```
#!pip install geopandas
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.patches as mpatches
import re
import geopandas as gpd
```

```
df = pd.read_csv("netflix.csv")
ChartNumber=""
```

Welcome To Netflix CaseStudy

Defining Problem Statement and Analysing basic metrics (10 Points)

- Help Netflix in deciding which type of shows/movies to produce
- How they can grow the business in different countries

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)

```
print(f'The data contains {df.shape[0]} rows and {df.shape[1]} columns')
print("\nHere are the columns it contains")
print(df.dtypes)

print('\nSome columns have null values.  Here are the details')

for idx, value in df.isna().sum().items():
    if(value>0):
        print(f"Column: {idx}, NA Values: {value}")
```

```
The data contains 8807 rows and 12 columns

Here are the columns it contains
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

Some columns have null values.  Here are the details
Column: director, NA Values: 2634
Column: cast, NA Values: 825
Column: country, NA Values: 831
Column: date_added, NA Values: 10
Column: rating, NA Values: 4
Column: duration, NA Values: 3
```

```
df
```

	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 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
			Jailbirds									Feuds,

```
df.type.unique()
```

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

- There are two type of shows – Movies and TV Show

```
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)
```

- There are 14 Ratings - 'PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R', 'TV-G', 'G', 'NC-17', 'NR', 'TV-Y7-FV', 'UR'
- TV-MA is the most common rating for both movies and TV Show

```
df.type.value_counts()
```

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

- There are 6131 movies and 2676 TV Shows
- Each TV show could be having 12 episodes, each lasting 40 minutes.

Lets clean NA values

```
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
```

We have a total of 8807 rows.

Following columns have null values director 2634 cast 825 country 831 date_added 10 rating 4 duration 3

From the above analysis, we see that

1. country has 831 Null values

- so 10% of rows dont have country.
- Country is a important parameter and we would like to fill these NA values probably.

2. There are 2634 entries with no director info, 825 with no cast.
   - These two columns are not that important for immedate analysis, and we would probably work without these columns.
3. date_added
   - is a relatively important field for us. So we would like to impute.
   - As we have all values in release year field, we can use it to fill date_added
4. Rating
   - only 4 rows having null
   -
     - can be given avg value of 3
5. Duration
   - only 3 rows having null
   - on further analysis, it was found that the duration values have gone to the rating column for these rows.
6. Director, cast - we will fill "other"

```
#Lets take care of null values.

#Handling null values For Countries- put ALL
df.country = df.country.fillna('ALL')

```

```
#Handling null values For rating
#Find the most popular rating for each type
df.groupby('type')['rating'].agg(lambda x: x.mode().iloc[0])
```

```
type
Movie      TV-MA
TV Show    TV-MA
Name: rating, dtype: object
```

- TV-MA is the most common rating for both movies and TV Show

```
#Rating- put TV-MA is most popular rating for both types of shows, so fill it across every null value in rating
df.rating = df.rating.fillna('TV-MA')
```

```
#Handling null values For date added by making it as the last day of release year, that is 31-12-<release year>
#before that though, we need to do some cleaning of the date_added column

#Some dates have trailing or leading spaces. remove them
df['date_added'] = df['date_added'].str.strip()
# Trying to convert the 'date_column' to datetime
df['converted_date'] = pd.to_datetime(df['date_added'], errors='coerce')
```

```
# Identifying rows that are strings (invalid dates)
invalid_dates = df[df['converted_date'].isna()]
invalid_dates
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	conv
6066	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Daniel Radcliffe, Jon Hamm, Adam Godley, Chris...	United Kingdom	NaN	2013	TV-MA	2 Seasons	British TV Shows, TV Comedies, TV Dramas	Set during the Russian Revolution, this comic ...	
6174	s6175	TV Show	Anthony Bourdain: Parts Unknown	NaN	Anthony Bourdain	United States	NaN	2018	TV-PG	5 Seasons	Docuseries	This CNN original series has chef Anthony Bour...	
6795	s6796	TV	Frasier	NaN	Kelsey Grammer, Jane Leeves	United	NaN	2003	TV-PG	11	Classic & Cult TV, TV	Frasier Crane is a snooty but	

```
#Fix invalid dates by using release_year
for r in invalid_dates.iterrows():
    print(r[0])
    #print(r[1]['release_year'])
    df.loc[r[0],'converted_date'] = pd.to_datetime(str(r[1]['release_year']) + '-12-01')
```

```
6066
6174
6795
6806
6901
7196
7254
7406
7847
8182
```

```
#verify we no longer have any na values in the converted_date column
df['converted_date'].isna().sum()
```

```
0
```

```
#date_added can be replaced with converted_date, which can then be dropped.
df['date_added'] = df['converted_date']
df.drop(columns='converted_date', inplace=True)
df
```

	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	2021-09-25	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	ALL	2021-09-24	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
			Jailbirds									Feuds,

```
#Fill director and cast columns
df.director= df.director.fillna('other')
df.cast= df.cast.fillna('other')
```

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

```
show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        3
listed_in       0
description     0
dtype: int64
```

```
#Duration column is next.
invalid_duration = df[df['duration'].isna()]

```

```
#The rows that have nan duration actually have wrong values in rating
#Duration values have gone into rating column !

#IF we had only one column to update we could have used this, but we need to update two columns
#df['durationN'].fillna(df['rating'], inplace=True)

for r in invalid_duration.iterrows():
    print('Updating index : ',r[0])
    #print(r[1]['release_year'])
    df.loc[r[0],'duration'] = r[1]['rating']
    df.loc[r[0],'rating'] = 'TV-MA' ##fill the most common rating
```

```
Updating index :  5541
Updating index :  5794
Updating index :  5813
```

```
#Verify we dont have any invalid duration
invalid_duration = df[df['duration'].isna()]
invalid_duration
```

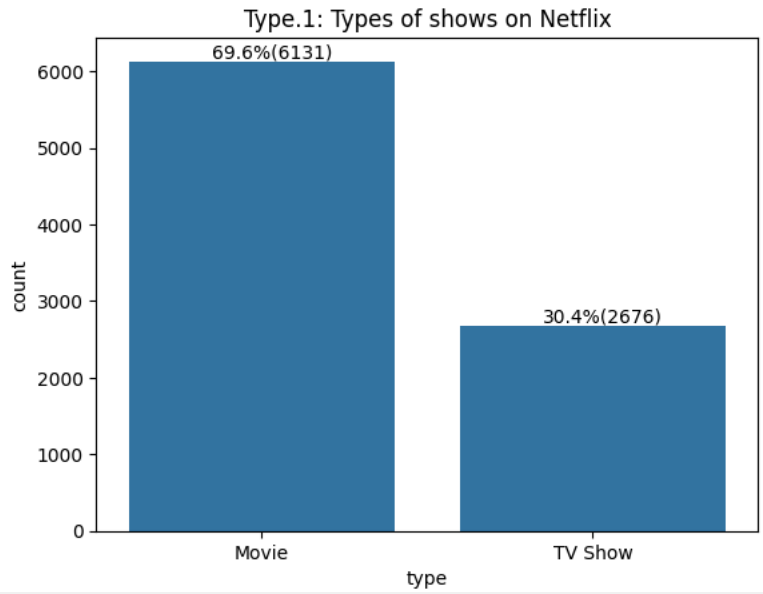
show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description

Data has been cleaned !

```
#We now got a clean dataframe, lets save it
df_cleaned = df.copy(deep=True)
df_cleaned.to_csv("df_cleaned.csv",mode='w')

```

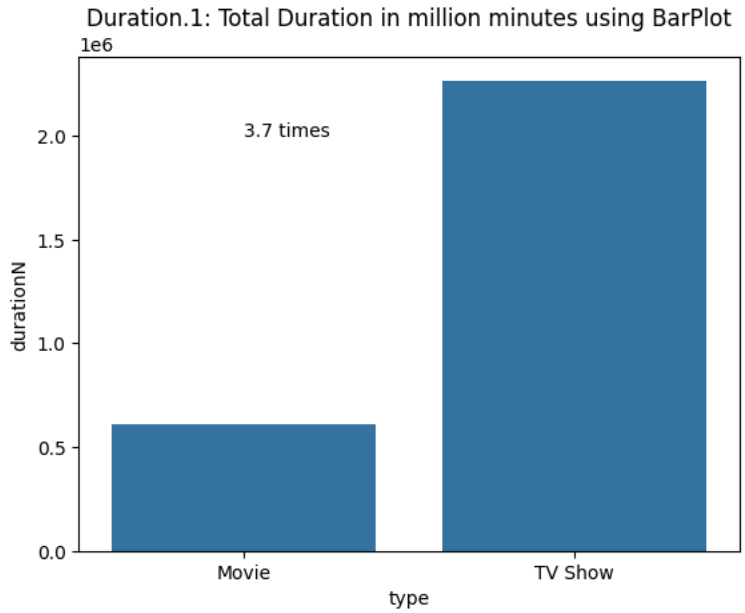
```
#Lets add a new duration column, durationN, standing for duration Normalized, to have only minutes based data.
#we will assume a season has 12 shows of 40 minutes each.

def normalizeDuration(x):
    #print(x)
    try:
        val = int(re.search(r'\d+',x).group())
    except:
        print("**************")
    if("Season" in x):
        val*=40*12
    #print(val)
    return val
dfWithDuration = df.copy(deep=True)
dfWithDuration['durationN'] = dfWithDuration['duration'].apply(normalizeDuration)

```

```
plot=sns.countplot(data=dfWithDuration, x='type')
ChartNumber="Type.1"
plt.title(f'{ChartNumber}: Types of shows on Netflix')
totalNumShows = dfWithDuration.shape[0]
for p in plot.patches:
 plot.annotate('{:}%({:})'.format(round(p.get_height()*100/totalNumShows,1),round(p.get_height())), (p.get_x()+0.25, p.get_height()+50))
plt.show()
```



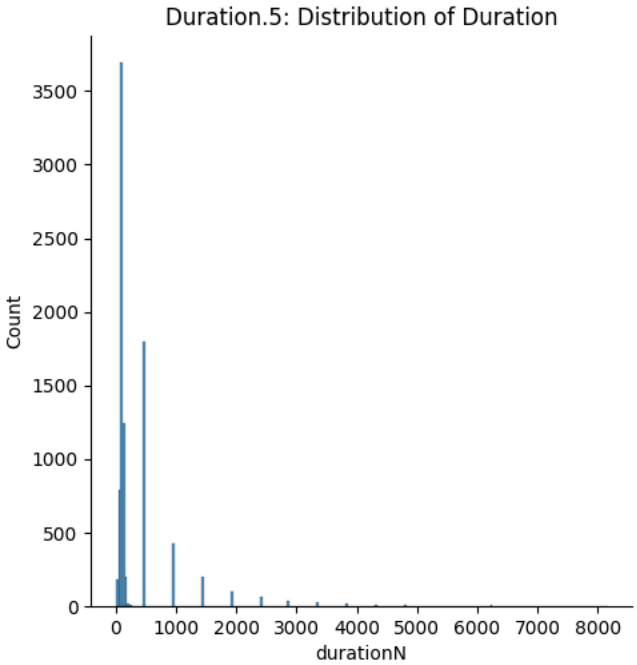

```
#Lets find total duration for each type
durationData = dfWithDuration.groupby('type').durationN.sum()
sns.barplot(data= durationData)
ChartNumber="Duration.1"
plt.title(f'{ChartNumber}: Total Duration in million minutes using BarPlot')
plt.text(0,2000000,f'{round(durationData.iloc[1]/durationData.iloc[0],1)} times')
```

```
Text(0, 2000000, '3.7 times')
```



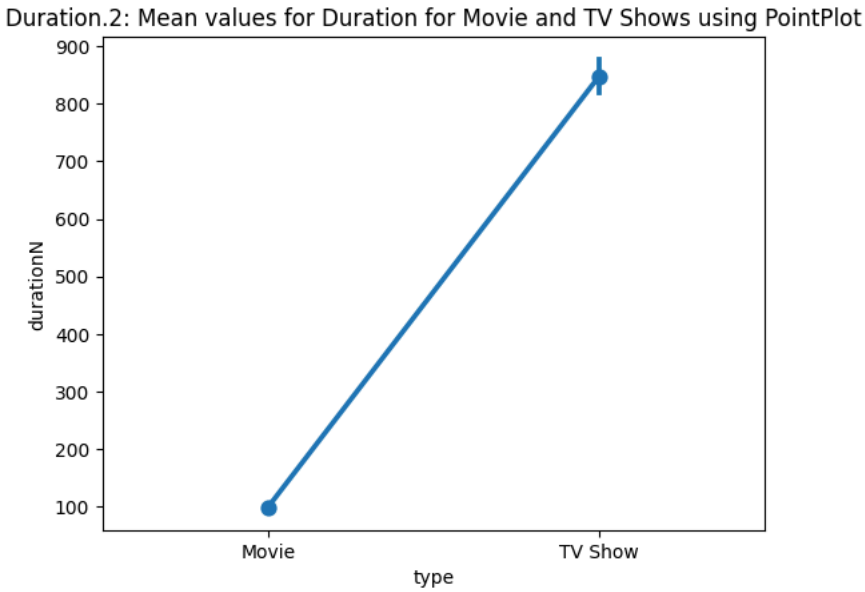

```
#plt.hexbin(x=df.show_id, y=df.durationN)
sns.displot(dfWithDuration.durationN)

# Set x-axis limits
#plt.xlim(0, 500)
ChartNumber="Duration.5"
plt.title(f'{ChartNumber}: Distribution of Duration')
```

```
Text(0.5, 1.0, 'Duration.5: Distribution of Duration')
```



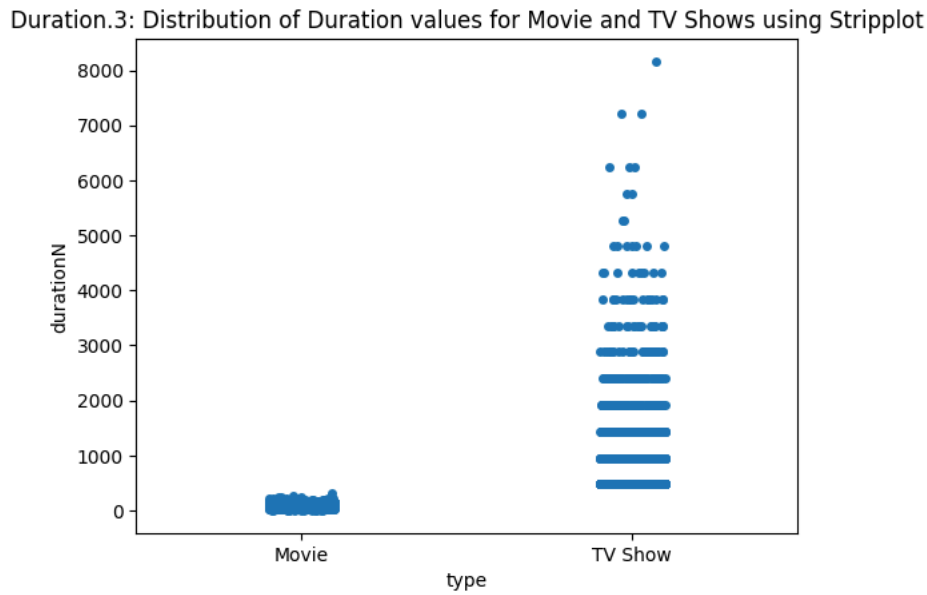

```
sns.pointplot(data= dfWithDuration, x='type', y='durationN')
ChartNumber="Duration.2"
plt.title(f'{ChartNumber}: Mean values for Duration for Movie and TV Shows using PointPlot')
```

```
Text(0.5, 1.0, 'Duration.2: Mean values for Duration for Movie and TV Shows using PointPlot')
```




```
sns.stripplot(data= dfWithDuration, x='type', y='durationN')
ChartNumber="Duration.3"
plt.title(f'{ChartNumber}: Distribution of Duration values for Movie and TV Shows using Stripplot')
```

```
Text(0.5, 1.0, 'Duration.3: Distribution of Duration values for Movie and TV Shows using Stripplot')
```



Interpretation - Assuming that each season has 12 episodes and each episode is 40 min on avg, we see that while number of TV shows are less than half of movies, they contain 3.7 times more 'viewing minutes' compared to movies. In other words, out of every 10 minutes of material on Netflix, 8 minutes is TV Show

Recommendation: Netflix should focus more on TV Shows

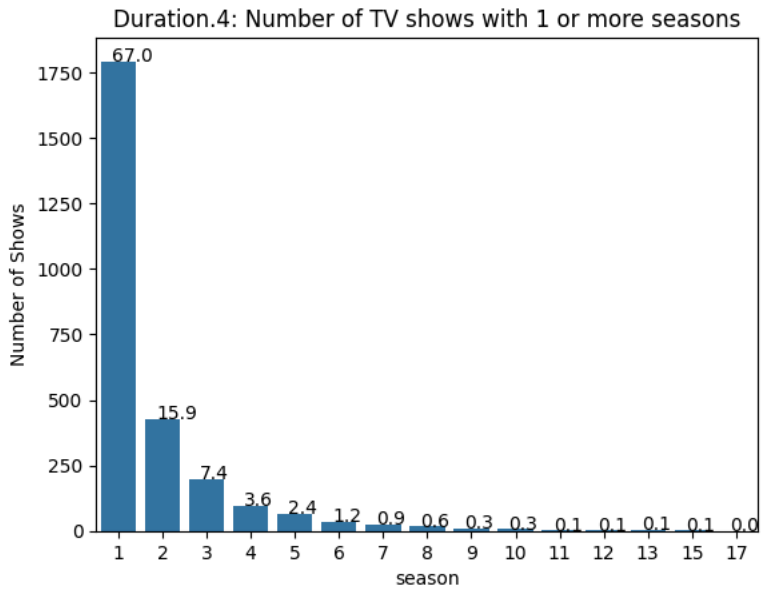
```
#Lets find out how many total tv shows have been there and how many lasted more than a season
countMoreThan1Show = dfWithDuration.loc[(dfWithDuration['durationN']>480) & (dfWithDuration['type']=='TV Show')].type.count()
countTotalShows = dfWithDuration.loc[(dfWithDuration['type']=='TV Show')].type.count()
countTotalShows
ratio = countMoreThan1Show/countTotalShows
ratio
```

```
0.3299701046337818
```

- 32 percent of TV Shows went on to create more than one season.
- A show will only go for a new season if it was successful in the first one.

```
#Lets plot the duration of all TV Shows in ascending manner to uncover its trend.
#

tvshowDuration_df  = dfWithDuration.loc[df['type']=='TV Show'].groupby('durationN', as_index=False).type.count()
#Create a new column called Season, that finds the number of seasons.  As we assumed each season to be 480min,
# we now divide duration to get back the number of seasons
tvshowDuration_df['season']= (tvshowDuration_df.durationN/480).astype(int)
tvshowDuration_df.rename(columns={'type':'Number of Shows'}, inplace=True)


```

```

plot = sns.barplot(data= tvshowDuration_df,x='season', y='Number of Shows')
ChartNumber="Duration.4"
plt.title(f'{ChartNumber}: Number of TV shows with 1 or more seasons')
total =  tvshowDuration_df['Number of Shows'].sum()

for p in plot.patches:
 plot.annotate('{:}'.format(round(p.get_height()*100/total,1)), (p.get_x()+0.25, p.get_height()+0.01))
plt.show()
```



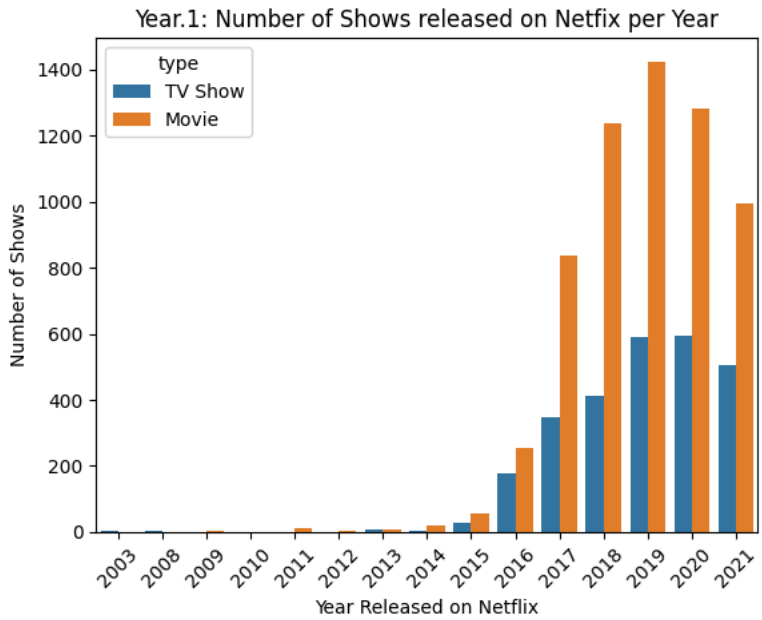
- The plot reveals that 33 percent of shows go on to make a new season after season 1.
- It can be construed that 33 percent of shows are successful

Recommendation: Netflix should encourage producers of TV Shows that have done more than one season to do more seasons

```
dfWithDuration['date_added_year'] = dfWithDuration.date_added.dt.year
dfWithDuration

```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	du
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	other	United States	2021-09-25	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...	
1	s2	TV Show	Blood & Water	other	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...	
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	ALL	2021-09-24	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...	

```
#Lets see if we have any insights to uncover by looking at the date_added for each type of show
dateAdded_df = dfWithDuration.groupby(['date_added_year','type'], as_index=False).show_id.count()
sns.barplot(data=dateAdded_df, x='date_added_year', y= 'show_id', hue='type')
plt.xticks(rotation=45)
plt.xlabel('Year Released on Netflix')
plt.ylabel('Number of Shows')
ChartNumber="Year.1"
plt.title(f'{ChartNumber}: Number of Shows released on Netfix per Year')
plt.show()
```




```
#Lets see if we have any insights to uncover by looking at the date_added for each type of show
dateAdded_df = dfWithDuration.groupby(['date_added','type'], as_index=False).show_id.count()
dateAdded_df

```

	date_added	type	show_id
0	2003-12-01	TV Show	2
1	2008-01-01	Movie	1
2	2008-02-04	TV Show	1
3	2008-12-01	TV Show	1
4	2009-05-05	Movie	1
...	...	...	...
2545	2021-09-23	Movie	1
2546	2021-09-23	TV Show	1
2547	2021-09-24	Movie	3
2548	2021-09-24	TV Show	7
2549	2021-09-25	Movie	1

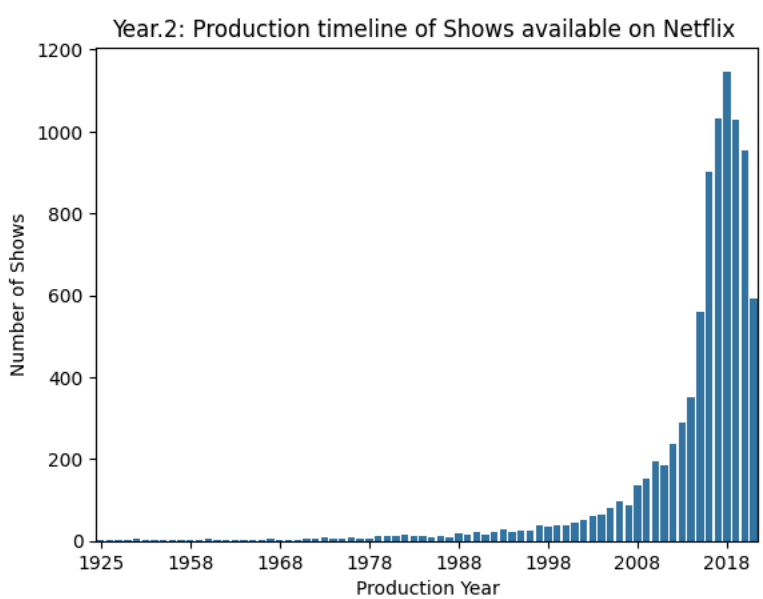
2550 rows × 3 columns

```
released_df = dfWithDuration.groupby('release_year').size().reset_index(name='show_count')
released_df
```

	release_year	show_count
0	1925	1
1	1942	2
2	1943	3
3	1944	3
4	1945	4
...	...	...
69	2017	1032
70	2018	1147
71	2019	1030
72	2020	953
73	2021	592

74 rows × 2 columns

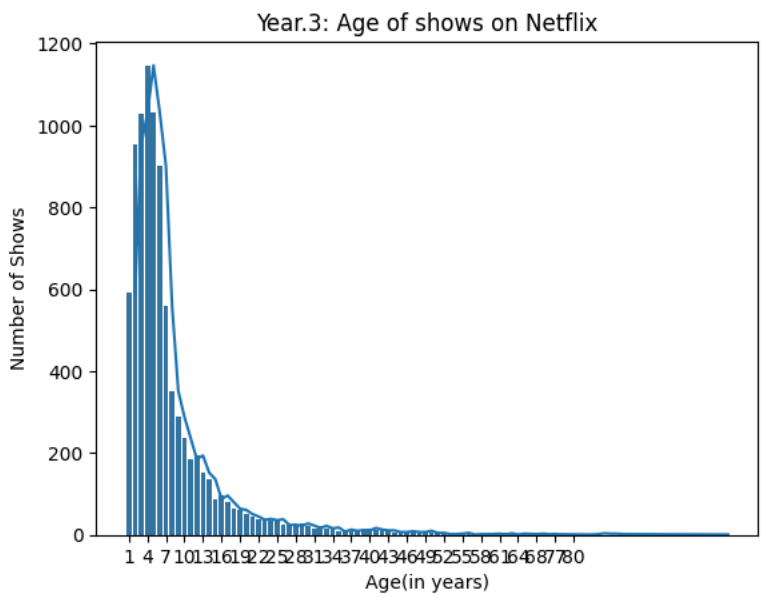
```
sns.barplot(data=released_df, x='release_year', y='show_count')

plt.xlabel('Production Year')
plt.ylabel('Number of Shows')
ChartNumber="Year.2"
plt.title(f'{ChartNumber}: Production timeline of Shows available on Netflix ')


# Calculate the desired xticks
tick_positions = range(0, len(released_df), 10)
tick_labels = [released_df['release_year'][i] for i in tick_positions]

# Set xticks to show only every 10th tick
plt.xticks(ticks=tick_positions, labels=tick_labels)
plt.show()
```



- Maximum number of shows were added to Netflix in 2019
- Netflix has a movie as old as 1925.

```
#We want to find ages
# 1. how many shows are 1 yr or less
# 2. how many shows are 2 yr to 1 yr
# 3. how many shows are 5 yr to 2 yr
# 4. how many shows are 5+ yrs
released_df['age'] = released_df.release_year.max()-released_df.release_year+1

```

```
sns.barplot(released_df,x='age',y='show_count')
sns.lineplot(data=released_df, x='age', y='show_count')
# Calculate the desired xticks
tick_positions = range(0,len(released_df),3)
tick_labels = [released_df['age'][len(released_df)-i-1] for i in tick_positions]

# Set xticks to show only every 10th tick
plt.xticks(ticks=tick_positions, labels=tick_labels)
ChartNumber="Year.3"
plt.title(f'{ChartNumber}: Age of shows on Netflix')
plt.xlabel('Age(in years)')
plt.ylabel('Number of Shows')
plt.show()
```



- Most shows on Netflix are not more than 10 yrs old
- Max number of shows are 4 yrs old.

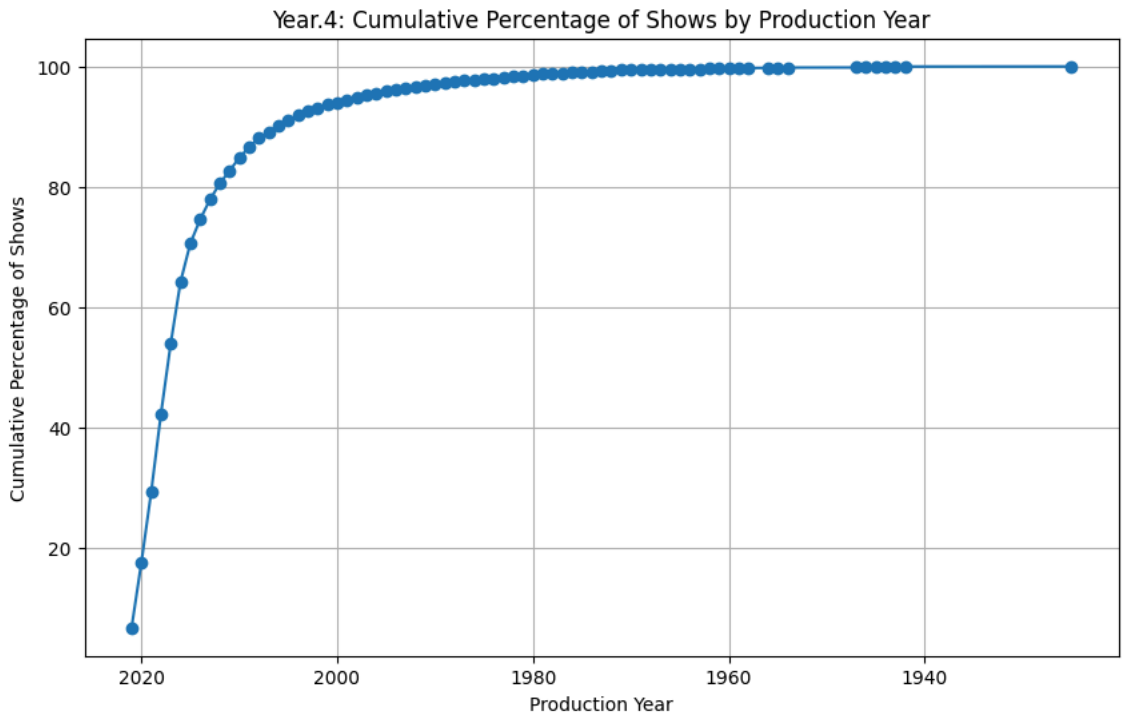
```
released_df
```

	release_year	show_count	age
0	1925	1	97
1	1942	2	80
2	1943	3	79
3	1944	3	78
4	1945	4	77
...	...	...	...
69	2017	1032	5
70	2018	1147	4
71	2019	1030	3
72	2020	953	2
73	2021	592	1

74 rows × 3 columns

```
# Sort the data by production year in descending order
released_df_sorted = released_df.sort_values(by='release_year', ascending=False)

# Calculate the cumulative sum of the number of shows
released_df_sorted['cumulative_shows'] = released_df_sorted['show_count'].cumsum()

# Calculate the cumulative percentage of the number of shows
released_df_sorted['cumulative_percentage'] = 100 * released_df_sorted['cumulative_shows'] / released_df_sorted['show_count'].sum()
```

```
# Plot the data
plt.figure(figsize=(10, 6))
plt.plot(released_df_sorted['release_year'], released_df_sorted['cumulative_percentage'], marker='o')
plt.xlabel('Production Year')
plt.ylabel('Cumulative Percentage of Shows')
ChartNumber="Year.4"
plt.title(f'{ChartNumber}: Cumulative Percentage of Shows by Production Year')
plt.gca().invert_xaxis()  # Invert the x-axis to show the latest year first
plt.grid(True)
plt.show()
```




```
pd.set_option('display.max_rows', None)
print(released_df_sorted)
pd.reset_option('display.max_rows')
```

```
    release_year  show_count  age  cumulative_shows  cumulative_percentage
73          2021         592    1               592               6.721926
72          2020         953    2              1545              17.542864
71          2019        1030    3              2575              29.238106
70          2018        1147    4              3722              42.261837
69          2017        1032    5              4754              53.979789
68          2016         902    6              5656              64.221642
67          2015         560    7              6216              70.580220
66          2014         352    8              6568              74.577041
65          2013         288    9              6856              77.847167
64          2012         237   10              7093              80.538208
63          2011         185   11              7278              82.638810
62          2010         194   12              7472              84.841603
61          2009         152   13              7624              86.567503
60          2008         136   14              7760              88.111729
59          2007          88   15              7848              89.110934
58          2006          96   16              7944              90.200976
57          2005          80   17              8024              91.109345
56          2004          64   18              8088              91.836040
55          2003          61   19              8149              92.528670
54          2002          51   20              8200              93.107755
53          2001          45   21              8245              93.618712
52          2000          37   22              8282              94.038833
51          1999          39   23              8321              94.481662
50          1998          36   24              8357              94.890428
49          1997          38   25              8395              95.321903
48          1996          24   26              8419              95.594414
47          1995          25   27              8444              95.878279
```

```
46          1994          22   28              8466              96.128080
45          1993          28   29              8494              96.446009
44          1992          23   30              8517              96.707165
43          1991          17   31              8534              96.900193
42          1990          22   32              8556              97.149994
41          1989          16   33              8572              97.331668
40          1988          18   34              8590              97.536051
39          1987           8   35              8598              97.626888
38          1986          13   36              8611              97.774498
37          1985          10   37              8621              97.888044
36          1984          12   38              8633              98.024299
35          1983          11   39              8644              98.149200
34          1982          17   40              8661              98.342228
33          1981          13   41              8674              98.489838
32          1980          11   42              8685              98.614738
31          1979          11   43              8696              98.739639
30          1978           7   44              8703              98.819121
29          1977           7   45              8710              98.898603
28          1976           9   46              8719              99.000795
27          1975           7   47              8726              99.080277
26          1974           7   48              8733              99.159759
25          1973          10   49              8743              99.273305
24          1972           5   50              8748              99.330078
23          1971           5   51              8753              99.386851
22          1970           2   52              8755              99.409561
21          1969           2   53              8757              99.432270
20          1968           3   54              8760              99.466334
19          1967           5   55              8765              99.523107
18          1966           1   56              8766              99.534461
```

```
# Find the year before which 80% of the shows are present
year_80 = released_df_sorted[released_df_sorted['cumulative_percentage'] <= 80]['release_year'].max()
year_80

```

```
2021
```

- 50% of shows are 5yr or less old
- 80% are 9 yr old or less
- Interpretation: It confirms that most shows are of recent production and customers prefer not to see very old movies or shows. Maybe only hit or classic shows will make it to Netflix which are more than 10 yrs old.

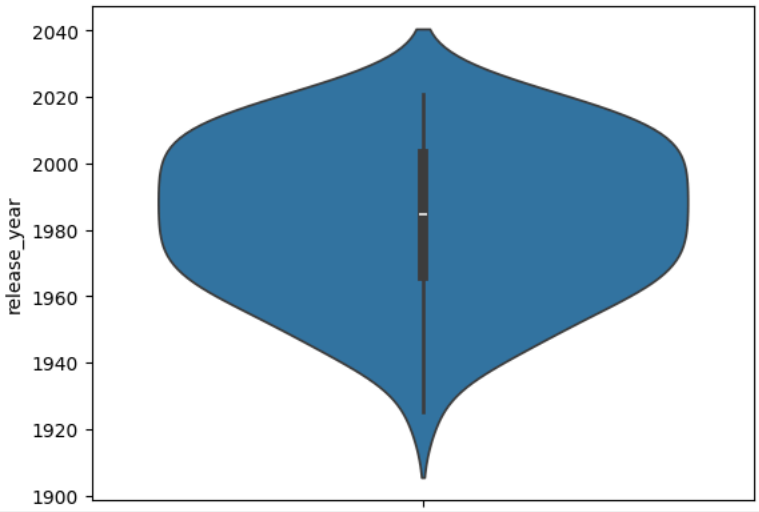
```
released_df
```

	release_year	show_count	age
0	1925	1	97
1	1942	2	80
2	1943	3	79
3	1944	3	78
4	1945	4	77
...	...	...	...
69	2017	1032	5
70	2018	1147	4
71	2019	1030	3
72	2020	953	2
73	2021	592	1

74 rows × 3 columns

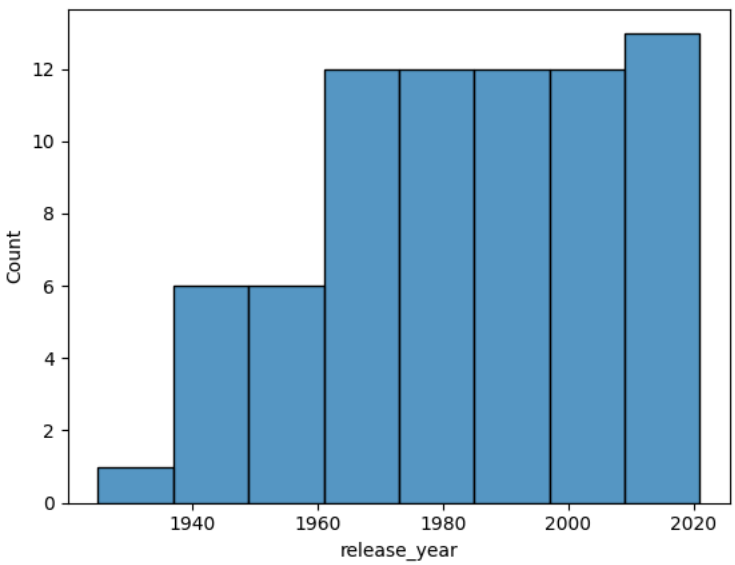
```
sns.violinplot(data=released_df, y='release_year')
```

```
<Axes: ylabel='release_year'>
```




```
sns.histplot(data=released_df, x='release_year')
```

```
<Axes: xlabel='release_year', ylabel='Count'>
```

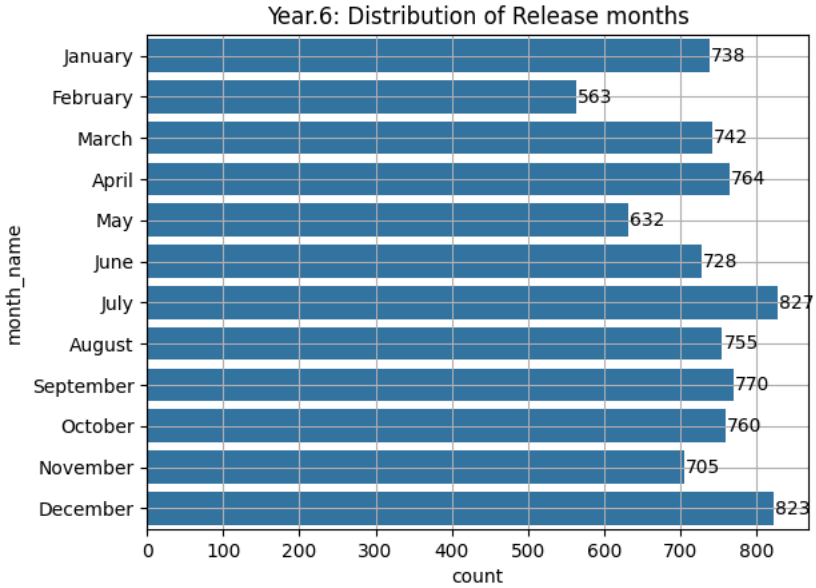



```
means = released_df.release_year.mode()
means
```

```
0     1925
1     1942
2     1943
3     1944
4     1945
      ...
69    2017
70    2018
71    2019
72    2020
73    2021
Name: release_year, Length: 74, dtype: int64
```

```
month_df = df[['date_added']].copy()
month_df['month_name'] = month_df.date_added.dt.month_name()
month_df['month_num'] = month_df.date_added.dt.month
plot = sns.countplot(data=month_df.sort_values(by='month_num'), y='month_name')
plt.grid(True)
ChartNumber="Year.6"
plt.title(f'{ChartNumber}: Distribution of Release months')
for patch in plot.patches:
    plt.text(patch.get_width() + 25, patch.get_y() + patch.get_height() / 2,
```

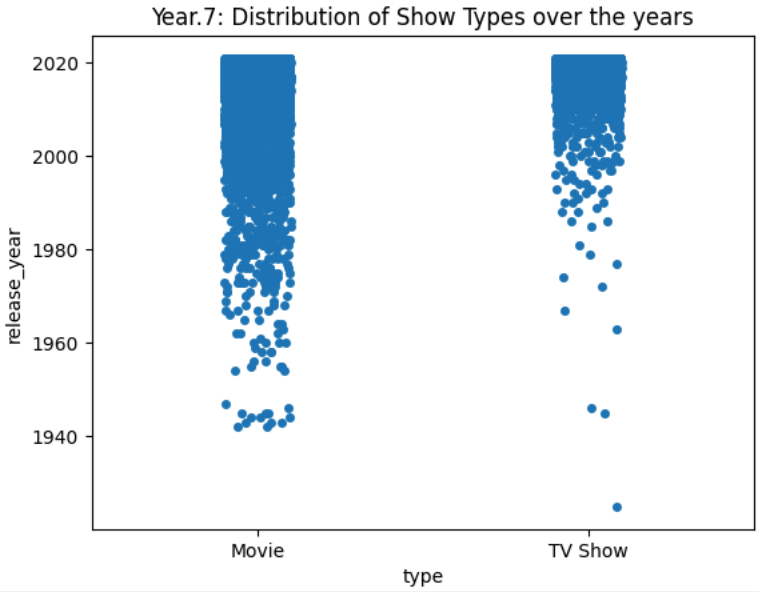
```
            f'{patch.get_width():.0f}',
            ha = 'center', va = 'center')
```



- Max number of shows are released in July and December. July is the start of the season maybe due to US independence day.
- Feb has the least number of releases

```
sns.stripplot(data=df, x='type', y='release_year')
ChartNumber="Year.7"
plt.title(f'{ChartNumber}: Distribution of Show Types over the years')
```

```
Text(0.5, 1.0, 'Year.7: Distribution of Show Types over the years')
```



- Very few TV shows of timeline older than 1980 are present when compared to movies.

```
Start coding or generate with AI.
```

```
listedin_df = df_cleaned[["show_id","type","listed_in","date_added"]]
listedin_df
```

	show_id	type	listed_in	date_added
0	s1	Movie	Documentaries	2021-09-25
1	s2	TV Show	International TV Shows, TV Dramas, TV Mysteries	2021-09-24
2	s3	TV Show	Crime TV Shows, International TV Shows, TV Act...	2021-09-24
3	s4	TV Show	Docuseries, Reality TV	2021-09-24
4	s5	TV Show	International TV Shows, Romantic TV Shows, TV ...	2021-09-24
...	...	...	...	...
8802	s8803	Movie	Cult Movies, Dramas, Thrillers	2019-11-20
8803	s8804	TV Show	Kids' TV, Korean TV Shows, TV Comedies	2019-07-01
8804	s8805	Movie	Comedies, Horror Movies	2019-11-01
8805	s8806	Movie	Children & Family Movies, Comedies	2020-01-11
8806	s8807	Movie	Dramas, International Movies, Music & Musicals	2019-03-02

8807 rows × 4 columns

```
#Reusable function
def splitAndExplode(xdf, columnName):
    # Split the CSV values into lists
    xdf[columnName] = xdf[columnName].str.split(',')
    # Use explode to create separate rows for each value
    exploded_df = xdf.explode(columnName)
    # Trim spaces after exploding
    exploded_df[columnName] = exploded_df[columnName].str.strip()
    return exploded_df
listedin_df = splitAndExplode(listedin_df.copy(deep=True), 'listed_in')
```

```
listedin_df
```

	show_id	type	listed_in	date_added
0	s1	Movie	Documentaries	2021-09-25
1	s2	TV Show	International TV Shows	2021-09-24
1	s2	TV Show	TV Dramas	2021-09-24
1	s2	TV Show	TV Mysteries	2021-09-24
2	s3	TV Show	Crime TV Shows	2021-09-24
...	...	...	...	...
8805	s8806	Movie	Children & Family Movies	2020-01-11
8805	s8806	Movie	Comedies	2020-01-11
8806	s8807	Movie	Dramas	2019-03-02
8806	s8807	Movie	International Movies	2019-03-02
8806	s8807	Movie	Music & Musicals	2019-03-02

19323 rows × 4 columns

```
listedin_df
```

	show_id	type	listed_in	date_added
0	s1	Movie	Documentaries	2021-09-25
1	s2	TV Show	International TV Shows	2021-09-24
1	s2	TV Show	TV Dramas	2021-09-24
1	s2	TV Show	TV Mysteries	2021-09-24
2	s3	TV Show	Crime TV Shows	2021-09-24
...	...	...	...	...
8805	s8806	Movie	Children & Family Movies	2020-01-11
8805	s8806	Movie	Comedies	2020-01-11
8806	s8807	Movie	Dramas	2019-03-02
8806	s8807	Movie	International Movies	2019-03-02
8806	s8807	Movie	Music & Musicals	2019-03-02

19323 rows × 4 columns

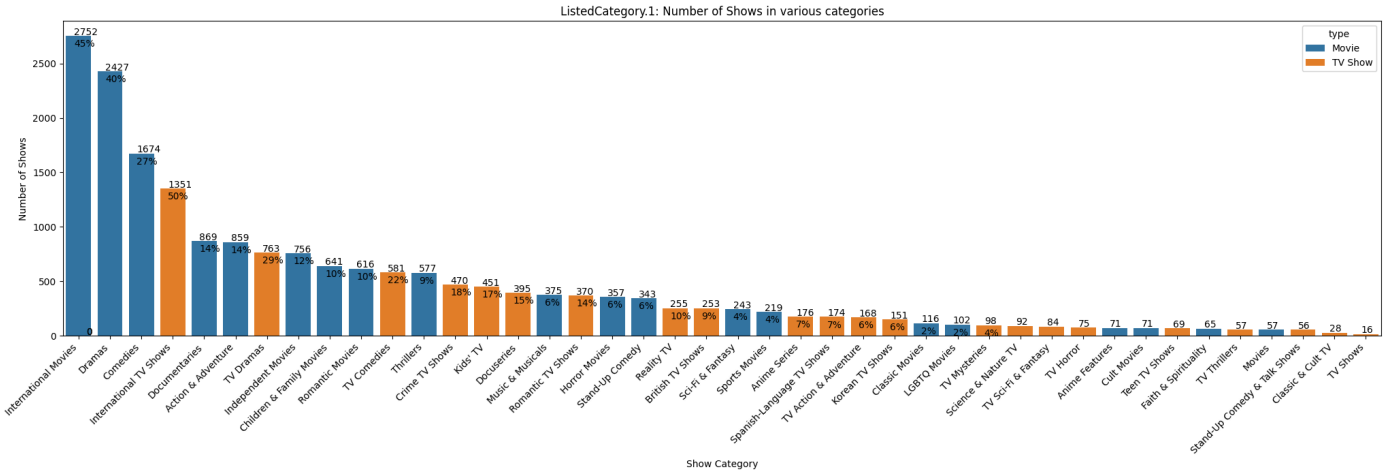
```
#Sort
listedin_sorted_df = listedin_df.groupby(['listed_in','type'], as_index=False).size().sort_values(by='size',ascending=False)
listedin_sorted_df
# category_counts = ['listed_in'].value_counts().sort_values(ascending=False).reset_index()
# category_counts
```

	listed_in	type	size
16	International Movies	Movie	2752
12	Dramas	Movie	2427
7	Comedies	Movie	1674
17	International TV Shows	TV Show	1351
10	Documentaries	Movie	869
0	Action & Adventure	Movie	859
34	TV Dramas	TV Show	763
15	Independent Movies	Movie	756
4	Children & Family Movies	Movie	641
24	Romantic Movies	Movie	616
33	TV Comedies	TV Show	581
41	Thrillers	Movie	577
8	Crime TV Shows	TV Show	470
18	Kids' TV	TV Show	451
11	Docuseries	TV Show	395
22	Music & Musicals	Movie	375
25	Romantic TV Shows	TV Show	370
14	Horror Movies	Movie	357
30	Stand-Up Comedy	Movie	343
23	Reality TV	TV Show	255
3	British TV Shows	TV Show	253
26	Sci-Fi & Fantasy	Movie	243
29	Sports Movies	Movie	219
2	Anime Series	TV Show	176
28	Spanish-Language TV Shows	TV Show	174
32	TV Action & Adventure	TV Show	168
19	Korean TV Shows	TV Show	151
6	Classic Movies	Movie	116
20	LGBTQ Movies	Movie	102
36	TV Mysteries	TV Show	98
27	Science & Nature TV	TV Show	92
37	TV Sci-Fi & Fantasy	TV Show	84
35	TV Horror	TV Show	75
1	Anime Features	Movie	71
9	Cult Movies	Movie	71
40	Teen TV Shows	TV Show	69
13	Faith & Spirituality	Movie	65
39	TV Thrillers	TV Show	57
21	Movies	Movie	57
31	Stand-Up Comedy & Talk Shows	TV Show	56
5	Classic & Cult TV	TV Show	28
38	TV Shows	TV Show	16

```
plt.figure(figsize=(25,6))
plot = sns.barplot(listedin_sorted_df,x='listed_in', y='size', hue='type')
plt.xticks(rotation=45, ha='right')
plt.xlabel('Show Category')
```

```
plt.ylabel('Number of Shows')
ChartNumber="ListedCategory.1"
plt.title(f'{ChartNumber}: Number of Shows in various categories')

nmovies = df.query('type=="Movie"').shape[0]
nshows=df.query('type=="TV Show"').shape[0]
print(f'Total movies {nmovies}')
print(f'Total TV Shows {nshows}')

for p in plot.patches:
    rvalue = p.get_facecolor()[0]
    #This will put the actual number
    plot.annotate('{:}'.format(round(p.get_height())), (p.get_x()+0.25, p.get_height()+10))
    #For percentage, we have to put a logic as total movies is different from total TV Shows.
    #TV shows and Movies will come in different color.  Based on color, we can decide how to find the percentage
    if(rvalue == 0.19460784313725488):
        plot.annotate('{:}%'.format(round(p.get_height()/nmovies*100)), (p.get_x()+0.25, p.get_height()-100))
    else:
        plot.annotate('{:}%'.format(round(p.get_height()/nshows*100)), (p.get_x()+0.25, p.get_height()-100))
plt.show()
```

```
Total movies 6131
Total TV Shows 2676
```



- Out of 6131 movies, 45% are international movies
- Out of 2676 TV Shows, 50% are international TV Shows
- Drama is the most popular category for both TV Shows and Movies, every 2 out of 5 movies is a drama, every 1 out of 3 TV Shows is a drama
- Every 4th movie is a comedy
- Every 5th tv show is a comedy

Recommendation 5: International movies, Drama and Comedies are the genres that should be encouraged more.

```

listedin_df['year'] = listedin_df.date_added.dt.year
```

```
listedin_df
```

	show_id	type	listed_in	date_added	year
0	s1	Movie	Documentaries	2021-09-25	2021
1	s2	TV Show	International TV Shows	2021-09-24	2021
1	s2	TV Show	TV Dramas	2021-09-24	2021
1	s2	TV Show	TV Mysteries	2021-09-24	2021
2	s3	TV Show	Crime TV Shows	2021-09-24	2021
...	...	...	...	...	...
8805	s8806	Movie	Children & Family Movies	2020-01-11	2020
8805	s8806	Movie	Comedies	2020-01-11	2020
8806	s8807	Movie	Dramas	2019-03-02	2019
8806	s8807	Movie	International Movies	2019-03-02	2019
8806	s8807	Movie	Music & Musicals	2019-03-02	2019

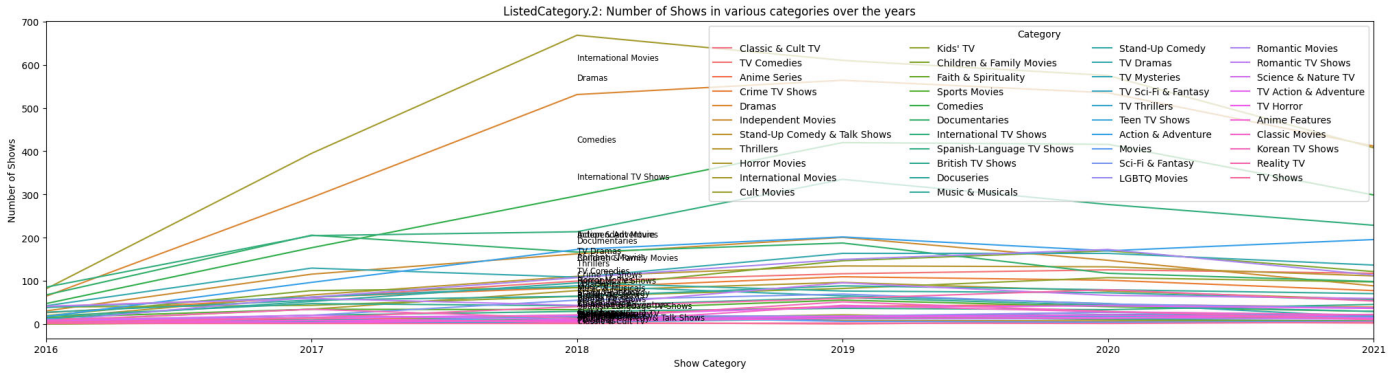
19323 rows × 5 columns

```
#sns.barplot(data = listedin_df.groupby(), x='year',
#sns.countplot(data=listedin_df, x='year', hue='listed_in')
listedin_groupByYear_df = listedin_df.groupby(['year','listed_in'], as_index=False).size()
listedin_groupByYear_df
```

	year	listed_in	size
0	2003	Classic & Cult TV	2
1	2003	TV Comedies	2
2	2008	Anime Series	1
3	2008	Crime TV Shows	1
4	2008	Dramas	1
...	...	...	...
336	2021	TV Sci-Fi & Fantasy	20
337	2021	TV Shows	5
338	2021	TV Thrillers	13
339	2021	Teen TV Shows	18
340	2021	Thrillers	112

341 rows × 3 columns

```

plt.figure(figsize=(25,6))

plt.xlabel('Show Category')
plt.ylabel('Number of Shows')
ChartNumber="ListedCategory.2"
plt.title(f'{ChartNumber}: Number of Shows in various categories over the years')
sns.lineplot(data=listedin_groupByYear_df,x='year',y='size', hue='listed_in')
#Legend with 4 columns
plt.legend(title='Category', ncol=4)
plt.xlim(2016,2021)

# Add labels to the lines
for category in listedin_groupByYear_df['listed_in'].unique():
    subset = listedin_groupByYear_df[listedin_groupByYear_df['listed_in'] == category]
    plt.text(subset['year'].iloc[-4], subset['size'].iloc[-3], category,
             horizontalalignment='left', size='small', color='black', weight='normal')
```



- International movies, Drama and Comedies are the three categories that saw maximum percentage rise in recent years.

```
ratings_df = df_cleaned.groupby('rating', as_index=False).size()
ratings_df
```

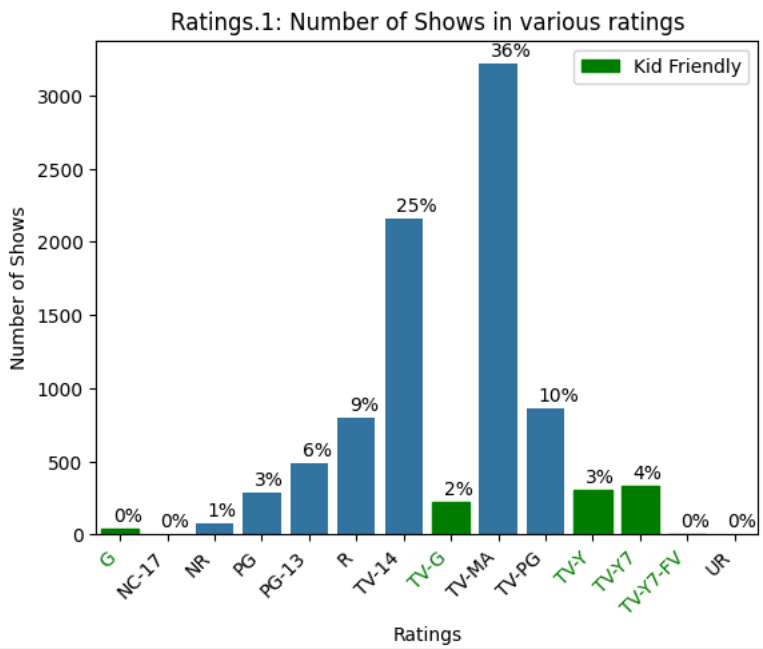
	rating	size
0	G	41
1	NC-17	3
2	NR	80
3	PG	287
4	PG-13	490
5	R	799
6	TV-14	2160
7	TV-G	220
8	TV-MA	3214
9	TV-PG	863
10	TV-Y	307
11	TV-Y7	334
12	TV-Y7-FV	6
13	UR	3

```
# List of kid-friendly ratings
kid_friendly_ratings = ['G', 'TV-G', 'TV-Y', 'TV-Y7', 'TV-Y7-FV']
```

```
plot = sns.barplot(data=ratings_df, x='rating', y='size')

plt.xlabel('Ratings')
plt.xticks(rotation=45, ha='right')
plt.ylabel('Number of Shows')
ChartNumber="Ratings.1"
plt.title(f'{ChartNumber}: Number of Shows in various ratings')

total_shows = nmovies+nshows
# Get the x-tick labels
```

```
xtick_labels = [tick.get_text() for tick in plot.get_xticklabels()]
i=0
for p in plot.patches:
    plot.annotate('{:}%'.format(round(p.get_height()/total_shows*100)), (p.get_x()+0.25, p.get_height()+50))
    if(xtick_labels[i] in kid_friendly_ratings):
        p.set_color("green")
    i=i+1

# Change the color of a kid friendly ticks
for tick in plot.get_xticklabels():
    if tick.get_text() in kid_friendly_ratings:
        tick.set_color('green')  # Change to desired color

# Add a small green color rectangle as a legend

green_patch = mpatches.Patch(color='green', label='Kid Friendly')
plt.legend(handles=[green_patch], loc='upper right')

```

```
<matplotlib.legend.Legend at 0x20820a213d0>
```




```


# List of kid-friendly ratings
kid_friendly_ratings = ['G', 'TV-G', 'TV-Y', 'TV-Y7', 'TV-Y7-FV']
# Filter DataFrame for kid-friendly ratings and sum the sizes
total_kid_shows = ratings_df[ratings_df['rating'].isin(kid_friendly_ratings)]['size'].sum()

print("Total number of shows appropriate for kids:", total_kid_shows)
print(f'Percentage of total shows that are kids friendly {round(total_kid_shows/len(df)*100)}')
```

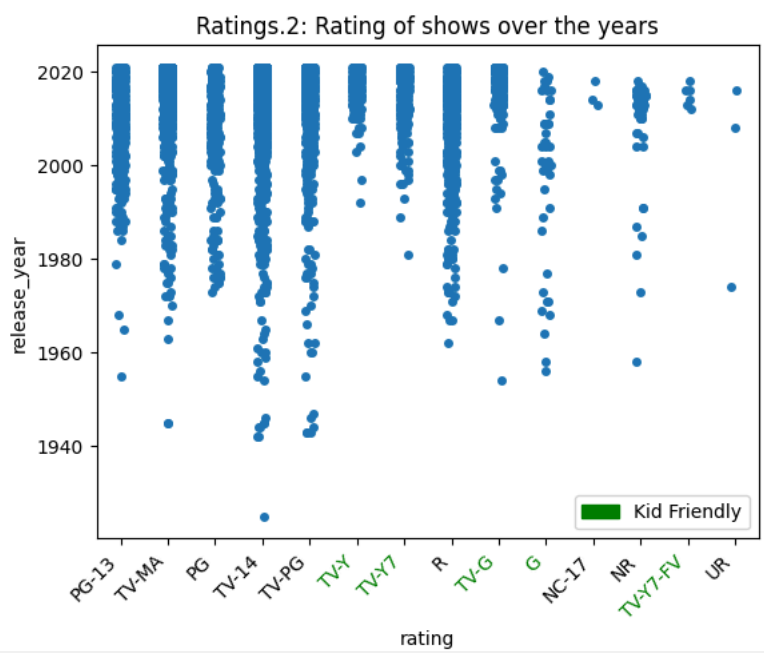
```
Total number of shows appropriate for kids: 908
Percentage of total shows that are kids friendly 10
```

- Only 10% of shows out of 8807 are kids-friendly
- 36% of shows on netflix are for mature audience, and overall 90% of shows are for adults only.

Recommendation 3: Most people using Netflix are looking for adult content. So quality content in these genres is always going to sell

```
ChartNumber="Ratings.2"
plt.title(f'{ChartNumber}: Rating of shows over the years')

plot= sns.stripplot(data=df, x='rating', y='release_year')
plt.xticks(rotation=45, ha='right')
# Change the color of a kid friendly ticks
for tick in plot.get_xticklabels():
    if tick.get_text() in kid_friendly_ratings:
        tick.set_color('green')  # Change to desired color
# Add a small green color rectangle as a legend

```

```
green_patch = mpatches.Patch(color='green', label='Kid Friendly')
plt.legend(handles=[green_patch], loc='lower right')
plt.show()
```



- TV14 and TV-PG rated shows are the most consistent over the years. It means these type of shows are “evergreen”
- Kid movies have lesser “shelf life”. Maybe we can infer that nobody wants to see old kid movies. This can explain why Netflix has less number of kid friendly movies. They don’t have longevity.

```
country_df = df[['type','title','country','date_added', 'listed_in','duration']]
country_df = splitAndExplode(country_df.copy(deep=True), 'country')
country_df.country.unique()
```

```
array(['United States', 'South Africa', 'ALL', 'India', 'Ghana',
       'Burkina Faso', 'United Kingdom', 'Germany', 'Ethiopia',
       'Czech Republic', 'Mexico', 'Turkey', 'Australia', 'France',
       'Finland', 'China', 'Canada', 'Japan', 'Nigeria', 'Spain',
       'Belgium', 'South Korea', 'Singapore', 'Italy', 'Romania',
       'Argentina', 'Venezuela', 'Hong Kong', 'Russia', '', 'Ireland',
       'Nepal', 'New Zealand', 'Brazil', 'Greece', 'Jordan', 'Colombia',
       'Switzerland', 'Israel', 'Taiwan', 'Bulgaria', 'Algeria', 'Poland',
       'Saudi Arabia', 'Thailand', 'Indonesia', 'Egypt', 'Denmark',
       'Kuwait', 'Netherlands', 'Malaysia', 'Vietnam', 'Hungary',
       'Sweden', 'Lebanon', 'Syria', 'Philippines', 'Iceland',
       'United Arab Emirates', 'Norway', 'Qatar', 'Mauritius', 'Austria',
       'Cameroon', 'Palestine', 'Uruguay', 'Kenya', 'Chile', 'Luxembourg',
       'Cambodia', 'Bangladesh', 'Portugal', 'Cayman Islands', 'Senegal',
       'Serbia', 'Malta', 'Namibia', 'Angola', 'Peru', 'Mozambique',
       'Belarus', 'Zimbabwe', 'Puerto Rico', 'Pakistan', 'Cyprus',
       'Guatemala', 'Iraq', 'Malawi', 'Paraguay', 'Croatia', 'Iran',
       'West Germany', 'Albania', 'Georgia', 'Soviet Union', 'Morocco',
       'Slovakia', 'Ukraine', 'Bermuda', 'Ecuador', 'Armenia', 'Mongolia',
       'Bahamas', 'Sri Lanka', 'Latvia', 'Liechtenstein', 'Cuba',
       'Nicaragua', 'Slovenia', 'Dominican Republic', 'Samoa',
       'Azerbaijan', 'Botswana', 'Vatican City', 'Jamaica', 'Kazakhstan',
       'Lithuania', 'Afghanistan', 'Somalia', 'Sudan', 'Panama', 'Uganda',
       'East Germany', 'Montenegro'], dtype=object)
```

```
country_df.shape[0]
```

```
10850
```

```
country_df
```

	type	title	country	date_added	listed_in	duration
0	Movie	Dick Johnson Is Dead	United States	2021-09-25	Documentaries	90 min
1	TV Show	Blood & Water	South Africa	2021-09-24	International TV Shows, TV Dramas, TV Mysteries	2 Seasons
2	TV Show	Ganglands	ALL	2021-09-24	Crime TV Shows, International TV Shows, TV Act...	1 Season
3	TV Show	Jailbirds New Orleans	ALL	2021-09-24	Docuseries, Reality TV	1 Season
4	TV Show	Kota Factory	India	2021-09-24	International TV Shows, Romantic TV Shows, TV ...	2 Seasons
...	...	...	...	...	...	...
8802	Movie	Zodiac	United States	2019-11-20	Cult Movies, Dramas, Thrillers	158 min
8803	TV Show	Zombie Dumb	ALL	2019-07-01	Kids' TV, Korean TV Shows, TV Comedies	2 Seasons
8804	Movie	Zombieland	United States	2019-11-01	Comedies, Horror Movies	88 min
8805	Movie	Zoom	United States	2020-01-11	Children & Family Movies, Comedies	88 min
8806	Movie	Zubaan	India	2019-03-02	Dramas, International Movies, Music & Musicals	111 min

10850 rows × 6 columns

```
pd.set_option('display.max_rows', None)

#lets look at shows that have International in their listed_in column.  I verified manually that these shows are specific to a country
#For ex, for rows in which country has India, and listed_in has International, all are Hindi or other Indian language based shows.
#Checked similar entries for South Korea to confirm that this interpretation of listed_in having International means its a local language
international_df = pd.DataFrame(columns=['country','tvshow','movies'])


for c in country_df.country.unique():
    tvshow = country_df[(country_df.country==c) & (country_df.listed_in.str.contains('International') & (country_df.type=='TV Show'))].sh

    movies = country_df[(country_df.country==c) & (country_df.listed_in.str.contains('International') & (country_df.type=='Movie'))].shap

    temp_df = pd.DataFrame({'country':[c],'tvshow':[tvshow],'movies':[movies]})
    international_df = pd.concat([international_df, temp_df])

international_df = international_df.reset_index(drop=True)
print(international_df)
pd.reset_option('display.max_rows')
```

```
                  country tvshow movies
0           United States     74    166
1            South Africa      9     39
2                     ALL    223    209
3                   India     66    864
4                   Ghana      0      5
5            Burkina Faso      0      1
6          United Kingdom    128    170
7                 Germany     35     94
8                Ethiopia      0      1
9          Czech Republic      3     10
10                 Mexico     43     70
11                 Turkey     30     80
12              Australia     31     30
13                 France     43    207
14                Finland      2      6
15                  China     40     71
16                 Canada     25     60
17                  Japan    151     72
18                Nigeria      9     88
19                  Spain     54    140
20                Belgium     11     58
21            South Korea    152     44
22              Singapore     19     14
23                  Italy     13     52
24                Romania      0     11
25              Argentina     16     58
26              Venezuela      0      3
27              Hong Kong      5     82
28                 Russia     10      6
29                            1      5
30                Ireland      3     12
31                  Nepal      0      2
32            New Zealand      2      9
33                 Brazil     26     43
34                 Greece      1      1
35                 Jordan      2      7
```

```
36              Colombia     31     15
37           Switzerland      1     14
38                Israel      8     14
39                Taiwan     70     17
40              Bulgaria      0      4
41               Algeria      0      3
42                Poland      9     26
43          Saudi Arabia      3      9
44              Thailand     24     40
45             Indonesia      2     80
46                 Egypt     15     99
47               Denmark     10     23
48                Kuwait      3      5
49           Netherlands      5     37
50              Malaysia      4     17
51               Vietnam      0      7
52               Hungary      0      4
53                Sweden     10     19
54               Lebanon      7     22
55                 Syria      1      1
56           Philippines      2     79
```

```
international_df['total']=international_df.movies+international_df.tvshow
international_df
```

	country	tvshow	movies	total
0	United States	74	166	240
1	South Africa	9	39	48
2	ALL	223	209	432
3	India	66	864	930
4	Ghana	0	5	5
...	...	...	...	...
119	Sudan	0	0	0
120	Panama	0	0	0
121	Uganda	0	0	0
122	East Germany	0	0	0
123	Montenegro	0	1	1

124 rows × 4 columns

```
#For the purpose of this analysis of comparing shows in International languages, I want to exclude the shows where we have ALL.  There is

# Get the index of rows where 'country' column has the value 'ALL'
all_index = international_df[international_df['country'] == 'ALL'].index
print(all_index)
# Drop the rows using the index
international_df = international_df.drop(all_index)
international_df
```

```
Index([2], dtype='int64')
```

	country	tvshow	movies	total
0	United States	74	166	240
1	South Africa	9	39	48
3	India	66	864	930
4	Ghana	0	5	5
5	Burkina Faso	0	1	1
...	...	...	...	...
119	Sudan	0	0	0
120	Panama	0	0	0
121	Uganda	0	0	0
122	East Germany	0	0	0
123	Montenegro	0	1	1

123 rows × 4 columns

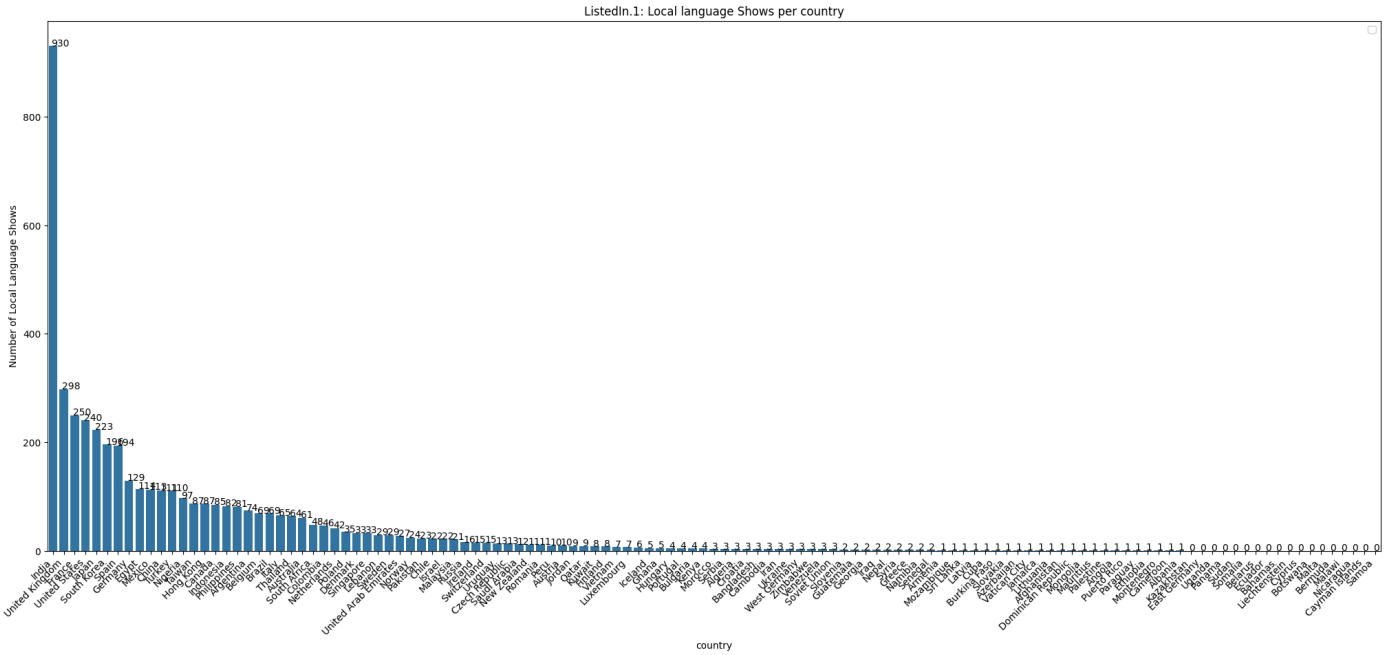
```
international_df.sort_values(by='total', ascending=False, inplace=True)
international_df
```

	country	tvshow	movies	total
3	India	66	864	930
6	United Kingdom	128	170	298
13	France	43	207	250
0	United States	74	166	240
17	Japan	151	72	223
...	...	...	...	...
98	Bermuda	0	0	0
87	Malawi	0	0	0
107	Nicaragua	0	0	0
72	Cayman Islands	0	0	0
110	Samoa	0	0	0

123 rows × 4 columns

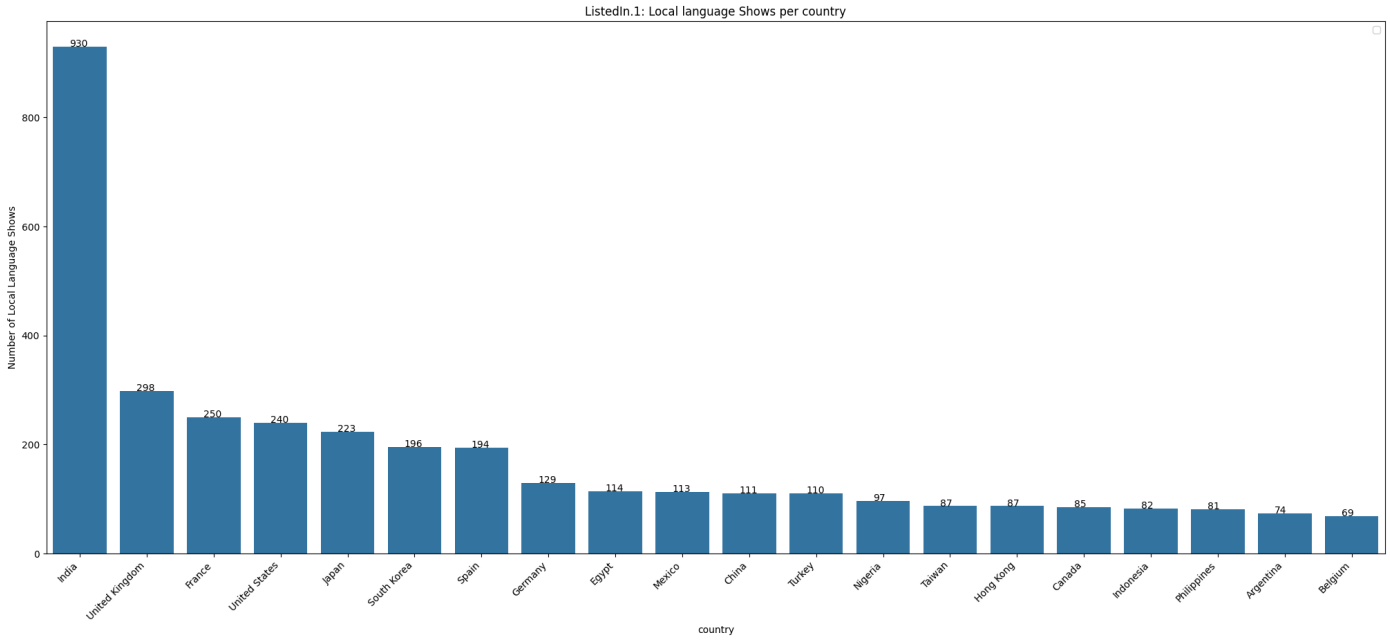
```
plt.figure(figsize=(25,10))
plot=sns.barplot(data=international_df, x='country', y='total')
plt.xticks(rotation=45,ha='right')
plt.ylabel('Number of Local Language Shows')
ChartNumber="ListedIn.1"
plt.title(f'{ChartNumber}: Local language Shows per country')
for p in plot.patches:
    plot.annotate('{:}'.format(round(p.get_height())), (p.get_x()+0.25, p.get_height()+0.25))
plt.legend()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_29808\1959597973.py:9: UserWarning: No artists with labels found to put in legend.  Note tha
  plt.legend()
```




```
plt.figure(figsize=(25,10))
plot=sns.barplot(data=international_df.iloc[:20], x='country', y='total')
plt.xticks(rotation=45,ha='right')
plt.ylabel('Number of Local Language Shows')
ChartNumber="ListedIn.1"
plt.title(f'{ChartNumber}: Local language Shows per country')
for p in plot.patches:
    plot.annotate('{:}'.format(round(p.get_height())), (p.get_x()+0.25, p.get_height()+0.25))
plt.legend()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_29808\2599052266.py:9: UserWarning: No artists with labels found to put in legend.  Note tha
  plt.legend()
```



- India has the maximum number of internation shows listed
- Japan and South Korea are Asian countries with more shows than China, despite having significantly less population than China

```
allCountries_df= pd.read_csv("allCountries_df.csv")
allCountries_df
```

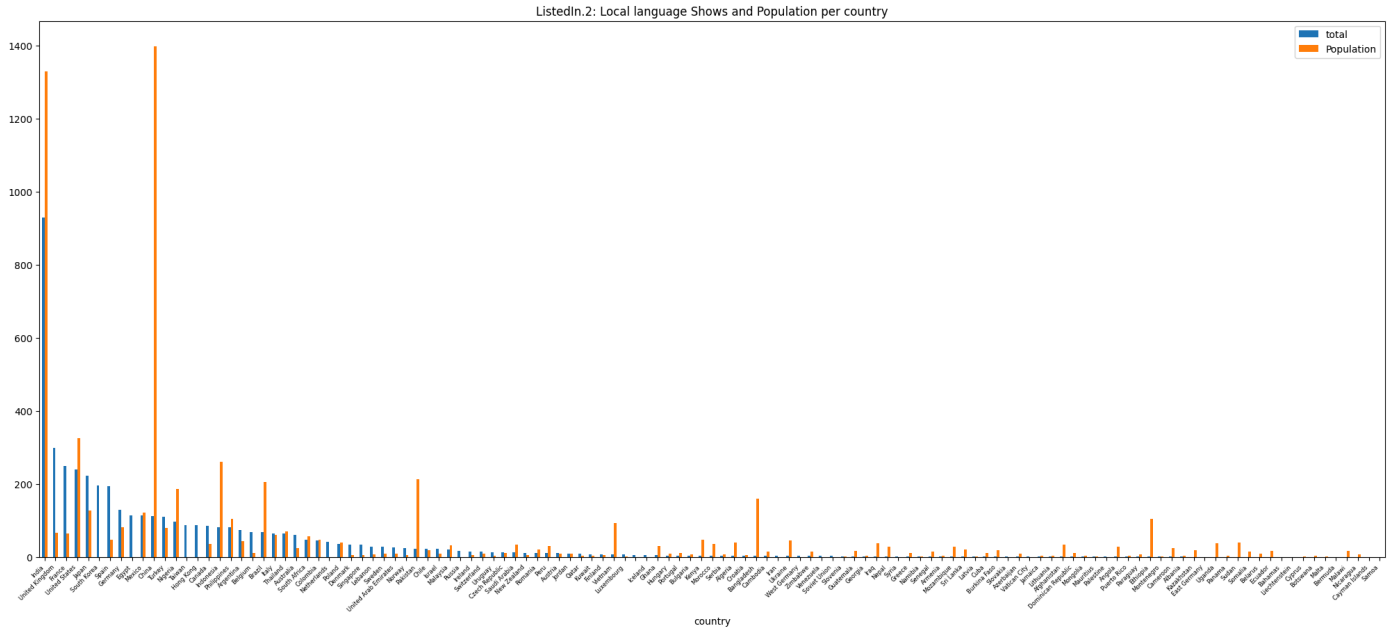
	Unnamed: 0	name	Population
0	0	Afghanistan	34.169723
1	1	Albania	2.881823
2	2	Algeria	39.917696
3	3	Andorra	0.000001
4	4	Angola	28.607677
...	...	...	...
197	197	Venezuela	0.000001
198	198	Vietnam	92.635828
199	199	Yemen	0.000001
200	200	Zambia	16.491819
201	201	Zimbabwe	14.295536

202 rows × 3 columns

```
international_df = international_df.merge(allCountries_df, how='left',left_on='country', right_on='name')
international_df.drop(columns=['Unnamed: 0','name'], inplace=True)

```

```
plt.figure(figsize=(25,10))
international_df.plot.bar(x='country',y=['total','Population'], figsize=(25,10))
ChartNumber="ListedIn.2"
plt.title(f'{ChartNumber}: Local language Shows and Population per country')
plt.xticks(rotation=45,ha='right',fontsize=6)
plt.show()
```

```
<Figure size 2500x1000 with 0 Axes>
```



- Largest number of local language movies are released in India
- While India has a fair ratio in terms of shows and population, some other countries dont. Lets findout

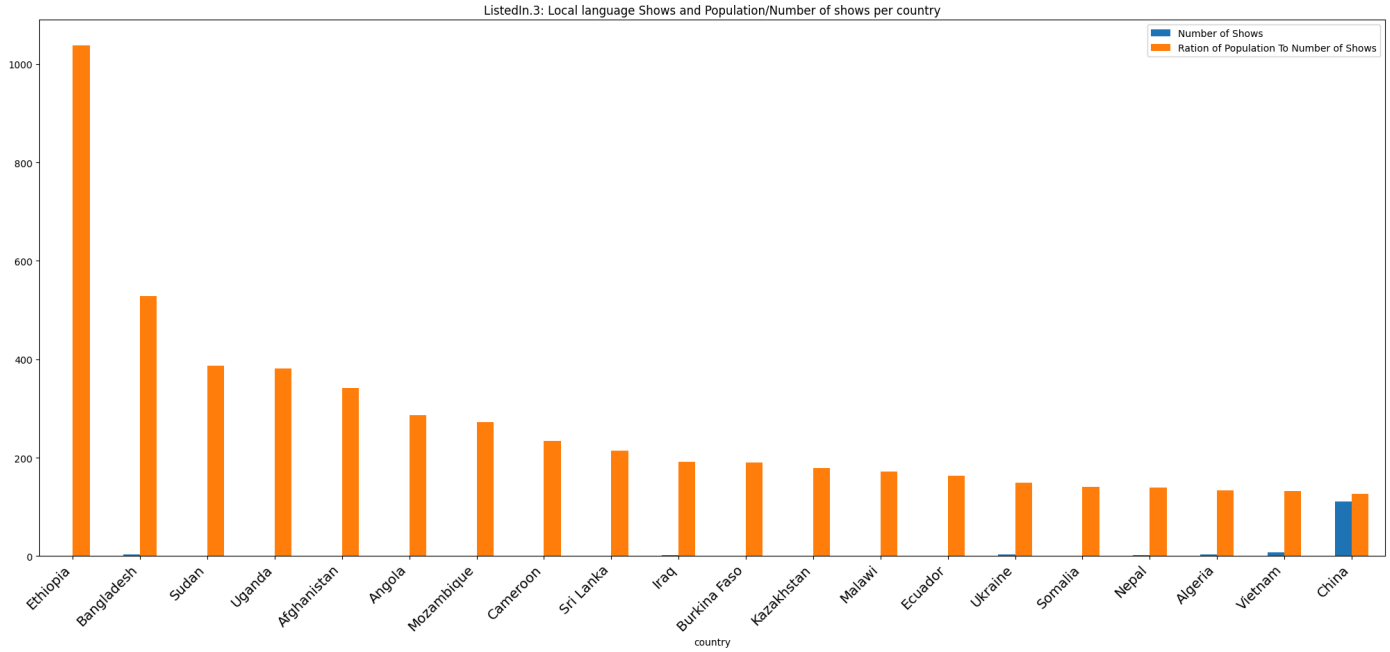
```
def calculateRatio(row):
    #print(row)
    if row.total!=0:
        return round(row.Population/row.total,2)
    else:
        return row.Population
international_df['pop_show_ratio'] = international_df.apply(calculateRatio, axis=1)*10
international_df
```

	country	tvshow	movies	total	Population	pop_show_ratio
0	India	66	864	930	1330.290669	14.30000
1	United Kingdom	128	170	298	65.428953	2.20000
2	France	43	207	250	63.897092	2.60000
3	United States	74	166	240	325.826238	13.60000
4	Japan	151	72	223	127.128592	5.70000
...	...	...	...	...	...	...
118	Bermuda	0	0	0	0.000001	0.00001
119	Malawi	0	0	0	17.158173	171.58173
120	Nicaragua	0	0	0	6.341277	63.41277
121	Cayman Islands	0	0	0	0.000001	0.00001
122	Samoa	0	0	0	0.204476	2.04476

123 rows × 6 columns

```
plt.figure(figsize=(25,10))
international_df.sort_values(by='pop_show_ratio', ascending=False).iloc[:20].plot.bar(x='country',y=['total','pop_show_ratio'], figsize=(
plt.xticks(rotation=45,ha='right',fontsize=14)
ChartNumber="ListedIn.3"
plt.title(f'{ChartNumber}: Local language Shows and Population/Number of shows per country')
plt.legend(['Number of Shows','Ration of Population To Number of Shows'])
plt.show()
```

```
<Figure size 2500x1000 with 0 Axes>
```



- On basis of ratio of population to number of local language show, Ethiopia, Bangladesh, Sudan and Uganda are countries to target.

Need more data: As this needs more analysis in terms of per capita income. How many people can even afford Netflix in these countries.

Recommendation 6: Netflix can look into how to reach out to some of the populous countries in Africa like Ethiopia, Sudan and Uganda, and Bangladesh.

Double-click (or enter) to edit

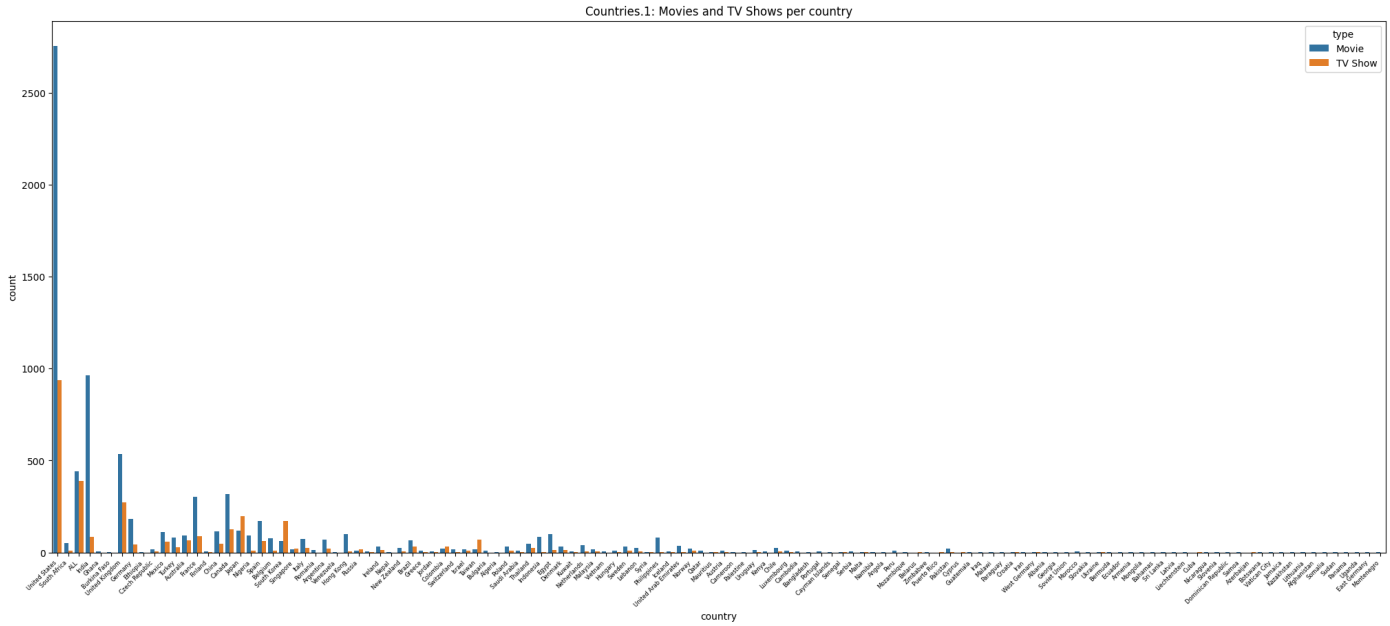
```
country_df[country_df.country=='India']
```

	type	title	country	date_added	listed_in	duration
4	TV Show	Kota Factory	India	2021-09-24	International TV Shows, Romantic TV Shows, TV ...	2 Seasons
24	Movie	Jeans	India	2021-09-21	Comedies, International Movies, Romantic Movies	166 min
29	Movie	Paranoia	India	2021-09-19	Thrillers	106 min
39	TV Show	Chhota Bheem	India	2021-09-16	Kids' TV	3 Seasons
50	TV Show	Dharmakshetra	India	2021-09-15	International TV Shows, TV Dramas, TV Sci-Fi &...	1 Season
...	...	...	...	...	...	...
8773	Movie	Yanda Kartavya Aahe	India	2018-01-01	Comedies, Dramas, International Movies	151 min
8775	TV Show	Yeh Meri Family	India	2018-08-31	International TV Shows, TV Comedies	1 Season
8798	Movie	Zed Plus	India	2019-12-31	Comedies, Dramas, International Movies	131 min
8799	Movie	Zenda	India	2018-02-15	Dramas, International Movies	120 min
8806	Movie	Zubaan	India	2019-03-02	Dramas, International Movies, Music & Musicals	111 min

1046 rows × 6 columns

```
plt.figure(figsize=(25,10))
sns.countplot(data=country_df, x='country',hue='type')
plt.xticks(rotation=45,ha='right',fontsize=6)
ChartNumber="Countries.1"
plt.title(f'{ChartNumber}: Movies and TV Shows per country')
```

```
Text(0.5, 1.0, 'Countries.1: Movies and TV Shows per country')
```



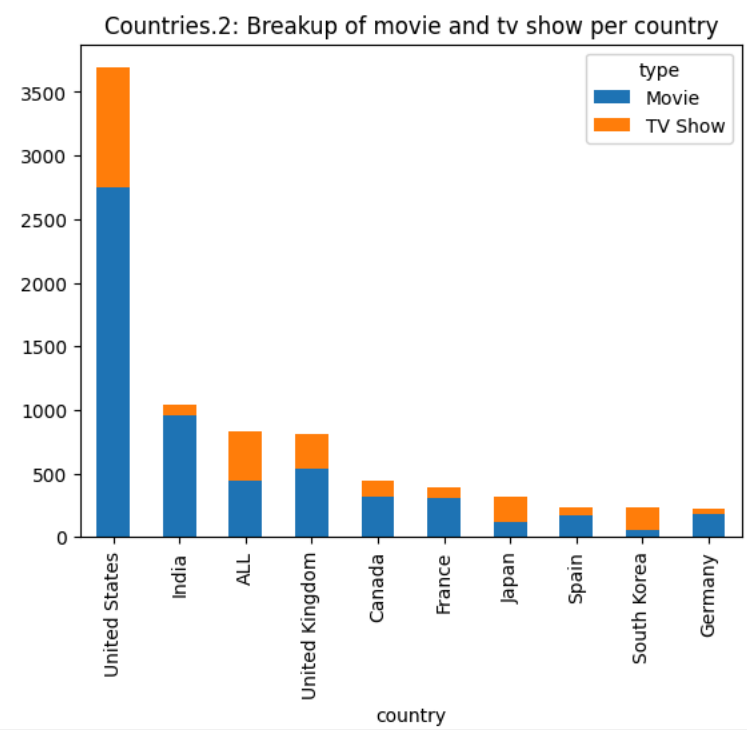

```
data = country_df[['type','country']]
data = data.groupby(['country','type'], as_index=False).size()
data = data.pivot(index='country',columns='type', values='size')
data
```

type country	Movie	TV Show
	6.0	1.0
ALL	440.0	391.0
Afghanistan	1.0	NaN
Albania	1.0	NaN
Algeria	3.0	NaN
...	...	...
Vatican City	1.0	NaN
Venezuela	4.0	NaN
Vietnam	7.0	NaN
West Germany	3.0	2.0
Zimbabwe	3.0	NaN

124 rows × 2 columns

```
data['TV Show'] =data['TV Show'].fillna(0)
data['Movie']=data['Movie'].fillna(0)
data['total']=data['TV Show']+data.Movie
data.sort_values(by='total', ascending=False, inplace=True)

```

```
plot = data.iloc[:10].plot.bar(stacked =True, y=['Movie','TV Show'])
ChartNumber="Countries.2"
plt.title(f'{ChartNumber}: Breakup of movie and tv show per country')

plt.show()
```



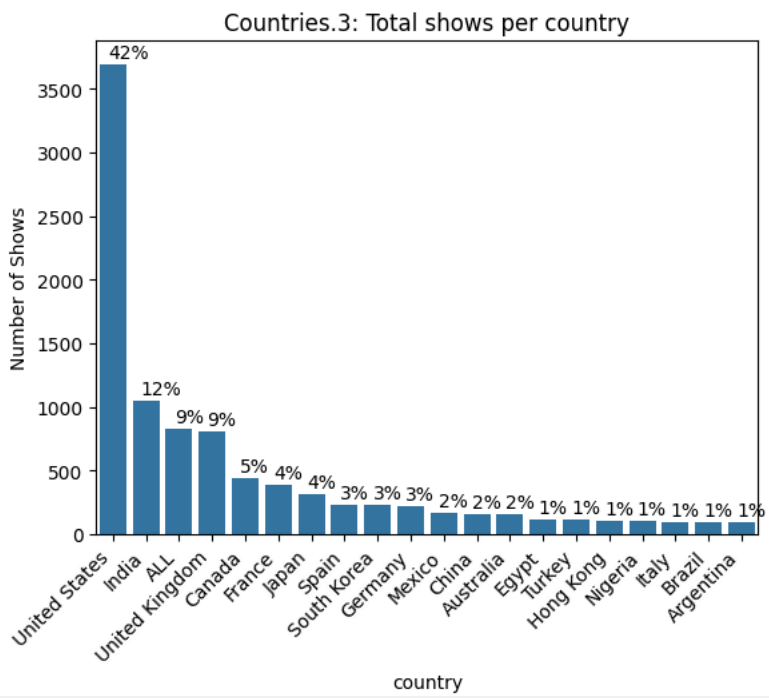
- Among countries where Netflix has 400+ shows, South korea has highest ratio of TV Show versus Movies. There aremany more TV shows released in S Korea as compared to Movies

```
data
```

type	Movie	TV Show	total
country			
United States	2752.0	938.0	3690.0
India	962.0	84.0	1046.0
ALL	440.0	391.0	831.0
United Kingdom	534.0	272.0	806.0
Canada	319.0	126.0	445.0
...	...	...	...
Slovakia	1.0	0.0	1.0
Ethiopia	1.0	0.0	1.0
Nicaragua	1.0	0.0	1.0
Ecuador	1.0	0.0	1.0
Liechtenstein	1.0	0.0	1.0

124 rows × 3 columns

```
plot=sns.barplot(data =data.iloc[:20], x='country', y='total')
plt.xticks(rotation=45, ha='right')
ChartNumber="Countries.3"
plt.title(f'{ChartNumber}: Total shows per country')
plt.ylabel("Number of Shows")
for p in plot.patches:
    plot.annotate('{:}%'.format(round(p.get_height()/total_shows*100)), (p.get_x()+0.25, p.get_height()+50))
plt.show()
```




```
total_shows
```

```
8807
```

- Out of total 8807 shows, 42% were released in United States. Next biggest country in terms of shows is India, UK and Canada.
- 9% of shows are not marked with country, so we can assume they were released in all countries.

Double-click (or enter) to edit

```
country_unique = country_df.country.unique()
# for c in allCountries_df.iterrows():
#     if(c[1]['name'] not in country_unique):
```

```
#         print (c[1]['name'], c[1]['Population'])

missingCountries = allCountries_df[~allCountries_df.name.isin(country_unique)].sort_values(by='Population', ascending=False)
missingCountries.head(10)

```

	Unnamed: 0	name	Population
179	179	Tanzania	53.410312
124	124	Myanmar	51.677032
194	194	Uzbekistan	31.188341
107	107	Madagascar	25.155487
131	131	Niger	20.498079
111	111	Mali	18.385580
200	200	Zambia	16.491819
35	35	Chad	14.348908
151	151	Rwanda	11.776857
72	72	Guinea	11.768478

```
#Fix country names
data = data.rename(index={'United States': 'United States of America'})
data

```

type	Movie	TV Show	total
country			
United States of America	2752.0	938.0	3690.0
India	962.0	84.0	1046.0
ALL	440.0	391.0	831.0
United Kingdom	534.0	272.0	806.0
Canada	319.0	126.0	445.0
...	...	...	...
Slovakia	1.0	0.0	1.0
Ethiopia	1.0	0.0	1.0
Nicaragua	1.0	0.0	1.0
Ecuador	1.0	0.0	1.0
Liechtenstein	1.0	0.0	1.0

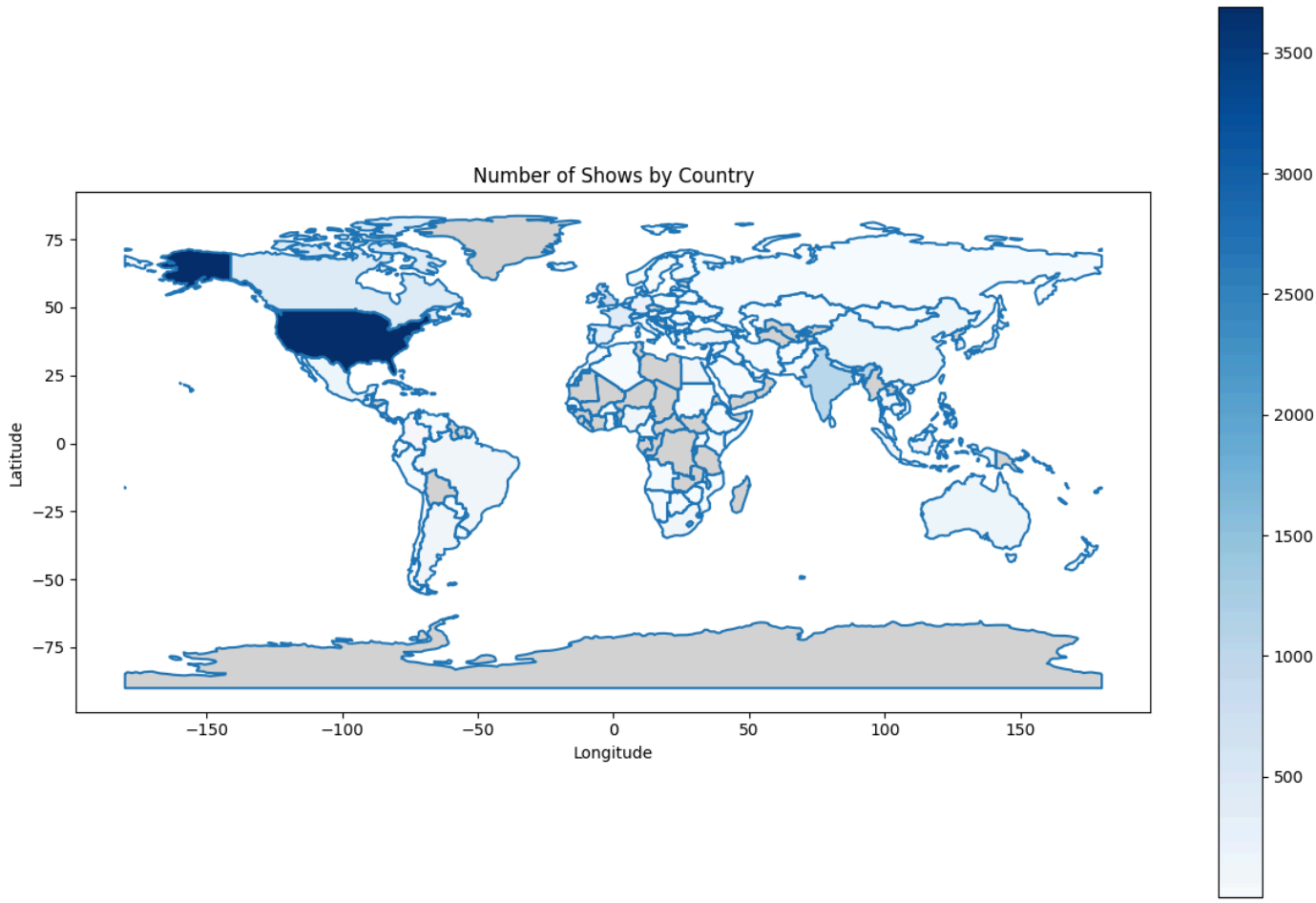
124 rows × 3 columns

```
country_df
```

	type	title	country	date_added	listed_in	duration
0	Movie	Dick Johnson Is Dead	United States	2021-09-25	Documentaries	90 min
1	TV Show	Blood & Water	South Africa	2021-09-24	International TV Shows, TV Dramas, TV Mysteries	2 Seasons
2	TV Show	Ganglands	ALL	2021-09-24	Crime TV Shows, International TV Shows, TV Act...	1 Season
3	TV Show	Jailbirds New Orleans	ALL	2021-09-24	Docuseries, Reality TV	1 Season
4	TV Show	Kota Factory	India	2021-09-24	International TV Shows, Romantic TV Shows, TV ...	2 Seasons
...	...	...	...	...	...	...
8802	Movie	Zodiac	United States	2019-11-20	Cult Movies, Dramas, Thrillers	158 min
8803	TV Show	Zombie Dumb	ALL	2019-07-01	Kids' TV, Korean TV Shows, TV Comedies	2 Seasons
8804	Movie	Zombieland	United States	2019-11-01	Comedies, Horror Movies	88 min
8805	Movie	Zoom	United States	2020-01-11	Children & Family Movies, Comedies	88 min
8806	Movie	Zubaan	India	2019-03-02	Dramas, International Movies, Music & Musicals	111 min

10850 rows × 6 columns

```

df = data

# Load the world map data from the local file
world = gpd.read_file('ne_110m_admin_0_countries/ne_110m_admin_0_countries.shp')

# Merge your data with the world data
world = world.merge(df, how='left', left_on='NAME', right_on='country')

# Plot the map
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
world.boundary.plot(ax=ax)
world.plot(column='total', ax=ax, legend=True, cmap='Blues', missing_kwds={"color": "lightgrey"})

# Customize the plot
plt.title('Number of Shows by Country')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

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



- Netflix is not yet present in quite a few countries in the African continent

```
Start coding or generate with AI.
```

```
df = df_cleaned.copy(deep=True)
```

```
director_df = df[['show_id', 'type','director','rating', 'duration','listed_in','country']].copy(deep=True)
director_df = splitAndExplode(director_df, 'director')
director_df
```

	show_id	type	director	rating	duration	listed_in	country
0	s1	Movie	Kirsten Johnson	PG-13	90 min	Documentaries	United States
1	s2	TV Show	other	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	South Africa
2	s3	TV Show	Julien Leclercq	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	ALL
3	s4	TV Show	other	TV-MA	1 Season	Docuseries, Reality TV	ALL
4	s5	TV Show	other	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	India
...	...	...	...	...	...	...	...
8802	s8803	Movie	David Fincher	R	158 min	Cult Movies, Dramas, Thrillers	United States
8803	s8804	TV Show	other	TV-Y7	2 Seasons	Kids' TV, Korean TV Shows, TV Comedies	ALL
8804	s8805	Movie	Ruben Fleischer	R	88 min	Comedies, Horror Movies	United States
8805	s8806	Movie	Peter Hewitt	PG	88 min	Children & Family Movies, Comedies	United States
8806	s8807	Movie	Mozez Singh	TV-14	111 min	Dramas, International Movies, Music & Musicals	India

9612 rows × 7 columns

```
director_count_df = director_df.groupby(['type','director'], as_index=False).size().sort_values(by='size', ascending=False)

director_count_df
```

	type	director	size
5077	TV Show	other	2446
4762	Movie	other	188
3582	Movie	Rajiv Chilaka	22
1817	Movie	Jan Suter	21
3633	Movie	Raúl Campos	19
...	...	...	...
2300	Movie	Kasper Barfoed	1
2301	Movie	Kasper Collin	1
2302	Movie	Kasra Farahani	1
2303	Movie	Katarina Launing	1
458	Movie	Aurora Guerrero	1

5078 rows × 3 columns

```
sns.barplot(director_count_df[director_count_df.type=='Movie'].iloc[:20].loc[director_count_df.director!='other'], x='director', y='size'
plt.xticks(rotation=45,ha='right',fontsize=6)
ChartNumber="Director.1"
plt.ylabel("Number of Movies")
plt.title(f'{ChartNumber}: Top 20 Movie Directors')
```

```
Text(0.5, 1.0, 'Director.1: Top 20 Movie Directors')
```

Recommendation 10: Top 10 movie directors with more than 10 titles to their names have been identified. Any movie by these directors should be prioritized