In [1]:

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

In [356]:

```
df = pd.read_csv("/Users/Chinmayi/Downloads/Netflix - Python.csv")
```

In [357]:

```
df
```

Out[357]:

	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	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV	To protect his family from a powerful

1. Problem Statement: Netflix wants to analyze the Movie/TV show data to get key insights on how to grow their business

Analysing basic metrics

In [25]:

```
df["type"].value_counts() # Number of TV shows and movies
```

Out[25]:

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

In [30]:

```
df["title"].count() # total number of TV shows and movies
```

Out[30]:

```
8807
```

In [26]:

```
df["country"].nunique() #no of unique countries
```

Out[26]:

```
748
```

In [27]:

```
df["country"].value_counts() #countries with most TV shows/Movies
```

Out[27]:

```
country
United States                             2818
India                                      972
United Kingdom                             419
Japan                                      245
South Korea                                199
                                          ... 
Romania, Bulgaria, Hungary                   1
Uruguay, Guatemala                           1
France, Senegal, Belgium                     1
Mexico, United States, Spain, Colombia       1
United Arab Emirates, Jordan                 1
Name: count, Length: 748, dtype: int64
```

In [35]:

```
df["release_year"].unique().min() #Oldest released movie/tv show
```

Out[35]:

```
1925
```

In [37]:

```
df["release_year"].unique().max() #most recently relesased movie/tv show
```

Out[37]:

```
2021
```

In [38]:

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

Out[38]:

```
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], dtype=int64)
```

2. Shape of data, missing values, attributes and overall summary

In [42]:

```
#shape of the data
df.shape
# The data contains 8807 rows and 12 columns
```

Out[42]:

```
(8807, 12)
```

In [358]:

```
# The data types of all attributes/columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

In [359]:

```
# converting into 'categorical' data types
df["type"] = df["type"].astype("category")
df["country"] = df["country"].astype("category")
df["rating"] = df["rating"].astype("category")
```

In [360]:

```
# convverting the data type of 'date_added' from 'object' to 'date_time'
df["date_added"] = df["date_added"].str.strip()
df["date_added"] = pd.to_datetime(df["date_added"])
```

In [361]:

```
# data types of all columns after making changes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   show_id       8807 non-null   object        
 1   type          8807 non-null   category      
 2   title         8807 non-null   object        
 3   director      6173 non-null   object        
 4   cast          7982 non-null   object        
 5   country       7976 non-null   category      
 6   date_added    8797 non-null   datetime64[ns]
 7   release_year  8807 non-null   int64         
 8   rating        8803 non-null   category      
 9   duration      8804 non-null   object        
 10  listed_in     8807 non-null   object        
 11  description   8807 non-null   object        
dtypes: category(3), datetime64[ns](1), int64(1), object(7)
memory usage: 676.6+ KB
```

In [362]:

```
# missing values for each column
df.isna().sum()
```

Out[362]:

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

Getting only the number from 'duration column'

In [363]:

```
df["duration"] = df["duration"].str.split(" ", expand = True)[0]
df["duration"] = df["duration"].astype("float64")
```

Preprocessing of Data

1. Imputing null values

In [364]:

```
#We will first fix the 'duration' column
# I found out that the 3 missing duration values were all from movies category so,
# I found out the average movie duration and filled this value
movie_duration = round(df[df["type"] == "Movie"]["duration"].mean(),0)
df["duration"].fillna(movie_duration, inplace = True)
```

In [365]:

```
# Since data type of 'rating' is 'category', we will use mode to fill the null values
rating_mode = df["rating"].mode()[0]
df["rating"].fillna(rating_mode, inplace = True)
```

In [366]:

```
# Since the data type of 'country' is categegorical, we will use backfill and forward fill
df["country"] = df["country"].fillna(method = "bfill")
```

In [367]:

```
df["date_added"] = df["date_added"].fillna(method = "ffill")
```

2. Unnesting

In [368]:

```
# Unnesting the 'cast' column and melting it in rows
cast = df["cast"].str.split(", ", expand = True)
pi = pd.concat([df, cast], axis = 1)
pi = pi.melt(id_vars = pi.columns[0:12].tolist(), value_name = "Cast")
pi.drop(pi[pi["Cast"].isna()].index, inplace = True)
pi.drop("cast", axis = 1, inplace =True)
```

In [369]:

```
# Unnesting the 'country' column and melting it in rows
country = pi["country"].str.split(", ", expand = True)
pi2 = pd.concat([pi,country],axis = 1)
pi2.drop(["country","variable"], axis = 1, inplace = True)
pi2 = pi2.melt(id_vars = pi2.columns[:11], value_name = "country").drop("variable",axis = 1)
pi2.drop(pi2[pi2["country"].isna()].index, inplace = True)
```

In [370]:

```
# Unnesting the 'listed_in' column and melting it in rows
listed_in = pi2["listed_in"].str.split(", ", expand = True)
pi3 = pd.concat([pi2,listed_in],axis = 1)
pi3.drop("listed_in", axis = 1, inplace = True)
pi3 = pi3.melt(id_vars = pi3.columns[:11], value_name = "genre").drop("variable",axis = 1)
pi3.drop(pi3[pi3["genre"].isna()].index, inplace = True)
```

In [371]:

```
# Saving the file in desktop
pi3.to_csv('/Users/Chinmayi/Downloads/cleaned_data.csv')
pi3 = pd.read_csv("/Users/Chinmayi/Downloads/cleaned_data.csv")
```

In [372]:

```
pi3.drop("Unnamed: 0", axis = 1, inplace = True)
```

In [373]:

```
# Getting the most popular director for each country
direct = pi3.groupby("country")[["title", "director"]].value_counts().reset_index().drop("count", axis = 1)
pop_dir = direct.groupby(["country"]).apply(lambda x:x["director"].value_counts().head(1)).reset_index().drop("cou
```

In [374]:

```
# Filling Nan 'director' values with most popular director in that country
qw = pi3.merge(pop_dir, how = 'left', on = "country")
qw["director_x"].fillna(qw["director_y"], inplace = True)
qw.drop("director_y", axis =1, inplace = True)
```

In [375]:

```
# Unnesting director column
director_x = qw["director_x"].str.split(", ", expand = True)
final = pd.concat([qw,director_x],axis = 1)
final.drop("director_x", axis = 1, inplace = True)
final = final.melt(id_vars = final.columns[:11], value_name = "director").drop("variable",axis = 1)
final.drop(final[final["director"].isna()].index, inplace = True)
```

In [379]:

```
# removing bad/wrong data
final.drop("Unnamed: 0", axis = 1, inplace = True)
final.drop(final[final["director"]=="Louis C.K."].index, inplace = True)
final["date_added"] = final["date_added"].str.strip()
final["date_added"] = pd.to_datetime(final["date_added"])
```

In [382]:

```
# Saving the final cleaned dataset
final.to_csv('/Users/Chinmayi/Downloads/final.csv')
```

In [353]:

```
# Reading the dataset
final = pd.read_csv("/Users/Chinmayi/Downloads/final.csv")
```

In [383]:

```
final
```

Out[383]:

	show_id	type	title	date_added	release_year	rating	duration	description	Cast	country	gen
0	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2.0	After crossing paths at a party, a Cape Town t...	Ama Qamata	South Africa	Internation TV Shov
1	s3	TV Show	Ganglands	2021-09-24	2021	TV-MA	1.0	To protect his family from a powerful drug lor...	Sami Bouajila	India	Crime T Shov
2	s5	TV Show	Kota Factory	2021-09-24	2021	TV-MA	2.0	In a city of coaching centers known to train I...	Mayur More	India	Internation TV Shov
3	s6	TV Show	Midnight Mass	2021-09-24	2021	TV-MA	1.0	The arrival of a charismatic young priest brin...	Kate Siegel	United States	TV Dram
4	s7	Movie	My Little Pony: A New Generation	2021-09-24	2021	PG	91.0	Equestria's divided. But a bright-eyed hero be...	Vanessa Hudgens	United States	Children Fami Movie
...	...	...	...	...	...	...	...	...	...	...	
2314812	s5888	Movie	Walt Disney Animation Studios Short Films Coll...	2015-10-25	2015	TV-Y	90.0	This collection of 12 short films from Disney ...	Dave Foley	United States	Children Fami Movie
2316010	s5888	Movie	Walt Disney Animation Studios Short Films Coll...	2015-10-25	2015	TV-Y	90.0	This collection of 12 short films from Disney ...	Derek Richardson	United States	Children Fami Movie
2316845	s5888	Movie	Walt Disney Animation Studios Short Films Coll...	2015-10-25	2015	TV-Y	90.0	This collection of 12 short films from Disney ...	Betty White	United States	Children Fami Movie
2317450	s5888	Movie	Walt Disney Animation Studios Short Films Coll...	2015-10-25	2015	TV-Y	90.0	This collection of 12 short films from Disney ...	Zachary Levi	United States	Children Fami Movie
2317921	s5888	Movie	Walt Disney Animation Studios Short Films Coll...	2015-10-25	2015	TV-Y	90.0	This collection of 12 short films from Disney ...	Mandy Moore	United States	Children Fami Movie

207137 rows × 12 columns

Statistical analysis

In [40]:

```
# Top 5 directors
final.groupby("director").apply(lambda x: x["title"].nunique()).sort_values(ascending = False).head(5)
```

Out[40]:

```
director
Marcus Raboy          963
Martin Campbell       255
Toshiya Shinohara     198
David Batty           181
Suhas Kadav           171
dtype: int64
```

In [42]:

```
# Top 5 countrie
final.groupby("country").apply(lambda x: x["title"].nunique()).sort_values(ascending = False).head(5)
```

Out[42]:

```
country
United States     3609
India             1083
United Kingdom     772
Canada             453
France             393
dtype: int64
```

In [45]:

```
# First movie/show added on netflix
final.loc[final["date_added"] == min(final["date_added"]),["title", "date_added"]].drop_duplicates()
```

Out[45]:

	title	date_added
5411	To and From New York	2008-01-01

In [201]:

```
# Most recent movie/show added on netflix
df.loc[df["date_added"] == max(df["date_added"]),["title", "date_added"]]
```

Out[201]:

	title	date_added
0	Dick Johnson Is Dead	2021-09-25

In [46]:

```
# Top 10 popular Actors/Actress
final.groupby("Cast").apply(lambda x: x["title"].nunique()).sort_values(ascending = False).head(10)
```

Out[46]:

```
Cast
Anupam Kher          43
Shah Rukh Khan       35
Julie Tejwani        33
Naseeruddin Shah     32
Takahiro Sakurai     32
Rupa Bhimani         31
Akshay Kumar         30
Om Puri              30
Yuki Kaji            29
Amitabh Bachchan     28
dtype: int64
```

In [61]:

```
# Aggregate quantitative details about the Movies
final.loc[final["type"]=="Movie", ["duration","release_year","title"]].drop_duplicates().describe()
```

Out[61]:

	duration	release_year
count	5656.000000	5656.000000
mean	101.355552	2012.911775
std	27.797722	9.599338
min	8.000000	1942.000000
25%	88.000000	2011.000000
50%	100.000000	2016.000000
75%	116.000000	2018.000000
max	312.000000	2021.000000

In [62]:

```
# Aggregate quantitative details about the TV Shows
final.loc[final["type"]=="TV Show", ["duration","release_year","title"]].drop_duplicates().describe()
```

Out[62]:

	duration	release_year
count	2323.000000	2323.000000
mean	1.837279	2016.504520
std	1.662850	5.257565
min	1.000000	1963.000000
25%	1.000000	2015.000000
50%	1.000000	2018.000000
75%	2.000000	2020.000000
max	17.000000	2021.000000

3. Value counts and Unique attributes

In [334]:

```
#Value counts of movies/tv shows,
final.groupby("type")["title"].apply(lambda x: x.nunique())
```

Out[334]:

```
type
Movie      5653
TV Show    2323
Name: title, dtype: int64
```

In [35]:

```
# value_counts of release years
final.groupby("release_year")["title"].apply(lambda x: x.nunique())
```

Out[35]:

```
release_year
1942       1
1944       1
1945       1
1946       1
1947       1
        ... 
2017     910
2018    1026
2019     917
2020     827
2021     494
Name: title, Length: 72, dtype: int64
```

In [10]:

```
# Unique years
final["release_year"].unique()
```

Out[10]:

```
array([2021, 1993, 2020, 2018, 1996, 1998, 1997, 2010, 2013, 2017, 1975,
       1978, 1983, 1987, 2012, 2001, 2014, 2002, 2003, 2004, 2011, 2008,
       2009, 2007, 2005, 2006, 1994, 2019, 2016, 2015, 1982, 1989, 1990,
       1991, 1999, 1986, 1992, 1984, 1980, 1961, 2000, 1995, 1985, 1976,
       1959, 1988, 1981, 1972, 1964, 1954, 1979, 1958, 1956, 1963, 1970,
       1973, 1974, 1960, 1966, 1971, 1962, 1969, 1977, 1967, 1968, 1965,
       1945, 1946, 1955, 1942, 1947, 1944], dtype=int64)
```

In [29]:

```
# value counts of rating category
final.groupby("rating")["title"].apply(lambda x: x.nunique())
```

Out[29]:

```
rating
G             40
NC-17          3
NR            63
PG           279
PG-13        477
R            790
TV-14       1954
TV-G         183
TV-MA       2884
TV-PG        719
TV-Y         267
TV-Y7        310
TV-Y7-FV       4
UR             3
Name: title, dtype: int64
```

In [31]:

```
# value_counts of countries
final.groupby("country")["title"].apply(lambda x: x.nunique())
```

Out[31]:

```
country
Afghanistan      1
Albania          1
Algeria          6
Angola           1
Argentina       88
                ..
Vatican City     1
Venezuela        2
Vietnam          7
West Germany     4
Zimbabwe         1
Name: title, Length: 113, dtype: int64
```

In [21]:

```
# Unique countries
final["country"].unique()
```

Out[21]:

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

In [320]:

```
# value_counts of genre
final.groupby("genre")["title"].apply(lambda x: x.nunique())
```

Out[320]:

```
genre
Action & Adventure               853
Anime Features                    68
Anime Series                     173
British TV Shows                 207
Children & Family Movies         608
Classic & Cult TV                 28
Classic Movies                   109
Comedies                        1662
Crime TV Shows                   394
Cult Movies                       70
Documentaries                    445
Docuseries                       188
Dramas                          2416
Faith & Spirituality              60
Horror Movies                    354
Independent Movies               753
International Movies            2574
International TV Shows          1240
Kids' TV                         408
Korean TV Shows                  147
LGBTQ Movies                      85
Movies                            50
Music & Musicals                 340
Reality TV                       163
Romantic Movies                  609
Romantic TV Shows                357
Sci-Fi & Fantasy                 240
Science & Nature TV               57
Spanish-Language TV Shows        162
Sports Movies                    165
Stand-Up Comedy                  342
Stand-Up Comedy & Talk Shows      49
TV Action & Adventure            166
TV Comedies                      555
TV Dramas                        756
TV Horror                         72
TV Mysteries                      93
TV Sci-Fi & Fantasy               82
TV Shows                          11
TV Thrillers                      54
Teen TV Shows                     66
Thrillers                        577
Name: title, dtype: int64
```

In [33]:

```
# Unique genres
final["genre"].unique()
```

Out[33]:

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

4. Visual Analysis

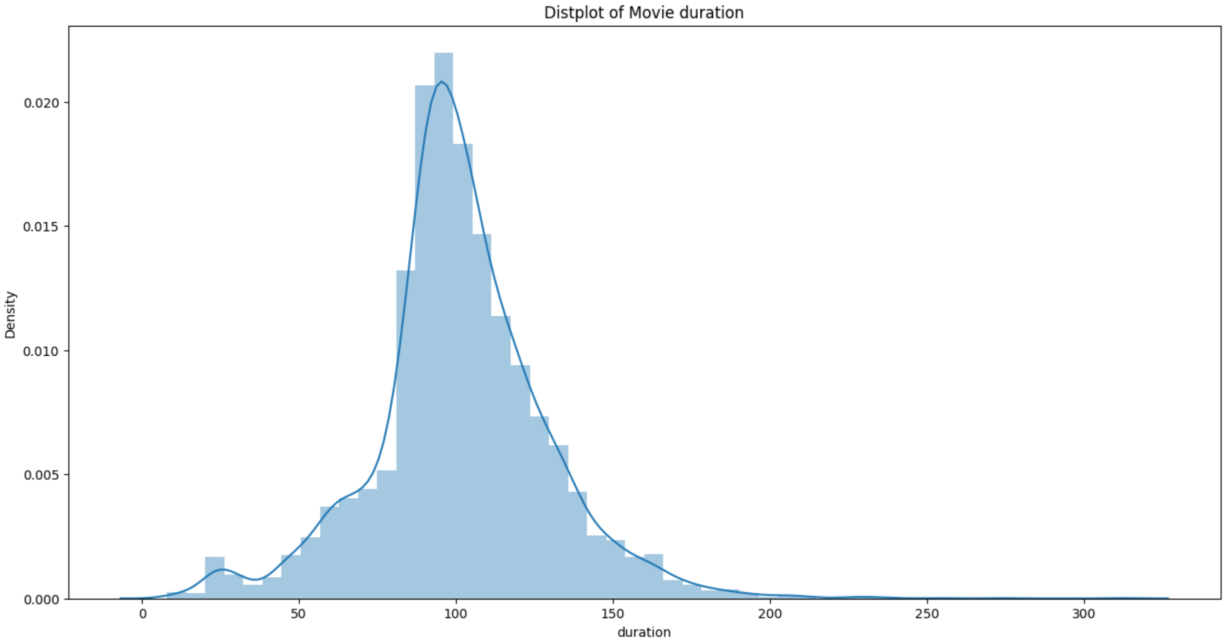
4.1. Univariate Analysis

In [139]:

```
# Distplot of Movie duration
data_ = final.loc[final["type"]=="Movie",["title", "duration"]].drop_duplicates()
plt.figure(figsize=(16, 8))
sns.distplot(data_["duration"])
plt.title("Distplot of Movie duration")

# Majority of the movies have a duration of about 100 mins (1h 40mins)
# and the graph says this duration drastically decreases as we move away from the 100mins mark.
```

Out[139]:

```
Text(0.5, 1.0, 'Distplot of Movie duration')
```

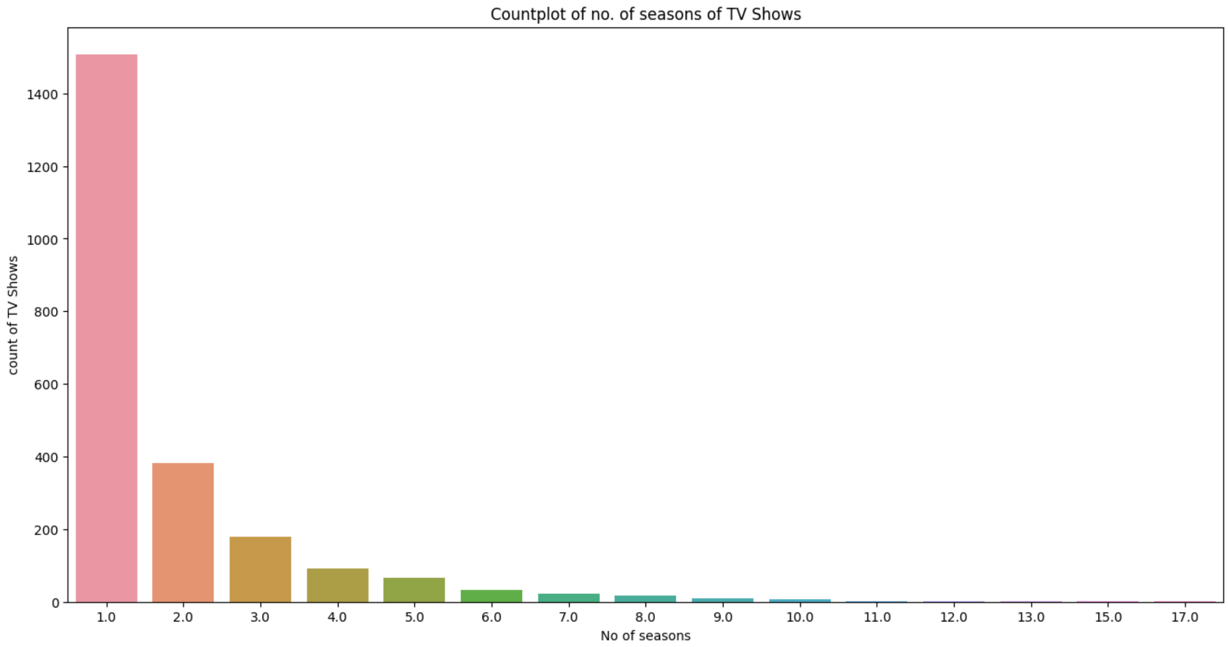


In [157]:

```
# Countplot of no. of seasons of TV Shows
data_ = final.loc[final["type"]=="TV Show",["title", "duration"]].drop_duplicates()["duration"].value_counts().res
plt.figure(figsize=(16, 8))
#plt.xticks(np.linspace(min(data_["duration"]), max(data_["duration"]), num=17))
sns.barplot(data = data_, x= "duration", y = "count")
plt.xlabel("No of seasons")
plt.ylabel("count of TV Shows")
plt.title("Countplot of no. of seasons of TV Shows")
# Majority of the TV shows have only 1 seasons. And after 5 seasons there are very few TV shows.
```

Out[157]:

```
Text(0.5, 1.0, 'Countplot of no. of seasons of TV Shows')
```

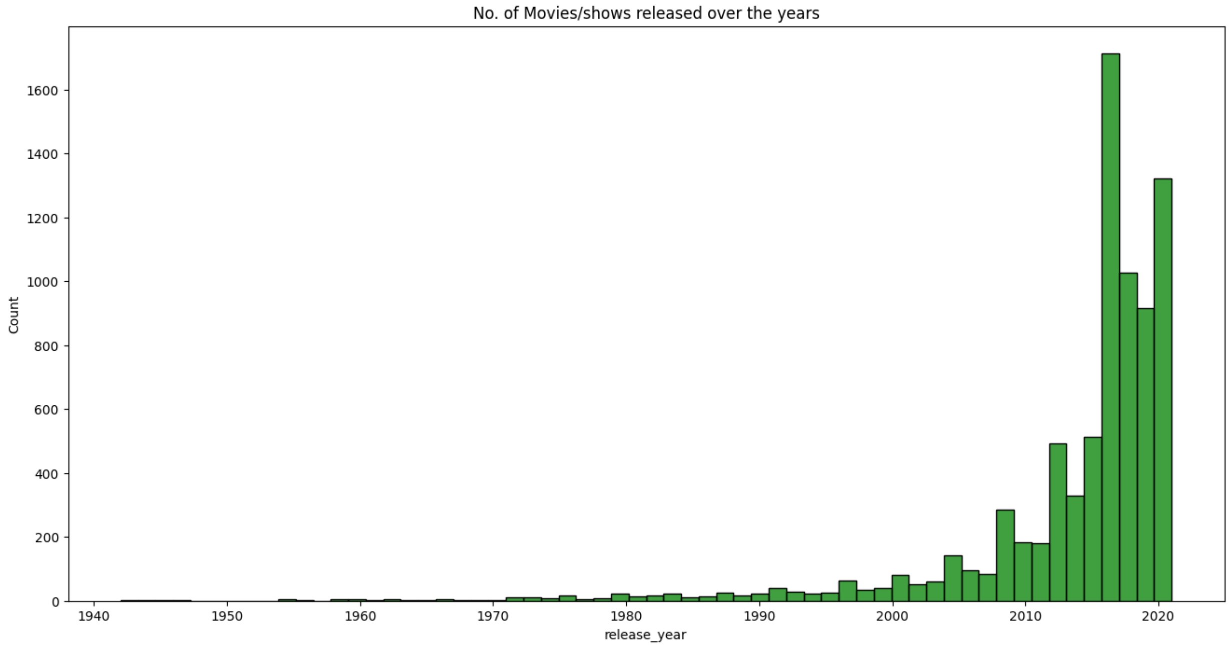


In [174]:

```
# Histogram of no. of seasons of TV Shows
data_ = final.loc[:,["title", "release_year"]].drop_duplicates()
plt.figure(figsize=(16, 8))
sns.histplot(data = data_, x= "release_year",bins = 60, color = "green")
plt.title("No. of Movies/shows released over the years")

# The no. of movies/shows released has increased exponentially over the years.
# It has peaked at the year 2019 and after that it has decreased.
```

Out[174]:

```
Text(0.5, 1.0, 'No. of Movies/shows released over the years')
```



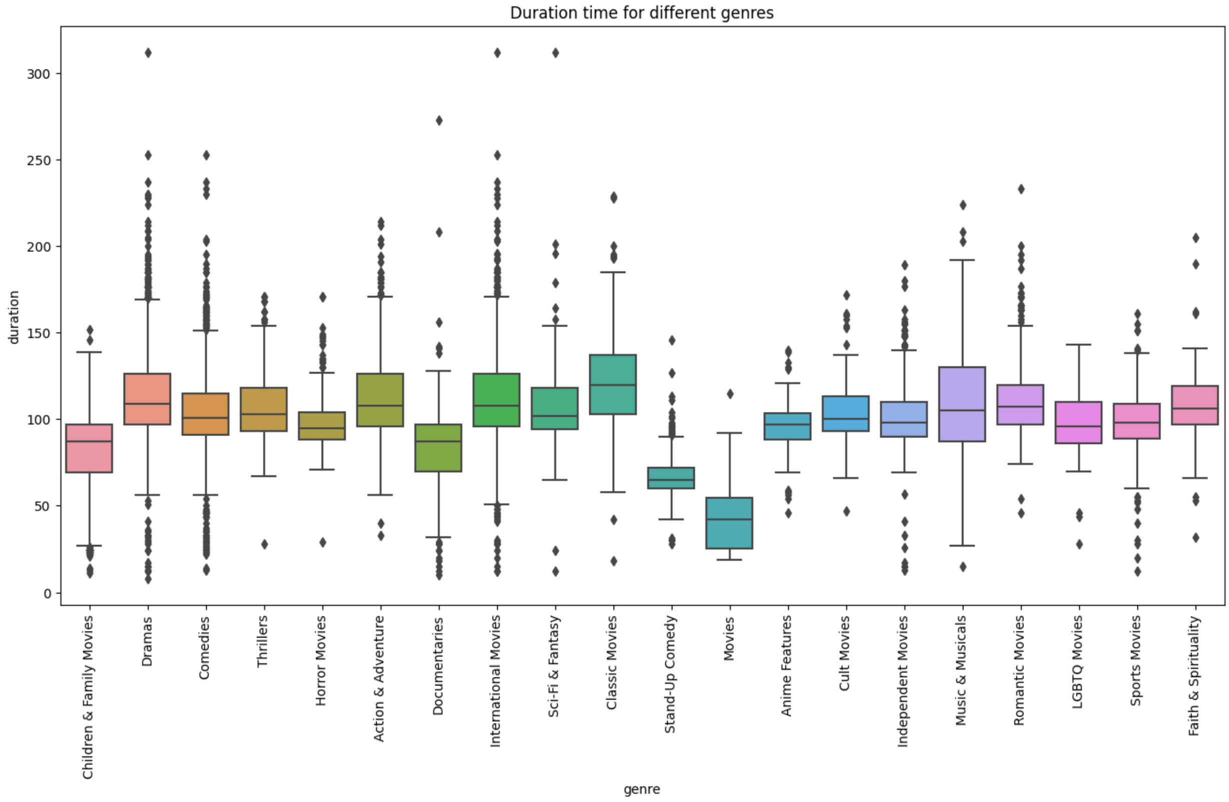
4.2. Categorical Data

In [192]:

```
#Duration time for different genres of Movies
plt.figure(figsize=(16, 8))
data_ = final.loc[final["type"]=="Movie", ["title", "genre", "duration"]].drop_duplicates()
plt.xticks(rotation=90)
sns.boxplot(data = data_, x = "genre", y = "duration")
plt.title("Duration time for different genres")

# We observe median duration of classical movies is the highest.
# The genre of 'Movies' has the least median duration. These genre of movies are mainly short movies which is of
# The genre 'Internation Movies' and 'Drama' have the biggest no. of outliers.
```

Out[192]:

```
Text(0.5, 1.0, 'Duration time for different genres')
```

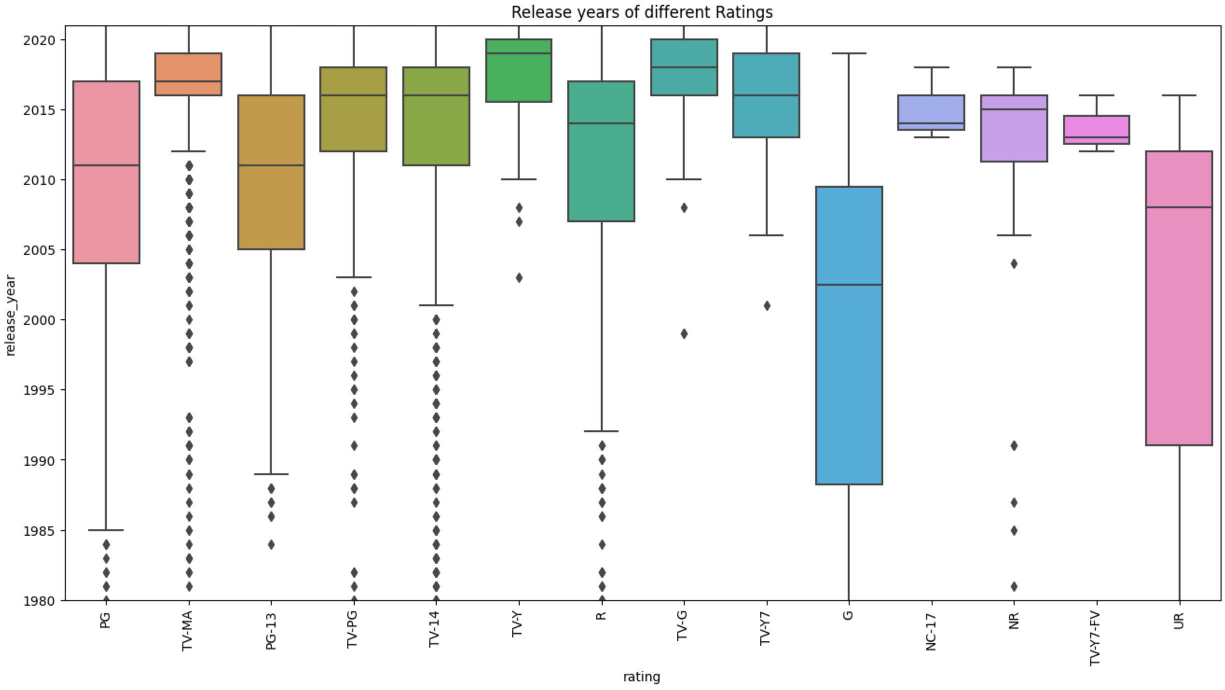


In [196]:

```
#Release years of different Ratings
plt.figure(figsize=(16, 8))
data_ = final.loc[final["type"]=="Movie", ["title", "rating", "release_year"]].drop_duplicates()
plt.xticks(rotation=90)
plt.ylim([1980,2021])
sns.boxplot(data = data_, x = "rating", y = "release_year")
plt.title("Release years of different Ratings")

# We observe that rating category 'G' and 'UR' are mostly for old movies/shows.
# The rating category 'TV-Y' and 'TV-G' are mostly for newer movies/shows.
```

Out[196]:

```
Text(0.5, 1.0, 'Release years of different Ratings')
```

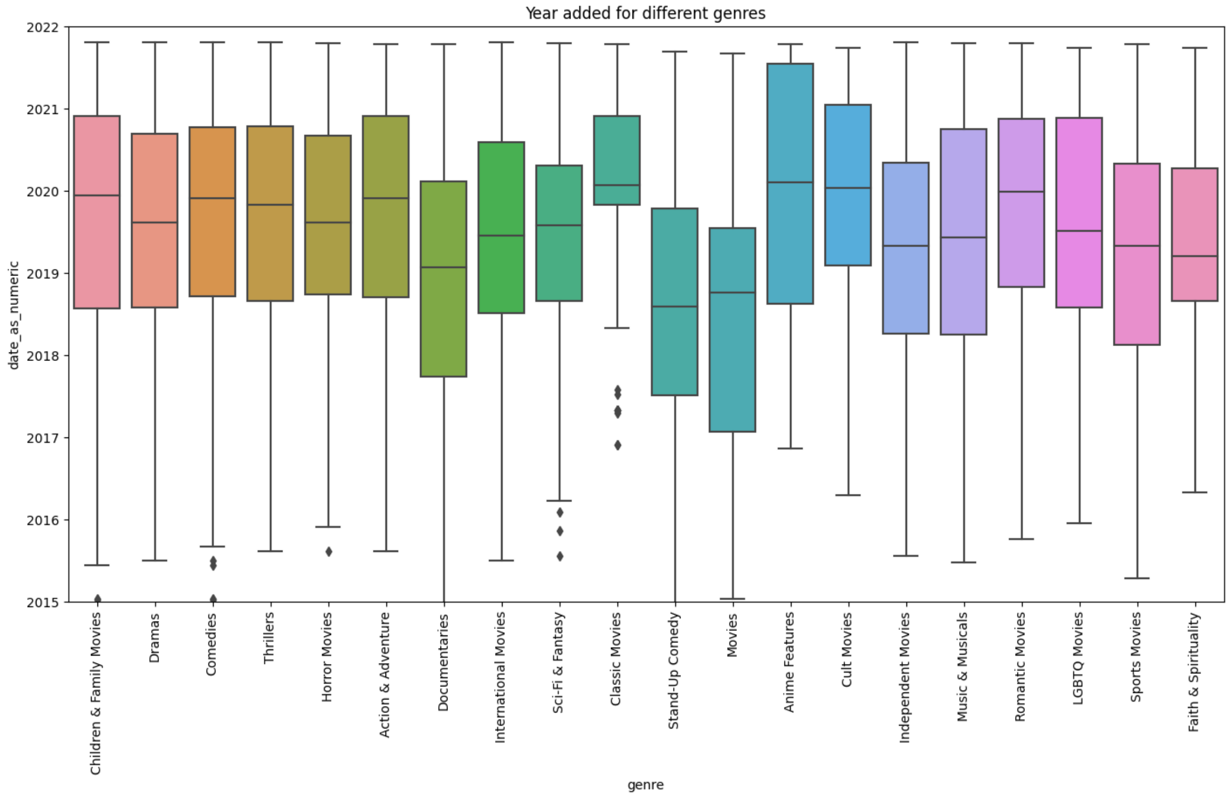


In [281]:

```
#Year added for different genres of Movies
plt.figure(figsize=(16, 8))
data_ = final.loc[final["type"]=="Movie", ["title", "genre", "date_added"]].drop_duplicates()
data_["date_as_numeric"] = data_["date_added"].dt.year + (data_["date_added"].dt.month*30)/365 + data_["date_adde
plt.xticks(rotation=90)
sns.boxplot(data = data_, x = "genre", y = "date_as_numeric")
plt.ylim([2015,2022])
plt.title("Year added for different genres")

# We see that 'Anime Features' genre has the highest median year and the box itself is above than any other genre
# This implies Anime genre is getting popular in recent times.
# The genre 'Movies' was mostly being added in the earlier days of Netflix.
# Classical Movies have been added recently.
```

Out[281]:

```
Text(0.5, 1.0, 'Year added for different genres')
```



4.3. Heatmaps and Pairplots

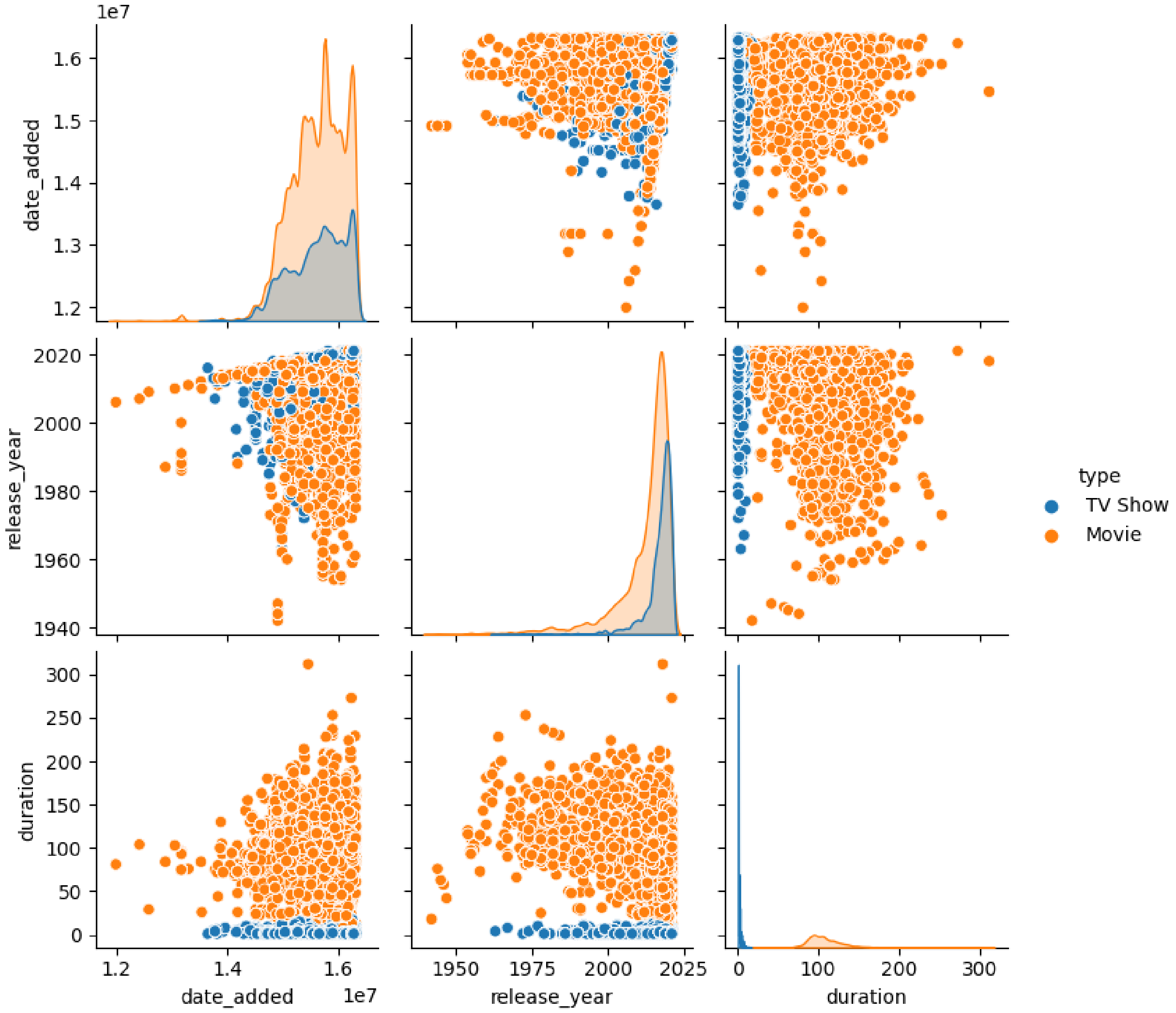
In [234]:

```
# Pairplot for numeric data
final2 = final.copy()
final2["date_added"] = final2["date_added"].apply(lambda x: x.value)/100000000000

plt.figure(figsize = (18,12))
sns.pairplot(final2, hue = "type")

# We see that TV shows duration mostly appear at 1, and movies mainly appear around 100.
# Most of the movies/shows have been added recently.
# The release years have been sparse before the year 2000, but after that it seems the number per year is uniform.
```

Out[234]:

```
<seaborn.axisgrid.PairGrid at 0x24856cce790>
```

```
<Figure size 1800x1200 with 0 Axes>
```

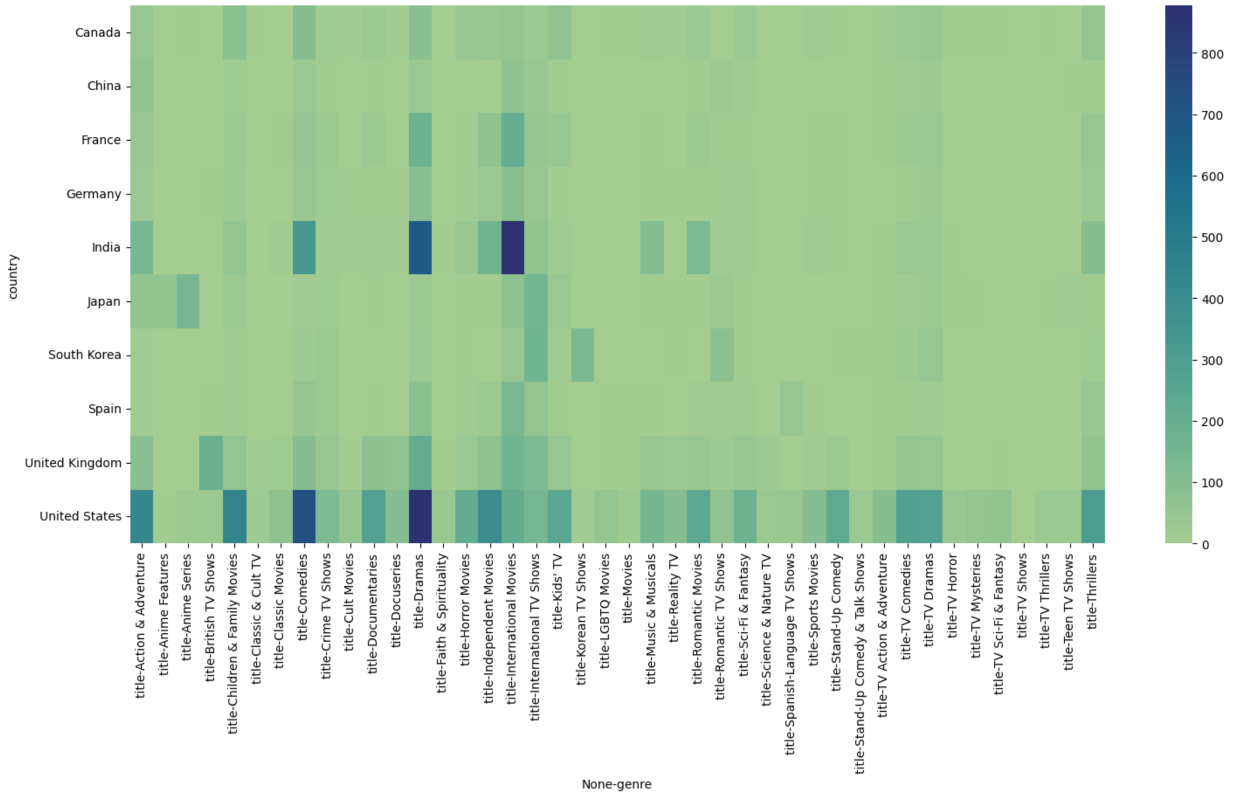


In [268]:

```
# Heatmap to show which genre is the most popular among the top 10 countries
top_country = final.groupby("country").apply(lambda x: x["title"].nunique()).sort_values(ascending = False).head(1
data_ = final.loc[final["country"].isin(top_country),["title", "country", "genre"]].drop_duplicates()
data_ = pd.pivot_table(data = data_, index = "country", columns = "genre", aggfunc = "count").fillna(0)
plt.figure(figsize = (18,8))
sns.heatmap(data_,cmap = "crest")

# In India, the genre 'International movies' and 'Dramas' seems to be most popular.
# In US, the genre 'Dramas' and 'Comedy' seems to be the most popular.
```

Out[268]:

```
<Axes: xlabel='None-genre', ylabel='country'>
```



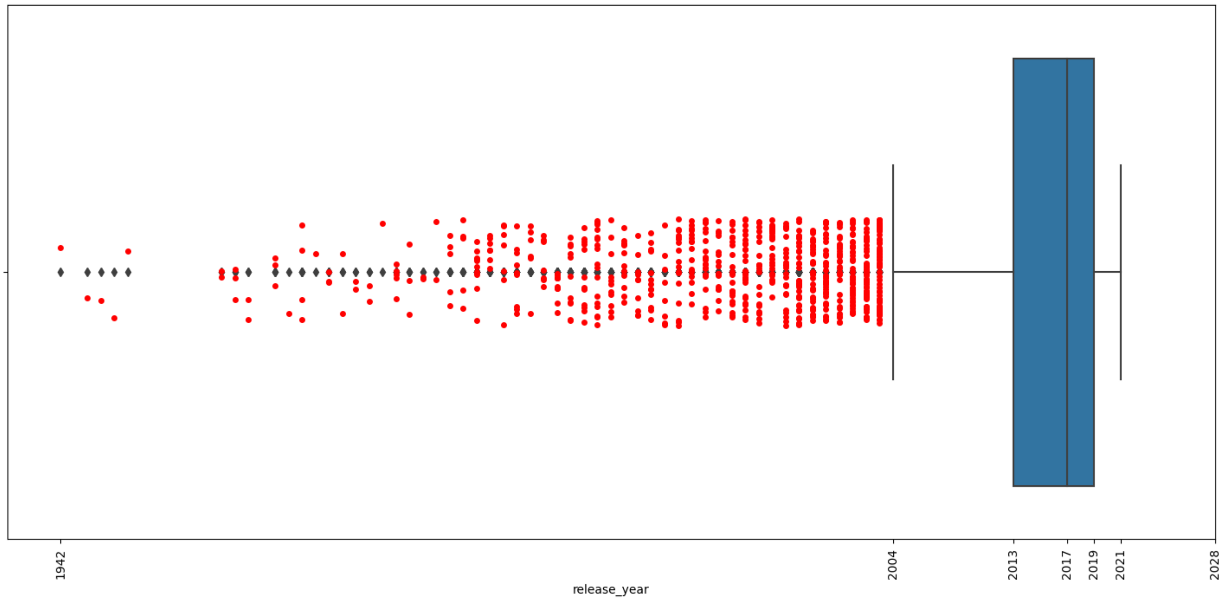
5. Missing values and outlier check

5.1 Missing values have already been addressed in the Preprocessing of the Data set

5.2 Outlier Check

In [317]:

```
# Checking for outliers in the release_year column
df = final.loc[:, ["title", "release_year"]].drop_duplicates()
outl = df["release_year"].describe()
Q1 = outl.loc["25%"]
Q3 = outl.loc["75%"]
iqr = Q3 - Q1
low = Q1 - 1.5*iqr
upp = Q3 + 1.5*iqr
outliers = df[(df["release_year"]<low) | (df["release_year"]>upp)]
plt.figure(figsize = (18,8))
plt.xticks(rotation=90)
sns.boxplot(x = df["release_year"])
sns.stripplot(x = outliers["release_year"], color = "red")
plt.xticks([df["release_year"].min(), low, Q1,df["release_year"].median(), Q3, upp, df["release_year"].max() ])
plt.show()

# Since most of the movies/shows have been added recently, there are no outliers above the upper whisker
# All the shows/movies in the outliers are from the year 1942 to 2004.
```



In [302]:

```
outliers
```

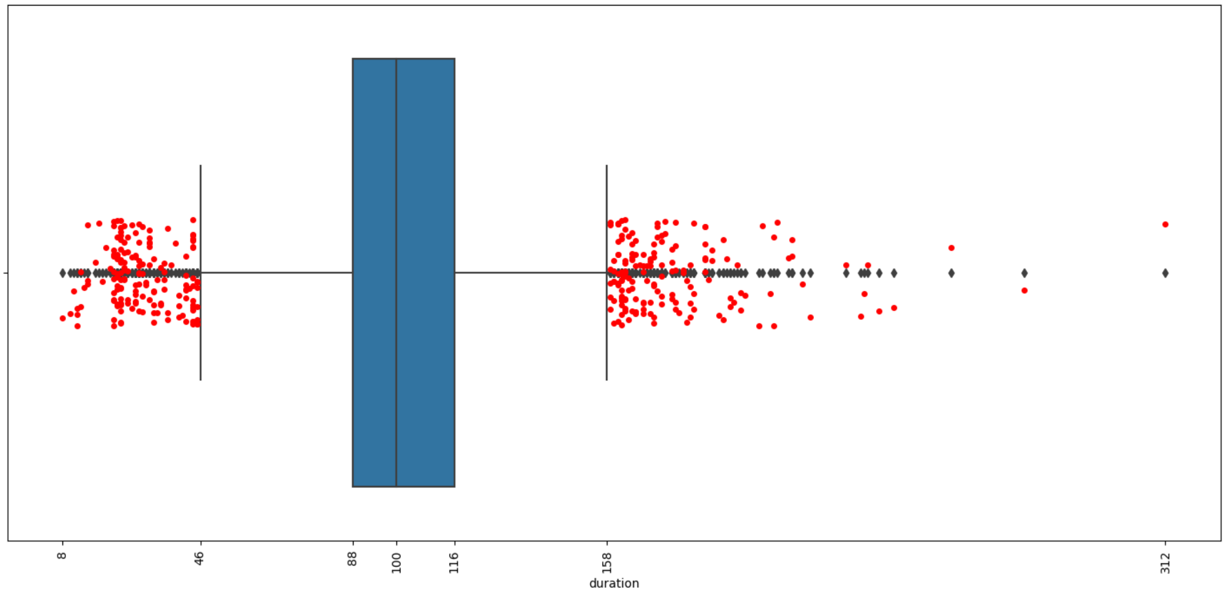
Out[302]:

	title	release_year
5	Sankofa	1993
16	Avvai Shanmughi	1996
18	Jeans	1998
20	Minsara Kanavu	1997
35	Jaws	1975
...	...	...
7934	Wyatt Earp	1994
7936	XXx	2002
7938	Y Tu Mamá También	2001
7940	Yaadein	2001
7962	Young Tiger	1973

700 rows × 2 columns

In [306]:

```
# Checking for outliers in the movies duration column
df = final.loc[final["type"] =="Movie", ["title", "duration"]].drop_duplicates()
outl = df["duration"].describe()
Q1 = outl.loc["25%"]
Q3 = outl.loc["75%"]
iqr = Q3 - Q1
low = Q1 - 1.5*iqr
upp = Q3 + 1.5*iqr
outliers = df[(df["duration"]<low) | (df["duration"]>upp)]
plt.figure(figsize = (18,8))
plt.xticks(rotation=90)
sns.boxplot(x = df["duration"])
sns.stripplot(x = outliers["duration"], color = "red")
plt.xticks([df["duration"].min(), low, Q1,df["duration"].median(), Q3, upp, df["duration"].max()])
plt.show()

# We see there are many outliers below the time duration of 46 mins.
# The outliers beyond upper whisker range from 158 - 312 mins.
```



In [307]:

```
outliers
```

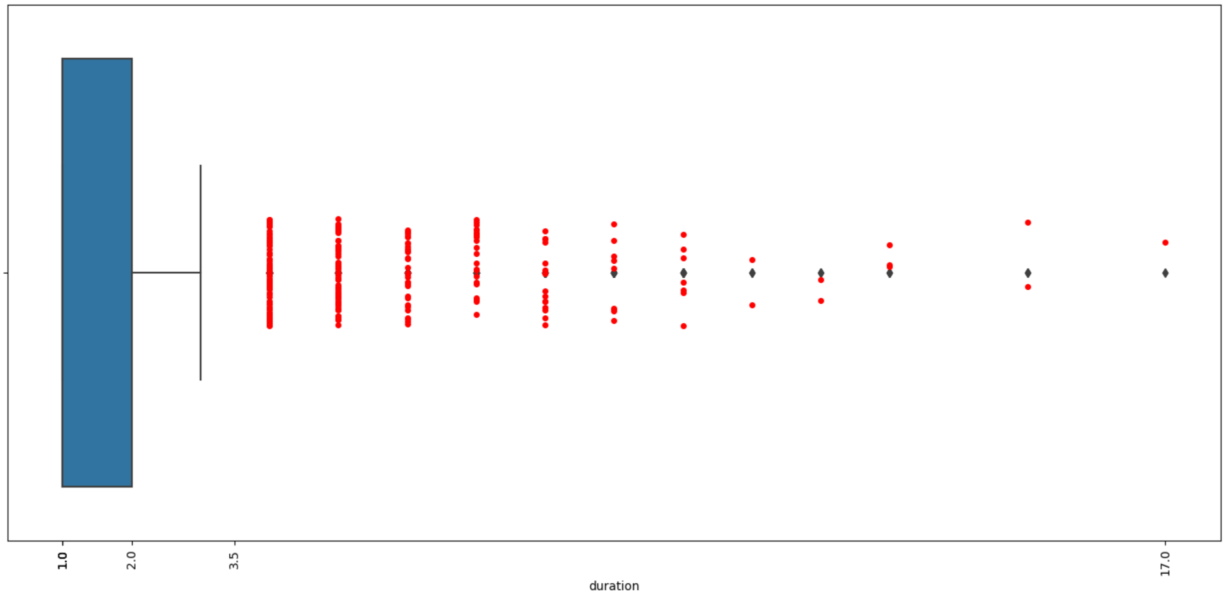
Out[307]:

	title	duration
16	Avvai Shanmughi	161.0
18	Jeans	166.0
62	A StoryBots Space Adventure	13.0
64	King of Boys	182.0
148	Once Upon a Time in America	229.0
...	...	...
7818	Trimurti	173.0
7827	Tukaram	162.0
7843	Under an Arctic Sky	40.0
7934	Wyatt Earp	191.0
7940	Yaadein	171.0

325 rows × 2 columns

In [309]:

```
# Checking for outliers in the movies duration column
df = final.loc[final["type"] =="TV Show", ["title", "duration"]].drop_duplicates()
outl = df["duration"].describe()
Q1 = outl.loc["25%"]
Q3 = outl.loc["75%"]
iqr = Q3 - Q1
low = Q1 - 1.5*iqr
upp = Q3 + 1.5*iqr
outliers = df[(df["duration"]<low) | (df["duration"]>upp)]
plt.figure(figsize = (18,8))
plt.xticks(rotation=90)
sns.boxplot(x = df["duration"])
sns.stripplot(x = outliers["duration"], color = "red")
plt.xticks([df["duration"].min(), Q1,df["duration"].median(), Q3, upp, df["duration"].max()])
plt.show()

# Most of the TV shows predominantly appear around 1 season mark.
# That is why there is no lower whisker, the median itself is 1.
# Outliers start appearing after season 4 or more.
```



In [310]:

```
outliers
```

Out[310]:

	title	duration
6	The Great British Baking Show	9.0
11	Dear White People	4.0
15	Resurrection: Ertugrul	5.0
48	Nailed It	6.0
58	Numberblocks	6.0
...	...	...
7741	The Twilight Zone (Original Series)	4.0
7756	The West Wing	7.0
7840	Ugly Duckling	4.0
7890	Weeds	8.0
7905	When Calls the Heart	5.0

255 rows × 2 columns

6.1 Insights on range of attributes

Release year: From the above boxplot to find the outliers in the release_year column, we see that the range of movie/show release year is from 1942 to 2021. The older movies/shows are less compared to recently released ones.

Movie duration: From the outlier boxplot mentioned above, we see that it ranges from as low as 8 mins to 312 mins!. However the ideal time duration for a movie is 100 mins(median).

TV show duration: From the above mentioned boxplots, we see that the number of seasons of TV show ranges from 1 to 17. Majority of them are 1 season shows. The number of shows which is aired for 4 or more seasons is very less.

Rating: The number of movies/shows for each rating range from 3 (NC-17, UR) to 2884 (TV-MA). Which means the succefull shows on Netflix are usually from the rating of TV-MA and TV-14.

Genre: The number of movies/shows for each genre is mapped. It is found that 'Internation Movies' genre has 2574(highest) count and 'TV Shows' genre has 11(least) count.

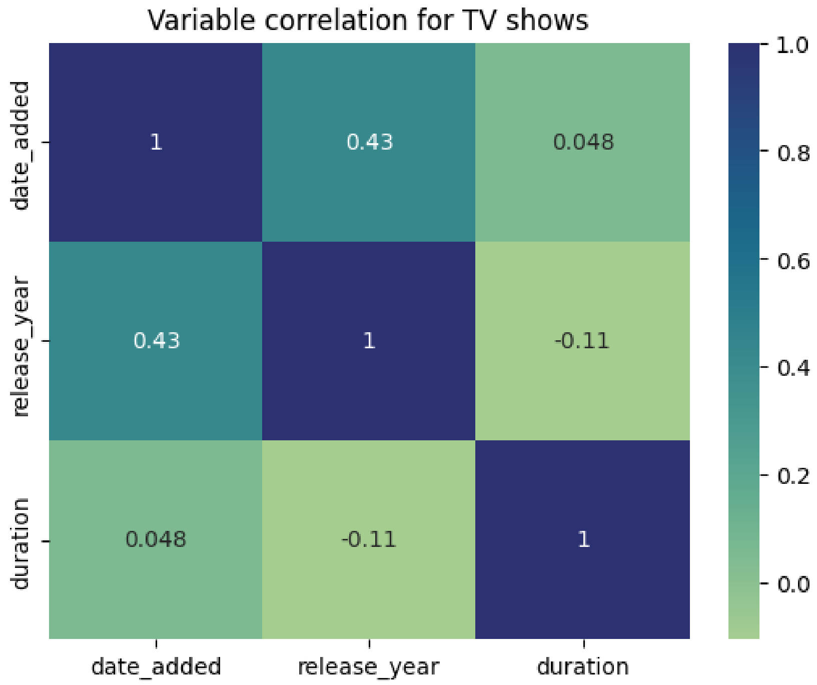
6.2 Distribution of variables and relation between them

In [326]:

```
# Variable correlation for TV shows
final2 = final.copy()
final2["date_added"] = final2["date_added"].apply(lambda x: x.value)/100000000000
final2 = final2.loc[final2["type"] =="TV Show", ["title","date_added","release_year", "duration"]].drop_duplicates
sns.heatmap(final2.corr(), cmap = "crest", annot = True)
plt.title("Variable correlation for TV shows")
```

Out[326]:

```
Text(0.5, 1.0, 'Variable correlation for TV shows')
```

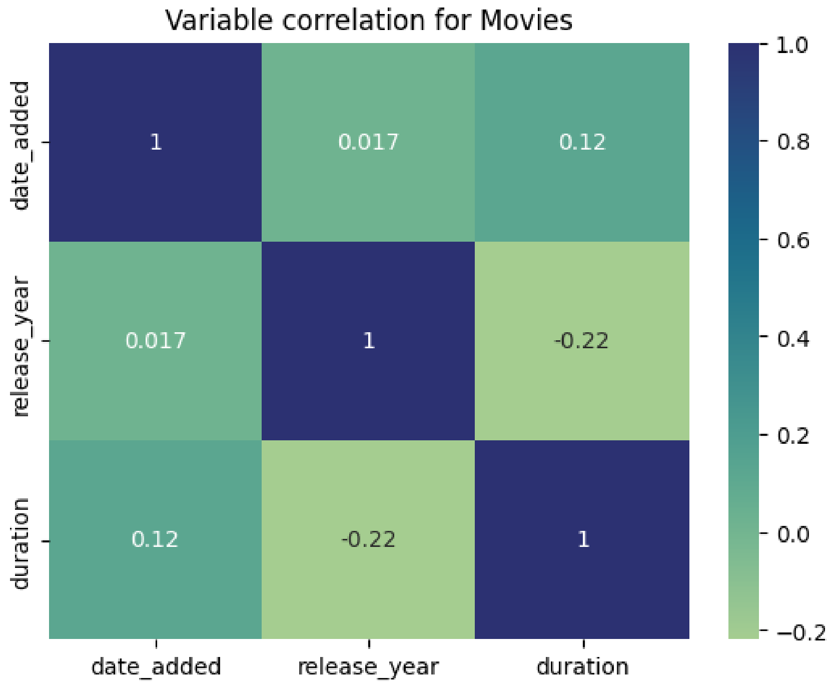


In [327]:

```
# Variable correlation for Movies
final2 = final.copy()
final2["date_added"] = final2["date_added"].apply(lambda x: x.value)/100000000000
final2 = final2.loc[final2["type"] =="Movie", ["title","date_added","release_year", "duration"]].drop_duplicates(
sns.heatmap(final2.corr(), cmap = "crest", annot = True)
plt.title("Variable correlation for Movies")
```

Out[327]:

```
Text(0.5, 1.0, 'Variable correlation for Movies')
```



It is seen that 'release year' and date added' variables are mildly related, which makes sense because older movies/shows added in the beginning, and over the years as and when new ones came, they were added on the platform. There is no relation between 'duration' and 'date added'. However 'duration' and 'release year' have negative correlation which means the duration of movies/shows have have slightly decreased over the years.

7. Business Insights

1. Country: There are 113 countries but most of the movies/shows come from these top 5 countries - US, India, UK, Canada and France.

2. Successfull directors: Marcus Raboy, Martin Campbell, Toshiya Shinohara

3. We see that 70% of the content on netflix is Movies and 30% is TV Shows.

In [337]:

```
final.groupby("type")["title"].apply(lambda x: x.nunique())*100/final.groupby("type")["title"].apply(lambda x: x.
```

Out[337]:

```
type
Movie      70.875125
TV Show    29.124875
Name: title, dtype: float64
```

4. Successfull Actors: Anupam Kher and Shah rukh khan have been featured in the most number of movies. And the top actors list is dominated my India.

5. Top Genre: The top 3 Genres are 'International Movies', 'Drama' and 'Comedy'.

6. Duration: The median duration for Movies and TV shows are 1h 40mins and 1 season respectively.

7. Genre: Anime and Classical Movie genre are becoming popular recently.

8. Genre duration: We observe median duration of 'classical movies' is the highest and the genre of 'Movies' is the least.

9. Favourite genre in the biggest markets: Popular genre in US is 'Drama' and in India it is 'International Movies'.

In [332]:

```python
# Let us look at the Director - Cast combination.
data_ = final.loc[:, ["Cast", "title", "director"]].drop_duplicates()
data_ = data_.groupby(["director","Cast"]).count().sort_values(by = "title", ascending = False).reset_index()
data_.head(20)
```

Out[332]:

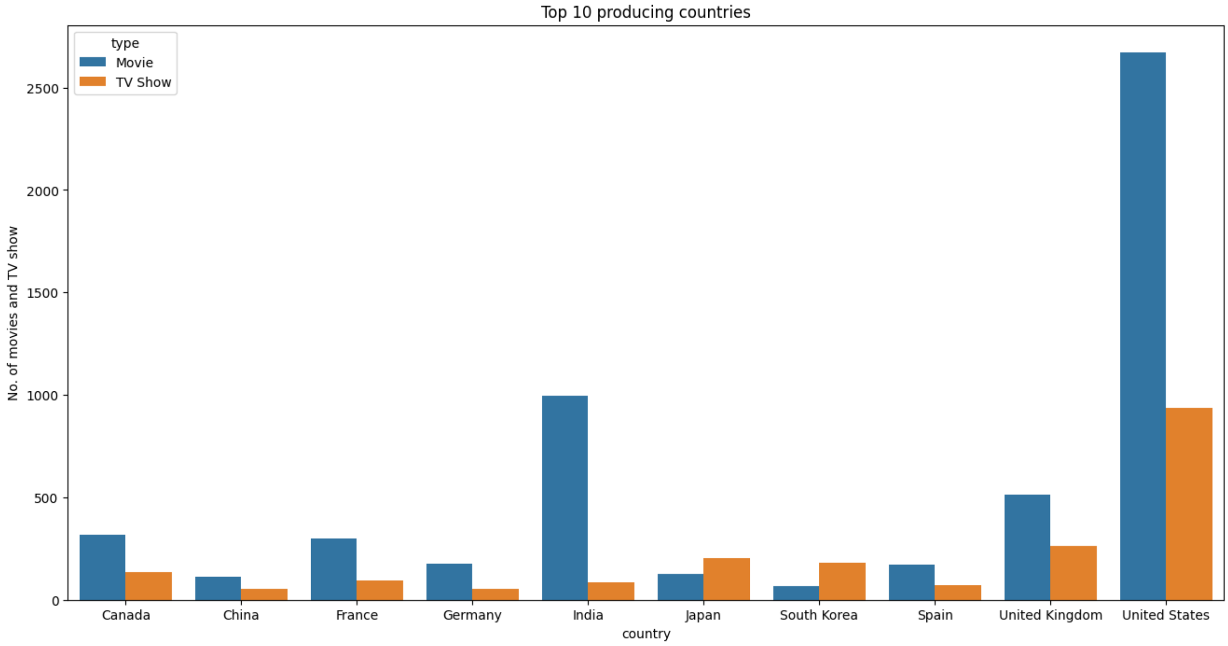
	director	Cast	title
0	Rajiv Chilaka	Rajesh Kava	22
1	Rajiv Chilaka	Julie Tejwani	21
2	Toshiya Shinohara	Takahiro Sakurai	21
3	Rajiv Chilaka	Rupa Bhimani	20
4	Rajiv Chilaka	Jigna Bhardwaj	20
5	Rajiv Chilaka	Vatsal Dubey	17
6	Toshiya Shinohara	Yuki Kaji	17
7	Toshiya Shinohara	Daisuke Ono	16
8	Toshiya Shinohara	Junichi Suwabe	15
9	Toshiya Shinohara	Yoshimasa Hosoya	15
10	Toshiya Shinohara	Yuichi Nakamura	15
11	Toshiya Shinohara	Jun Fukuyama	14
12	Rajiv Chilaka	Swapnil	14
13	Toshiya Shinohara	Ai Kayano	14
14	Rajiv Chilaka	Mousam	14
15	Toshiya Shinohara	Hiroshi Kamiya	13
16	Martin Campbell	David Attenborough	13
17	Toshiya Shinohara	Natsuki Hanae	12
18	Toshiya Shinohara	Kana Hanazawa	12
19	Toshiya Shinohara	Nobuhiko Okamoto	12

10. Director - Cast combo: We see that the which Actor/Director combination have been featured the most.

11. In Japan and South Korea, TV shows are more popular than movies. Rest of the remaining top countries, movies are more popular than TV shows.

In [96]:

```
#Top 10 countries and their distribution of movies and TV shows
data_ = final.loc[:, ["type", "title", "country"]].drop_duplicates()
data_ = data_.groupby(["country", "type"])["title"].count().reset_index()
top_country = final.groupby("country").apply(lambda x: x["title"].nunique()).sort_values(ascending = False).head(
data_ = data_[data_["country"].isin(top_country)]

plt.figure(figsize=(16, 8))
sns.barplot(data = data_, x = "country", y ="title", hue = "type")
plt.ylabel("No. of movies and TV show")
plt.title("Top 10 producing countries")
plt.show()
```



In [235]:

```
final
```

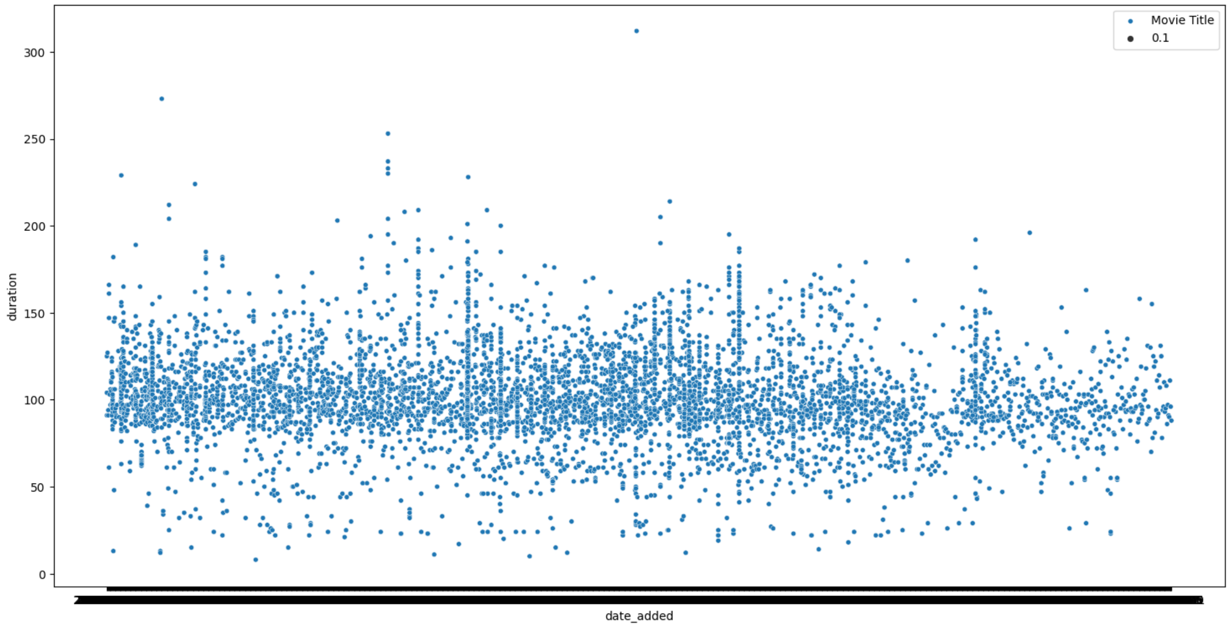
Out[235]:

	show_id	type	title	date_added	release_year	rating	duration	description	Cast	country	genre
0	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2.0	After crossing paths at a party, a Cape Town t...	Ama Qamata	South Africa	Internationa TV Shows
1	s3	TV Show	Ganglands	2021-09-24	2021	TV-MA	1.0	To protect his family from a powerful drug lor...	Sami Bouajila	India	Crime TV Shows
2	s5	TV Show	Kota Factory	2021-09-24	2021	TV-MA	2.0	In a city of coaching centers known to train I...	Mayur More	India	Internationa TV Shows
3	s6	TV Show	Midnight Mass	2021-09-24	2021	TV-MA	1.0	The arrival of a charismatic young priest brin...	Kate Siegel	United States	TV Dramas
4	s7	Movie	My Little Pony: A New Generation	2021-09-24	2021	PG	91.0	Equestria's divided. But a bright-eyed hero be...	Vanessa Hudgens	United States	Children & Family Movies
...	...	...	...	...	...	...	...	...	...	...	..
207135	s5888	Movie	Walt Disney Animation Studios Short Films Coll...	2015-10-25	2015	TV-Y	90.0	This collection of 12 short films from Disney ...	Dave Foley	United States	Children & Family Movies
207136	s5888	Movie	Walt Disney Animation Studios Short Films Coll...	2015-10-25	2015	TV-Y	90.0	This collection of 12 short films from Disney ...	Derek Richardson	United States	Children & Family Movies
207137	s5888	Movie	Walt Disney Animation Studios Short Films Coll...	2015-10-25	2015	TV-Y	90.0	This collection of 12 short films from Disney ...	Betty White	United States	Children & Family Movies
207138	s5888	Movie	Walt Disney Animation Studios Short Films Coll...	2015-10-25	2015	TV-Y	90.0	This collection of 12 short films from Disney ...	Zachary Levi	United States	Children & Family Movies
207139	s5888	Movie	Walt Disney Animation Studios Short Films Coll...	2015-10-25	2015	TV-Y	90.0	This collection of 12 short films from Disney ...	Mandy Moore	United States	Children & Family Movies

207137 rows × 12 columns

In [108]:

```
data_ = final.loc[final["type"]=="Movie", ["title", "date_added", "duration"]].drop_duplicates()
plt.figure(figsize=(18, 9))
sns.scatterplot(data = data_, x = "date_added", y ="duration", size = 0.1, label = "Movie Title")
plt.show()
```



8. Recommendations

1. Country: There are 113 countries but not all of them give the most return. We should focus the content more on important countries which - US, India, UK, Canada and France.

2. Successful directors: Since certain director's movie/show are featured more than others, Netflix can make original movies/show by hiring the top directors. For example: Marcus Raboy, Martin Campbell, Toshiya Shinohara.

3. Successful Actors: If Netflix has the budget to pay for star - studded cast, it can hire popular actors/actress to attract more people into the platform. For example: Anupam Kher, Shah Rukh Khan, Takahiro Sakurai etc,.

4. Director - Cast combo: If Netflix has budget constraint, it can hire successful yet lesser know Director- Cast combination. The best combination is mentioned in the table above.

5. Targeting the right genre for specific countries: Netflix can recommend popular genre to the audience of that country. For example: US - Drama, comedy, India - International Movies, UK - 'British TV Shows', Japan - Anime etc,.

6. Duration: Netflix can give more preference to movies whose duration is around 1h 40mins, and shows with 1 or 2 seasons. Since data suggests, this is the ideal duration.

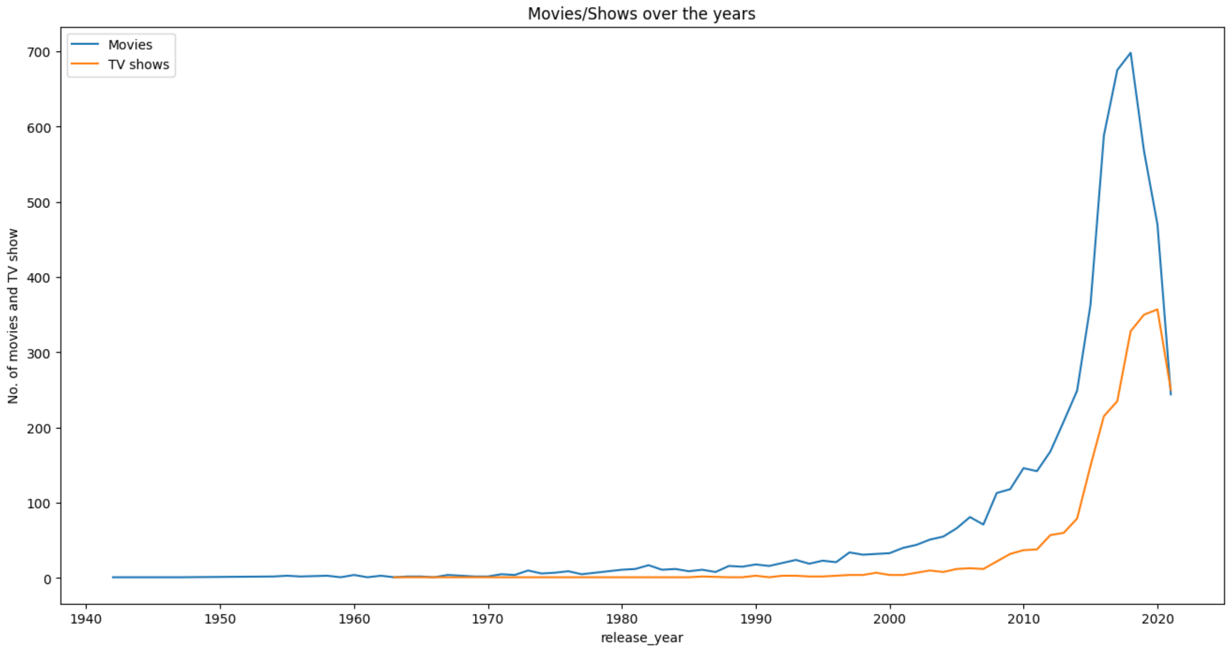
7. Netflix can produce or sponsor more towards specific genres of movies/show. From the data it is visible that specific genre like 'Anime' and 'classical movies' are getting popular recently throughout the world.

8. In countries like Japan and South Korea, Netflix should recommend more TV shows rather than wasting resources on Movies.

9. Should put more content on the platform overall: Because after 2019, the no. of movies/shows added have decreased. People expect latest content.

In [109]:

```
# Movies/Shows released over the years
plt.figure(figsize=(16, 8))
sns.lineplot(data = final[final["type"]=="Movie"].groupby("release_year")["title"].apply(lambda x: x.nunique()), 
sns.lineplot(data = final[final["type"]=="TV Show"].groupby("release_year")["title"].apply(lambda x: x.nunique())
plt.ylabel("No. of movies and TV show")
plt.title("Movies/Shows over the years")
plt.show()
```



10 Rating: If Netflix does produce its original content it should prefer TV-Y, TV-G rating category. Since they are more popular recently.

In [ ]:

```
```

In [ ]:

```
```