In [8]:

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

1) Uploading Netflix Data

In [9]:

```
!wget "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/original/netflix.csv" -O netflix.csv
```

```
--2023-02-23 14:33:34--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/original/netflix.csv
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 99.84.178.226, 99.84.178.172, 99.84.178.93, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|99.84.178.226|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3399671 (3.2M) [text/plain]
Saving to: 'netflix.csv'

netflix.csv         100%[===================>]   3.24M  --.-KB/s    in 0.03s

2023-02-23 14:33:34 (97.4 MB/s) - 'netflix.csv' saved [3399671/3399671]
```

In [10]:

```
netflix_data = pd.read_csv('netflix.csv')
nd = pd.read_csv('netflix.csv')
```

2)Analyzing Data

In [12]:

```
netflix_data.head()
```

Out[12]:

	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...

In [13]:

```
netflix_data.shape
```

Out[13]:

```
(8807, 12)
```

In [14]:

```
netflix_data.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
```

In [15]:

```
netflix_data["show_id"].unique()
```

Out[15]:

```
array(['s1', 's2', 's3', ..., 's8805', 's8806', 's8807'], dtype=object)
```

In [16]:

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

Out[16]:

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

In [17]:

```
netflix_data["title"].unique()
```

Out[17]:

```
array(['Dick Johnson Is Dead', 'Blood & Water', 'Ganglands', ...,
       'Zombieland', 'Zoom', 'Zubaan'], dtype=object)
```

In [18]:

```
netflix_data["director"].unique()
```

Out[18]:

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

In [19]:

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

Out[19]:

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

In [20]:

```
netflix_data["country"].unique()
```

Out[20]:

```
array(['United States', 'South Africa', nan, 'India',
       'United States, Ghana, Burkina Faso, United Kingdom, Germany, Ethiopia',
       'United Kingdom', 'Germany, Czech Republic', 'Mexico', 'Turkey',
       'Australia', 'United States, India, France', 'Finland',
       'China, Canada, United States',
       'South Africa, United States, Japan', 'Nigeria', 'Japan',
       'Spain, United States', 'France', 'Belgium',
       'United Kingdom, United States', 'United States, United Kingdom',
       'France, United States', 'South Korea', 'Spain',
       'United States, Singapore', 'United Kingdom, Australia, France',
       'United Kingdom, Australia, France, United States',
       'United States, Canada', 'Germany, United States',
       'South Africa, United States', 'United States, Mexico',
       'United States, Italy, France, Japan',
       'United States, Italy, Romania, United Kingdom',
       'Australia, United States', 'Argentina, Venezuela',
       'United States, United Kingdom, Canada', 'China, Hong Kong',
       'Russia', 'Canada', 'Hong Kong', 'United States, China, Hong Kong',
       'Italy, United States', 'United States, Germany',
       'United Kingdom, Canada, United States', ', South Korea',
       'Ireland', 'India, Nepal',
       'New Zealand, Australia, France, United States', 'Italy',
       'Italy, Brazil, Greece', 'Argentina', 'Jordan', 'Colombia',
       'United States, Japan', 'Belgium, United Kingdom',
       'Switzerland, United Kingdom, Australia', 'Israel, United States',
       'Canada, United States', 'Brazil', 'Argentina, Spain', 'Taiwan',
       'United States, Nigeria', 'Bulgaria, United States',
       'Spain, United Kingdom, United States', 'United States, China',
       'United States, France',
       'Spain, France, United Kingdom, United States',
       ', France, Algeria', 'Poland', 'Germany',
       'France, Israel, Germany, United States, United Kingdom',
       'New Zealand', 'Saudi Arabia', 'Thailand', 'Indonesia',
       'Egypt, Denmark, Germany', 'United States, Switzerland',
       'Hong Kong, Canada, United States', 'Kuwait, United States',
       'France, Canada, United States, Spain',
       'France, Netherlands, Singapore', 'France, Belgium',
       'Ireland, United States, United Kingdom', 'Egypt', 'Malaysia',
       'Israel', 'Australia, New Zealand', 'United Kingdom, Germany',
       'Belgium, Netherlands', 'South Korea, Czech Republic',
       'Australia, Germany', 'Vietnam', 'United Kingdom, Belgium',
       'United Kingdom, Australia, United States',
       'France, Japan, United States',
       'United Kingdom, Germany, Spain, United States',
       'United Kingdom, United States, France, Italy',
       'United States, Germany, Canada',
       'United States, France, Italy, United Kingdom',
       'United States, United Kingdom, Germany, Hungary',
       'United States, New Zealand', 'Sweden', 'China', 'Lebanon',
       'Romania', 'Finland, Germany', 'Lebanon, Syria', 'Philippines',
       'Iceland', 'Denmark', 'United States, India',
       'Philippines, Singapore, Indonesia',
       'China, United States, Canada', 'Lebanon, United Arab Emirates',
       'Canada, United States, Denmark', 'United Arab Emirates',
       'Mexico, France, Colombia', 'Netherlands',
       'Germany, United States, France', 'United States, Bulgaria',
       'United Kingdom, France, Germany, United States',
       'Norway, Denmark', 'Syria, France, Lebanon, Qatar',
       'United States, Czech Republic', 'Mauritius',
       'Canada, South Africa', 'Austria', 'Mexico, Brazil',
       'Germany, France', 'Mexico, United States',
       'United Kingdom, France, Spain, United States',
       'United States, Australia',
       'United States, United Kingdom, France', 'United States, Russia',
       'United States, United Kingdom, New Zealand',
       'Australia, United Kingdom', 'Canada, Nigeria, United States',
       'France, United States, United Kingdom, Canada',
       'France, United Kingdom', 'India, United Kingdom',
       'Canada, United States, Mexico',
       'United Kingdom, Germany, United States',
```

'United Kingdom, Germany, United States',
'Czech Republic, United Kingdom, United States',
'China, United Kingdom', 'Italy, United Kingdom', 'China, Taiwan',
'United States, Brazil, Japan, Spain, India',
'United States, China, United Kingdom', 'Cameroon',
'Lebanon, Palestine, Denmark, Qatar', 'Japan, United States',
'Uruguay, Germany', 'Egypt, Saudi Arabia',
'United Kingdom, France, Poland, Germany, United States',
'Ireland, Switzerland, United Kingdom, France, United States',
'United Kingdom, South Africa, France',
'Ireland, United Kingdom, France, Germany',
'Russia, United States', 'United Kingdom, United States, France',
'United Kingdom,', 'United States, India, United Kingdom', 'Kenya',
'Spain, Argentina', 'India, United Kingdom, France, Qatar',
'Belgium, France', 'Argentina, Chile', 'United States, Thailand',
'Chile, Brazil', 'United States, Colombia',
'Canada, United States, United Kingdom', 'Uruguay', 'Luxembourg',
'United States, Cambodia, Romania', 'Bangladesh',
'Spain, Belgium, United States',
'United Kingdom, United States, Australia',
'Canada, United States, France', 'Portugal, United States',
'Portugal, Spain', 'India, United States',
'United Kingdom, Ireland', 'United Kingdom, Spain, United States',
'Hungary, United States', 'United States, South Korea',
'Canada, United States, Cayman Islands', 'India, France',
'France, Canada', 'Canada, Hungary, United States', 'Norway',
'Canada, United Kingdom, United States',
'United Kingdom, Germany, France, United States',
'Denmark, United States', 'Senegal', 'France, Algeria',
'United Kingdom, Finland, Germany, United States, Australia, Japan, France, Ireland',
'Philippines, Canada, United Kingdom, United States',
'Ireland, France, Iceland, United States, Mexico, Belgium, United Kingdom, Hong Kong',
'Singapore', 'Kuwait', 'United States, France, Serbia',
'United States, Italy', 'Spain, Italy',
'United States, Ireland, United Kingdom, India',
'United Kingdom, Singapore', 'Hong Kong, United States',
'United States, Malta, France, United Kingdom',
'United States, China, Canada', 'Canada, United States, Ireland',
'Lebanon, Canada, France', 'Japan, Canada, United States',
'Spain, France, Canada',
'Denmark, Singapore, Canada, United States',
'United States, France, Denmark', 'United States, China, Colombia',
'Spain, Thailand, United States', 'Mexico, Spain',
'Ireland, Luxembourg, Belgium', 'China, United States',
'Canada, Belgium', 'Canada, United Kingdom',
'Lebanon, United Arab Emirates, France, Switzerland, Germany',
'France, Belgium, Italy',
'Lebanon, United States, United Arab Emirates', 'Lebanon, France',
'France, Lebanon', 'France, Lebanon, United Kingdom',
'France, Norway, Lebanon, Belgium',
'Sweden, Czech Republic, United Kingdom, Denmark, Netherlands',
'United States, United Kingdom, India', 'Indonesia, Netherlands',
'Turkey, South Korea', 'Serbia, United States', 'Namibia',
'United Kingdom, Kenya', 'United Kingdom, France, Germany, Spain',
'United Kingdom, France, United States, Belgium, Luxembourg, China, Germany',
'Thailand, United States',
'United States, France, Canada, Belgium', 'United Kingdom, China',
'Germany, China, United Kingdom',
'Australia, New Zealand, United States',
'Hong Kong, Iceland, United States', 'France, Australia, Germany',
'United States, Belgium, Canada, France', 'South Africa, Angola',
'United States, Philippines',
'United States, United Kingdom, Canada, China',
'United States, Canada, United Kingdom', 'Turkey, United States',
'Peru, Germany, Norway', 'Mozambique', 'Brazil, France',
'China, Spain, South Korea, United States', 'Spain, Germany',
'Hong Kong, China', 'France, Belgium, Luxembourg, Cambodia,',
'United Kingdom, Australia', 'Belarus',
'Indonesia, United Kingdom',
'Switzerland, France, Belgium, United States', 'Ghana',
'Spain, France, Canada, United States', 'Chile, Italy',
'United Kingdom, Nigeria', 'Chile', 'France, Egypt',

'Egypt, France', 'France, Brazil, Spain, Belgium',
'Egypt, Algeria', 'Canada, South Korea, United States',
'Nigeria, United Kingdom', 'United States, France, Canada',
'Poland, United States',
'United Arab Emirates, Jordan, Lebanon, Saudi Arabia',
'United States, Mexico, Spain, Malta',
'Saudi Arabia, United Arab Emirates', 'Zimbabwe',
'United Kingdom, Germany, United Arab Emirates, New Zealand',
'Romania, United States', 'Canada, Nigeria',
'Saudi Arabia, Netherlands, Germany, Jordan, United Arab Emirates, United States',
'United Kingdom, Spain', 'Finland, France',
'United Kingdom, Germany, United States, France',
'India, United Kingdom, China, Canada, Japan, South Korea, United States',
'Italy, United Kingdom, France', 'United States, Mexico, Colombia',
'Turkey, India', 'Italy, Turkey',
'United Kingdom, United States, Japan',
'France, Belgium, United States',
'Puerto Rico, United States, Colombia', 'Uruguay, Argentina',
'United States, United Kingdom, Japan', 'United States, Argentina',
'United Kingdom, Italy', 'Ireland, United Kingdom',
'United Kingdom, France, Belgium, Canada, United States',
'Netherlands, Germany, Denmark, United Kingdom', 'Hungary',
'Austria, Germany', 'Taiwan, China',
'United Kingdom, United States, Ireland',
'South Korea, United States', 'Brazil, United Kingdom',
'Pakistan, United States', 'Romania, France, Switzerland, Germany',
'Romania, United Kingdom', 'France, Malta, United States',
'Cyprus',
'United Kingdom, France, Belgium, Ireland, United States',
'United States, Norway, Canada', 'Kenya, United States',
'France, South Korea, Japan, United States', 'Taiwan, Malaysia',
'Uruguay, Argentina, Germany, Spain',
'United States, United Kingdom, France, Germany, Japan',
'United States, France, Japan',
'United Kingdom, France, United States',
'Spain, France, United States',
'Indonesia, South Korea, Singapore', 'United States, Spain',
'Netherlands, Germany, Italy, Canada',
'Spain, Germany, Denmark, United States', 'Norway, Sweden',
'South Korea, Canada, United States, China',
'Argentina, Uruguay, Serbia', 'France, Japan',
'Mauritius, South Africa', 'United States, Poland',
'United Kingdom, United States, Germany, Denmark, Belgium, Japan',
'India, Germany', 'India, United Kingdom, Canada, United States',
'Philippines, United States', 'Romania, Bulgaria, Hungary',
'Uruguay, Guatemala', 'France, Senegal, Belgium',
'United Kingdom, Canada', 'Mexico, United States, Spain, Colombia',
'Canada, Norway', 'Singapore, United States',
'Finland, Germany, Belgium', 'United Kingdom, France',
'United States, Chile', 'United Kingdom, Japan, United States',
'Spain, United Kingdom', 'Argentina, United States, Mexico',
'United States, South Korea, Japan', 'Canada, Australia',
'United Kingdom, Hungary, Australia', 'Italy, Belgium',
'United States, United Kingdom, Germany', 'Switzerland',
'Singapore, Malaysia',
'France, Belgium, Luxembourg, Romania, Canada, United States',
'South Africa, Nigeria', 'Spain, France',
'United Kingdom, Hong Kong', 'Pakistan', 'Brazil, United States',
'Denmark, Brazil, France, Portugal, Sweden', 'India, Turkey',
'Malaysia, Singapore, Hong Kong', 'Philippines, Singapore',
'Australia, Canada', 'Taiwan, China, France, United States',
'Germany, Italy', 'Colombia, Peru, United Kingdom',
'Thailand, China, United States', 'Argentina, United States',
'Sweden, United States', 'Uruguay, Spain, Mexico',
'France, Luxembourg, Canada', 'Denmark, Spain', 'Chile, Argentina',
'United Kingdom, Belgium, Sweden', 'Canada, Brazil',
'Italy, France', 'Canada, Germany',
'Pakistan, United Arab Emirates', 'Ghana, United States',
'Mexico, Finland', 'United Arab Emirates, United Kingdom, India',
'Netherlands, Belgium', 'United States, Taiwan',
'Austria, Iraq, United States', 'United Kingdom, Malawi',
'Paraguay, Argentina', 'United Kingdom, Russia, United States',
'India, Pakistan', 'Indonesia, Singapore', 'Spain, Belgium',

'India, Pakistan', 'Indonesia, Singapore', 'Spain, Belgium',
'Iceland, Sweden, Belgium', 'Croatia', 'Uruguay, Argentina, Spain',
'United Kingdom, Ireland, United States',
'Canada, Germany, France, United States', 'United Kingdom, Japan',
'Norway, Denmark, Netherlands, Sweden',
'Hong Kong, China, United States', 'Ireland, Canada',
'Italy, Switzerland, France, Germany', 'Mexico, Netherlands',
'United States, Sweden', 'Germany, France, Russia',
'France, Iran, United States', 'United Kingdom, India',
'Russia, Poland, Serbia', 'Spain, Portugal', 'Peru',
'Mexico, Argentina',
'United Kingdom, Canada, United States, Cayman Islands',
'Indonesia, United States',
'United States, Israel, United Kingdom, Canada',
'Norway, Iceland, United States', 'Czech Republic, United States',
'United Kingdom, India, United States',
'United Kingdom, West Germany', 'India, Australia',
'United States,', 'Belgium, United Kingdom, United States',
'India, Germany, Austria',
'United States, Brazil, South Korea, Mexico, Japan, Germany',
'Spain, Mexico', 'China, Japan', 'Argentina, France',
'China, United States, United Kingdom',
'France, Luxembourg, United States',
'China, United States, Australia', 'Colombia, Mexico',
'United States, Canada, Ireland', 'Chile, Peru',
'Argentina, Italy', 'Canada, Japan, United States',
'United Kingdom, Canada, United States, Germany',
'Italy, Switzerland, Albania, Poland',
'United States, Japan, Canada', 'Cambodia',
'Italy, United States, Argentina',
'Saudi Arabia, Syria, Egypt, Lebanon, Kuwait',
'United States, Canada, Indonesia, United Kingdom, China, Singapore',
'Spain, Colombia',
'United Kingdom, South Africa, Australia, United States',
'Bulgaria', 'Argentina, Brazil, France, Poland, Germany, Denmark',
'United Kingdom, Spain, United States, Germany',
'Philippines, Qatar', 'Netherlands, Belgium, Germany, Jordan',
'United Arab Emirates, United States', 'Norway, Germany, Sweden',
'South Korea, China', 'Georgia', 'Soviet Union, India',
'Australia, United Arab Emirates', 'Canada, Germany, South Africa',
'South Korea, China, United States', 'India, Soviet Union',
'India, Mexico', 'Georgia, Germany, France',
'United Arab Emirates, Romania', 'India, Malaysia',
'Germany, Jordan, Netherlands', 'Turkey, France, Germany, Poland',
'Greece, United States', 'France, United Kingdom, United States',
'Norway, Germany', 'France, Morocco', 'Cambodia, United States',
'United States, Denmark', 'United States, Colombia, Mexico',
'United Kingdom, Italy, Israel, Peru, United States',
'Argentina, Uruguay, Spain, France',
'United Kingdom, France, United States, Belgium',
'France, Canada, China, Cambodia',
'United Kingdom, France, Belgium, United States', 'Chile, France',
'Netherlands, United States', 'France, United Kingdom, India',
'Czech Republic, Slovakia', 'Singapore, France',
'Spain, Switzerland', 'United States, Australia, China',
'South Africa, United States, Germany',
'United States, United Kingdom, Australia',
'Spain, Italy, Argentina', 'Chile, Spain, Argentina, Germany',
'West Germany', 'Austria, Czech Republic', 'Lebanon, Qatar',
'United Kingdom, Jordan, Qatar, Iran',
'France, South Korea, Japan', 'Israel, Germany, France',
'Canada, Japan, Netherlands', 'United States, Hungary',
'France, Germany', 'France, Qatar',
'United Kingdom, Germany, Canada', 'Ireland, South Africa',
'Chile, United States, France', 'Belgium, France, Netherlands',
'United Kingdom, Ukraine, United States',
'Germany, Australia, France, China', 'Norway, United States',
'United States, Bermuda, Ecuador',
'United States, Hungary, Ireland, Canada',
'United Kingdom, Egypt, United States',
'United States, France, United Kingdom', 'Spain, Mexico, France',
'United States, South Africa', 'Hong Kong, China, Singapore',

'South Africa, China, United States', 'Denmark, France, Poland',
'New Zealand, United Kingdom',
'Netherlands, Denmark, South Africa', 'Iran, France',
'United Kingdom, United States, France, Germany',
'Australia, France', 'Ireland, United Kingdom, United States',
'United Kingdom, France, Germany', 'Canada, Luxembourg',
'Brazil, Netherlands, United States, Colombia, Austria, Germany',
'France, Canada, Belgium', 'Canada, France',
'Bulgaria, United States, Spain, Canada', 'Sweden, Netherlands',
'France, United States, Mexico',
'Australia, United Kingdom, United Arab Emirates, Canada',
'Australia, Armenia, Japan, Jordan, Mexico, Mongolia, New Zealand, Philippines, South Africa, Sweden, United States, Uruguay',
'India, Iran', 'France, Belgium, Spain',
'Denmark, Sweden, Israel, United States', 'United States, Iceland',
'United Kingdom, Russia',
'United States, Israel, Italy, South Africa',
'Netherlands, Denmark, France, Germany', 'South Korea, Japan',
'United Kingdom, Pakistan', 'France, New Zealand',
'United Kingdom, Czech Republic, United States, Germany, Bahamas',
'China, Germany, India, United States', 'Germany, Sri Lanka',
'United States, India, Bangladesh',
'United States, Canada, France', 'Brazil, France, Germany',
'Germany, United States, Hong Kong, Singapore',
'France, Germany, Switzerland',
'Germany, France, Luxembourg, United Kingdom, United States',
'United Kingdom, Canada, Italy', 'Czech Republic, France',
'Taiwan, Hong Kong, United States, China', 'Germany, Australia',
'United Kingdom, Poland, United States', 'Denmark, Zimbabwe',
'United Kingdom, South Africa',
'Finland, Sweden, Norway, Latvia, Germany',
'South Africa, United States, New Zealand, Canada',
'United States, Italy, United Kingdom, Liechtenstein',
'Denmark, France, Belgium, Italy, Netherlands, United States, United Kingdom',
'United States, Australia, Mexico',
'United Kingdom, Czech Republic, Germany, United States',
'France, China, Japan, United States',
'United States, South Korea, China', 'Germany, Belgium',
'Pakistan, Norway, United States',
'United States, Canada, Belgium, United Kingdom', 'Venezuela',
'Canada, France, Italy, Morocco, United States',
'Canada, Spain, France', 'United States, Indonesia',
'Spain, France, Italy',
'United Arab Emirates, United States, United Kingdom',
'United Kingdom, Israel, Russia', 'Spain, Cuba',
'United States, Brazil', 'United States, France, Mexico',
'United States, Nicaragua',
'United Kingdom, United States, Spain, Germany, Greece, Canada',
'Italy, Canada, France',
'United Kingdom, Denmark, Canada, Croatia', 'Italy, Germany',
'United States, France, United Kingdom, Japan',
'United States, United Kingdom, Denmark, Sweden',
'United States, United Kingdom, Italy',
'United States, France, Canada, Spain',
'Russia, United States, China', 'United States, Canada, Germany',
'Ireland, United States', 'United States, United Arab Emirates',
'United States, Ireland',
'Ireland, United Kingdom, Italy, United States', 'Poland,',
'Slovenia, Croatia, Germany, Czech Republic, Qatar',
'Canada, United Kingdom, Netherlands',
'United States, Spain, Germany', 'India, Japan',
'China, South Korea, United States',
'United Kingdom, France, Belgium',
'Canada, Ireland, United States',
'United Kingdom, United States, Dominican Republic',
'United States, Senegal', 'Germany, United Kingdom, United States',
'South Africa, Germany, Netherlands, France',
'Canada, United States, United Kingdom, France, Luxembourg',
'Ireland, United States, France', 'Germany, United States, Canada',
'United Kingdom, Germany, Canada, United States',
'United States, France, Canada, Lebanon, Qatar',
'Netherlands, Belgium, United Kingdom, United States',
'France, Belgium, China, United States',
'United States, Chile, Israel',

'United States, Chile, Israel',
'United Kingdom, Norway, Denmark, Germany, Sweden',
'Norway, Denmark, Sweden', 'China, India, Nepal',
'Colombia, Mexico, United States', 'United Kingdom, South Korea',
'Denmark, China', 'United States, Greece, Brazil',
'South Korea, France',
'United States, Australia, Samoa, United Kingdom',
'Germany, United Kingdom', 'Argentina, Chile, Peru',
'Turkey, Azerbaijan', 'Poland, West Germany',
'Germany, United States, Sweden', 'Canada, Spain',
'United States, Cambodia', 'United States, Greece',
'Norway, United Kingdom, France, Ireland',
'United Kingdom, Poland', 'Israel, Sweden, Germany, Netherlands',
'Switzerland, France', 'Italy, India', 'United States, Botswana',
'Chile, Argentina, France, Spain, United States',
'United States, India, South Korea, China',
'Denmark, Germany, Belgium, United Kingdom, France',
'Denmark, Germany, Belgium, United Kingdom, France, Sweden',
'France, Switzerland, Spain, United States, United Arab Emirates',
'Brazil, India, China, United States',
'Denmark, France, United States, Sweden', 'Australia, Iraq',
'China, Morocco, Hong Kong', 'Canada, United States, Germany',
'United Kingdom, Thailand', 'Venezuela, Colombia',
'Colombia, United States',
'France, Germany, Czech Republic, Belgium',
'Switzerland, Vatican City, Italy, Germany, France',
'Portugal, France, Poland, United States',
'United States, New Zealand, Japan',
'United States, Netherlands, Japan, France', 'India, Switzerland',
'Canada, India', 'United States, Morocco',
'Singapore, Japan, France',
'Canada, Mexico, Germany, South Africa',
'United Kingdom, United States, Canada',
'Germany, France, United States, Canada, United Kingdom',
'United States, Uruguay', 'India, Canada',
'Ireland, Canada, United Kingdom, United States',
'United States, Germany, Australia', 'Australia, France, Ireland',
'Australia, India', 'United States, United Kingdom, Canada, Japan',
'Sweden, United Kingdom, Finland', 'Hong Kong, Taiwan',
'United States, United Kingdom, Spain, South Korea', 'Guatemala',
'Ukraine',
'Italy, South Africa, West Germany, Australia, United States',
'United States, Germany, United Kingdom, Australia',
'Italy, France, Switzerland', 'Canada, France, United States',
'Switzerland, United States', 'Thailand, Canada, United States',
'China, Hong Kong, United States', 'United Kingdom, New Zealand',
'Czech Republic, United Kingdom, France',
'Australia, United Kingdom, Canada', 'Jamaica, United States',
'Australia, United Kingdom, United States, New Zealand, Italy, France',
'France, United States, Canada',
'United Kingdom, France, Canada, Belgium, United States',
'Denmark, United Kingdom, Sweden', 'United States, Hong Kong',
'United States, Kazakhstan',
'Argentina, France, United States, Germany, Qatar',
'United States, Germany, United Kingdom',
'United States, Germany, United Kingdom, Italy',
'United States, New Zealand, United Kingdom',
'Finland, United States', 'Spain, France, Uruguay',
'France, Canada, United States', 'United States, Canada, China',
'Ireland, Canada, Luxembourg, United States, United Kingdom, Philippines, India',
'United States, Czech Republic, United Kingdom', 'Israel, Germany',
'Mexico, France',
'Israel, Germany, Poland, Luxembourg, Belgium, France, United States',
'Austria, United States', 'United Kingdom, Lithuania',
'United States, Greece, United Kingdom',
'United Kingdom, China, United States, India',
'United States, Sweden, Norway',
'United Kingdom, United States, Morocco',
'United States, United Kingdom, Morocco',
'Spain, Canada, United States',
'United States, India, United Arab Emirates',
'United Kingdom, Canada, France, United States',

```
'India, Germany, France',
'Belgium, Ireland, Netherlands, Germany, Afghanistan',
'France, Canada, Italy, United States, China',
'Ireland, United Kingdom, Greece, France, Netherlands',
'Denmark, Indonesia, Finland, Norway, United Kingdom, Israel, France, United States, Germany, Netherlands',
'New Zealand, United States',
'United States, Australia, South Africa, United Kingdom',
'United States, Germany, Mexico',
'Somalia, Kenya, Sudan, South Africa, United States',
'United States, Canada, Japan, Panama',
'United Kingdom, Spain, Belgium', 'Serbia, South Korea, Slovenia',
'Denmark, United Kingdom, South Africa, Sweden, Belgium',
'Germany, Canada, United States',
'Ireland, Canada, United States, United Kingdom',
'New Zealand, United Kingdom, Australia',
'United Kingdom, Australia, Canada, United States',
'Germany, United States, Italy', 'United States, Venezuela',
'United Kingdom, Canada, Japan',
'United Kingdom, United States, Czech Republic',
'United Kingdom, China, United States',
'United Kingdom, Brazil, Germany',
'United Kingdom, Namibia, South Africa, Zimbabwe, United States',
'Canada, United States, India, United Kingdom',
'Switzerland, United Kingdom, United States',
'United Kingdom, India, Sweden',
'United States, Brazil, India, Uganda, China',
'Peru, United States, United Kingdom',
'Germany, United States, United Kingdom, Canada',
'Canada, India, Thailand, United States, United Arab Emirates',
'United States, East Germany, West Germany',
'France, Netherlands, South Africa, Finland',
'Egypt, Austria, United States', 'Russia, Spain',
'Croatia, Slovenia, Serbia, Montenegro', 'Japan, Canada',
'United States, France, South Korea, Indonesia',
'United Arab Emirates, Jordan'], dtype=object)
```

In [21]:

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

Out[21]:

```
array(['September 25, 2021', 'September 24, 2021', 'September 23, 2021',
       ..., 'December 6, 2018', 'March 9, 2016', 'January 11, 2020'],
      dtype=object)
```

In [22]:

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

Out[22]:

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

In [23]:

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

Out[23]:

```
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)
```

In [24]:

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

Out[24]:

```
array(['90 min', '2 Seasons', '1 Season', '91 min', '125 min',
       '9 Seasons', '104 min', '127 min', '4 Seasons', '67 min', '94 min',
       '5 Seasons', '161 min', '61 min', '166 min', '147 min', '103 min',
       '97 min', '106 min', '111 min', '3 Seasons', '110 min', '105 min',
       '96 min', '124 min', '116 min', '98 min', '23 min', '115 min',
       '122 min', '99 min', '88 min', '100 min', '6 Seasons', '102 min',
       '93 min', '95 min', '85 min', '83 min', '113 min', '13 min',
       '182 min', '48 min', '145 min', '87 min', '92 min', '80 min',
       '117 min', '128 min', '119 min', '143 min', '114 min', '118 min',
       '108 min', '63 min', '121 min', '142 min', '154 min', '120 min',
       '82 min', '109 min', '101 min', '86 min', '229 min', '76 min',
       '89 min', '156 min', '112 min', '107 min', '129 min', '135 min',
       '136 min', '165 min', '150 min', '133 min', '70 min', '84 min',
       '140 min', '78 min', '7 Seasons', '64 min', '59 min', '139 min',
       '69 min', '148 min', '189 min', '141 min', '130 min', '138 min',
       '81 min', '132 min', '10 Seasons', '123 min', '65 min', '68 min',
       '66 min', '62 min', '74 min', '131 min', '39 min', '46 min',
       '38 min', '8 Seasons', '17 Seasons', '126 min', '155 min',
       '159 min', '137 min', '12 min', '273 min', '36 min', '34 min',
       '77 min', '60 min', '49 min', '58 min', '72 min', '204 min',
       '212 min', '25 min', '73 min', '29 min', '47 min', '32 min',
       '35 min', '71 min', '149 min', '33 min', '15 min', '54 min',
       '224 min', '162 min', '37 min', '75 min', '79 min', '55 min',
       '158 min', '164 min', '173 min', '181 min', '185 min', '21 min',
       '24 min', '51 min', '151 min', '42 min', '22 min', '134 min',
       '177 min', '13 Seasons', '52 min', '14 min', '53 min', '8 min',
       '57 min', '28 min', '50 min', '9 min', '26 min', '45 min',
       '171 min', '27 min', '44 min', '146 min', '20 min', '157 min',
       '17 min', '203 min', '41 min', '30 min', '194 min', '15 Seasons',
       '233 min', '237 min', '230 min', '195 min', '253 min', '152 min',
       '190 min', '160 min', '208 min', '180 min', '144 min', '5 min',
       '174 min', '170 min', '192 min', '209 min', '187 min', '172 min',
       '16 min', '186 min', '11 min', '193 min', '176 min', '56 min',
       '169 min', '40 min', '10 min', '3 min', '168 min', '312 min',
       '153 min', '214 min', '31 min', '163 min', '19 min', '12 Seasons',
       nan, '179 min', '11 Seasons', '43 min', '200 min', '196 min',
       '167 min', '178 min', '228 min', '18 min', '205 min', '201 min',
       '191 min'], dtype=object)
```

In [25]:

```
netflix_data["listed_in"].unique()
```

Out[25]:

```
array(['Documentaries', 'International TV Shows, TV Dramas, TV Mysteries',
       'Crime TV Shows, International TV Shows, TV Action & Adventure',
       'Docuseries, Reality TV',
       'International TV Shows, Romantic TV Shows, TV Comedies',
       'TV Dramas, TV Horror, TV Mysteries', 'Children & Family Movies',
       'Dramas, Independent Movies, International Movies',
       'British TV Shows, Reality TV', 'Comedies, Dramas',
       'Crime TV Shows, Docuseries, International TV Shows',
       'Dramas, International Movies',
       'Children & Family Movies, Comedies',
       'British TV Shows, Crime TV Shows, Docuseries',
       'TV Comedies, TV Dramas', 'Documentaries, International Movies',
       'Crime TV Shows, Spanish-Language TV Shows, TV Dramas',
       'Thrillers',
       'International TV Shows, Spanish-Language TV Shows, TV Action & Adventure',
       'International TV Shows, TV Action & Adventure, TV Dramas',
       'Comedies, International Movies',
       'Comedies, International Movies, Romantic Movies',
       'Docuseries, International TV Shows, Reality TV',
       'Comedies, International Movies, Music & Musicals', 'Comedies',
       'Horror Movies, Sci-Fi & Fantasy', 'TV Comedies',
       'British TV Shows, International TV Shows, TV Comedies',
       'International TV Shows, TV Dramas, TV Thrillers', "Kids' TV",
       'Dramas, International Movies, Thrillers',
       'Action & Adventure, Dramas, International Movies',
```

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

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

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

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

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

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

```
'Children & Family Movies, Faith & Spirituality',
'Classic Movies, Comedies, Sports Movies',
'Comedies, Dramas, Sports Movies',
'Action & Adventure, Romantic Movies, Sci-Fi & Fantasy',
'Classic & Cult TV, TV Sci-Fi & Fantasy',
'Comedies, Cult Movies, LGBTQ Movies',
'Comedies, Horror Movies, Sci-Fi & Fantasy',
'Action & Adventure, Comedies, Horror Movies',
'Classic & Cult TV, Crime TV Shows, TV Dramas',
'Action & Adventure, Documentaries, Sports Movies',
'International Movies, LGBTQ Movies, Romantic Movies',
'Cult Movies, Dramas, Thrillers'], dtype=object)
```

After checking each column data from the dataset provided by Netflix here are some insights of my understanding.

> **Understanding each column data / attributes and the information in it.**
>
> - ***show_id:*** **This column contains a unique id for each type of data which is either a movie or a TV show. This is a unique id provided by Netflix team to recognise the content.**
> - ***type:*** **This column defines whether the uploaded title is a Movie or a TV Show.**
> - ***title:*** **This column contains the name of the TV Show or a Movie.**
> - ***director:*** **This column contains the name of the director of the type TV Show or Movie.**
> - ***cast:*** **This column contains the list of name(s) of all or one actor(s) of type Movie or TV Show.**
> - ***country:*** **This column contains the name(s) country (or countries) in which a movie or tv show has released in.**
> - ***date_added:*** **This column contains the date on which a movie or a tv show was added on the platform of Netflix.**
> - ***release_year:*** **This column contains the year in which the movie or tv show had originally released and nothing to do with the date added on Netflix.**
> - ***rating:*** **This column cotains the rating certificate by a particular country's movie board authority, and we can also see that the ratings are different in various coutries.**
> - ***duration:*** **This column contains the runtime of a movie and number of seasons if it is a TV Show.**
> - ***listed_in:*** **This column contains the genre(s) of each movie and tv show that was added on Netflix.**
> - ***description:*** **This column contains basic summary of each movie or tv show.**

In [26]:

```
Total_records_in_data = netflix_data.shape[0]
Total_records_in_data
```

Out[26]:

```
8807
```

In [27]:

```
Total_attributes_in_data = netflix_data.shape[1]
Total_attributes_in_data
```

Out[27]:

```
12
```

In [28]:

```
netflix_data.columns
```

Out[28]:

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

> **In the original dataset provided to us we have a total of 8807 records of Movies and TV shows combined together and each record has a total of 12 attributes.**

3) Cleaning Data

Before cleaning the data we need to understand the main purpose of cleaning the data. Though all the information provided in the Netflix Dataset is very important, we need to make sure that the records are worked upon to create better

understanding through visual or non visual analysis to understand the trends and inclination of business with time.

We shall perform the necessary cleaning steps as follow:

> ***3.1) Split Data from selected columns***
>
> ***3.2) Split the data of one column to multiple columns***
>
> ***3.3) Delete unwanted column***
>
> ***3.4) Find and Impute Null Value***
>
> ***3.5) Converting Data Type of Columns***

3.1) Split Data from Selected Columns:

> The columns like "director", "cast", "country" and "listed_in" have multiple data in one records so we shall further split the data into multiple records.
>
> - "director": Some movies or tv shows have more than 1 director so it would be just replicate the entire record and create a new record for each director separately.
> - "cast": Some movies or tv shows have more than 1 actor so creating a separate record for each actor by just replicating other column data will help us the data of each actor separately later in the analysis.
> - "country": A few movies or tv shows were released in more than one country so having a separate record for each country would be good.
> - "listed_in": A movie or tv show can have more than one genre so splitting them all into multiple records will be a better help in the analysis.

So now that we defined the columns with multiple data let's go ahead and split them into mulitple records.

In [30]:

```
netflix_data = netflix_data.assign(director = netflix_data["director"].str.split(", ")).explode("director")
netflix_data = netflix_data.assign(cast = netflix_data["cast"].str.split(", ")).explode("cast")
netflix_data = netflix_data.assign(country = netflix_data["country"].str.split(", ")).explode("country")
netflix_data = netflix_data.assign(listed_in = netflix_data["listed_in"].str.split(", ")).explode("listed_in")
```

In [31]:

```
netflix_data.head(20)
```

Out[31]:

	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	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Mysteries	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Khosi Ngema	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Khosi Ngema	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Khosi Ngema	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Mysteries	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Gail Mabalane	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Gail Mabalane	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Gail Mabalane	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Mysteries	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Thabang Molaba	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
1	s2	TV Show	Blood & Water	NaN	Thabang Molaba	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Thabang Molaba	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Mysteries	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Dillon Windvogel	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Dillon Windvogel	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Dillon Windvogel	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Mysteries	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Natasha Thahane	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Natasha Thahane	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Natasha Thahane	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Mysteries	After crossing paths at a party, a Cape Town t...
1	s2	TV Show	Blood & Water	NaN	Arno Greeff	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...

In [32]:

```
netflix_data.reset_index(drop = True, inplace = True)
```

In [33]:

```
netflix_data.head(10)
```

Out[33]:

	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	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...
2	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...
3	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Mysteries	After crossing paths at a party, a Cape Town t...
4	s2	TV Show	Blood & Water	NaN	Khosi Ngema	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...
5	s2	TV Show	Blood & Water	NaN	Khosi Ngema	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...
6	s2	TV Show	Blood & Water	NaN	Khosi Ngema	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Mysteries	After crossing paths at a party, a Cape Town t...
7	s2	TV Show	Blood & Water	NaN	Gail Mabalane	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...
8	s2	TV Show	Blood & Water	NaN	Gail Mabalane	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...
9	s2	TV Show	Blood & Water	NaN	Gail Mabalane	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	TV Mysteries	After crossing paths at a party, a Cape Town t...

3.2) Split the data of one column to multiple columns:

As we can see the data in date_added can be split into individual into the year_added_to_netflix and month_added_to_netflix for better analysis. We can also further split the column duration as duration_in_minutes for Movie and no_of_seasons for TV Show where we can use the int values for better analysis.

In [34]:

```
netflix_data.dtypes
```

```
netflix_data.dtypes
```

Out[34]:

```
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
```

As we can observe the dtype of date_added column is Object intead of date we need to first convert the type of column to date first before going further. Also let us convert the type of column duration to string.

In [35]:

```
netflix_data["date_added"] = pd.to_datetime(netflix_data["date_added"])
```

> **Now that we have converted the data type or dtype of the column "date_added", looking at the dates adding two new columns will help us for analysis better. The two new columns that we are about to create are**
>
> - ***month_added_on_netflix*: We shall fill this column with data that holds the name of the month in which a movie or tv show was added on Netflix.**
> - ***year_added_on_netflix*: We shall fill this column with data that holds the year in which a movie or tv show was added on Netflix no matter when the orignal release_year of the movie or tv show is.**

In [37]:

```
netflix_data.insert(7, "month_added_on_netflix", netflix_data["date_added"].dt.month_name())
netflix_data.head(3)
```

Out[37]:

	show_id	type	title	director	cast	country	date_added	month_added_on_netflix	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	September	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	South Africa	2021-09-24	September	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...
2	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	2021-09-24	September	2021	TV-MA	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...

In [38]:

```
netflix_data.insert(8, "year_added_on_netflix", netflix_data["date_added"].dt.year)
netflix_data.head(3)
```

Out[38]:

	show_id	type	title	director	cast	country	date_added	month_added_on_netflix	year_added_on_netflix	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	September	2021.0	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	South Africa	2021-09-24	September	2021.0	2021	TV-MA	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...
2	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	2021-09-24	September	2021.0	2021	TV-MA	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...

> **Now that we have extracted the month and year of the movie or TV show that was added on Netflix. Let's look at the next column to further split and why we need to.**

The duration column consists of runtime in minutes of a movies but it also contains data like number of seasons for a TV Show. So, instead of having both information in one column which can create confusion while trying analysing data, it would be good we can futher split them into two new columns.

The two new columns we are creating now are

- **no_of_seasons (To hold number of seasons of a TV Show)**
- **duration_in_minutes (To hold the runtime of a Movie)**

> **So before splitting let's first make sure the dtype or Data Type of the column is converted to string before we can go to further split it.**

In [36]:

```
netflix_data["duration"] = netflix_data["duration"].astype(str)
```

In [39]:

```
def splitdata(x):
  if x == np.NaN:
    return 0
  else:
    y = x.split()[0]
    return y
```

In [40]:

```
netflix_data.insert(11, "duration_in_minutes", netflix_data[netflix_data["type"]=="Movie"]["duration"].apply(splitdata))
```

In [41]:

```
netflix_data.insert(11, "no_of_seasons", netflix_data[netflix_data["type"]=="TV Show"]["duration"].apply(splitdata))
```

In [42]:

```
netflix_data.head(3)
```

Out[42]:

	show_id	type	title	director	cast	country	date_added	month_added_on_netflix	year_added_on_netflix	release_year	rating	no_of_seasons	duration_in_minutes	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	September	2021.0	2020	PG-13	NaN	90	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	2021-09-24	September	2021.0	2021	TV-MA	2	NaN	2 Seasons	International TV Shows	After crossing paths at a party, a Cape Town t...
2	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	2021-09-24	September	2021.0	2021	TV-MA	2	NaN	2 Seasons	TV Dramas	After crossing paths at a party, a Cape Town t...

3.3) Delete unwanted column:

As we check the data in each column we can see the last column "description" there isn't much we can utlise from the data at this moment. We can also consider to remove the duration column now that we have added the data of duration in minutes and season into separate columns. So we shall delete (drop) the columns.

In [43]:

```
netflix_data.drop(["description","duration"],axis = 1, inplace=True)
netflix_data.head(3)
```

Out[43]:

	show_id	type	title	director	cast	country	date_added	month_added_on_netflix	year_added_on_netflix	release_year	rating	no_of_seasons	duration_in_minutes	listed_in
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	September	2021.0	2020	PG-13	NaN	90	Documentaries
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	2021-09-24	September	2021.0	2021	TV-MA	2	NaN	International TV Shows

2	show_id	TV Show	Blood & Water	direction	Ama Qamata	cast	South Africa	2021-09-24	month_added	September	netflix	year_added_on_netflix	release_year	TV-MA	no_of_seasons	duration_in_minutes	listed_in	TV Dramas, International TV Shows					

3.4) Find and Impute Null (NaN) Values

Analyse the Null Values in various columns

In [44]:

```
no_of_records = netflix_data.shape[0]
no_of_attributes = netflix_data.shape[1]
no_of_records, no_of_attributes
```

Out[44]:

```
(201991, 14)
```

In [45]:

```
netflix_data.isna().sum()
```

Out[45]:

```
show_id                      0
type                         0
title                        0
director                 50643
cast                      2146
country                  11897
date_added                 158
month_added_on_netflix     158
year_added_on_netflix      158
release_year                 0
rating                      67
no_of_seasons           145843
duration_in_minutes      56148
listed_in                    0
dtype: int64
```

From the above data we can analyse that there are a total of 201991 records. The columns with NaN values are

1. **"director"**
2. **"cast"**
3. **"country"**
4. **"date_added"**
5. **"month_added_on_netflix"**
6. **"year_added_on_netflix"**
7. **"rating"**
8. **"no_of_seasons"**
9. **"duration_in_minutes"**

Delete Records:

> **If we check the column with the lowest number of missing values we can see that columns "date_added", "month_added_on_netflix", "year_added_on_netflix" and "rating" have missing values less than 1% of the total records in the netflix_data dataframe. So deleting these records will have negligible affect on analysing the data or making any business decisions.**

In [46]:

```
total_records_after_splitting_data = netflix_data.shape[0]
total_records_after_splitting_data
```

Out[46]:

```
201991
```

201991

In [47]:

```
netflix_data.isna().sum()
```

Out[47]:

```
show_id                      0
type                         0
title                        0
director                 50643
cast                      2146
country                  11897
date_added                 158
month_added_on_netflix     158
year_added_on_netflix      158
release_year                 0
rating                      67
no_of_seasons           145843
duration_in_minutes      56148
listed_in                    0
dtype: int64
```

In [48]:

```
netflix_data.dropna(subset=["date_added","month_added_on_netflix","year_added_on_netflix","rating"], inplace = True)
netflix_data.isnull().sum()
```

Out[48]:

```
show_id                      0
type                         0
title                        0
director                 50425
cast                      2146
country                  11894
date_added                   0
month_added_on_netflix       0
year_added_on_netflix        0
release_year                 0
rating                       0
no_of_seasons           145834
duration_in_minutes      55932
listed_in                    0
dtype: int64
```

Impute/Fill Data

> **We created two new columns "no_of_seasons" and "duration_in_minutes"**

- **The column "no_of_seasons" we created hold the number of seasons of all type column "TV Show". So we can fill the null values in other records with 0 keeping mind the the missing column values the type is Movie and so "no_of_seasons" doesn't make sense to a type Movie.**
- **The column "duration_in_minutes" was provided to us for the type Movie, but when we created this new column all the records where the type is TV Show will show as null in this column so we can fill it with 0 as the minutes section doesn't apply to all records with type TV Show.**

In [49]:

```
netflix_data[["no_of_seasons","duration_in_minutes"]] = netflix_data[["no_of_seasons","duration_in_minutes"]].fillna(0)
```

In [50]:

```
netflix_data.shape[0]
```

Out[50]:

```
201766
```

In [51]:

```
netflix_data.isnull().sum()
```

Out[51]:

```
show_id                     0
type                        0
title                       0
director                50425
cast                     2146
country                 11894
date_added                  0
month_added_on_netflix      0
year_added_on_netflix       0
release_year                0
rating                      0
no_of_seasons               0
duration_in_minutes         0
listed_in                   0
dtype: int64
```

> **Fill null values in "director" column:**

- **As we can see there are a total of 50425 null values in "director" column.**
- **Finding the mode of the directors from overall column and filling the null values isn't the right way to fill as a director might change by year and also by country.**
- **So considering the columns "release_year" and "country" we shall find the mode of director and then fill the null values with the best possible mode value.**

In [52]:

```
for i in netflix_data[(netflix_data["director"].isnull())]["release_year"].unique():
  for j in netflix_data[(netflix_data["director"].isnull())]["country"].unique():
    if j != np.NaN:
      director_mode = netflix_data[(netflix_data["release_year"]==i) & (netflix_data["country"]==j)]["director"].mode()
      if len(director_mode) != 0:
        netflix_data.loc[(netflix_data["release_year"]==i) & (netflix_data["country"]==j), "director"] = netflix_data.loc[(netflix_data["release_year"]==i) & (netflix_data["country"]==j),
"director"].fillna(director_mode[0])
```

In [53]:

```
netflix_data.isna().sum()
```

Out[53]:

```
show_id                     0
type                        0
title                       0
director                 7700
cast                     2146
country                 11894
date_added                  0
month_added_on_netflix      0
year_added_on_netflix       0
release_year                0
rating                      0
no_of_seasons               0
duration_in_minutes         0
listed_in                   0
dtype: int64
```

> **The remaining records with null values are 7700 under "director" column. So we shall fill it with the value as "Unknown Director" to make sure we don't just fill any random director name.**

In [54]:

```
netflix_data["director"].fillna("Unknown Director", inplace= True)
```

In [55]:

```
netflix_data.isna().sum()
```

Out[55]:

```
show_id                     0
type                        0
title                       0
director                    0
cast                     2146
country                 11894
date_added                  0
month_added_on_netflix      0
year_added_on_netflix       0
release_year                0
rating                      0
no_of_seasons               0
duration_in_minutes         0
listed_in                   0
dtype: int64
```

> **Fill null values in "country" column:**

- **As we can see there are a total of 11,894 null values in the column.**
- **Finding a mode of the entire column of country and filling won't be the right way to impute the null values.**
- **We shall consider the combination of two columns "director" and "release_year" to find the mode of country for better analysis and cleaner data and better update of null values.**

In [56]:

```
for i in netflix_data[(netflix_data["country"].isnull())]["director"].unique():
    for j in netflix_data[(netflix_data["country"].isnull())]["release_year"].unique():
      country_mode = netflix_data[(netflix_data["director"]==i) & (netflix_data["release_year"]==j)]["country"].mode()
      if len(country_mode) != 0:
        netflix_data.loc[(netflix_data["director"]==i) & (netflix_data["release_year"]==j), "country"] = netflix_data.loc[(netflix_data["director"]==i) & (netflix_data["release_year"]==j)
, "country"].fillna(country_mode[0])
```

In [57]:

```
netflix_data.isna().sum()
```

Out[57]:

```
show_id                     0
type                        0
title                       0
director                    0
cast                     2146
country                  6110
date_added                  0
month_added_on_netflix      0
year_added_on_netflix       0
release_year                0
rating                      0
no_of_seasons               0
duration_in_minutes         0
listed_in                   0
dtype: int64
```

> **The remaining records with null values are 6100 under "country" column. So we shall fill it with the value as "Unknown Country" to make sure we don't just fill any random country name.**

In [58]:

```
netflix_data["country"].fillna("Unknown Country", inplace= True)
```

In [59]:

```
netflix_data.isna().sum()
```

Out[59]:

```
show_id                    0
type                       0
title                      0
director                   0
cast                    2146
country                    0
date_added                 0
month_added_on_netflix     0
year_added_on_netflix      0
release_year               0
rating                     0
no_of_seasons              0
duration_in_minutes        0
listed_in                  0
dtype: int64
```

> **Fill null values in "cast" column:**

- **As we can see there are a total of 2146 null values in "director" column.**
- **Finding the mode of the cast from overall column and filling the null values isn't the right way to fill as a cast might change by year and also by country.**
- **So considering the columns "release_year" and "country" we shall find the mode of cast and then fill the null values with the best possible mode value.**

In [60]:

```
for i in netflix_data[(netflix_data["cast"].isnull())]["country"].unique():
  if i != "Unknown Country":
    for j in netflix_data[(netflix_data["cast"].isnull())]["release_year"].unique():
      cast_mode = netflix_data[(netflix_data["country"]==i) & (netflix_data["release_year"]==j)]["cast"].mode()
      if len(cast_mode) != 0:
        netflix_data.loc[(netflix_data["country"]==i) & (netflix_data["release_year"]==j), "cast"] = netflix_data.loc[(netflix_data["country"]==i) & (netflix_data["release_year"]==j), "ca
st"].fillna(cast_mode[0])
```

In [61]:

```
netflix_data.isnull().sum()
```

Out[61]:

```
show_id                    0
type                       0
title                      0
director                   0
cast                     216
country                    0
date_added                 0
month_added_on_netflix     0
year_added_on_netflix      0
release_year               0
rating                     0
no_of_seasons              0
duration_in_minutes        0
listed_in                  0
dtype: int64
```

> **Now we are left with 216 null values under cast, so we can fill those recods with "Unknown Cast" as it is less than 1% of the entire data.**

In [62]:

```
netflix_data["cast"].fillna("Unknown Cast", inplace= True)
```

In [63]:

```
#Final check of Null values in each column
netflix_data.isnull().sum()
```

Out[63]:

```
show_id                    0
type                       0
title                      0
director                   0
cast                       0
country                    0
date_added                 0
month_added_on_netflix     0
year_added_on_netflix      0
release_year               0
rating                     0
no_of_seasons              0
duration_in_minutes        0
listed_in                  0
dtype: int64
```

3.5) Converting Data Type of Columns:

Most of the columns are of type *Object*. But, type Object can be anything between a String, *Unicode* or even a mixed type. So it's better we convert it into type *String*.

Image for Reference of Type Object:

Pandas dtype	Python type	NumPy type	Usage
object	str or mixed	string_, unicode_, mixed types	Text or mixed numeric and non-numeric values

> **We should also convert a few columns from type object to type int for better understanding of data.**

In [64]:

```
netflix_data.dtypes
```

Out[64]:

```
show_id                           object
type                              object
title                             object
director                          object
cast                              object
country                           object
date_added                datetime64[ns]
month_added_on_netflix            object
year_added_on_netflix            float64
release_year                       int64
rating                            object
no_of_seasons                     object
duration_in_minutes               object
listed_in                         object
dtype: object
```

> **There seems to be some values in column "duration_in_minutes" as "nan". So let's replace it with a value 0**

In [65]:

```
netflix_data[netflix_data["duration_in_minutes"] == "nan"]
```

Out[65]:

	show_id	type	title	director	cast	country	date_added	month_added_on_netflix	year_added_on_netflix	release_year	rating	no_of_seasons	duration_in_minutes	listed_in
126537	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	2017-04-04	April	2017.0	2017	74 min	0	nan	Movies
131603	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	2016-09-16	September	2016.0	2010	84 min	0	nan	Movies
131737	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	2016-08-15	August	2016.0	2015	66 min	0	nan	Movies

In [66]:

```
netflix_data.loc[netflix_data["duration_in_minutes"] == "nan","duration_in_minutes"] = 0
```

In [67]:

```
netflix_data = netflix_data.astype({"show_id": str, "type": str, "title": str, "director": str, "cast": str, "country": str,
                                    "month_added_on_netflix": str, "year_added_on_netflix": int, "rating": str,
                                    "no_of_seasons": int, "duration_in_minutes": int, "listed_in": str})
```

In [68]:

```
netflix_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 201766 entries, 0 to 201990
Data columns (total 14 columns):
 #   Column                  Non-Null Count   Dtype         
---  ------                  --------------   -----         
 0   show_id                 201766 non-null  object        
 1   type                    201766 non-null  object        
 2   title                   201766 non-null  object        
 3   director                201766 non-null  object        
 4   cast                    201766 non-null  object        
 5   country                 201766 non-null  object        
 6   date_added              201766 non-null  datetime64[ns]
 7   month_added_on_netflix  201766 non-null  object        
 8   year_added_on_netflix   201766 non-null  int64         
 9   release_year            201766 non-null  int64         
 10  rating                  201766 non-null  object        
 11  no_of_seasons           201766 non-null  int64         
 12  duration_in_minutes     201766 non-null  int64         
 13  listed_in               201766 non-null  object        
dtypes: datetime64[ns](1), int64(4), object(9)
memory usage: 23.1+ MB
```

4) Non-Graphical Analysis

In [69]:

```
#Entire Netflix Data
netflix_data
```

Out[69]:

	show_id	type	title	director	cast	country	date_added	month_added_on_netflix	year_added_on_netflix	release_year	rating	no_of_seasons	duration_in_minutes	listed_in
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	Andre Robinson	United States	2021-09-25	September	2021	2020	PG-13	0	90	Documentaries
1	s2	TV Show	Blood & Water	Adze Ugah	Ama Qamata	South Africa	2021-09-24	September	2021	2021	TV-MA	2	0	International TV Shows
2	s2	TV Show	Blood & Water	Adze Ugah	Ama Qamata	South Africa	2021-09-24	September	2021	2021	TV-MA	2	0	TV Dramas
3	s2	TV Show	Blood & Water	Adze Ugah	Ama Qamata	South Africa	2021-09-24	September	2021	2021	TV-MA	2	0	TV Mysteries
4	s2	TV Show	Blood & Water	Adze Ugah	Khosi Ngema	South Africa	2021-09-24	September	2021	2021	TV-MA	2	0	International TV Shows
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
201986	s8807	Movie	Zubaan	Mozez Singh	Anita Shabdish	India	2019-03-02	March	2019	2015	TV-14	0	111	International Movies
201987	s8807	Movie	Zubaan	Mozez Singh	Anita Shabdish	India	2019-03-02	March	2019	2015	TV-14	0	111	Music & Musicals
201988	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	2019-03-02	March	2019	2015	TV-14	0	111	Dramas

	show_id	type	title	director	cast	country	date_added	month_added_on_netflix	year_added_on_netflix	release_year	rating	no_of_seasons	duration_in_minutes	listed_in
201989	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	2019-03-02	March	2019	2015	TV-14	0	111	Music & Musicals

201766 rows × 14 columns

> **So after the cleaning of data and imputing null / missing values the total number of records now is 2,01,766. It has increased as we had created separate records for each director, cast, country and listed_in columns.**

In [70]:

```
#Total number of records in Netflix Data after cleaning and imputing data
total_records = netflix_data.shape[0]
total_records
```

Out[70]:

```
201766
```

> **So in the data cleaning process we had deleted a couple of columns and added 4 new columns so now the total number of attributes as increased to 14**

In [71]:

```
#Each record consists of the following attributes or type of data in it
attributes_in_data = netflix_data.columns
number_of_attributes = len(netflix_data.columns)
print("Number of Attributes:",number_of_attributes,"\nList of the Attributes:\n",attributes_in_data)
```

```
Number of Attributes: 14
List of the Attributes:
 Index(['show id', 'type', 'title', 'director', 'cast', 'country', 'date added',
       'month_added_on_netflix', 'year_added_on_netflix', 'release_year',
       'rating', 'no_of_seasons', 'duration_in_minutes', 'listed_in'],
      dtype='object')
```

In [72]:

```
#Types of Content on Netflix (This data includes duplicate Titles due to splitting of data into multiple records)
types_of_content = netflix_data["type"].unique()
print("Types of Content:",types_of_content[0],"&", types_of_content[1])
```

```
Types of Content: Movie & TV Show
```

> **Though we know that the total content that was released on Netflix it would make it easier to know how many of those are movie and how many of those are tv shows to make better business decisions for future.**

In [73]:

```
#Total count of types of content that were released on Netflix
Total_count_types_of_content = netflix_data.groupby("type")["title"].nunique()
print("Total number of Content release on Netflix:",Total_count_types_of_content.sum())
print("Total number of Movies released on Netflix:",Total_count_types_of_content.values[0])
print("Total number of TV Shows released on Netflix:",Total_count_types_of_content.values[1])
```

```
Total number of Content release on Netflix: 8793
Total number of Movies released on Netflix: 6129
Total number of TV Shows released on Netflix: 2664
```

> **When did Netflix start adding content on their platform?**

In [74]:

In [74]:

```
#First Content Release Year
first_content_release_year_on_netflix = netflix_data[netflix_data["year_added_on_netflix"] == min(netflix_data["year_added_on_netflix"])]["year_added_on_netflix"]
print("Year when first content was added on Netflix:",first_content_release_year_on_netflix.values[0])
```

```
Year when first content was added on Netflix: 2008
```

> **When was the last content uploaded as per the dataset?**

In [75]:

```
#Latest Content Release Year
last_content_release_year_on_netflix = netflix_data[netflix_data["year_added_on_netflix"] == max(netflix_data["year_added_on_netflix"])]["year_added_on_netflix"]
print("Year when latest content was added on Netflix:",last_content_release_year_on_netflix.values[0])
```

```
Year when latest content was added on Netflix: 2021
```

> **How much of the content that was uploaded on Netflix is older than the time from when they started uploading in 2008?**
>
> **How many of those are movies and how many of those are TV Shows?**

In [76]:

```
#Total Content that have originally released before the first content added on Netflix
total_count_released_before_netflix = netflix_data[netflix_data["release_year"] < min(netflix_data["year_added_on_netflix"])]["title"].nunique()
total_count_movie_released_before_netflix = netflix_data[(netflix_data["release_year"] < min(netflix_data["year_added_on_netflix"])) & (netflix_data["type"] == "Movie")]["title"].nunique(
)
total_count_tvshow_released_before_netflix = netflix_data[(netflix_data["release_year"] < min(netflix_data["year_added_on_netflix"])) & (netflix_data["type"] == "TV Show")]["title"].nuniqu
e()
print("Total number of content that were released before Netflix started adding content on their platform:", total_count_released_before_netflix)
print("Total number of movies that released before Netflix started adding content on their platform:", total_count_movie_released_before_netflix)
print("Total number of tv shows that released before Netflix started adding content on their platform:", total_count_tvshow_released_before_netflix)
```

```
Total number of content that were released before Netflix started adding content on their platform: 1045
Total number of movies that released before Netflix started adding content on their platform: 922
Total number of tv shows that released before Netflix started adding content on their platform: 123
```

> **How many directors content have been uploaded on Netflix till date?**
>
> **How many of those are Movie directors and how many are TV Show directors?**

In [77]:

```
#Total Unique Directors
total_unique_directors = netflix_data.groupby("type")["director"].nunique().sort_values(ascending = False)
print("Total Overall Unique Directors:",total_unique_directors.sum())
print("Total Unique Movie Directors:",total_unique_directors.values[0])
print("Total Unique TV Show Directors:", total_unique_directors.values[1])
```

```
Total Overall Unique Directors: 5340
Total Unique Movie Directors: 4786
Total Unique TV Show Directors: 554
```

> **How many cast content have been uploaded on Netflix till date?**
>
> **How many of those are Movie cast and how many are TV Show cast?**

In [78]:

```
#Total Unique Cast
total_unique_cast = netflix_data.groupby("type")["cast"].nunique().sort_values(ascending = False)
print("Total Overall Unique Cast:",total_unique_cast.sum())
print("Total Unique Movie Cast:",total_unique_cast.values[0])
print("Total Unique TV Show Cast:", total_unique_cast.values[1])
```

```
Total Overall Unique Cast: 40800
Total Unique Movie Cast: 25964
Total Unique TV Show Cast: 14836
```

> **Who is the top director who has released most movies on Netflix?**

In [79]:

```
#Director with most directed movies released on Netflix
director_with_most_movies = netflix_data[netflix_data["type"] == "Movie"].groupby("director")["title"].nunique().sort_values(ascending = False)[1:2]
print("Name of the top Movie director:",director_with_most_movies.index[0])
print("Number of Movies",director_with_most_movies.index[0], "directed:",director_with_most_movies.values[0])
```

```
Name of the top Movie director: Rajiv Chilaka
Number of Movies Rajiv Chilaka directed: 22
```

> **Who is the top director who has released most tv shows on Netflix?**

In [80]:

```
#Director with most directed TV Shows released on Netflix
director_with_most_movies = netflix_data[netflix_data["type"] == "TV Show"].groupby("director")["title"].nunique().sort_values(ascending = False)[1:2]
print("Name of the top TV Show director:",director_with_most_movies.index[0])
print("Number of TV Shows",director_with_most_movies.index[0], "directed:",director_with_most_movies.values[0])
```

```
Name of the top TV Show director: Greg Rankin
Number of TV Shows Greg Rankin directed: 174
```

> **Which movie cast has got the highest number of content uploaded on Netflix?**

In [81]:

```
#Cast with most movies acted in that released on Netflix
cast_with_most_movies = netflix_data[netflix_data["type"] == "Movie"].groupby("cast")["title"].nunique().sort_values(ascending = False)
print("Name of the top Movie Actor:",cast_with_most_movies.index[0])
print(cast_with_most_movies.index[0], "acted in",cast_with_most_movies.values[0],"Movies.")
```

```
Name of the top Movie Actor: James Franco
James Franco acted in 88 Movies.
```

> **Which tv show cast has got the highest number of content uploaded on Netflix?**

In [82]:

```
#Cast with most TV Shows acted in that released on Netflix
cast_with_most_tvshows = netflix_data[netflix_data["type"] == "TV Show"].groupby("cast")["title"].nunique().sort_values(ascending = False)
print("Name of the top TV Show Actor:",cast_with_most_tvshows.index[0])
print(cast_with_most_tvshows.index[0], "acted in",cast_with_most_tvshows.values[0],"TV Shows.")
```

```
Name of the top TV Show Actor: Andre Robinson
Andre Robinson acted in 43 TV Shows.
```

> **In which year did Netflix upload highest number of Movies on their platform?**
>
> **How many movies did they upload in that year?**

In [83]:

```
#Year with the most released Movies on Netflix
year_with_most_movies_released_on_netflix = netflix_data[netflix_data["type"] == "Movie"].groupby("year_added_on_netflix")["title"].nunique().sort_values(ascending = False)
print("Year with most Movie releases:", year_with_most_movies_released_on_netflix.index[0])
print("A total of",year_with_most_movies_released_on_netflix.values[0],"movies released in the year of", year_with_most_movies_released_on_netflix.index[0] )
```

```
Year with most Movie releases: 2019
A total of 1424 movies released in the year of 2019
```

> **In which month did Netflix upload highest number of Movies on their platform?**
>
> **How many movies did they upload in that month?**

In [85]:

```
#Month with the most released Movies on Netflix
month_with_most_movies_released_on_netflix = netflix_data[netflix_data["type"] == "Movie"].groupby("month_added_on_netflix")["title"].nunique().sort_values(ascending = False)
print("Month with most Movie releases:", month_with_most_movies_released_on_netflix.index[0])
print("A total of",month_with_most_movies_released_on_netflix.values[0], "Movies released on netflix in the month of",month_with_most_movies_released_on_netflix.index[0],"in the period of"
,last_content_release_year_on_netflix.values[0]-first_content_release_year_on_netflix.values[0],"years, since Netflix has been adding content on it's platform.")
```

```
Month with most Movie releases: July
A total of 565 Movies released on netflix in the month of July in the period of 13 years, since Netflix has been adding content on it's platform.
```

> **What is the oldest movie to have gotten released on Netflix and when was it originally release and the date on which it was uploaded on netflix?**

In [89]:

```
#Oldest Movie released on Netlfix
old_movie_name_on_netflix = netflix_data[(netflix_data["release_year"] == min(netflix_data[netflix_data["type"] == "Movie"]["release_year"])) & (netflix_data["type"] == "Movie")]["title"]
old_movie_dateadded_on_netflix = netflix_data[(netflix_data["release_year"] == min(netflix_data[netflix_data["type"] == "Movie"]["release_year"])) & (netflix_data["type"] == "Movie")]["da
te_added"].dt.date
old_movie_releaseyear_on_netflix = netflix_data[(netflix_data["release_year"] == min(netflix_data[netflix_data["type"] == "Movie"]["release_year"])) & (netflix_data["type"] == "Movie")]["
release_year"]
print("Oldest Movie to get released on Netflix:", old_movie_name_on_netflix.values[0])
print("Date on which", old_movie_name_on_netflix.values[0], "was added on netflix", old_movie_dateadded_on_netflix.values[0])
print(old_movie_name_on_netflix.values[0],"was orginally released in the year:", old_movie_releaseyear_on_netflix.values[0])
```

```
Oldest Movie to get released on Netflix: Prelude to War
Date on which Prelude to War was added on netflix 2017-03-31
Prelude to War was orginally released in the year: 1942
```

> **What is the latest movie to have gotten released on Netflix and when was it originally release and the date on which it was uploaded on netflix?**

In [90]:

```
#Latest Movie released on Netlfix
last_movie_name_on_netflix = netflix_data[(netflix_data["release_year"] == max(netflix_data[netflix_data["type"] == "Movie"]["release_year"])) & (netflix_data["type"] == "Movie")]["title"
]
last_movie_dateadded_on_netflix = netflix_data[(netflix_data["release_year"] == max(netflix_data[netflix_data["type"] == "Movie"]["release_year"])) & (netflix_data["type"] == "Movie")]["d
ate_added"].dt.date
last_movie_releaseyear_on_netflix = netflix_data[(netflix_data["release_year"] == max(netflix_data[netflix_data["type"] == "Movie"]["release_year"])) & (netflix_data["type"] == "Movie")][
"release_year"]
```

```
print("Oldest Movie to get released on Netflix:", last_movie_name_on_netflix.values[0])
print("Date on which", last_movie_name_on_netflix.values[0], "was added on netflix", last_movie_dateadded_on_netflix.values[0])
print(last_movie_name_on_netflix.values[0],"was orginally released in the year:", last_movie_releaseyear_on_netflix.values[0])
```

```
Oldest Movie to get released on Netflix: My Little Pony: A New Generation
Date on which My Little Pony: A New Generation was added on netflix 2021-09-24
My Little Pony: A New Generation was orginally released in the year: 2021
```

> **In which year did Netflix upload highest number of TV Shows on their platform?**
>
> **How many tv shows did they upload in that year?**

In [84]:

```
#Year with the most released TV Shows on Netflix
year_with_most_tvshows_released_on_netflix = netflix_data[netflix_data["type"] == "TV Show"].groupby("year_added_on_netflix")["title"].nunique().sort_values(ascending = False)
print("Year with most TV Show releases:", year_with_most_tvshows_released_on_netflix.index[0])
print("A total of", year_with_most_tvshows_released_on_netflix.values[0],"TV shows released in the year of",year_with_most_tvshows_released_on_netflix.index[0])
```

```
Year with most TV Show releases: 2020
A total of 595 TV shows released in the year of 2020
```

> **In which month did Netflix upload highest number of TV Shows on their platform?**
>
> **How many tv shows did they upload in that month?**

In [86]:

```
#Month with the most released TV Shows on Netflix
month_with_most_tvshows_released_on_netflix = netflix_data[netflix_data["type"] == "TV Show"].groupby("month_added_on_netflix")["title"].nunique().sort_values(ascending = False)
print("Year with most TV Show releases:", month_with_most_tvshows_released_on_netflix.index[0])
print("A total of",month_with_most_tvshows_released_on_netflix.values[0], "Movies released on netflix in the month of",month_with_most_tvshows_released_on_netflix.index[0],"in the period o
f",last_content_release_year_on_netflix.values[0]-first_content_release_year_on_netflix.values[0],"years, since Netflix has been adding content on it's platform.")
```

```
Year with most TV Show releases: December
A total of 265 Movies released on netflix in the month of December in the period of 13 years, since Netflix has been adding content on it's platform.
```

> **What is the oldest tv show to have gotten released on Netflix and when was it originally release and the date on which it was uploaded on netflix?**

In [91]:

```
#Oldest TV Show released on Netlfix
old_tvshow_name_on_netflix = netflix_data[(netflix_data["release_year"] == min(netflix_data[netflix_data["type"] == "TV Show"]["release_year"])) & (netflix_data["type"] == "TV Show")]["ti
tle"]
old_tvshow_dateadded_on_netflix = netflix_data[(netflix_data["release_year"] == min(netflix_data[netflix_data["type"] == "TV Show"]["release_year"])) & (netflix_data["type"] == "TV Show")
]["date_added"].dt.date
old_tvshow_releaseyear_on_netflix = netflix_data[(netflix_data["release_year"] == min(netflix_data[netflix_data["type"] == "TV Show"]["release_year"])) & (netflix_data["type"] == "TV Show"
)]["release_year"]
print("Oldest TV Show to get released on Netflix:", old_tvshow_name_on_netflix.values[0])
print("Date on which", old_tvshow_name_on_netflix.values[0], "was added on netflix", old_tvshow_dateadded_on_netflix.values[0])
print(old_tvshow_name_on_netflix.values[0],"was orginally released in the year:", old_tvshow_releaseyear_on_netflix.values[0])
```

```
Oldest TV Show to get released on Netflix: Pioneers: First Women Filmmakers*
Date on which Pioneers: First Women Filmmakers* was added on netflix 2018-12-30
Pioneers: First Women Filmmakers* was orginally released in the year: 1925
```

> **What is the latest tv show to have gotten released on Netflix and when was it originally release and the date on which it was uploaded on netflix?**

In [92]:

```
#Latest TV Show released on Netlfix
last_tvshow_name_on_netflix = netflix_data[(netflix_data["release_year"] == max(netflix_data[netflix_data["type"] == "TV Show"]["release_year"])) & (netflix_data["type"] == "TV Show")]["title"]
last_tvshow_dateadded_on_netflix = netflix_data[(netflix_data["release_year"] == max(netflix_data[netflix_data["type"] == "TV Show"]["release_year"])) & (netflix_data["type"] == "TV Show")]["date_added"].dt.date
last_tvshow_releaseyear_on_netflix = netflix_data[(netflix_data["release_year"] == max(netflix_data[netflix_data["type"] == "TV Show"]["release_year"])) & (netflix_data["type"] == "TV Show")]["release_year"]
print("Latest TV Show to get released on Netflix:", last_tvshow_name_on_netflix.values[0])
print("Date on which", last_tvshow_name_on_netflix.values[0], "was added on netflix", last_tvshow_dateadded_on_netflix.values[0])
print(last_tvshow_name_on_netflix.values[0],"was orginally released in the year:", last_tvshow_releaseyear_on_netflix.values[0])
```

```
Latest TV Show to get released on Netflix: Blood & Water
Date on which Blood & Water was added on netflix 2021-09-24
Blood & Water was orginally released in the year: 2021
```

> **What was the first content that was uploaded on Netflix and the date on which it was uploaded?**

In [87]:

```
#Date, Name and Type  of first content release on Netflix
date_of_first_content_added_on_netflix = netflix_data[netflix_data["date_added"] == min(netflix_data["date_added"])]["date_added"].dt.date
name_of_first_content_added_on_netflix = netflix_data[netflix_data["date_added"] == min(netflix_data["date_added"])]["title"]
type_of_first_content_added_on_netflix = netflix_data[netflix_data["date_added"] == min(netflix_data["date_added"])]["type"]
print("Date on which first content was added on Netflix:",date_of_first_content_added_on_netflix.values[0])
print("Name of first content was added on Netflix:",name_of_first_content_added_on_netflix.values[0])
print("Type of first content was added on Netflix:",type_of_first_content_added_on_netflix.values[0])
```

```
Date on which first content was added on Netflix: 2008-01-01
Name of first content was added on Netflix: To and From New York
Type of first content was added on Netflix: Movie
```

> **What was the latest content that was uploaded on Netflix and the date on which it was uploaded?**

In [88]:

```
#Date, Name and Type  of Latest content release on Netflix
date_of_last_content_added_on_netflix = netflix_data[netflix_data["date_added"] == max(netflix_data["date_added"])]["date_added"].dt.date
name_of_last_content_added_on_netflix = netflix_data[netflix_data["date_added"] == max(netflix_data["date_added"])]["title"]
type_of_last_content_added_on_netflix = netflix_data[netflix_data["date_added"] == max(netflix_data["date_added"])]["type"]
print("Date on which latest content was added on Netflix:",date_of_last_content_added_on_netflix.values[0])
print("Name of latest content was added on Netflix:",name_of_last_content_added_on_netflix.values[0])
print("Type of latest content was added on Netflix:",type_of_last_content_added_on_netflix.values[0])
```

```
Date on which latest content was added on Netflix: 2021-09-25
Name of latest content was added on Netflix: Dick Johnson Is Dead
Type of latest content was added on Netflix: Movie
```

> **Top 10 Data**
>
> - **Directors**
> - **Cast**
> - **Countries**
> - **Genre**

In [93]:

```
#Top 10 Overall Directors unique title they directed avoiding Unknown Directors
top_10_directors = netflix_data.groupby("director")["title"].nunique().sort_values(ascending = False)[1:11]
print("Top 10 Directors by Unique Title:\n",top_10_directors)
```

```
Top 10 Directors by Unique Title:
 director
Greg Rankin       186
Noam Murro        179
Danny Cannon      161
Jeremy Grant      103
Akiva Schaffer     86
Kevin Smith        69
Thomas Astruc      60
Eli Roth           43
Garth Davis        42
Toa Fraser         40
Name: title, dtype: int64
```

In [94]:

```
#Top 10 Cast by unique Title
top_10_cast = netflix_data.groupby("cast")["title"].nunique().sort_values(ascending = False)[0:10]
print("Top 10 Cast by Unique Title:\n",top_10_cast)
```

```
Top 10 Cast by Unique Title:
 cast
James Franco          104
Unknown Cast           91
Mahershala Ali         75
Tiffany Haddish        73
Andre Robinson         72
David Spade            52
Fortune Feimster       52
Harley Quinn Smith     48
Anupam Kher            46
Shah Rukh Khan         35
Name: title, dtype: int64
```

In [95]:

```
#Top 10 Countries by both TV Show and Movie releases on Netflix
top_10_country = netflix_data.groupby("country")["title"].nunique().sort_values(ascending = False)[:10]
print("Top 10 Countries by Unique Title:\n",top_10_country)
```

```
Top 10 Countries by Unique Title:
 country
United States      3692
India              1077
United Kingdom      803
Canada              446
France              395
Unknown Country     368
Japan               321
South Korea         232
Spain               232
Germany             226
Name: title, dtype: int64
```

In [96]:

```
#Top 10 Genre by both TV Show and Movie releases on Netflix
top_10_genre = netflix_data.groupby("listed_in")["title"].nunique().sort_values(ascending = False)[:10]
print("Top 10 Genres by Unique Title:\n",top_10_genre)
```

```
Top 10 Genres by Unique Title:
 listed_in
International Movies        2752
Dramas                      2426
Comedies                    1674
International TV Shows      1349
Documentaries                869
Action & Adventure           859
TV Dramas                    762
Independent Movies           756
Children & Family Movies     641
Romantic Movies              616
```

```
Name: title, dtype: int64
```

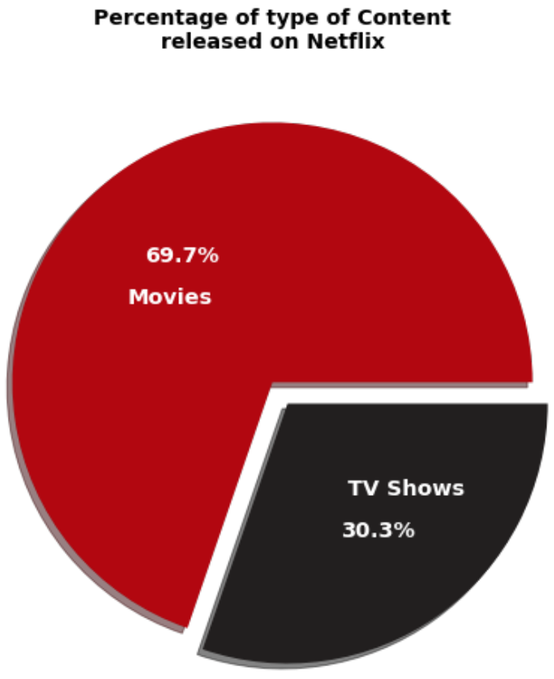
5) Data Visualization

5.1) Pie Chart

- **Type of content released most on Netflix.**

In [97]:

```
plt.figure(figsize = (10,8))
netflix_red = "#b20710"
netflix_grey = "#221f1f"
plt.pie(Total_count_types_of_content.values, labels = ["Movies", "TV Shows"], labeldistance = 0.4,
        shadow = True, explode = (0,0.1), autopct="%.1f%%", colors = [netflix_red,netflix_grey],
        textprops ={"color":"white", "fontweight":"bold","fontsize":"x-large"})
plt.title("Percentage of type of Content\nreleased on Netflix", fontsize = 14, fontweight = "bold")
plt.show()
```

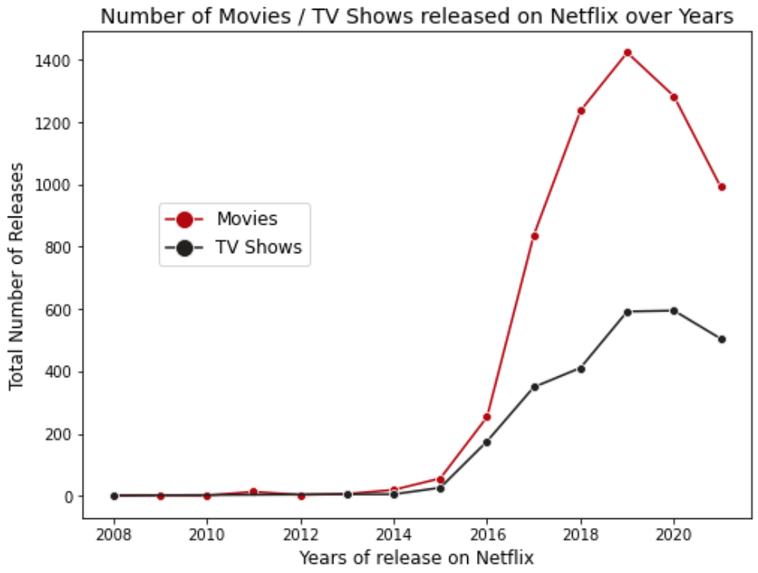


5.2) Line Plot

- **Yearly total releases of Movies / TV Shows.**

In [98]:

```
yearwise_number_of_unique_movies_released = netflix_data[netflix_data["type"] == "Movie"].groupby("year_added_on_netflix")["title"].nunique()
yearwise_number_of_unique_tvshows_released = netflix_data[netflix_data["type"] == "TV Show"].groupby("year_added_on_netflix")["title"].nunique()
plt.figure(figsize = (8,6))
netflix_red = "#b20710"
netflix_grey = "#221f1f"
sns.lineplot(x = yearwise_number_of_unique_movies_released.index, y = yearwise_number_of_unique_movies_released.values, marker="o", color = netflix_red);
sns.lineplot(x = yearwise_number_of_unique_tvshows_released.index, y = yearwise_number_of_unique_tvshows_released.values, marker="o", color = netflix_grey);
plt.xlabel("Years of release on Netflix", fontsize = 12)
plt.ylabel("Total Number of Releases", fontsize = 12)
plt.legend(["Movies","TV Shows"], loc = "lower left", bbox_to_anchor = (0.1, 0.5), markerscale = 2, fontsize=12)
plt.title("Number of Movies / TV Shows released on Netflix over Years", fontsize = 14)
plt.show()
```



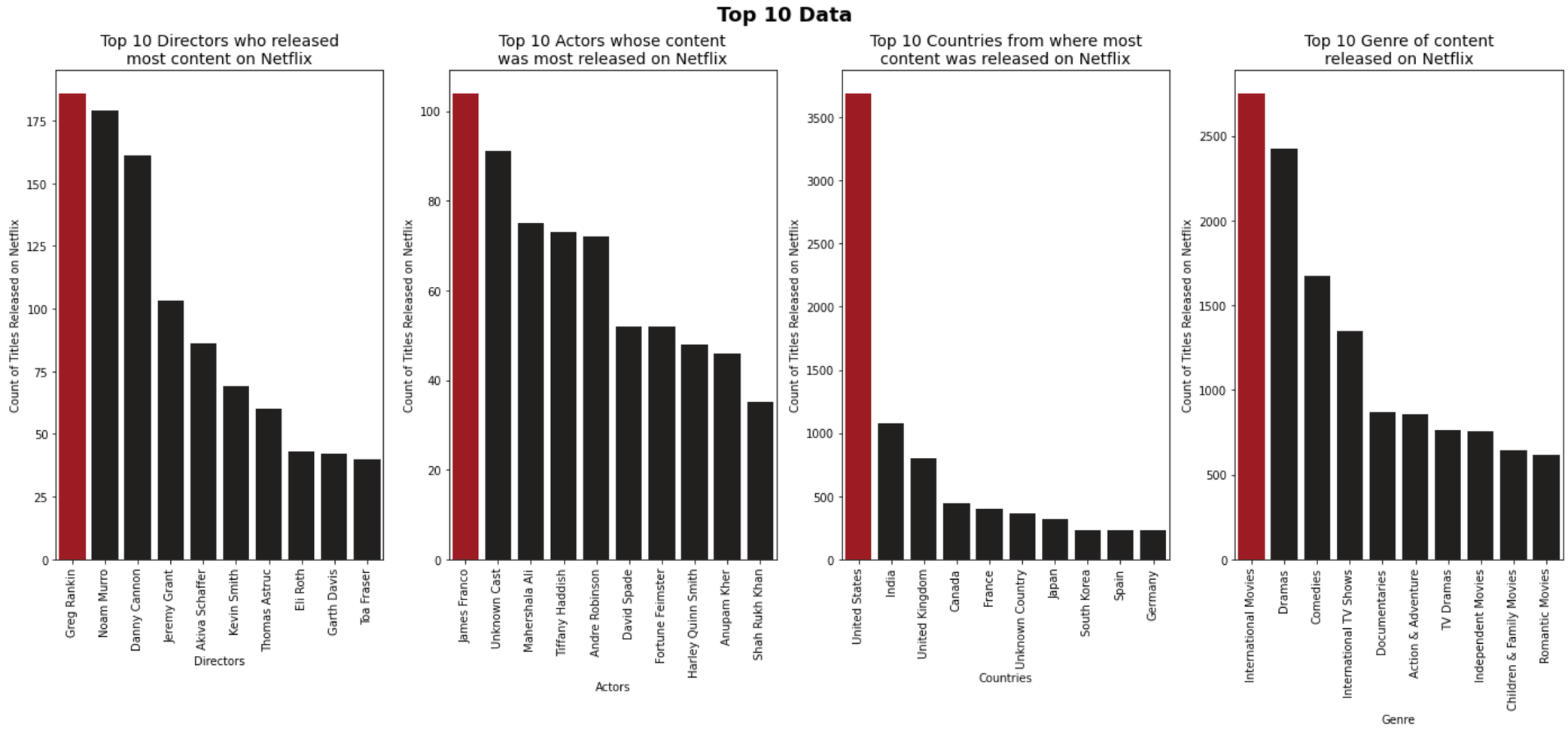
5.3) Barplot: Top 10 Data

- **Top 10 Directors who released most content on Netflix**
- **Top 10 Actors whose content was most released on Netflix**
- **Top 10 Countries from where most content was released on Netflix**
- **Top 10 Genre of content released on Netflix.**

In [99]:

```
plt.figure(figsize = (24,8))
netflix_red = "#b20710"
netflix_grey = "#221f1f"


#Bar Plot for Top 10 Directors
plt.subplot(1,4,1)
top_10_directors_data = pd.DataFrame(top_10_directors).reset_index()
top10directors_cp = [netflix_grey if(x < max(top_10_directors_data["title"])) else netflix_red for x in top_10_directors_data["title"]]
sns.barplot(data = top_10_directors_data, x = "director", y = "title", palette = top10directors_cp )
plt.xticks(rotation = 90)
plt.xlabel("Directors")
plt.title("Top 10 Directors who released\nmost content on Netflix", size = 14)
plt.ylabel("Count of Titles Released on Netflix")

#Bar Plot for Top 10 Actors
plt.subplot(1,4,2)
top_10_actors_data = pd.DataFrame(top_10_cast).reset_index()
top10actors_cp= [netflix_grey if(x < max(top_10_actors_data["title"])) else netflix_red for x in top_10_actors_data["title"]]
sns.barplot(data = top_10_actors_data, x = "cast", y = "title", palette = top10actors_cp )
plt.xticks(rotation = 90)
plt.xlabel("Actors")
plt.title("Top 10 Actors whose content\nwas most released on Netflix", size = 14)
plt.ylabel("Count of Titles Released on Netflix")

#Bar Plot for Top 10 Countries
plt.subplot(1,4,3)
top_10_country_data = pd.DataFrame(top_10_country).reset_index()
top10country_cp = [netflix_grey if(x < max(top_10_country_data["title"])) else netflix_red for x in top_10_country_data["title"]]
sns.barplot(data = top_10_country_data, x = "country", y = "title", palette = top10country_cp )
plt.xticks(rotation = 90)
plt.xlabel("Countries")
plt.title("Top 10 Countries from where most\ncontent was released on Netflix", size = 14)
plt.ylabel("Count of Titles Released on Netflix")
```

```
#Bar Plot for Top 10 Genre
plt.subplot(1,4,4)
top_10_genre_data = pd.DataFrame(top_10_genre).reset_index()
top10genre_cp = [netflix_grey if(x < max(top_10_genre_data["title"])) else netflix_red for x in top_10_genre_data["title"]]
sns.barplot(data = top_10_genre_data, x = "listed_in", y = "title", palette = top10genre_cp )
plt.title("Top 10 Genre of content\nreleased on Netflix", size = 14)
plt.xticks(rotation = 90)
plt.xlabel("Genre")
plt.ylabel("Count of Titles Released on Netflix")

plt.suptitle("Top 10 Data", size = 18, fontweight = "bold")
plt.show()
```



5.4) Distplot

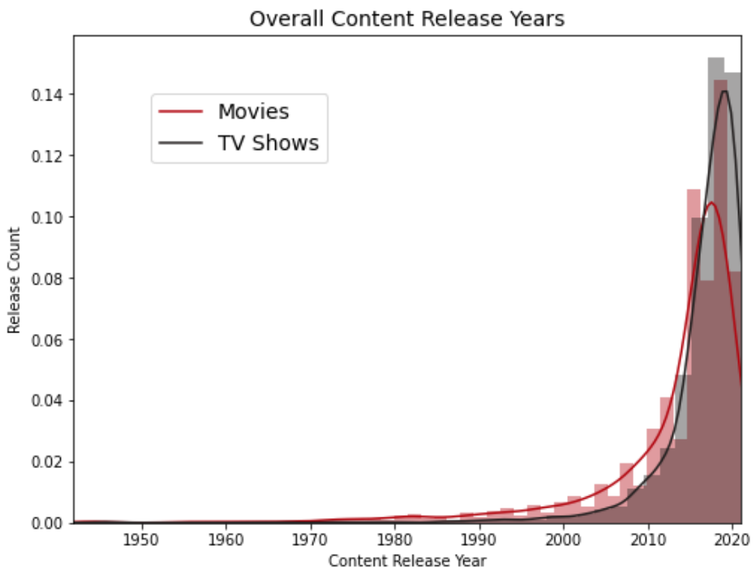
- **Let's check the releases per year for movies**
- **Let's check the releases per year for tv shows**
- **These are general content release years and not netflix release years**

In [140]:

```
movie_title_duplicates_removed_data = netflix_data[netflix_data["type"]=="Movie"].drop_duplicates(subset="title")
tvshow_title_duplicates_removed_data = netflix_data[netflix_data["type"]=="TV Show"].drop_duplicates(subset="title")
max_year = [max(tvshow_title_duplicates_removed_data["release_year"]) if max(tvshow_title_duplicates_removed_data["release_year"]) > max(movie_title_duplicates_removed_data["release_year"]
) else max(movie_title_duplicates_removed_data["release_year"])]
min_year = [min(tvshow_title_duplicates_removed_data["release_year"]) if min(tvshow_title_duplicates_removed_data["release_year"]) > min(movie_title_duplicates_removed_data["release_year"]
) else min(movie_title_duplicates_removed_data["release_year"])]
plt.figure(figsize = (8,6))
netflix_red = "#b20710"
```

```
netflix_grey = "#221f1f"
sns.distplot(movie_title_duplicates_removed_data["release_year"], color = netflix_red, label = "Movies")
sns.distplot(tvshow_title_duplicates_removed_data["release_year"], color = netflix_grey, label = "TV Shows")
plt.xlim(min_year[0],max_year[0])
plt.legend(["Movies","TV Shows"], loc = "upper left", bbox_to_anchor = (0.1, 0.9), fontsize=14)
plt.xlabel("Content Release Year")
plt.ylabel("Release Count")
plt.title("Overall Content Release Years",fontsize=14)
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

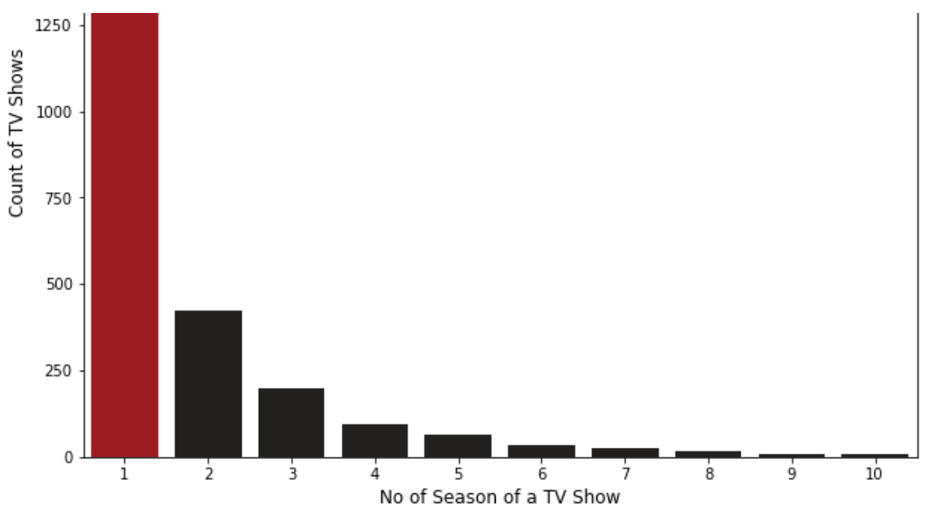


5.5) Countplot

- **Count the number of TV Shows by their number of seasons**

In [101]:

```
plt.figure(figsize = (10,8))
netflix_red = "#b20710"
netflix_grey = "#221f1f"

#Count Plot to count the number of TV Shows by seasons
tvshows_without_duplicate_titles = netflix_data[(netflix_data["no_of_seasons"]!=0) & (netflix_data["type"]=="TV Show") ].drop_duplicates(subset="title")
topseasons_cp = [netflix_grey if(x < tvshows_without_duplicate_titles["no_of_seasons"].value_counts().values[0]) else netflix_red for x in tvshows_without_duplicate_titles["no_of_seasons"].value_counts().values]
sns.countplot(data = tvshows_without_duplicate_titles, x = "no_of_seasons", palette = topseasons_cp)
plt.xlim(-0.5,9.5)
plt.xlabel("No of Season of a TV Show", fontsize = 12)
plt.ylabel("Count of TV Shows", fontsize = 12)

plt.show()
```

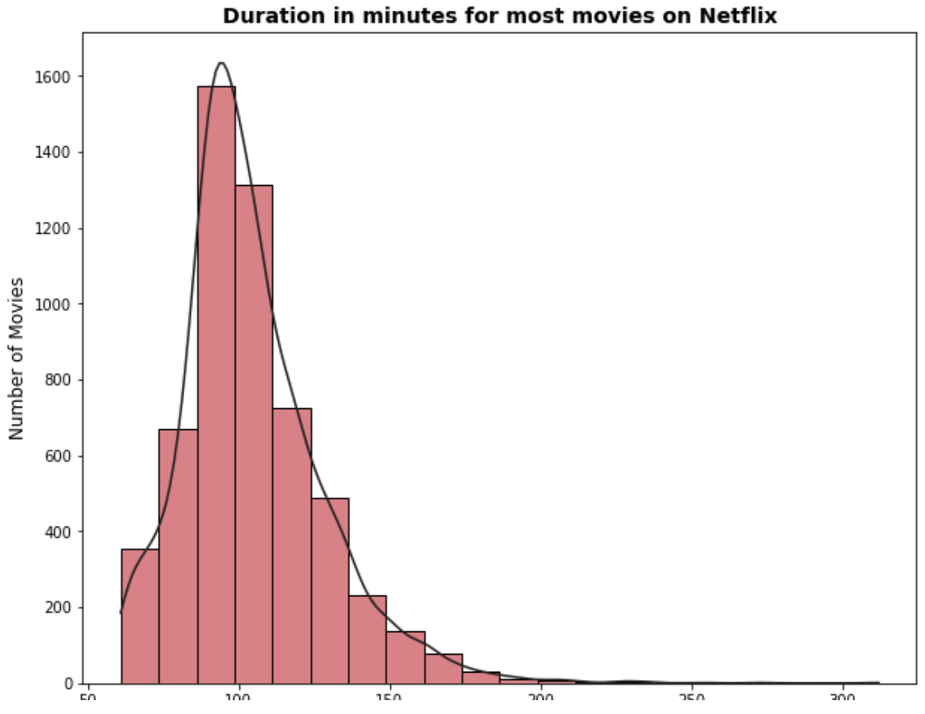


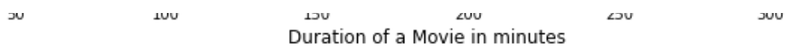
5.6) Histogram

- **Let's find the value of movie duration time to understand which duration movies are most released on Netflix**

In [102]:

```
plt.figure(figsize = (10,8))
netflix_red = "#b20710"
netflix_grey = "#221f1f"

duration_of_movie = netflix_data[(netflix_data["duration_in_minutes"]>60) & (netflix_data["type"] == "Movie")].drop_duplicates(subset = "title")
topduration_cp = [netflix_grey if(x < duration_of_movie["duration_in_minutes"].value_counts().values[0]) else netflix_red for x in duration_of_movie["duration_in_minutes"].value_counts().
values]
line = sns.histplot(data = duration_of_movie, x = "duration_in_minutes", bins = 20, kde = True, color = netflix_red)
line.lines[0].set_color(netflix_grey)
plt.xlabel("Duration of a Movie in minutes", fontsize=12)
plt.ylabel("Number of Movies", fontsize=12)

plt.title("Duration in minutes for most movies on Netflix", fontsize = 14, fontweight = "bold")
plt.show()
```



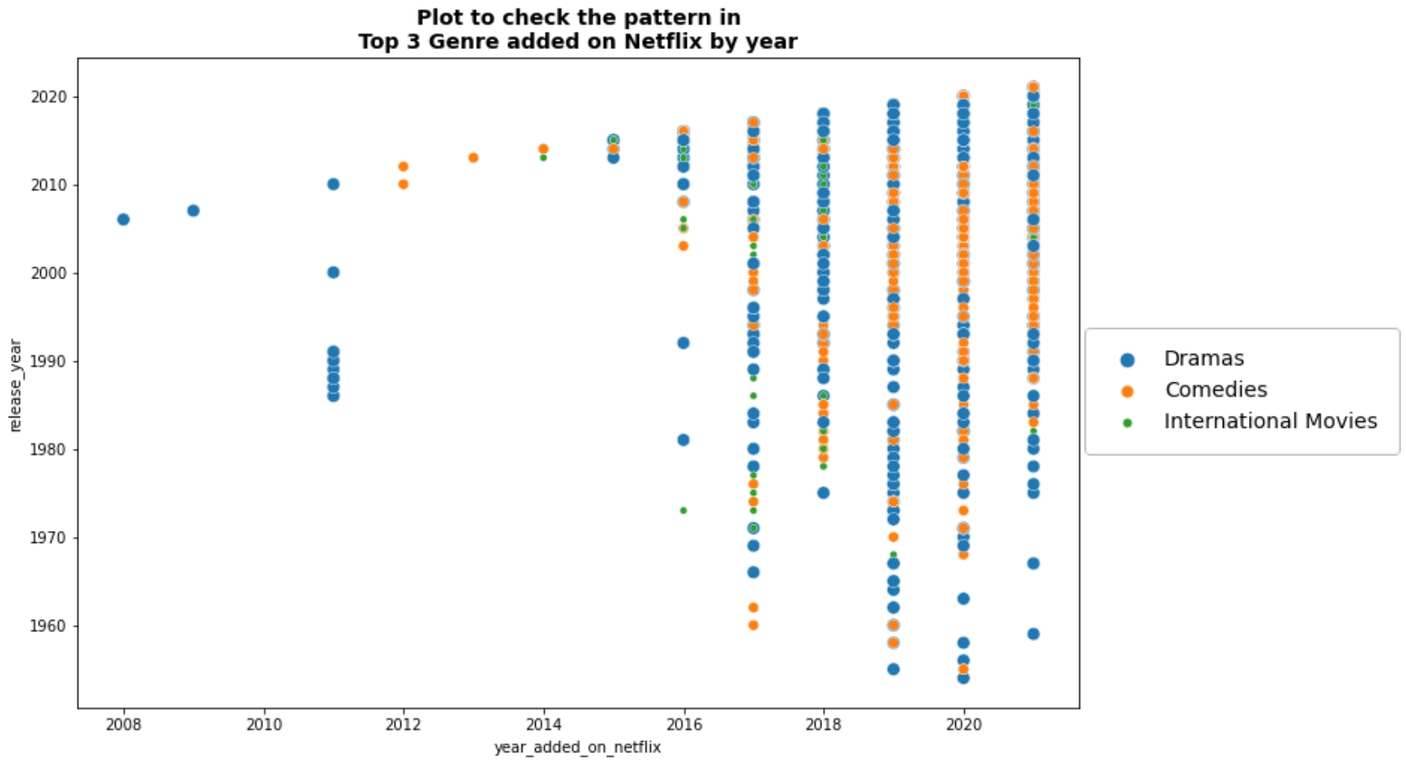


5.7) Scatter Plot

- **Check the top 3 genre release pattern by year on Netflix**

In [103]:

```
top_3_genre_df = netflix_data[(netflix_data["listed_in"].isin(top_10_genre[:3].index))].drop_duplicates(subset="title")
plt.figure(figsize = (12,8))
netflix_red = "#b20710"
netflix_grey = "#221f1f"
sns.scatterplot(data = top_3_genre_df, x = "year_added_on_netflix", y = "release_year", size = "listed_in", sizes = (25,75), hue="listed_in")
plt.legend(loc = "upper right", bbox_to_anchor = (1.33, 0.6), fontsize=14, borderpad= 1)
plt.title("Plot to check the pattern in\nTop 3 Genre added on Netflix by year", fontsize = 14, fontweight = "bold")
plt.show()
```

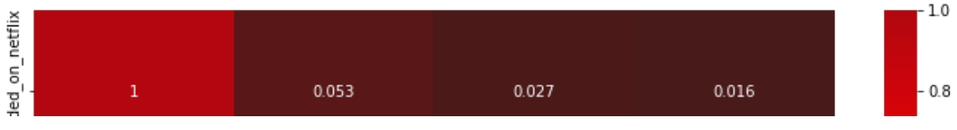


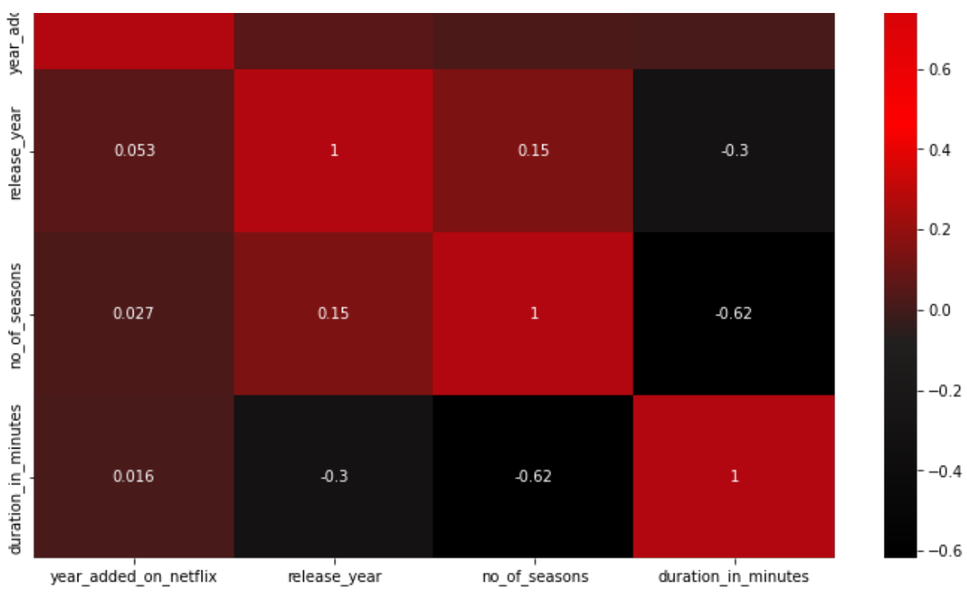
5.8) Correlation Heat Map

This is to understand the immediate % of relation between the immediately related data

In [104]:

```
netflix_data_no_duplicate = netflix_data.drop_duplicates(subset="title")
netflix_correlation = netflix_data_no_duplicate.corr()
plt.figure(figsize = (12,8))
netflix_red = "#b20710"
netflix_grey = "#221f1f"
cmap = clr.LinearSegmentedColormap.from_list('netflix_colors', ["black",netflix_grey,"red",netflix_red])
sns.heatmap(netflix_data.corr(), annot=True, cmap = cmap)
plt.show()
```





5.9) Pair Plot

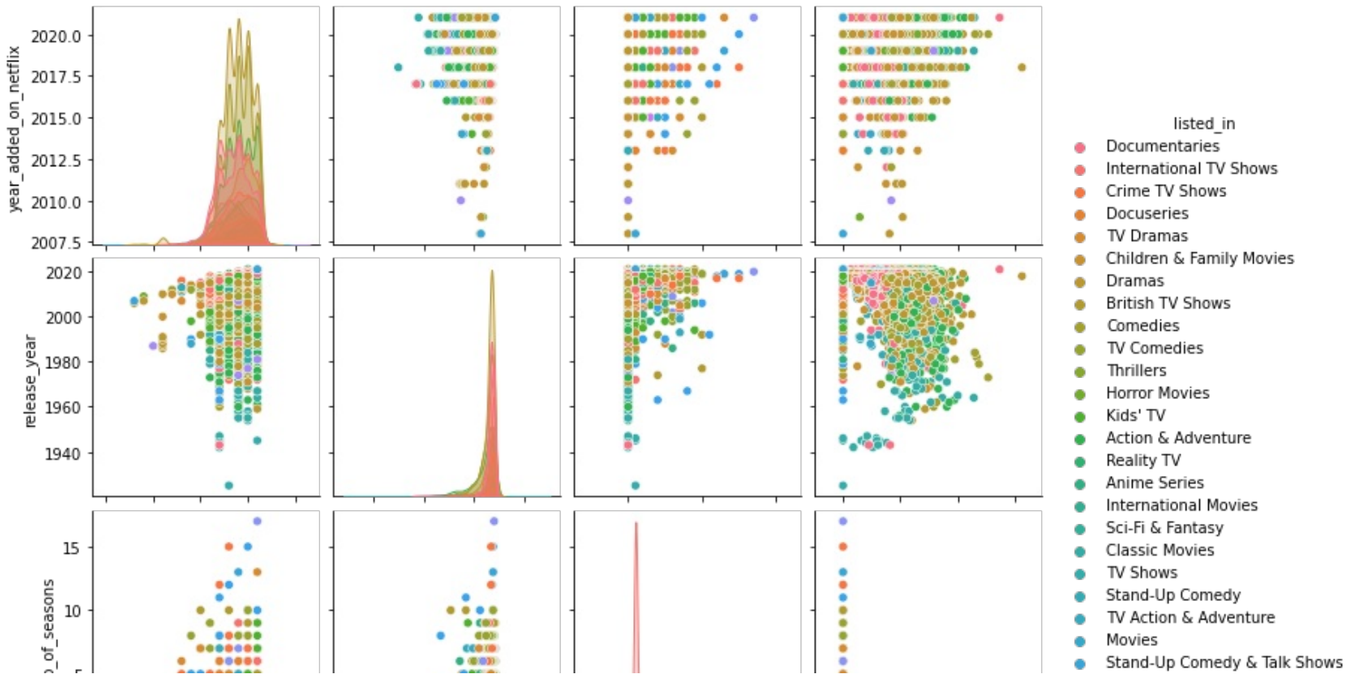
How Genre of both Movies and TV Shows are ploted compared to other columns like

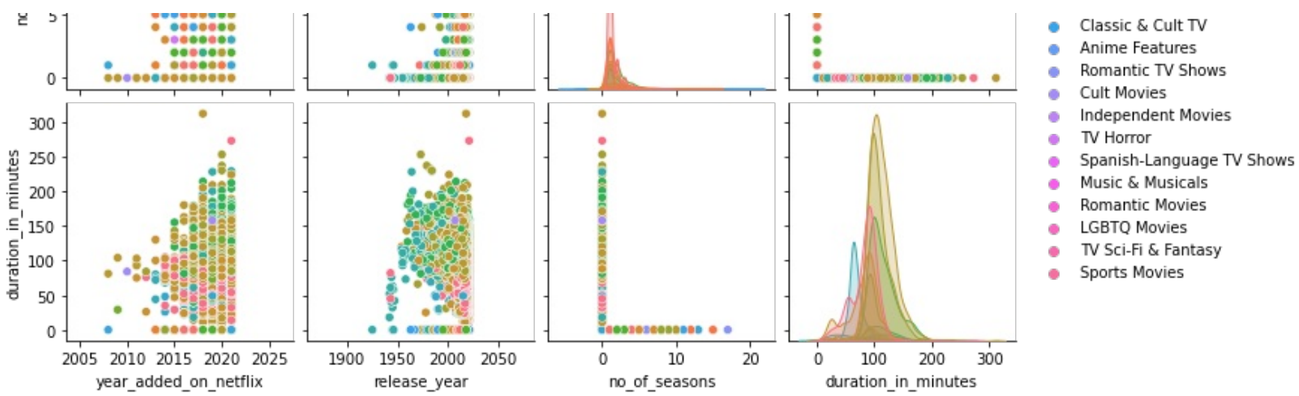
- **Year they were added on netflix**
- **The year in which they originally got released**
- **no of seasons in case of a TV Show genre**
- **duration in minutes in case of a movie genre**

In [105]:

```
plt.figure(figsize=(30,20))
netflix_duplicate_free_data = netflix_data.drop_duplicates(subset = "title")
sns.pairplot(data = netflix_duplicate_free_data, hue = "listed_in")
plt.show()
```

```
<Figure size 2160x1440 with 0 Axes>
```





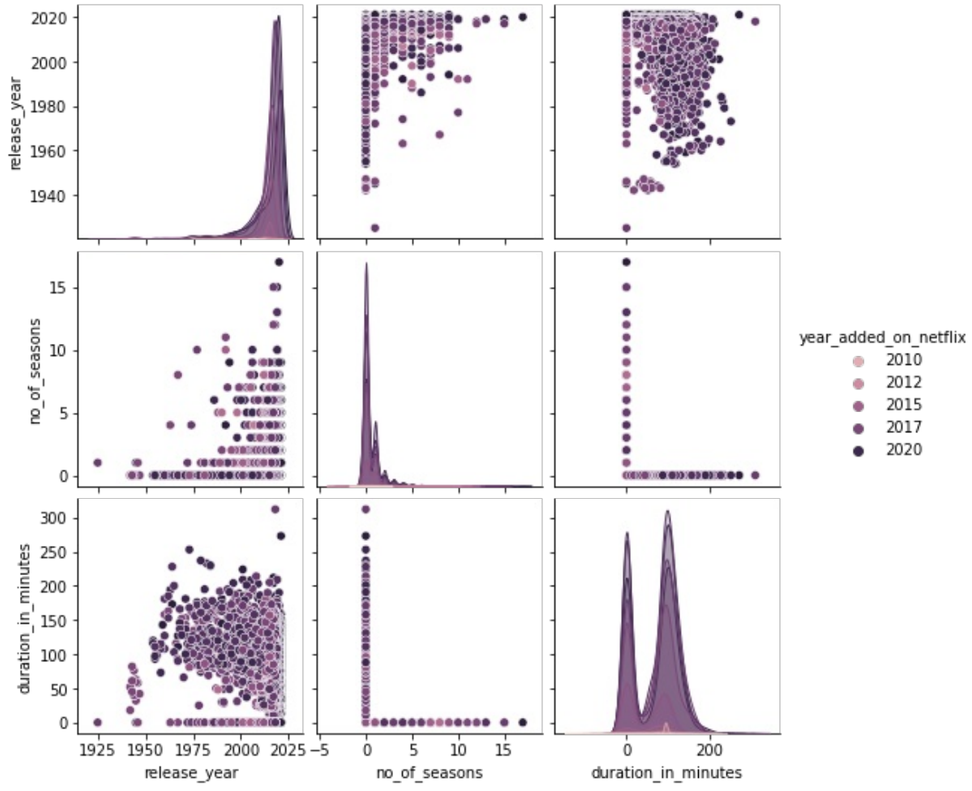
How year in which a Movie and TV Shows were added on Netflix are ploted compared to other columns like

- **The year in which they originally got released**
- **no of seasons in case of a TV Show release year on Netflix**
- **duration in minutes in case of a movie release year on Netflix**

In [106]:

```
plt.figure(figsize=(30,20))
netflix_duplicate_free_data = netflix_data.drop_duplicates(subset = "title")
sns.pairplot(data = netflix_duplicate_free_data, hue = "year_added_on_netflix")
plt.show()
```

```
<Figure size 2160x1440 with 0 Axes>
```

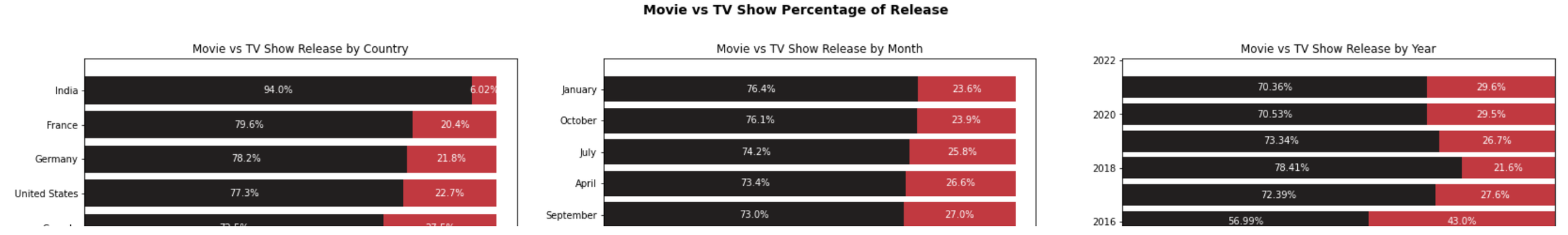


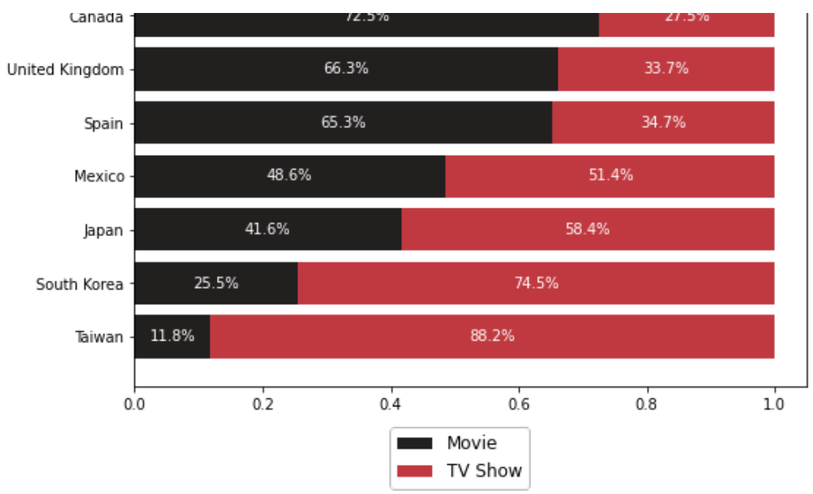
5.10) Bar Plot Horizontal

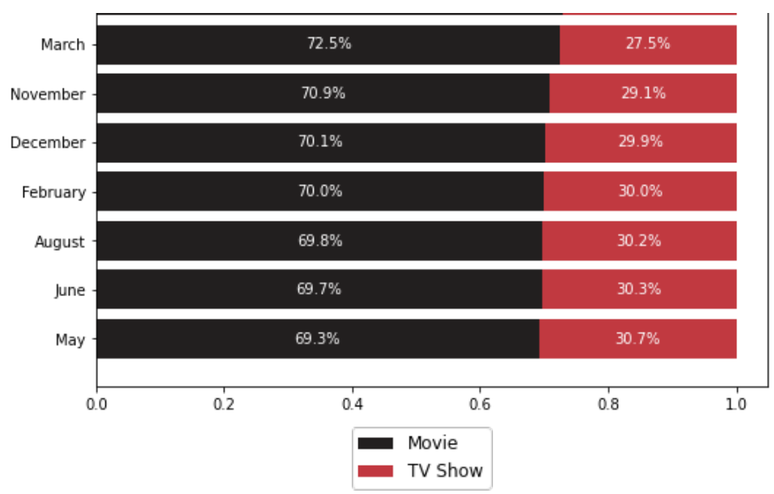
In [107]:

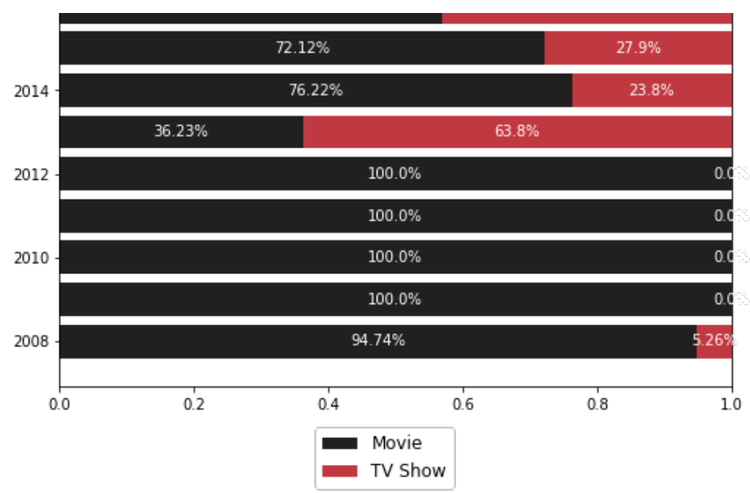
```
plt.figure(figsize=(28,8))
```

```
netflix_red = "#b20710"
netflix_grey = "#221f1f"

#Top Countries by Movies % release
plt.subplot(1,3,1)
top_countries = netflix_data[netflix_data["country"] != "Unknown Country"]['country'].value_counts()[:11].index
countrywise_release_data = netflix_data[(netflix_data["country"] != "Unknown Country")][['type', 'country']].groupby('country')['type'].value_counts().unstack().loc[top_countries]
countrywise_release_data['sum'] = countrywise_release_data.sum(axis=1)
countrywise_release_data_ratio = (countrywise_release_data.T / countrywise_release_data['sum']).T[['Movie', 'TV Show']].sort_values(by='Movie',ascending=False)[::-1]
plt.barh(countrywise_release_data_ratio.index, countrywise_release_data_ratio['Movie'], color=netflix_grey, label='Movie')
plt.barh(countrywise_release_data_ratio.index, countrywise_release_data_ratio['TV Show'], left=countrywise_release_data_ratio['Movie'], color=netflix_red, alpha=0.8, label='TV Show')
for i in countrywise_release_data_ratio.index:
    plt.annotate(f"{countrywise_release_data_ratio['Movie'][i]*100:.3}%", xy=(countrywise_release_data_ratio['Movie'][i]/2, i),
                 va = 'center', ha='center',fontsize = 10, color='white')
for i in countrywise_release_data_ratio.index:
    plt.annotate(f"{countrywise_release_data_ratio['TV Show'][i]*100:.3}%", xy=(countrywise_release_data_ratio['Movie'][i]+countrywise_release_data_ratio['TV Show'][i]/2, i),
                 va = 'center', ha='center',fontsize=10, color='white',)
plt.legend(loc = "upper right", bbox_to_anchor = (0.6, -0.05), fontsize=12)
plt.title("Movie vs TV Show Release by Country", fontsize = 12)

#Top Months by Movies % release
plt.subplot(1,3,2)
top_month_release = netflix_data["month_added_on_netflix"].value_counts().index
monthwise_release_data = netflix_data[["type", "month_added_on_netflix"]].groupby("month_added_on_netflix")["type"].value_counts().unstack().loc[top_month_release]
monthwise_release_data["sum"] = monthwise_release_data.sum(axis=1)
monthwise_release_data_ratio = (monthwise_release_data.T / monthwise_release_data["sum"]).T[["Movie", "TV Show"]].sort_values(by="Movie",ascending=False)[::-1]
plt.barh(monthwise_release_data_ratio.index, monthwise_release_data_ratio["Movie"], color=netflix_grey, label="Movie")
plt.barh(monthwise_release_data_ratio.index, monthwise_release_data_ratio["TV Show"], left=monthwise_release_data_ratio["Movie"], color=netflix_red, alpha=0.8, label="TV Show")
for i in monthwise_release_data_ratio.index:
    plt.annotate(f"{monthwise_release_data_ratio['Movie'][i]*100:.3}%", xy=(monthwise_release_data_ratio["Movie"][i]/2, i),
                 va = "center", ha="center",fontsize = 10, color="white")
for i in monthwise_release_data_ratio.index:
    plt.annotate(f"{monthwise_release_data_ratio['TV Show'][i]*100:.3}%", xy=(monthwise_release_data_ratio["Movie"][i]+monthwise_release_data_ratio["TV Show"][i]/2, i),
                 va = "center", ha="center",fontsize=10, color="white",)
plt.legend(loc = "upper right", bbox_to_anchor = (0.6, -0.05), fontsize=12)
plt.title("Movie vs TV Show Release by Month", fontsize = 12)

#Top Year by Movies % release
plt.subplot(1,3,3)
top_year_release = netflix_data["year_added_on_netflix"].value_counts().index
yearwise_release_data = netflix_data[["type", "year_added_on_netflix"]].groupby("year_added_on_netflix")["type"].value_counts().unstack().loc[top_year_release]
yearwise_release_data["sum"] = yearwise_release_data.sum(axis=1)
yearwise_release_data_ratio = (yearwise_release_data.T / yearwise_release_data["sum"]).T[["Movie", "TV Show"]].sort_values(by="Movie",ascending=False)[::-1]
yearwise_release_data_ratio["TV Show"] = yearwise_release_data_ratio["TV Show"].fillna(0)
plt.barh(yearwise_release_data_ratio.index, yearwise_release_data_ratio["Movie"], color=netflix_grey, label="Movie")
plt.barh(yearwise_release_data_ratio.index, yearwise_release_data_ratio["TV Show"], left=yearwise_release_data_ratio["Movie"], color=netflix_red, alpha=0.8, label="TV Show")
for i in yearwise_release_data_ratio.index:
    plt.annotate(f"{yearwise_release_data_ratio['Movie'][i]*100:.4}%", xy=(yearwise_release_data_ratio["Movie"][i]/2, i),
                 va = "center", ha="center",fontsize = 10, color="white")
for i in yearwise_release_data_ratio.index:
    plt.annotate(f"{yearwise_release_data_ratio['TV Show'][i]*100:.3}%", xy=(yearwise_release_data_ratio["Movie"][i]+yearwise_release_data_ratio["TV Show"][i]/2, i),
                 va = "center", ha="center",fontsize=10, color="white",)
plt.legend(loc = "upper right", bbox_to_anchor = (0.6, -0.05), fontsize=12)
plt.title("Movie vs TV Show Release by Year", fontsize = 12)

plt.suptitle("Movie vs TV Show Percentage of Release", fontsize=14, fontweight ="bold")
plt.show()
```









Looking at the visualisation above we can understand the following:

1. **Movie vs TV Show Release by Top 10 countries:**
   - **India has released the highest number of movies till date on Netflix.**
   - **Mexico has released almost equal number of movies and TV shows on Netflix.**
   - **Taiwan has released most TV shows.**
2. **Movie vs TV Show releases by Month of release on Netflix**
   - **We can observe that almost every month other than May, June and August more than 70% releases have been movies.**
3. **Movie vs TV show release on Netflix by Year**
   - **We can see that initial 5 years the content released by Netflix was movies.**
   - **Netflix started uploading TV shows more from 2013.**
   - **Since 2018 we can observe that the percent of TV shows have been constantly increase year by year and at the same time the content of upload of movies is reducing at the same level. But having said, it is not a significant change.**

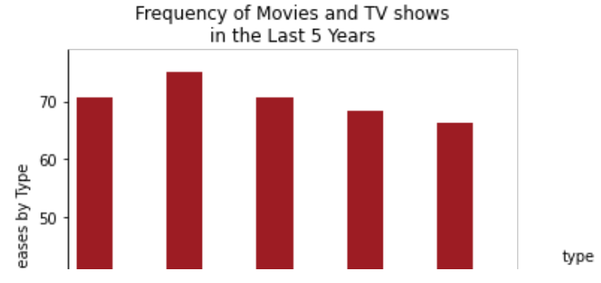
5.11) Dodge Bar Chart

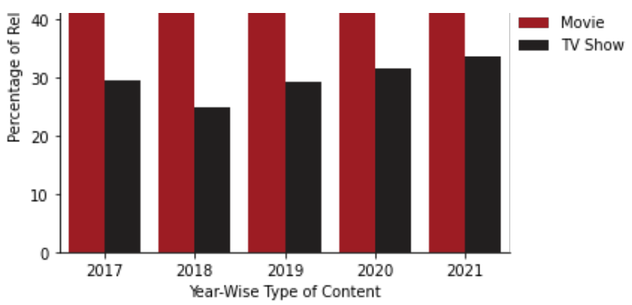
Let's check the last 5 years content uploade differentiation between Movies and TV Shows

In [108]:

```
plt.figure(figsize=(10,8))
netflix_red = "#b20710"
netflix_grey = "#221f1f"

#Last 5 Years TV Shows vs Movies released on Netflix
netflix_drop_duplicate = netflix_data.drop_duplicates(subset = "title")
last_5_years_data_by_type = netflix_drop_duplicate[netflix_drop_duplicate["year_added_on_netflix"]>2016].groupby("year_added_on_netflix")["type"].value_counts(normalize = True).mul(100).re
name('percent').reset_index().round(2)
dbc = sns.catplot(data = last_5_years_data_by_type, x = "year_added_on_netflix", y = "percent", hue ="type", kind = "bar", legend = True, palette=[netflix_red,netflix_grey])
dbc.set_axis_labels("Year-Wise Type of Content","Percentage of Releases by Type")

plt.title("Frequency of Movies and TV shows\nin the Last 5 Years")
plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```

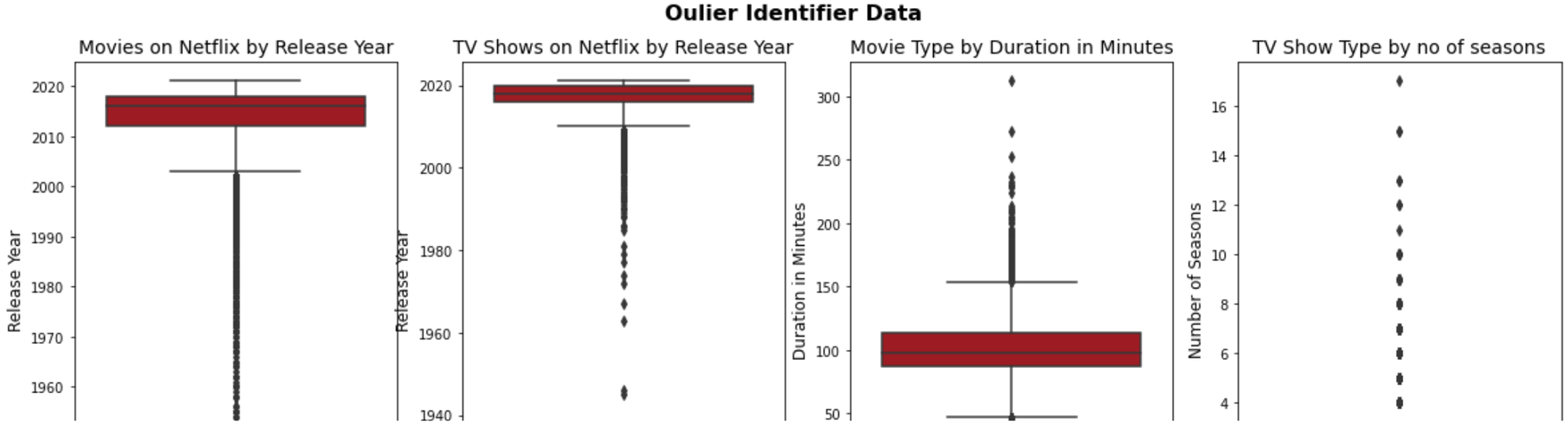


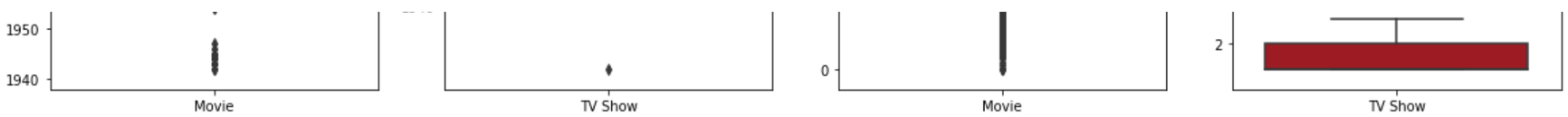


5.12) Outliers Identification through Box Plot

In [139]:

```
netflix_no_duplicate = netflix_data.drop_duplicates(subset = "title")
netflix_movie_no_duplicate = netflix_no_duplicate[netflix_no_duplicate["type"] == "Movie"]
netflix_tvshow_no_duplicate = netflix_no_duplicate[netflix_no_duplicate["type"] == "TV Show"]

plt.figure(figsize=(20,6))
netflix_red = "#b20710"
netflix_grey = "#221f1f"

plt.subplot(1,4,1)
sns.boxplot(data = netflix_movie_no_duplicate, x = "type", y ="release_year", color= netflix_red)
plt.title("Movies on Netflix by Release Year", fontsize = 14)
plt.xlabel("")
plt.ylabel("Release Year", fontsize = 12)

plt.subplot(1,4,2)
sns.boxplot(data = netflix_tvshow_no_duplicate, x = "type", y ="release_year", color = netflix_red)
plt.title("TV Shows on Netflix by Release Year", fontsize = 14)
plt.xlabel("")
plt.ylabel("Release Year", fontsize = 12)

plt.subplot(1,4,3)
sns.boxplot(data = netflix_movie_no_duplicate, x = "type", y ="duration_in_minutes", color = netflix_red)
plt.title("Movie Type by Duration in Minutes", fontsize = 14)
plt.xlabel("")
plt.ylabel("Duration in Minutes", fontsize = 12)

plt.subplot(1,4,4)
sns.boxplot(data = netflix_tvshow_no_duplicate[netflix_tvshow_no_duplicate["no_of_seasons"]!=0], x = "type", y ="no_of_seasons", color = netflix_red)
plt.title("TV Show Type by no of seasons", fontsize = 14)
plt.xlabel("")
plt.ylabel("Number of Seasons", fontsize = 12)

plt.suptitle("Oulier Identifier Data", fontweight = "bold", fontsize = 16)
plt.show()
```





Understand the outlier data is very important to understand the pattern and content being uploaded by netflix. We can undertand the following from the above plotting.

- **The most Movies released on Netflix are from the original release year sometime after 2012 till around 2018. The rest of the movies are distributed outside can be considered as outliers.**
- **The most TV Shows released on Netflix are from the original release year sometime after 2018 till around 2020. The rest of the movies are distributed outside can be considered as outliers.**
- **The movies that most released on Netflix have a runtime or are of duration between 90 minutes to 120 minutes.**
- **The TV Shows that most released on Netflix have a seasons between 1 to 2 seasons.**

6) Recommendations:

1. **Netflix should consider to release movies between the duration of 90 minutes to 120 minutes as that seems to be working presently for them.**
2. **As most of the movies that seems to working for netflix are from after 2012, they can try to consider to remove any of the old movies that are not working for them any more and consider to onboard more latest movies.**
3. **Netflix can cosider onboarding more movies from India as it seems that movies are being most viewed in India over all.**
4. **Netflix can cosider onboarding more TV Shows from Taiwan as it seems that movies are being most viewed in Taiwan over all.**
5. **It is recommended that Netflix can consider uploading more "International Movie" Genre as looks like the audience from various countries might be watch more due to which it is working for them.**
6. **Though TV Shows are slowing pickup it is recommended that you keep onboarding Movies on a regular basis and not reduce the number as they seemed to have always worked more than TV Shows.**
7. **It is recommended to onboard more comedy genre content as we can see that there has been a sudden increase especially in the last 3 years.**

In [136]:

```
```

In [ ]:

```
```