# Business Case: Netflix - Data Exploration and Visualisation

**Problem Statement:** Analyze the data and generate insights that could help Netflix decide which type of shows/movies to produce and how they can grow the business in different countries.

## Defining Problem Statement and Analysing basic metrics:

### Importing required libraries and downloading the data

```
!pip install -U gdown
import gdown
import pandas as pd
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.11/dist-packages (5.2.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.11/dist-packages (from gdown) (4.13.4)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from gdown) (3.18.0)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.11/dist-packages (from gdown) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from gdown) (4.67.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4->gdown) (2.7)
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4->gdown) (4.14.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests[socks]->gdown) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests[socks]->gdown) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests[socks]->gdown) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests[socks]->gdown) (2025.7.14)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.11/dist-packages (from requests[socks]->gdown) (1.7.1)
```

Google Drive file ID and download URL. Download the file.

```
[112] file_id = '13nPuO4TwmN21vZcz0ApLbtyBJcoVLqQE'
      url = f'https://drive.google.com/uc?id={file_id}'
      gdown.download(url, 'netflix_titles.csv', quiet=False)
```

```
Downloading...
From: https://drive.google.com/uc?id=13nPuO4TwmN21vZcz0ApLbtyBJcoVLqQE
To: /content/netflix_titles.csv
100%|██████████| 3.40M/3.40M [00:00<00:00, 75.9MB/s]
'netflix_titles.csv'
```

### Loading the dataset and checking its length

Load the CSV into a DataFrame

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

# Step 6: Display first few rows
df.head(10)
```

| | 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... |
| 3 | s4 | TV Show | Jailbirds New Orleans | NaN | NaN | NaN | September 24, 2021 | 2021 | TV-MA | 1 Season | Docuseries, Reality TV | Feuds, flirtations and toilet talk go down amo... |
| 4 | s5 | TV Show | Kota Factory | NaN | Mayur More, Jitendra Kumar, Ranjan Raj, Alam K... | India | September 24, 2021 | 2021 | TV-MA | 2 Seasons | International TV Shows, Romantic TV Shows, TV ... | In a city of coaching centers known to train I... |
| 5 | s6 | TV Show | Midnight Mass | Mike Flanagan | Kate Siegel, Zach Gilford, Hamish Linklater, H... | NaN | September 24, 2021 | 2021 | TV-MA | 1 Season | TV Dramas, TV Horror, TV Mysteries | The arrival of a charismatic young priest brin... |
| 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 min | Children & Family Movies | Equestria's divided. But a bright-eyed hero be... |
| 7 | s8 | Movie | Sankofa | Haile Gerima | Kofi Ghanaba, Oyafunmike Ogunlano, Alexandra D... | United States, Ghana, Burkina Faso, United Kin... | September 24, 2021 | 1993 | TV-MA | 125 min | Dramas, Independent Movies, International Movies | On a photo shoot in Ghana, an American model s... |
| 8 | s9 | TV Show | The Great British Baking Show | Andy Devonshire | Mel Giedroyc, Sue Perkins, Mary Berry, Paul Ho... | United Kingdom | September 24, 2021 | 2021 | TV-14 | 9 Seasons | British TV Shows, Reality TV | A talented batch of amateur bakers face off in... |
| 9 | s10 | Movie | The Starling | Theodore Melfi | Melissa McCarthy, Chris O'Dowd, Kevin Kline, T... | United States | September 24, 2021 | 2021 | PG-13 | 104 min | Comedies, Dramas | A woman adjusting to life after a loss contend... |

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

```
[114] len(df)
```

```
8807
```

# 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.

## Observations from the dataset:

1. There are a total 8807 titles and 12 features or columns.
2. Null values in a few columns director,cast, country.
3. Nested values in a few columns like director,cast, listed_in, etc.

Getting all the attributes of the data and checking its shape:

```
[115] df.columns
```

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

The shape of data

```
[116] df.ndim
```

```
2
```

## Data type of all the attributes

```
[117] 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
```

It is clear that except release_year every attribute data type is an object which is a string in Numpy Pandas. But the values in the "duration" must be int. To deal with this type conversion is the way to go. We will do this later part of this document.

Describing the dataset to get all statistical info about the numerical columns.

```
[118] df.describe(include='all').T
```

| | count | unique | top | freq | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **show_id** | 8807 | 8807 | s8807 | 1 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **type** | 8807 | 2 | Movie | 6131 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **title** | 8807 | 8807 | Zubaan | 1 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **director** | 6173 | 4528 | Rajiv Chilaka | 19 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **cast** | 7982 | 7692 | David Attenborough | 19 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **country** | 7976 | 748 | United States | 2818 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **date_added** | 8797 | 1767 | January 1, 2020 | 109 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **release_year** | 8807.0 | NaN | NaN | NaN | 2014.180198 | 8.819312 | 1925.0 | 2013.0 | 2017.0 | 2019.0 | 2021.0 |
| **rating** | 8803 | 17 | TV-MA | 3207 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **duration** | 8804 | 220 | 1 Season | 1793 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **listed_in** | 8807 | 514 | Dramas, International Movies | 362 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **description** | 8807 | 8775 | Paranormal activity at a lush, abandoned prope... | 4 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

Checking the Null values:

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

| | 0 |
|---|---|
| **show_id** | 0 |
| **type** | 0 |
| **title** | 0 |
| **director** | 2634 |
| **cast** | 825 |
| **country** | 831 |
| **date_added** | 10 |
| **release_year** | 0 |
| **rating** | 4 |
| **duration** | 3 |
| **listed_in** | 0 |
| **description** | 0 |

**dtype:** int64

The number of Null values are 2634 in director, 825 in cast, 831 in country, 10 in data_added, 4 in rating, 3 in duration.

### Dealing with the nested values: Starting with Unnesting the directors column:

```
[120] # unnesting the directors column -
      constraint1 = df['director'].apply(lambda x: str(x).split(', ')).tolist() #Splitting individual director by ',' and adding them to a list.
```

Nested director values are now converted into list of values

Each of the comma separated values are transformed into different columns

Creating a new DataFrame (df_new1) using constraint1 as the data and df['title'] as the index.

```
df_new1 = pd.DataFrame(constraint1, index=df['title'])
df_new1.head(20)
```

| title | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dick Johnson Is Dead | Kirsten Johnson | None | None | None | None | None | None | None | None | None | None | None | None |
| Blood & Water | nan | None | None | None | None | None | None | None | None | None | None | None | None |
| Ganglands | Julien Leclercq | None | None | None | None | None | None | None | None | None | None | None | None |
| Jailbirds New Orleans | nan | None | None | None | None | None | None | None | None | None | None | None | None |
| Kota Factory | nan | None | None | None | None | None | None | None | None | None | None | None | None |
| Midnight Mass | Mike Flanagan | None | None | None | None | None | None | None | None | None | None | None | None |
| My Little Pony: A New Generation | Robert Cullen | José Luis Ucha | None | None | None | None | None | None | None | None | None | None | None |
| Sankofa | Haile Gerima | None | None | None | None | None | None | None | None | None | None | None | None |
| The Great British Baking Show | Andy Devonshire | None | None | None | None | None | None | None | None | None | None | None | None |
| The Starling | Theodore Melfi | None | None | None | None | None | None | None | None | None | None | None | None |
| Vendetta: Truth, Lies and The Mafia | nan | None | None | None | None | None | None | None | None | None | None | None | None |
| Bangkok Breaking | Kongkiat Komesiri | None | None | None | None | None | None | None | None | None | None | None | None |
| Je Suis Karl | Christian Schwochow | None | None | None | None | None | None | None | None | None | None | None | None |
| Confessions of an Invisible Girl | Bruno Garotti | None | None | None | None | None | None | None | None | None | None | None | None |
| Crime Stories: India Detectives | nan | None | None | None | None | None | None | None | None | None | None | None | None |
| Dear White People | nan | None | None | None | None | None | None | None | None | None | None | None | None |
| Europe's Most Dangerous Man: Otto Skorzeny in Spain | Pedro de Echave García | Pablo Azorín Williams | None | None | None | None | None | None | None | None | None | None | None |
| Falsa identidad | nan | None | None | None | None | None | None | None | None | None | None | None | None |
| Intrusion | Adam Salky | None | None | None | None | None | None | None | None | None | None | None | None |
| Jaguar | nan | None | None | None | None | None | None | None | None | None | None | None | None |

Reshaping the data by stacking:

```
[123] df_new1 = df_new1.stack()
      df_new1[:30]
```

| | | 0 |
|---|---|---|
| | title | |
| Dick Johnson Is Dead | 0 | Kirsten Johnson |
| Blood & Water | 0 | nan |
| Ganglands | 0 | Julien Leclercq |
| Jailbirds New Orleans | 0 | nan |
| Kota Factory | 0 | nan |
| Midnight Mass | 0 | Mike Flanagan |
| My Little Pony: A New Generation | 0 | Robert Cullen |
| | 1 | José Luis Ucha |
| Sankofa | 0 | Haile Gerima |
| The Great British Baking Show | 0 | Andy Devonshire |
| The Starling | 0 | Theodore Melfi |
| Vendetta: Truth, Lies and The Mafia | 0 | nan |
| Bangkok Breaking | 0 | Kongkiat Komesiri |
| Je Suis Karl | 0 | Christian Schwochow |
| Confessions of an Invisible Girl | 0 | Bruno Garotti |
| Crime Stories: India Detectives | 0 | nan |
| Dear White People | 0 | nan |
| Europe's Most Dangerous Man: Otto Skorzeny in Spain | 0 | Pedro de Echave García |
| | 1 | Pablo Azorín Williams |
| Falsa identidad | 0 | nan |
| Intrusion | 0 | Adam Salky |
| Jaguar | 0 | nan |

Cleaning the Data Set by

1. Re-setting the index of the data frame and
2. Renaming the column to "Director". and
3. Removes a default-level column (level_1)

```
df_new1=pd.DataFrame(df_new1.reset_index())
df_new1.head(30)
```

| | title | level_1 | 0 |
|---|---|---|---|
| 0 | Dick Johnson Is Dead | 0 | Kirsten Johnson |
| 1 | Blood & Water | 0 | nan |
| 2 | Ganglands | 0 | Julien Leclercq |
| 3 | Jailbirds New Orleans | 0 | nan |
| 4 | Kota Factory | 0 | nan |
| 5 | Midnight Mass | 0 | Mike Flanagan |
| 6 | My Little Pony: A New Generation | 0 | Robert Cullen |
| 7 | My Little Pony: A New Generation | 1 | José Luis Ucha |
| 8 | Sankofa | 0 | Haile Gerima |
| 9 | The Great British Baking Show | 0 | Andy Devonshire |
| 10 | The Starling | 0 | Theodore Melfi |
| 11 | Vendetta: Truth, Lies and The Mafia | 0 | nan |
| 12 | Bangkok Breaking | 0 | Kongkiat Komesiri |
| 13 | Je Suis Karl | 0 | Christian Schwochow |
| 14 | Confessions of an Invisible Girl | 0 | Bruno Garotti |
| 15 | Crime Stories: India Detectives | 0 | nan |
| 16 | Dear White People | 0 | nan |
| 17 | Europe's Most Dangerous Man: Otto Skorzeny in ... | 0 | Pedro de Echave García |
| 18 | Europe's Most Dangerous Man: Otto Skorzeny in ... | 1 | Pablo Azorín Williams |
| 19 | Falsa identidad | 0 | nan |
| 20 | Intrusion | 0 | Adam Salky |
| 21 | Jaguar | 0 | nan |
| 22 | Monsters Inside: The 24 Faces of Billy Milligan | 0 | Olivier Megaton |

Now:

1. Resetting the index,
2. renaming the '0' column to 'Director' and
3. Dropping the unnecessary 'level_1' column.

```
[126] df_new1=pd.DataFrame(df_new1.reset_index())
      df_new1.rename(columns={0:'Director'}, inplace=True)
      df_new1.drop(['level_1'], axis=1, inplace=True)
```

```
df_new1.head(50)
```

| | index | title | Director |
|---|---|---|---|
| **0** | 0 | Dick Johnson Is Dead | Kirsten Johnson |
| **1** | 1 | Blood & Water | nan |
| **2** | 2 | Ganglands | Julien Leclercq |
| **3** | 3 | Jailbirds New Orleans | nan |
| **4** | 4 | Kota Factory | nan |
| **5** | 5 | Midnight Mass | Mike Flanagan |
| **6** | 6 | My Little Pony: A New Generation | Robert Cullen |
| **7** | 7 | My Little Pony: A New Generation | José Luis Ucha |
| **8** | 8 | Sankofa | Haile Gerima |
| **9** | 9 | The Great British Baking Show | Andy Devonshire |
| **10** | 10 | The Starling | Theodore Melfi |
| **11** | 11 | Vendetta: Truth, Lies and The Mafia | nan |
| **12** | 12 | Bangkok Breaking | Kongkiat Komesiri |
| **13** | 13 | Je Suis Karl | Christian Schwochow |
| **14** | 14 | Confessions of an Invisible Girl | Bruno Garotti |
| **15** | 15 | Crime Stories: India Detectives | nan |
| **16** | 16 | Dear White People | nan |
| **17** | 17 | Europe's Most Dangerous Man: Otto Skorzeny in ... | Pedro de Echave García |
| **18** | 18 | Europe's Most Dangerous Man: Otto Skorzeny in ... | Pablo Azorín Williams |
| **19** | 19 | Falsa identidad | nan |
| **20** | 20 | Intrusion | Adam Salky |

Below is how all the unnesting code looks like in one go:

```
# unnesting the directors column -
constraint1=df['director'].apply(lambda x: str(x).split(', ')).tolist()
#Splitting individual director by , and adding them to a list.
df_new1=pd.DataFrame(constraint1, index=df['title'])
df_new1=df_new1.stack()
df_new1=pd.DataFrame(df_new1.reset_index())
df_new1.rename(columns={0:'Director'}, inplace=True)
df_new1.drop(['level_1'], axis=1, inplace=True)
df_new1.head(20)
```

| | title | Director |
|---|---|---|
| 0 | Dick Johnson Is Dead | Kirsten Johnson |
| 1 | Blood & Water | nan |
| 2 | Ganglands | Julien Leclercq |
| 3 | Jailbirds New Orleans | nan |
| 4 | Kota Factory | nan |
| 5 | Midnight Mass | Mike Flanagan |
| 6 | My Little Pony: A New Generation | Robert Cullen |
| 7 | My Little Pony: A New Generation | José Luis Ucha |
| 8 | Sankofa | Haile Gerima |
| 9 | The Great British Baking Show | Andy Devonshire |
| 10 | The Starling | Theodore Melfi |
| 11 | Vendetta: Truth, Lies and The Mafia | nan |
| 12 | Bangkok Breaking | Kongkiat Komesiri |
| 13 | Je Suis Karl | Christian Schwochow |
| 14 | Confessions of an Invisible Girl | Bruno Garotti |
| 15 | Crime Stories: India Detectives | nan |
| 16 | Dear White People | nan |
| 17 | Europe's Most Dangerous Man: Otto Skorzeny in ... | Pedro de Echave García |
| 18 | Europe's Most Dangerous Man: Otto Skorzeny in ... | Pablo Azorín Williams |

Unnesting the 'cast' column just like 'Director' column:

```
# unnesting the cast column -
constraint2=df['cast'].apply(lambda x: str(x).split(', ')).tolist()
df_new2=pd.DataFrame(constraint2, index=df['title'])
df_new2=df_new2.stack()
df_new2=pd.DataFrame(df_new2.reset_index())
df_new2.rename(columns={0:'Cast'}, inplace=True)
df_new2.drop(['level_1'], axis=1, inplace=True)
df_new2.head(40)
```

| | title | Cast |
|---|---|---|
| 0 | Dick Johnson Is Dead | nan |
| 1 | Blood & Water | Ama Qamata |
| 2 | Blood & Water | Khosi Ngema |
| 3 | Blood & Water | Gail Mabalane |
| 4 | Blood & Water | Thabang Molaba |
| 5 | Blood & Water | Dillon Windvogel |
| 6 | Blood & Water | Natasha Thahane |
| 7 | Blood & Water | Arno Greeff |
| 8 | Blood & Water | Xolile Tshabalala |
| 9 | Blood & Water | Getmore Sithole |
| 10 | Blood & Water | Cindy Mahlangu |
| 11 | Blood & Water | Ryle De Morny |
| 12 | Blood & Water | Greteli Fincham |
| 13 | Blood & Water | Sello Maake Ka-Ncube |
| 14 | Blood & Water | Odwa Gwanya |
| 15 | Blood & Water | Mekaila Mathys |
| 16 | Blood & Water | Sandi Schultz |
| 17 | Blood & Water | Duane Williams |
| 18 | Blood & Water | Shamilla Miller |
| 19 | Blood & Water | Patrick Mofokeng |

Unnesting the 'listed_in' column just like 'Director' column:

```
# unnesting the listed_in column -
constraint3=df['listed_in'].apply(lambda x: str(x).split(', ')).tolist()
df_new3=pd.DataFrame(constraint3, index=df['title'])
df_new3=df_new3.stack()
df_new3=pd.DataFrame(df_new3.reset_index())
df_new3.rename(columns={0:'Genre'}, inplace=True)
df_new3.drop(['level_1'], axis=1, inplace=True)
df_new3.head(20)
```

| | title | Genre |
|---|---|---|
| 0 | Dick Johnson Is Dead | Documentaries |
| 1 | Blood & Water | International TV Shows |
| 2 | Blood & Water | TV Dramas |
| 3 | Blood & Water | TV Mysteries |
| 4 | Ganglands | Crime TV Shows |
| 5 | Ganglands | International TV Shows |
| 6 | Ganglands | TV Action & Adventure |
| 7 | Jailbirds New Orleans | Docuseries |
| 8 | Jailbirds New Orleans | Reality TV |
| 9 | Kota Factory | International TV Shows |
| 10 | Kota Factory | Romantic TV Shows |
| 11 | Kota Factory | TV Comedies |
| 12 | Midnight Mass | TV Dramas |
| 13 | Midnight Mass | TV Horror |
| 14 | Midnight Mass | TV Mysteries |
| 15 | My Little Pony: A New Generation | Children & Family Movies |
| 16 | Sankofa | Dramas |
| 17 | Sankofa | Independent Movies |
| 18 | Sankofa | International Movies |
| 19 | The Great British Baking Show | British TV Shows |

Unnesting the 'country' column just like 'Director' column:

```
#unnesting the country column
constraint4 = df['country'].apply(lambda x: str(x).split(', ')).tolist()
df_new4 = pd.DataFrame(constraint4, index = df['title'])
df_new4 = df_new4.stack()
df_new4 = pd.DataFrame(df_new4.reset_index())
df_new4.rename(columns={0:'Country'}, inplace=True )
df_new4.drop(['level_1'], axis=1, inplace=True)
df_new4.head(20)
```

| | title | Country |
|---|---|---|
| 0 | Dick Johnson Is Dead | United States |
| 1 | Blood & Water | South Africa |
| 2 | Ganglands | nan |
| 3 | Jailbirds New Orleans | nan |
| 4 | Kota Factory | India |
| 5 | Midnight Mass | nan |
| 6 | My Little Pony: A New Generation | nan |
| 7 | Sankofa | United States |
| 8 | Sankofa | Ghana |
| 9 | Sankofa | Burkina Faso |
| 10 | Sankofa | United Kingdom |
| 11 | Sankofa | Germany |
| 12 | Sankofa | Ethiopia |
| 13 | The Great British Baking Show | United Kingdom |
| 14 | The Starling | United States |
| 15 | Vendetta: Truth, Lies and The Mafia | nan |
| 16 | Bangkok Breaking | nan |
| 17 | Je Suis Karl | Germany |
| 18 | Je Suis Karl | Czech Republic |
| 19 | Confessions of an Invisible Girl | nan |

Checking which country is contributing more to Netflix:

```
df_new4.groupby(['Country']).agg({"title":"nunique"}).reset_index().sort_values(by=['title'],
                                                                                 ascending=False)
```

| | Country | title |
|---|---|---|
| 119 | United States | 3689 |
| 47 | India | 1046 |
| 127 | nan | 831 |
| 117 | United Kingdom | 804 |
| 22 | Canada | 445 |
| ... | ... | ... |
| 101 | Somalia | 1 |
| 106 | Sri Lanka | 1 |
| 114 | Uganda | 1 |
| 122 | Vatican City | 1 |
| 120 | United States, | 1 |

128 rows × 2 columns

The United States stands as the country with the highest number of titles in Netflix.

Now merging all the Unnested data frames:

```
# merging the unnested director data with unnested cast data
df_new5=df_new2.merge(df_new1, on=['title'], how='inner')
# merging the above merged data with unnested genre data
df_new6=df_new5.merge(df_new3, on=['title'], how='inner')
# merging the above merged data with unnested country data
df_new=df_new6.merge(df_new4, on=['title'], how='inner')
```

Replacing all the nan director, country and cast values with Unknown:

```
#replacing the null values in below columns with 'Unknown'
df_new['Cast'].fillna('Unknown', inplace=True)
df_new['Director'].fillna('Unknown', inplace=True)
df_new['Country'].fillna('Unknown', inplace=True)
df_new.head(20)
```

| | title | Cast | Director | Genre | Country |
|---|---|---|---|---|---|
| 0 | Dick Johnson Is Dead | nan | Kirsten Johnson | Documentaries | United States |
| 1 | Blood & Water | Ama Qamata | nan | International TV Shows | South Africa |
| 2 | Blood & Water | Ama Qamata | nan | TV Dramas | South Africa |
| 3 | Blood & Water | Ama Qamata | nan | TV Mysteries | South Africa |
| 4 | Blood & Water | Khosi Ngema | nan | International TV Shows | South Africa |
| 5 | Blood & Water | Khosi Ngema | nan | TV Dramas | South Africa |
| 6 | Blood & Water | Khosi Ngema | nan | TV Mysteries | South Africa |
| 7 | Blood & Water | Gail Mabalane | nan | International TV Shows | South Africa |
| 8 | Blood & Water | Gail Mabalane | nan | TV Dramas | South Africa |
| 9 | Blood & Water | Gail Mabalane | nan | TV Mysteries | South Africa |
| 10 | Blood & Water | Thabang Molaba | nan | International TV Shows | South Africa |
| 11 | Blood & Water | Thabang Molaba | nan | TV Dramas | South Africa |
| 12 | Blood & Water | Thabang Molaba | nan | TV Mysteries | South Africa |
| 13 | Blood & Water | Dillon Windvogel | nan | International TV Shows | South Africa |
| 14 | Blood & Water | Dillon Windvogel | nan | TV Dramas | South Africa |
| 15 | Blood & Water | Dillon Windvogel | nan | TV Mysteries | South Africa |
| 16 | Blood & Water | Natasha Thahane | nan | International TV Shows | South Africa |
| 17 | Blood & Water | Natasha Thahane | nan | TV Dramas | South Africa |
| 18 | Blood & Water | Natasha Thahane | nan | TV Mysteries | South Africa |
| 19 | Blood & Water | Arno Greeff | nan | International TV Shows | South Africa |

Checking for any null values in this new data frame before merging this into the original dataframe:

```
[136] df_new.isna().sum()
```

| | 0 |
|---|---|
| title | 0 |
| Cast | 0 |
| Director | 0 |
| Genre | 0 |
| Country | 0 |

**dtype:** int64

Merging the Unnested data with the original data:

```
df_final=df_new.merge(df[['show_id', 'type', 'title', 'date_added',
                          'release_year', 'rating', 'duration']],
                      on=['title'], how='left')
df_final.head()
```

| | title | Cast | Director | Genre | Country | show_id | type | date_added | release_year | rating | duration |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Dick Johnson Is Dead | nan | Kirsten Johnson | Documentaries | United States | s1 | Movie | September 25, 2021 | 2020 | PG-13 | 90 min |
| 1 | Blood & Water | Ama Qamata | nan | International TV Shows | South Africa | s2 | TV Show | September 24, 2021 | 2021 | TV-MA | 2 Seasons |
| 2 | Blood & Water | Ama Qamata | nan | TV Dramas | South Africa | s2 | TV Show | September 24, 2021 | 2021 | TV-MA | 2 Seasons |
| 3 | Blood & Water | Ama Qamata | nan | TV Mysteries | South Africa | s2 | TV Show | September 24, 2021 | 2021 | TV-MA | 2 Seasons |
| 4 | Blood & Water | Khosi Ngema | nan | International TV Shows | South Africa | s2 | TV Show | September 24, 2021 | 2021 | TV-MA | 2 Seasons |

# Non-Graphical Analysis: value counts and unique attributes:

Checking what percentage of Movies and TV shows in the dataset:

```
df['type'].value_counts(normalize = True)*100
```

| type | proportion |
|---|---|
| Movie | 69.615079 |
| TV Show | 30.384921 |

**dtype:** float64

This shows Movies contribute to almost 70% of the Netflix content.

Checking the null values again:

```
[139] df_final.isna().sum() * 100.00 / df_final.shape[0]
```

| | 0 |
|---|---|
| title | 0.000000 |
| Cast | 0.000000 |
| Director | 0.000000 |
| Genre | 0.000000 |
| Country | 0.000000 |
| show_id | 0.000000 |
| type | 0.000000 |
| date_added | 0.078221 |
| release_year | 0.000000 |
| rating | 0.033170 |
| duration | 0.001485 |

**dtype:** float64

Null values appearing in date_added, rating and duration columns. But in low numbers. So we can drop them to clean the dataset.

```
[141] df_final.dropna(subset=['duration', 'rating', 'release_year'], axis=0, inplace=True)
```

Checking Number of unique values per column:

```
df_final.nunique()
```

| | 0 |
|---|---|
| **title** | 8807 |
| **Cast** | 36440 |
| **Director** | 4994 |
| **Genre** | 42 |
| **Country** | 128 |
| **show_id** | 8807 |
| **type** | 2 |
| **date_added** | 1767 |
| **release_year** | 74 |
| **rating** | 17 |
| **duration** | 220 |

**dtype:** int64

Replacing missing (NaN) values in the 'Country' column with the most frequent country:

```
df_final['Country'].fillna(df_final['Country'].value_counts().idxmax(), inplace=True)
```

It is observed that the 'date_added' is not in a proper format for calculations. So cleaning it by replacing the date_added with just the 'year_added':

```
df_final['year_added']=df_final['date_added'].str.split(',', expand=True)[1]
df_final.head()
```

| | title | Cast | Director | Genre | Country | show_id | type | date_added | release_year | rating | duration | year_added |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Dick Johnson Is Dead | nan | Kirsten Johnson | Documentaries | United States | s1 | Movie | September 25, 2021 | 2020 | PG-13 | 90 min | 2021 |
| 1 | Blood & Water | Ama Qamata | nan | International TV Shows | South Africa | s2 | TV Show | September 24, 2021 | 2021 | TV-MA | 2 Seasons | 2021 |
| 2 | Blood & Water | Ama Qamata | nan | TV Dramas | South Africa | s2 | TV Show | September 24, 2021 | 2021 | TV-MA | 2 Seasons | 2021 |
| 3 | Blood & Water | Ama Qamata | nan | TV Mysteries | South Africa | s2 | TV Show | September 24, 2021 | 2021 | TV-MA | 2 Seasons | 2021 |
| 4 | Blood & Water | Khosi Ngema | nan | International TV Shows | South Africa | s2 | TV Show | September 24, 2021 | 2021 | TV-MA | 2 Seasons | 2021 |

**Creating separate data frames for TV shows and movies:**

```
[144] df_shows=df_final[df_final['type']=='TV Show']
      df_movies=df_final[df_final['type']=='Movie']
```

```
ct = pd.crosstab(df_final['type'], df_new['Genre'])
ct.loc['Movie'].sort_values(ascending=False)
```

| Genre | Movie |
|---|---|
| Dramas | 29768 |
| International Movies | 28211 |
| Comedies | 20829 |
| Action & Adventure | 12216 |
| Independent Movies | 9834 |
| Children & Family Movies | 9771 |
| Thrillers | 7107 |
| Romantic Movies | 6412 |
| Horror Movies | 4571 |
| Sci-Fi & Fantasy | 4037 |
| Music & Musicals | 3077 |
| Documentaries | 2407 |
| Sports Movies | 1531 |
| Classic Movies | 1434 |
| Cult Movies | 1077 |
| Anime Features | 1045 |
| LGBTQ Movies | 838 |
| Faith & Spirituality | 719 |
| Stand-Up Comedy | 540 |
| Movies | 407 |

The Genre with the highest number of movies is "Dramas".

```
ct.loc['TV Show'].sort_values(ascending=False)
```

| Genre | TV Show |
|---|---|
| International TV Shows | 12823 |
| TV Dramas | 8942 |
| TV Comedies | 4956 |
| Crime TV Shows | 4733 |
| Kids' TV | 4561 |
| Romantic TV Shows | 3049 |
| Anime Series | 2291 |
| TV Action & Adventure | 2288 |
| Spanish-Language TV Shows | 2126 |
| British TV Shows | 1808 |
| TV Mysteries | 1281 |
| Korean TV Shows | 1122 |
| TV Sci-Fi & Fantasy | 1045 |
| TV Horror | 941 |
| Docuseries | 845 |
| TV Thrillers | 768 |
| Teen TV Shows | 742 |
| Reality TV | 735 |
| TV Shows | 337 |
| Classic & Cult TV | 272 |
| Stand-Up Comedy & Talk Shows | 268 |
| Science & Nature TV | 157 |

The Genre with the highest number of TV Shows is "International TV Shows".

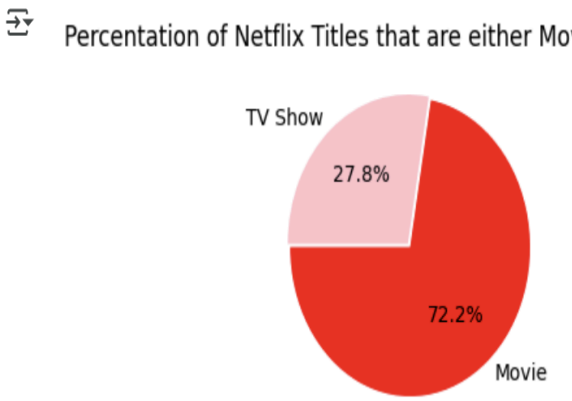
## Visual Analysis - Univariate, Bivariate after pre-processing of the data:

Importing required libraries:

```
[147] import matplotlib
      import matplotlib.pyplot as plt
      import seaborn as sns
```

Visualization using Univariate Analysis: Lets analyze the type of content in Netflix by using Pie Plot:

```
[148] plt.figure(figsize=(6,3))
      plt.title("Percentation of Netflix Titles that are either Movies or TV Shows")
      g=plt.pie(df_final.type.value_counts(),explode=(0.015,0.015),
                labels=df_final.type.value_counts().index, colors=['red','pink'],autopct='%1.1f%%', startangle=180)
```




There are far more movies than TV Shows.

Analysing the amount of content added throughout the years:

```
df_year = df_final.year_added.value_counts().to_frame().reset_index().rename(columns={"index": "count", "year_added":"year"})
df_year = df_year.sort_values(by='year', ascending=False)
df_year
```

| | year | count |
|---|---|---|
| 2 | 2021 | 36516 |
| 1 | 2020 | 46025 |
| 0 | 2019 | 46970 |
| 3 | 2018 | 35770 |
| 4 | 2017 | 25199 |
| 5 | 2016 | 8523 |
| 6 | 2015 | 1560 |
| 7 | 2014 | 450 |
| 9 | 2013 | 207 |
| 10 | 2012 | 36 |
| 8 | 2011 | 438 |
| 12 | 2010 | 20 |
| 11 | 2009 | 30 |
| 13 | 2008 | 19 |

Analysing the number of movies added throughout the years:

```
df_movies_year = df_movies.year_added.value_counts().to_frame().reset_index().rename(columns={"index": "count", "year_added":"year"})
df_movies_year = df_movies_year.sort_values(by='year', ascending=False)
df_movies_year
```

| | year | count |
|---|---|---|
| 3 | 2021 | 25691 |
| 1 | 2020 | 32462 |
| 0 | 2019 | 34446 |
| 2 | 2018 | 28049 |
| 4 | 2017 | 18242 |
| 5 | 2016 | 4856 |
| 6 | 2015 | 1125 |
| 8 | 2014 | 343 |
| 9 | 2013 | 75 |
| 10 | 2012 | 36 |
| 7 | 2011 | 438 |
| 12 | 2010 | 20 |
| 11 | 2009 | 30 |
| 13 | 2008 | 18 |

More no.of movies are added in the year 2019.

Analysing the number of shows added throughout the years:

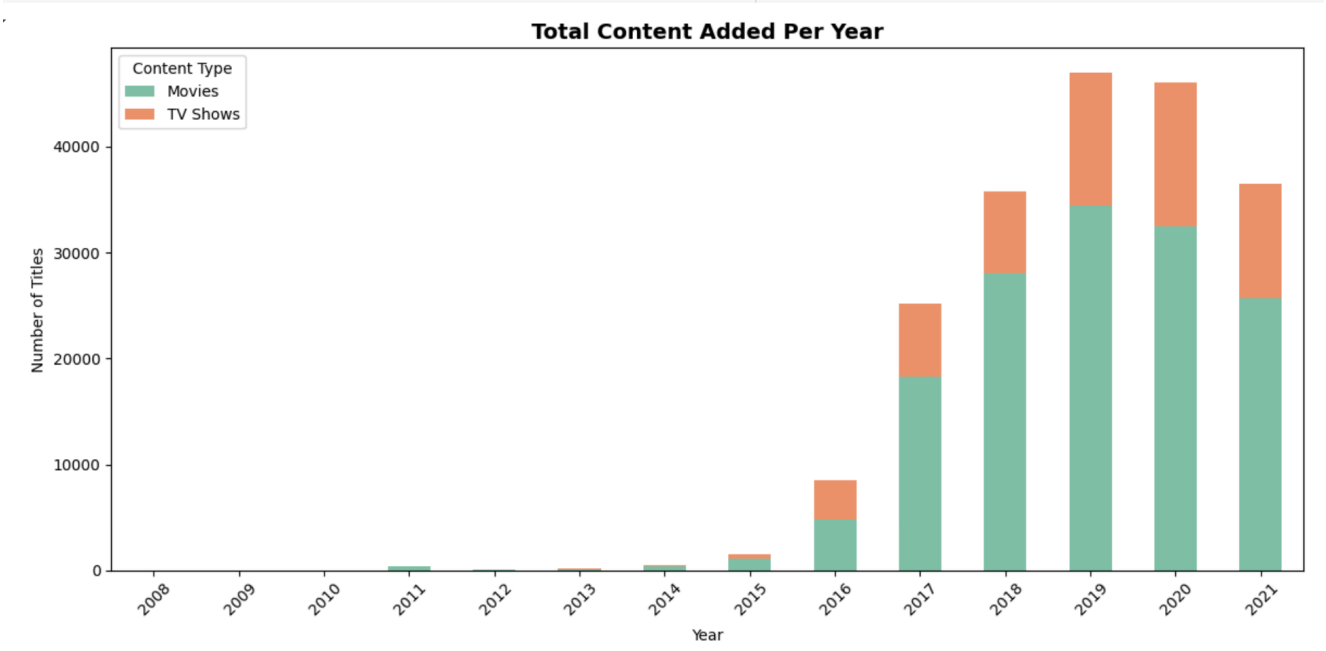
```
df_shows_year = df_shows.year_added.value_counts().to_frame().reset_index().rename(columns={"index": "count", "year_added":"year"})
df_shows_year = df_shows_year.sort_values(by='year', ascending=False)
df_shows_year
```

| | year | count |
|---|---|---|
| 2 | 2021 | 10825 |
| 0 | 2020 | 13563 |
| 1 | 2019 | 12524 |
| 3 | 2018 | 7721 |
| 4 | 2017 | 6957 |
| 5 | 2016 | 3667 |
| 6 | 2015 | 435 |
| 8 | 2014 | 107 |
| 7 | 2013 | 132 |
| 9 | 2008 | 1 |

More no.of shows are added in the year 2020.

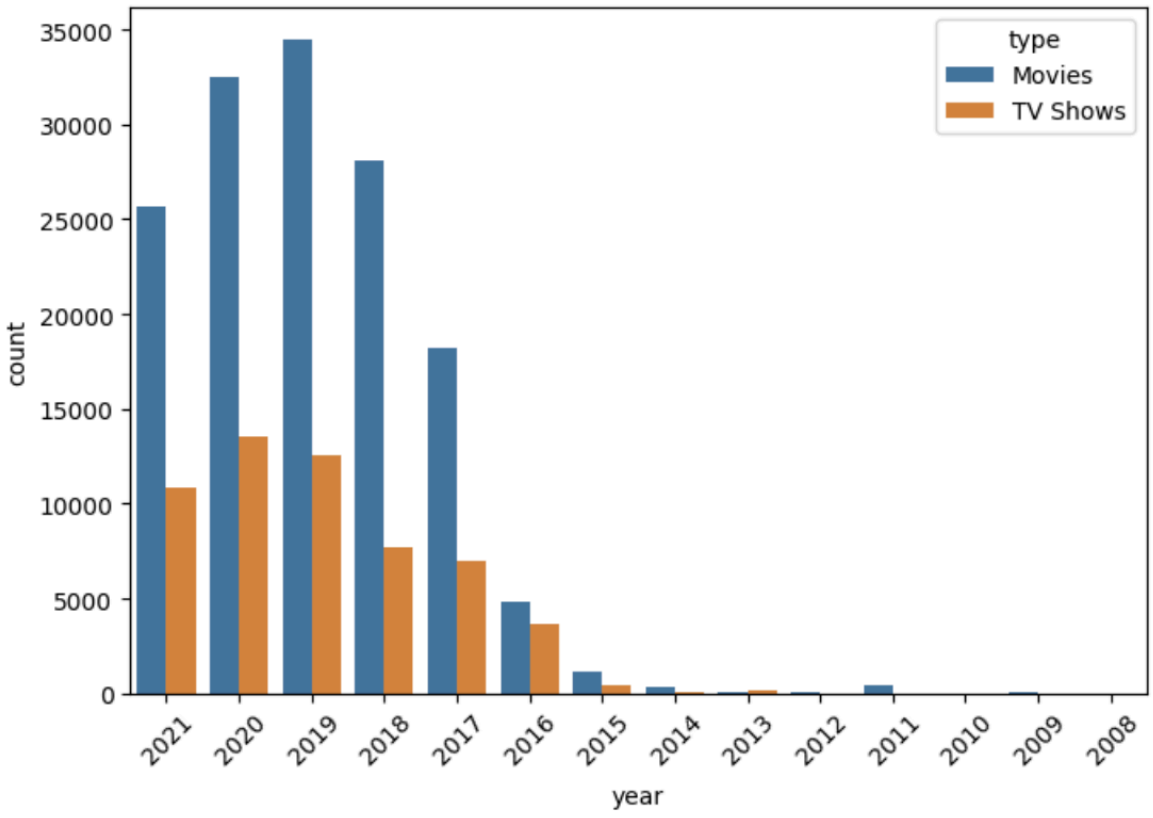
Plotting a Bar plot visualizing the amount of content added in years for both movies and tv shows in a single bar per year:

```
import pandas as pd
import matplotlib.pyplot as plt

# Add labels
df_movies_year['type'] = 'Movies'
df_shows_year['type'] = 'TV Shows'

# Combine the datasets
df_combined = pd.concat([df_movies_year, df_shows_year])
df_combined['year'] = df_combined['year'].astype(str)

# Pivot for stacked bar
df_pivot = df_combined.pivot_table(index='year', columns='type', values='count', fill_value=0)

# Plotting
df_pivot.plot(kind='bar', stacked=True, color=['#66c2a5', '#fc8d62'], figsize=(12, 6))

# Styling
plt.title('Total Content Added Per Year', fontsize=14, weight='bold')
plt.xlabel('Year')
plt.ylabel('Number of Titles')
plt.xticks(rotation=45)
plt.legend(title='Content Type')
plt.tight_layout()
plt.show()
```



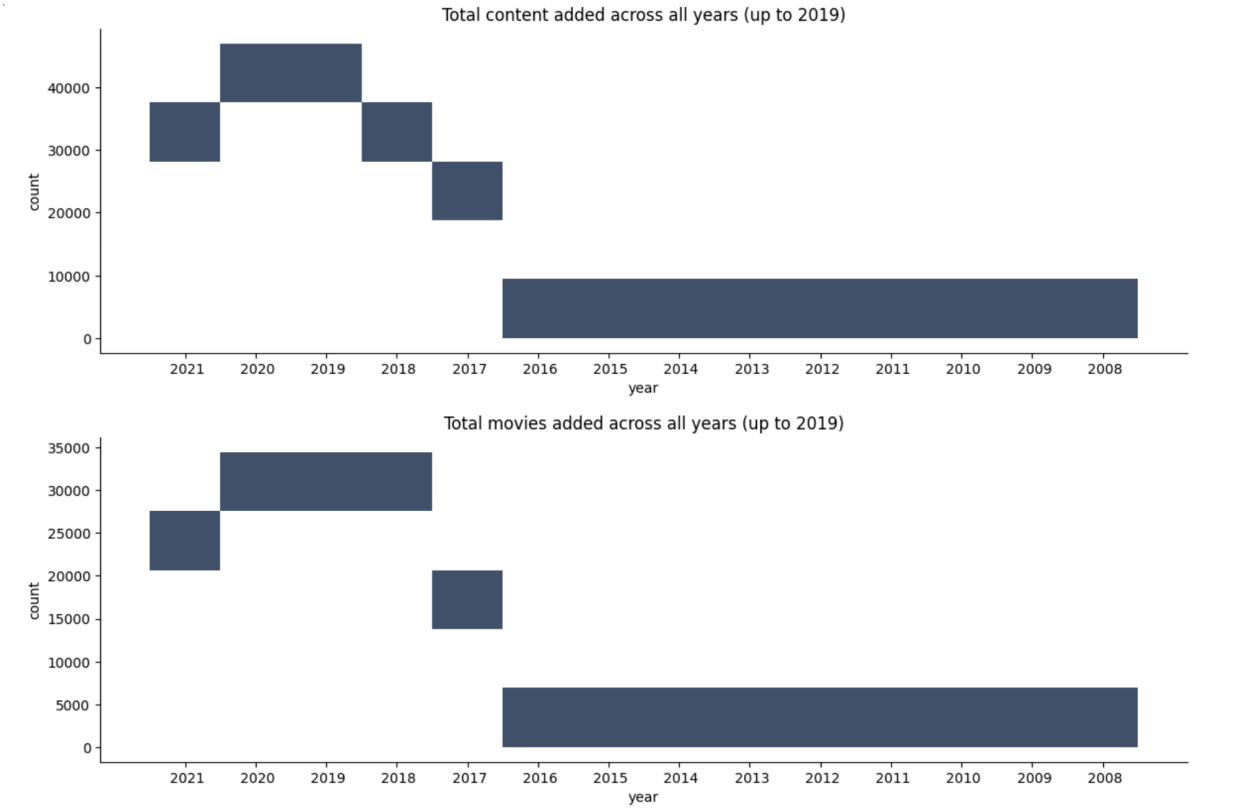
Below the same data is represented in different bars for movies and tv shows per year:

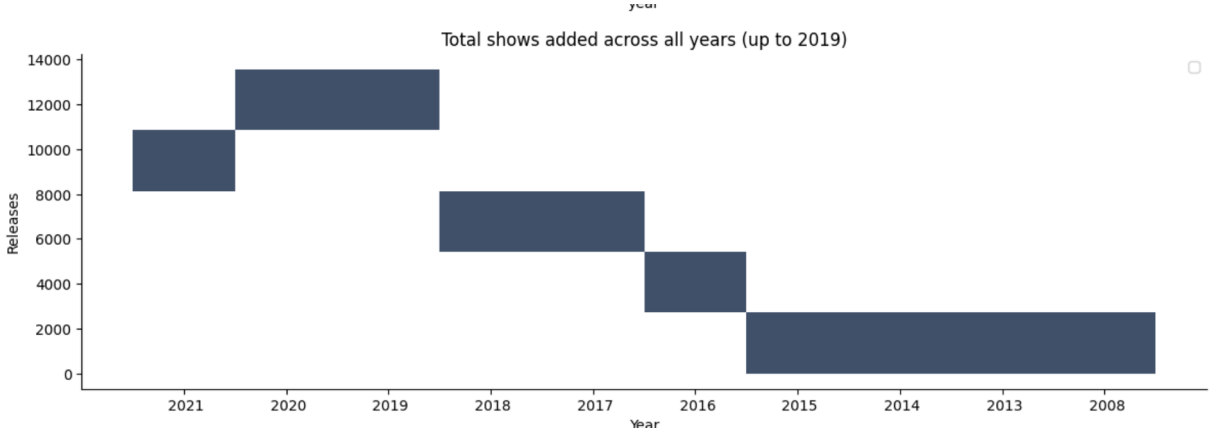
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Combine both datasets with a label
df_movies_year['type'] = 'Movies'
df_shows_year['type'] = 'TV Shows'

df_combined = pd.concat([df_movies_year, df_shows_year])

fig, ax = plt.subplots(figsize=(7, 5))

sns.barplot(data=df_combined, x='year', y='count', hue='type', ax=ax)

ax.set_xticklabels(ax.get_xticklabels(), rotation=45)
plt.tight_layout()
plt.show()
```



Plotting a displot to analyse visually how the metrics between number of titles and year are distributed:

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

#plt.figure(figsize=(12, 6))
#fig, ax = plt.subplots(figsize=(7, 5))
sns.displot(data=df_year, x='year', y='count', height=4, aspect=3)
plt.title("Total content added across all years (up to 2019)")
sns.displot(data=df_movies_year, x='year', y='count', height=4, aspect=3)
plt.title("Total movies added across all years (up to 2019)")
sns.displot (data=df_shows_year, x='year', y='count', height=4, aspect=3)
#ax.set_xticklabels(ax.get_xticklabels(), rotation=45)
#ax.set_xticks(np.arange(2008, 2021, 1))
plt.title("Total shows added across all years (up to 2019)")
plt.legend(['Total','Movie','TV Show'])
plt.ylabel("Releases")
plt.xlabel("Year")
plt.show()
```

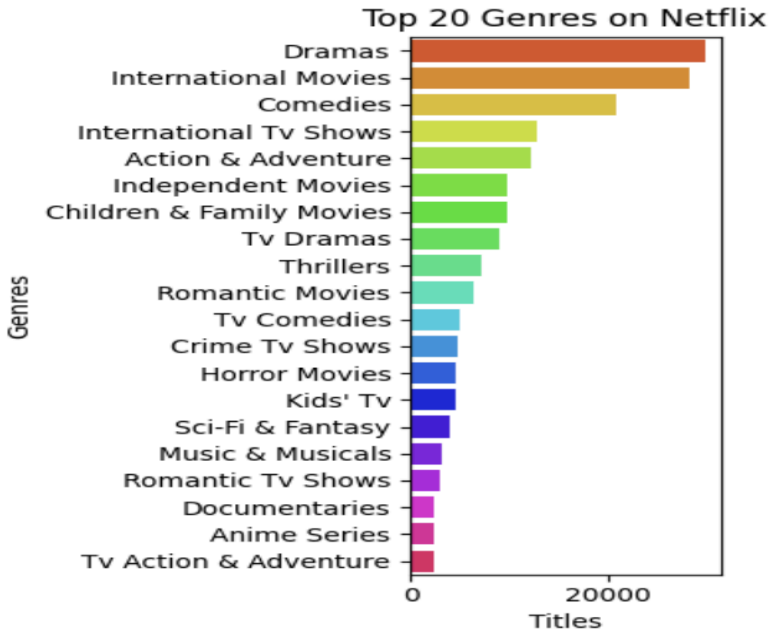





**Bivariate Analysis:**

Now analysing what are the top Genres on the Netflix through Countplot:

```
filtered_genres = df_final.set_index('title')['Genre'].str.split(', ', expand=True).stack()
filtered_genres = filtered_genres.str.strip().str.title()  # remove spaces & unify case
filtered_genres = filtered_genres.reset_index(drop=True)

# Get top 20 genres with truly unique names
top_genres = filtered_genres.value_counts().nlargest(20).index.drop_duplicates()
palette = dict(zip(top_genres, sns.color_palette("hsv", len(top_genres))))

plt.figure(figsize=(4, 5))
sns.countplot(y=filtered_genres, order=top_genres, palette=palette)
plt.title('Top 20 Genres on Netflix')
plt.xlabel('Titles')
plt.ylabel('Genres')
plt.tight_layout()
plt.show()
```



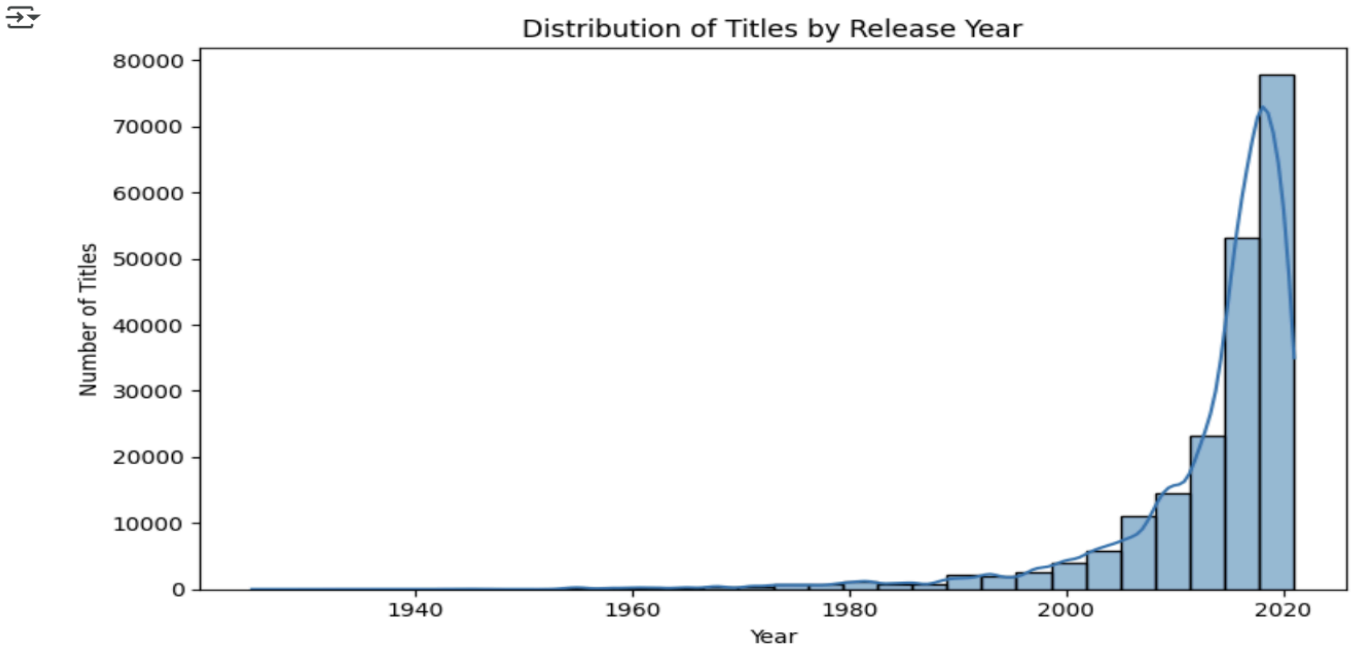
Confirming the same with non-graphical analysis below:

```
genre_counts = df_final['Genre'].value_counts()

print(genre_counts)
```

```
Genre
Dramas                            29768
International Movies              28211
Comedies                          20829
International TV Shows            12823
Action & Adventure                12216
Independent Movies                 9834
Children & Family Movies           9771
TV Dramas                          8942
Thrillers                          7107
Romantic Movies                    6412
TV Comedies                        4956
Crime TV Shows                     4733
Horror Movies                      4571
Kids' TV                           4561
Sci-Fi & Fantasy                   4037
Music & Musicals                   3077
Romantic TV Shows                  3049
Documentaries                      2407
Anime Series                       2291
TV Action & Adventure              2288
Spanish-Language TV Shows          2126
British TV Shows                   1808
Sports Movies                      1531
Classic Movies                     1434
TV Mysteries                       1281
Korean TV Shows                    1122
Cult Movies                        1077
Anime Features                     1045
TV Sci-Fi & Fantasy                1045
TV Horror                           941
Docuseries                          845
LGBTQ Movies                        838
TV Thrillers                        768
Teen TV Shows                       742
Reality TV                          735
Faith & Spirituality                719
Stand-Up Comedy                     540
Movies                              407
TV Shows                            337
Classic & Cult TV                   272
Stand-Up Comedy & Talk Shows        268
Science & Nature TV                 157
```

Now Plotting a histogram over 'release_year':

```
plt.figure(figsize=(8,5))
sns.histplot(data=df_final, x='release_year', bins=30, kde=True)
plt.title('Distribution of Titles by Release Year')
plt.xlabel('Year')
plt.ylabel('Number of Titles')
plt.tight_layout()
plt.show()
```



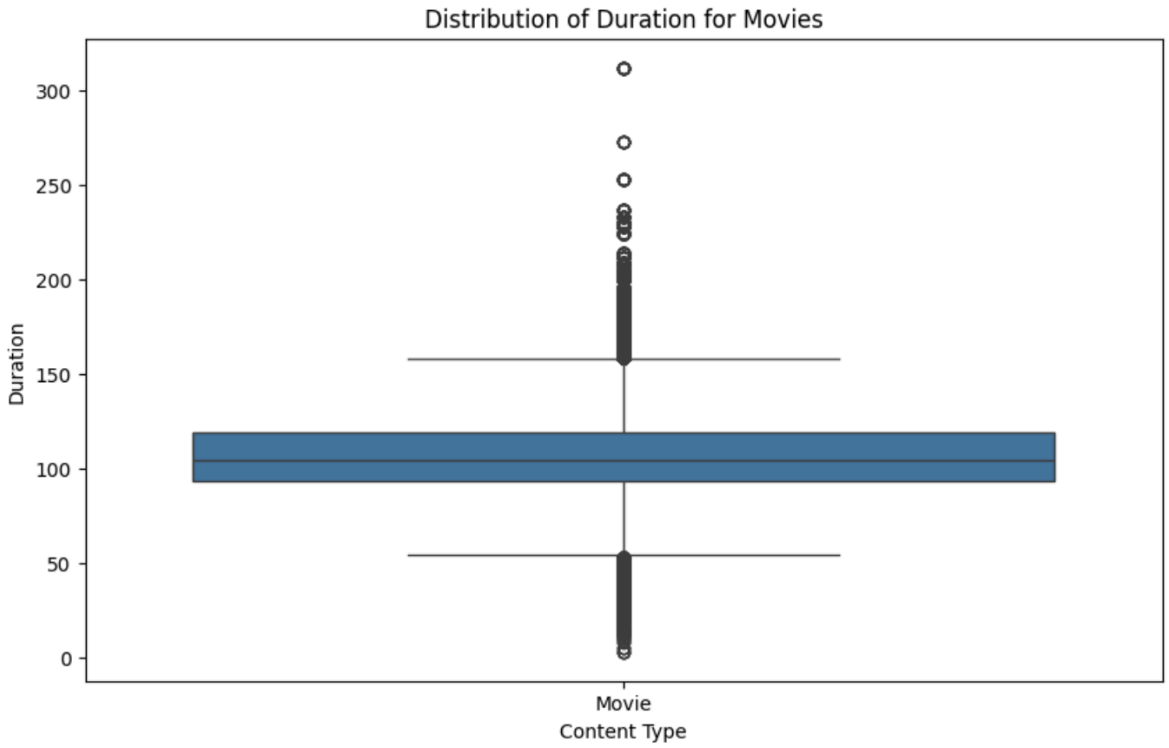

It is clearly evident that more titles started releasing on Netflix just before the Covid Pandemic time i.e., 2019.

## Boxplot and Outlier check:

Now plotting a box plot to analyse the duration of the content separately for Movies and TV Shows.

Let's start with analysing the movies first.

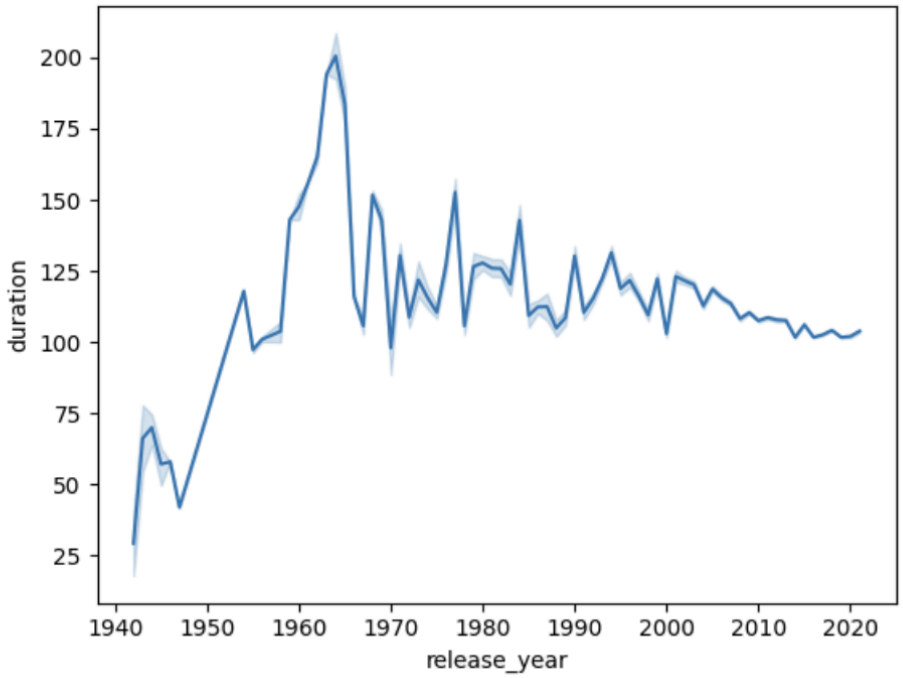
```
netflix_movies_df = df_final[df_final.type.str.contains("Movie")]
netflix_movies_df['duration'] = netflix_movies_df['duration'].str.extract('(\d+)', expand=False).astype(int)

# Creating a boxplot for movie duration
plt.figure(figsize=(10, 6))
sns.boxplot(data=netflix_movies_df, x='type', y='duration')
plt.xlabel('Content Type')
plt.ylabel('Duration')
plt.title('Distribution of Duration for Movies')
plt.show()
```




Analysing the movie box plot, we can see that most movies fall within a reasonable duration range, with few outliers exceeding approximately 2.5 hours. This suggests that most movies on Netflix are designed to fit within a standard viewing time.

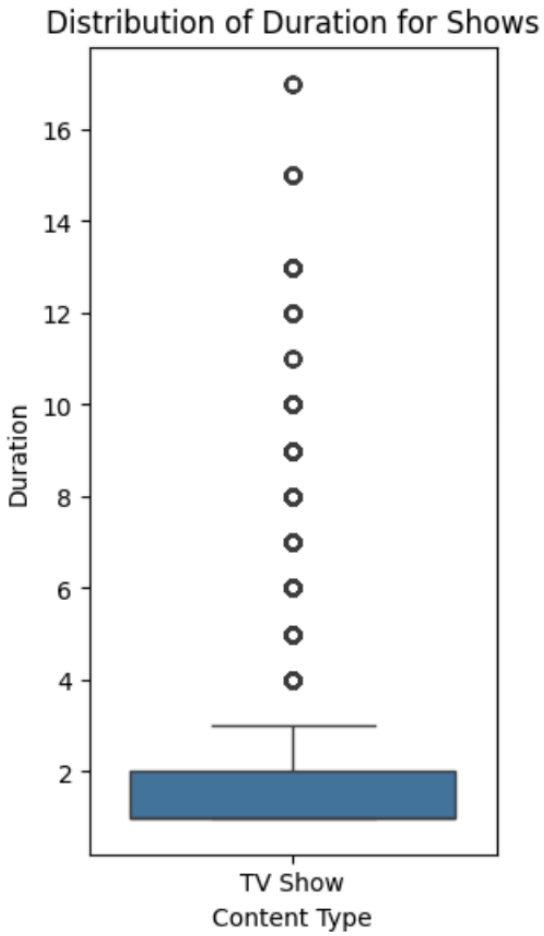
```
sns.lineplot(data=netflix_movies_df, x='release_year', y='duration')
```

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



Analysing the movies line plot, we can see the huge variations in the duration of movies has been significantly reduced over time and from the past 10 years, most of them were confined between 100 to 125 minutes range.

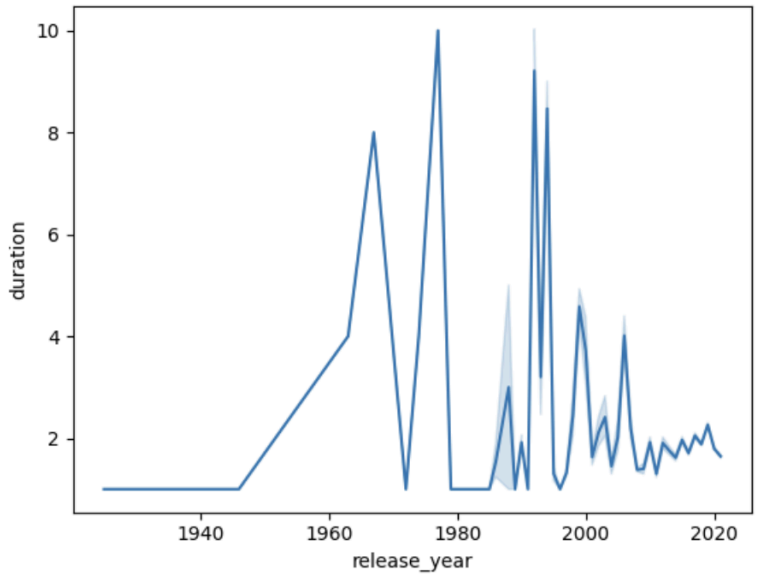
```
netflix_shows_df = df_final[df_final.type.str.contains("TV Show")]
netflix_shows_df['duration'] = netflix_shows_df['duration'].str.extract('(\d+)', expand=False).astype(int)

# Creating a boxplot for movie duration
plt.figure(figsize=(3, 6))
sns.boxplot(data=netflix_shows_df, x='type', y='duration')
plt.xlabel('Content Type')
plt.ylabel('Duration')
plt.title('Distribution of Duration for Shows')
plt.show()
```




For TV shows, the box plot reveals that most shows have one to four seasons, with very few outliers having longer durations. This aligns with the earlier trends, indicating that Netflix focuses on shorter series formats.

```
sns.lineplot(data=netflix_shows_df, x='release_year', y='duration')
```

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




Variation in the duration of tv shows has also been reduced over time. Most of them are not too lengthy from the past few years.

## Missing Values:

```
print('\nColumns with missing value:')
print(df_final.isnull().any())
```

```
Columns with missing value:
title           False
Cast            False
Director        False
Genre           False
Country         False
show_id         False
type            False
date_added       True
release_year    False
rating          False
duration        False
year_added       True
dtype: bool
```

Both date_added and year_added are the same but in different formats. So let's drop the date_added column.

```
df_final = df_final.drop(columns='date_added')
df_final.T.apply(lambda x: x.isnull().sum(), axis = 1)
```

| | 0 |
|---|---|
| **title** | 0 |
| **Cast** | 0 |
| **Director** | 0 |
| **Genre** | 0 |
| **Country** | 0 |
| **show_id** | 0 |
| **type** | 0 |
| **release_year** | 0 |
| **rating** | 0 |
| **duration** | 0 |
| **year_added** | 158 |

**dtype:** int64

The number of missing values in the year_added column is very less. So they can be dropped.

```
df_final.dropna(subset=['year_added'], inplace=True)
df_final.isnull().sum()
```

| | 0 |
|---|---|
| **title** | 0 |
| **Cast** | 0 |
| **Director** | 0 |
| **Genre** | 0 |
| **Country** | 0 |
| **show_id** | 0 |
| **type** | 0 |
| **release_year** | 0 |
| **rating** | 0 |
| **duration** | 0 |
| **year_added** | 0 |

**dtype:** int64

## Business Insights :

With the help of this article, we have been able to learn about-

1. Quantity: Our analysis revealed that Netflix had added more movies than TV shows, aligning with the expectation that movies dominate their content library.
2. Content Addition: July emerged as the month when Netflix adds the most content, closely followed by December, indicating a strategic approach to content release.
3. Genre Correlation: Strong positive associations were observed between various genres, such as TV dramas and international TV shows, romantic and international TV shows, and independent movies and dramas. These correlations provide insights into viewer preferences and content interconnections.
4. Movie Lengths: The analysis of movie durations indicated a peak around the 1960s, followed by a stabilization around 100 minutes, highlighting a trend in movie lengths over time.
5. TV Show Episodes: Most TV shows on Netflix have less than four seasons, suggesting a preference for shorter series among viewers.
6. Common Themes: Words like love, life, family, and adventure were frequently found in titles and descriptions, capturing recurring themes in Netflix content.
7. Rating Distribution: The distribution of ratings over the years offers insights into the evolving content landscape and audience reception.
8. Data-Driven Insights: Our data analysis journey showcased the power of data in unravelling the mysteries of Netflix's content landscape, providing valuable insights for viewers and content creators.
9. Continued Relevance: As the streaming industry evolves, understanding these patterns and trends becomes increasingly essential for navigating the dynamic landscape of Netflix and its vast library.
10. Happy Streaming: We hope this blog has been an enlightening and entertaining journey into the world of Netflix, and we encourage you to explore the captivating stories within its ever-changing content offerings. Let the data guide your streaming adventures!

# RECOMMENDATIONS:

1. Netflix has to focus on TV Shows also because there are people who will like to see tv shows rather than movies
2. By approaching the top director we can plan some more movies/tv shows in order to increase the popularity
3. Not only reaching top directors we can also see the director with less no.of movies and having high ratings as there may be some financial issues or anything so in order to get good content netflix can reach to them and Netflix can produce the movie and give the director a chance.
4. We have seen most no of international movies genre so need to give priority to other genres like horror,comedy..etc
5. In TV Shows we may focus on thriller genre which will be helpful for having more no of seasons
6. Most of the movies released in ott is in a year 2019 so we need to go on increasing this value in order to attract people by showing that
7. Getting subscription is useful as netflix is releasing more movies per year
8. Mainly the release in OTT should focus on the festival holidays, year end and week ends which is to be mainly focussed.
9. Some movies can be released directly into ott which has some positive talk which may help in improving subscriptions
10. Should focus on a actor who has immense following and make use of it by doing a TV Shows or web series
11. Advertisements in the country which has very few movies released should be increased and attract people of that country by making their native TV Shows.