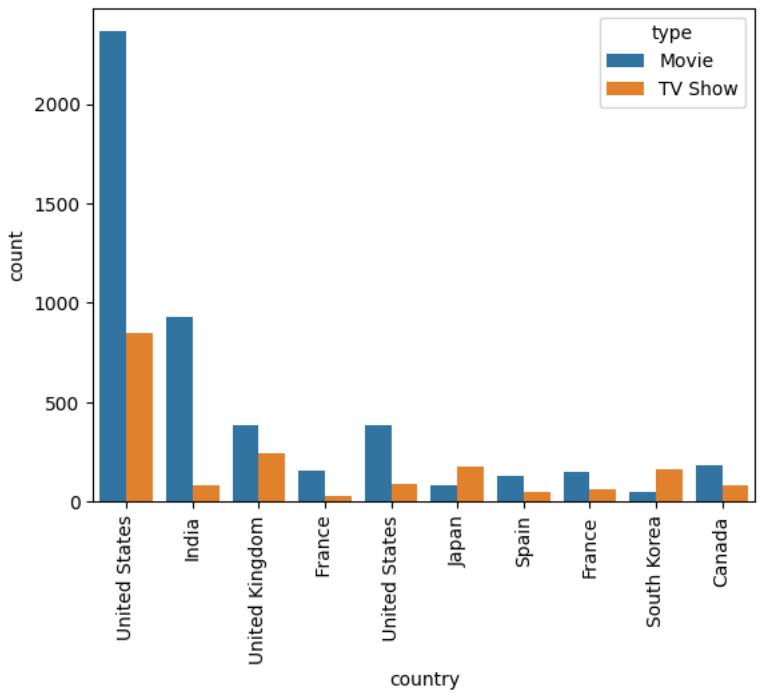
```
  x = x.replace(']','')
  return x

temp6['country'] =  temp6['country'].apply(str_split)
temp6 = temp6.explode('country')
temp6['country'] = temp6['country'].apply(str_remove)

arr = temp6['country'].value_counts().iloc[:10].index
```

```
<ipython-input-119-9a300aed1a67>:11: 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-c
  temp6['country'] =  temp6['country'].apply(str_split)
```

```
arr
```

```
Index(['United States', 'India', 'United Kingdom', 'Canada', 'France', 'Japan',
       'Spain', 'South Korea', 'Germany', 'Mexico'],
      dtype='object')
```

```
temp7 = temp6[temp6['country'].isin(arr)]
sns.countplot(x = 'country',data = temp3,hue = 'type')
plt.xticks(rotation = 90)
plt.show()
```



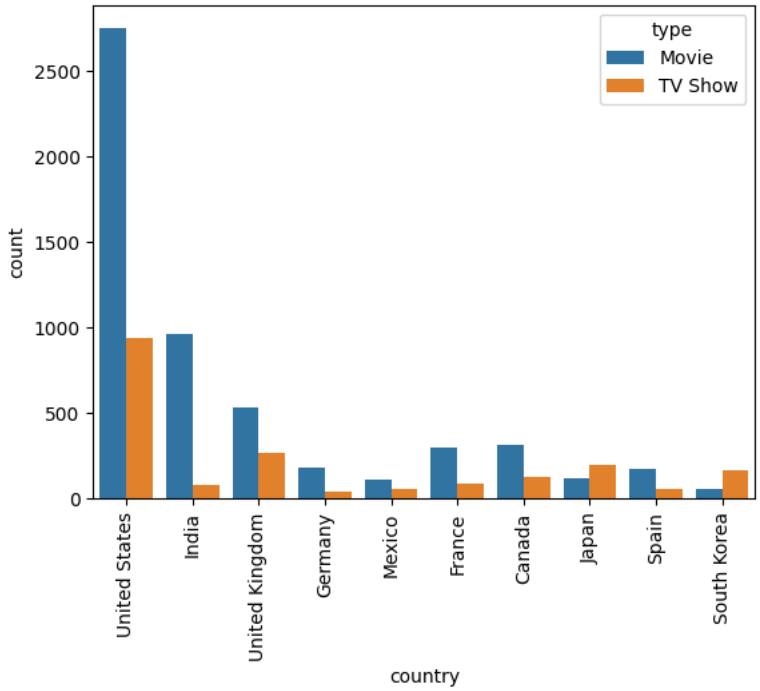
INFERENCE 1: Netflix has produced most movies/shows for its audience in USA followed by India and U.K

INFERENCE 2: Audience preferred choice is movies in most countries except for Japan and South Korea where TV shows are preferred over movies

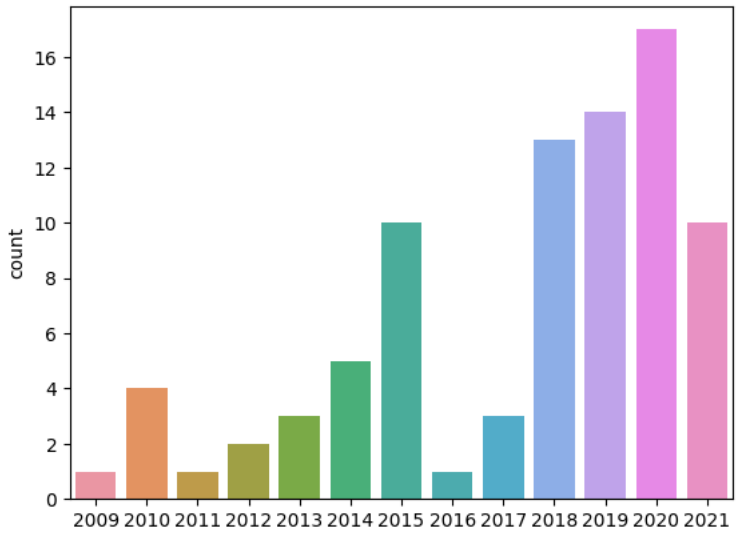
8. Recommendations: Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand

```
t6 = pd.read_csv("netflix.csv")
t6 = t6.dropna(subset = 'country')
def str_split(x):
  return str(x).split(', ')

def str_remove(x):
  x = x.replace('[','')
```

```
  x = x.replace(']','')
  return x

t6['country'] =  t6['country'].apply(str_split)
t6 = t6.explode('country')
t6['country'] = t6['country'].apply(str_remove)

arr = t6['country'].value_counts().iloc[:10].index
t6 = t6[t6['country'].isin(arr)]
sns.countplot(x = 'country',data = t6,hue = 'type')
plt.xticks(rotation = 90)
plt.show()
```




```
t7 = t6[(t6['type'] == 'TV Show') & (t6['country'] == 'India')]
sns.barplot(x = t7['release_year'].value_counts().index,y = t7['release_year'].value_counts())
plt.ylabel('count')
plt.show()
```



Actionable Insights: The craze for TV Shows are increasing in India in recent years.Netflix can produce some good TV Show content to target the massive Indian population.

Ratings specific EDA

```
rat=df['rating'].nunique(dropna=False)
rat
```

```
18
```

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

```
array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R',
       'TV-G', 'G', 'NC-17', '74 min', '84 min', '66 min', 'NR', nan,
       'TV-Y7-FV', 'UR'], dtype=object)
```

```
df["rating"].value_counts() > 6
```

```
TV-MA        True
TV-14        True
TV-PG        True
R            True
PG-13        True
TV-Y7        True
TV-Y         True
PG           True
TV-G         True
NR           True
G            True
TV-Y7-FV    False
NC-17       False
UR          False
74 min      False
84 min      False
66 min      False
Name: rating, dtype: bool
```

```
df["release_year"].value_counts().head(10)
```

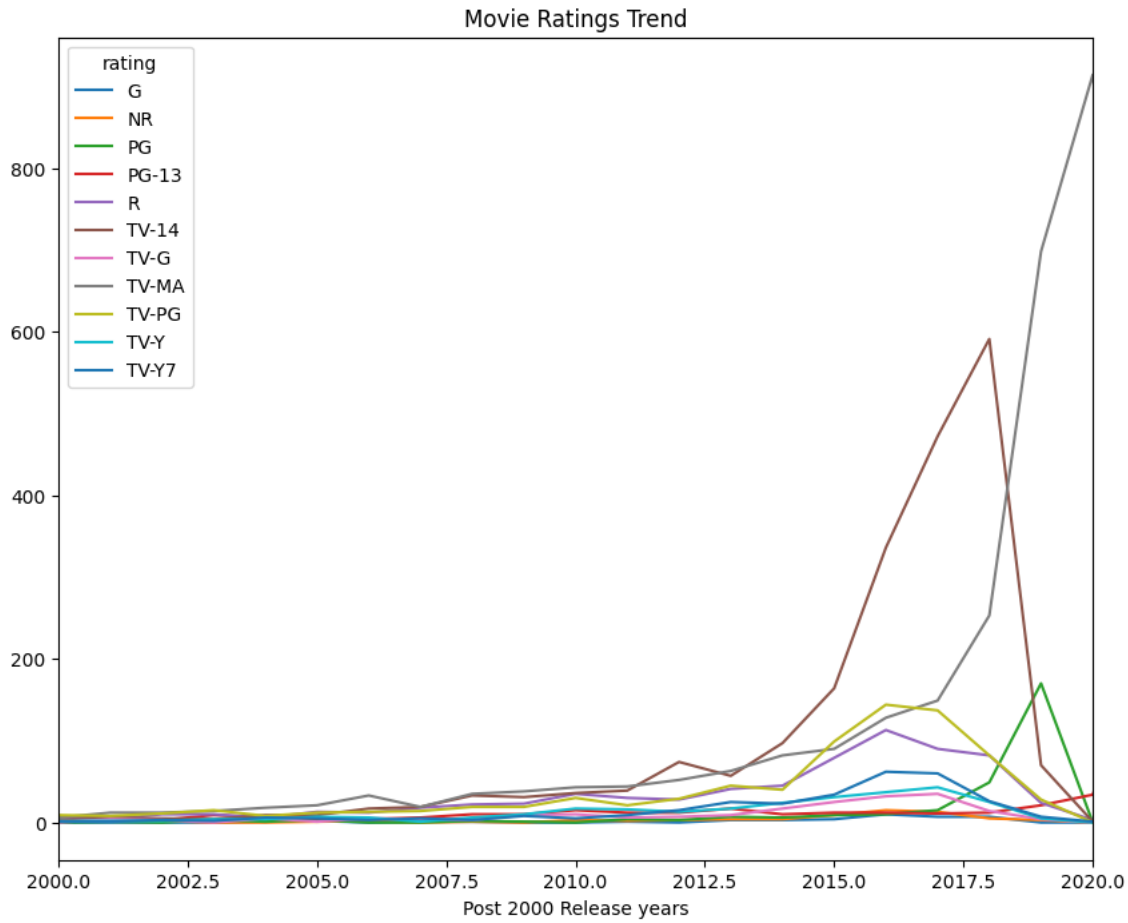
```
2018    1147
2017    1032
2019    1030
2020     953
2016     902
2021     592
2015     560
2014     352
2013     288
2012     237
Name: release_year, dtype: int64
```

```
nf_rating_copy = df
nf_rat = df.groupby('rating')[['release_year']].count()
nf_rat = nf_rat.drop(['74 min','84 min','66 min','UR','TV-Y7-FV','NC-17'])
nf_rat.reset_index(inplace = True)
nf_rat.columns = ['rating','release_year counts']
nf_rat
```

```
nf_rating=nf_rating_copy.merge(nf_rat,  how='inner')
nf_rating.head()
nf_rating.shape
```

```
(8788, 13)
```

```
pd.crosstab(df['release_year'] ,nf_rating['rating'] ).plot(kind= 'line',figsize = (10,8),title = 'Movie Ratings Trend')
plt.xlim(2000,2020)
plt.xlabel('Post 2000 Release years')
plt.show()
```



Actionable Insights: I have removed the outliers i.e the values ('74 min','84 min','66 min','UR','TV-Y7-FV','NC-17') those were not contributing enough to the plot. Also, as the values/counts were almost non changing for years less than 2000, I have visualized after 2000. From above line plot, as we can clear see that TV-14(unsuitable for childer under 14) has been decreasing lately due to advancements in internet technology and TV-MA(content for mature adults) has been more preferred now a days owing to lockdown restriction and self-isolations due to COVID. Hence Netflix should focus more on content related to TV-MA ratings

Actionable Insights:Out of total 962 Indian movies, most of the movies have duration in 100-150 mins. Netflix should focus on this range while producing more in this range as eveident from above pairplot. Movies with rating TV-14(547) i.e content for children above 14 are mostly preffered as opposed to the overall movie ratings throughout the world where TV-MA(232) is dominant. Netflix should focus more on this aspect with respect to Indian audience.

Actionable Insights: Out of total 84 tv shows, the tv shows which are mostly enjoyed and hence has more seasons as compared to others are given in nf_India_tv dataframe. As it's eveident from the pairplot for TV shows, Netflix should focus mostly on TV-MA (34) and TV-14 (25) i.e TV shows for mature adults and under 14 years as they are mostly preferred by the audience

✓ 2s completed at 10:28 PM