Netflix - Data Exploration and Visualisation

Problem Statement

Deciding on Which Type of Shows/Movies to Include

Analysing Basic Metrics

In [30]:
```
import datetime
start_time = datetime.datetime.now()  # Setup a timestamp for the start of the script
```

In [31]:
```
# Importing required modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

In [32]:
```
# Retreiving the Netflix dataset
# netflix_df = pd.read_csv('netflix.csv')
netflix_df = pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/00
netflix_df.head(2)
```

Out[32]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	du
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	S

In [33]:
```
rows, columns = netflix_df.shape
print(f'The dataset has {rows} rows and {columns} columns')
```

```
The dataset has 8807 rows and 12 columns
```

In [34]:
```
# Quick Overview of the dataset
netflix_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 [35]:
```
# Finding the percentage of Null Values
netflix_df.isnull().sum()/netflix_df.shape[0]*100
```

Out[35]:
```
show_id          0.000000
type             0.000000
title            0.000000
director        29.908028
cast             9.367549
country          9.435676
date_added       0.113546
release_year     0.000000
rating           0.045418
duration         0.034064
listed_in        0.000000
description      0.000000
dtype: float64
```

Inference: 30% of Directors, 10 % of Cast and Country and 0.1% of Added Date values are not available

Action Item: Dropping the NA fields or filling the default value is not recommended. Since it's a kind of grouping NA values together. To resolve that the imputation needs to be done

In [36]:
```
# Columns having nested values and their count
for col in netflix_df.columns:
    if(netflix_df[col].dtype == 'object'):
        print(f'{col} : {netflix_df[col].str.contains(",").sum()}')
```

```
show_id : 0
type : 0
title : 138
director : 614
cast : 7101
country : 1320
date_added : 8797
rating : 0
duration : 0
listed_in : 6787
description : 6448
```

Inference:

- Assuming Title and Date Added are not having the nested values based on the analysis did on sample data.
- Nested Columns: Director, Cast, Country, Listed In/Genre
- Other Columns needs a pre-processing because of a nested values

In [37]:
```
netflix_df.nunique()
```

Out[37]:
```
show_id         8807
type               2
title           8807
director        4528
cast            7692
country          748
date_added      1767
release_year      74
rating            17
duration         220
listed_in        514
description     8775
dtype: int64
```

Inference:

- Title and Show ID can be used as a primary key since all the values are unique.
- Only Movies and Series Type data are available.
- The shows are categorized for 17 different Category of people (No Nested Values available)

Pre-Processing of Data

In [38]:
```
# Dropping Description column - Not required for analysis without NLP
netflix_df.drop(['description'], axis=1, inplace=True)
```

In [39]:
```
# remove strings from the duration column values and make it int datatype
netflix_df['duration'] = netflix_df['duration'].str.replace('Seasons', '').str.replace(
netflix_df['duration'] = netflix_df['duration'].astype(float)
```

In [40]:
```
# Change the datatype of the date_added column to datetime
netflix_df['date_added'] = pd.to_datetime(netflix_df['date_added'].str.strip())
```

In [41]:
```
# Setting Show ID as index
netflix_df.set_index('show_id', inplace=True)
netflix_df.head(2)
```

Out[41]:

	type	title	director	cast	country	date_added	release_year	rating	durat
show_id									
s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	2020	PG-13	(
s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	TV-MA	

In [42]:
```
# Function to convert Nested Values to Rows
def convert_nested_columns_to_rows(df, column):
    nested_df = df[column].str.split(',', expand=True)
    nested_df = pd.DataFrame(nested_df.stack()).reset_index()
    nested_df.columns = ['show_id', f'{column}_no', column]
    nested_df.drop(columns=[f'{column}_no'], inplace=True)
    return nested_df

# Nested Columns - Director, Cast, Country, Listed In
director_df = convert_nested_columns_to_rows(netflix_df, 'director')
cast_df = convert_nested_columns_to_rows(netflix_df, 'cast')
country_df = convert_nested_columns_to_rows(netflix_df, 'country')
listed_in_df = convert_nested_columns_to_rows(netflix_df, 'listed_in')

# Merging the Nested Columns with the main dataset
director_cast_df = pd.merge(director_df, cast_df, on='show_id', how='outer')
country_listed_in_df = pd.merge(country_df, listed_in_df, on='show_id', how='outer')
merged_df = pd.merge(director_cast_df, country_listed_in_df, on='show_id', how='outer')
merged_df.drop_duplicates(inplace=True)
merged_df.head()
```

Out[42]:

	show_id	director	cast	country	listed_in
0	s1	Kirsten Johnson	NaN	United States	Documentaries
1	s10	Theodore Melfi	Melissa McCarthy	United States	Comedies
2	s10	Theodore Melfi	Melissa McCarthy	United States	Dramas
3	s10	Theodore Melfi	Chris O'Dowd	United States	Comedies
4	s10	Theodore Melfi	Chris O'Dowd	United States	Dramas

In [43]:
```
# Filtering the main dataset without nested columns
filterd_netflix_df = netflix_df.loc[:, netflix_df.columns.difference(['director', 'cast

# Merging the main dataset with the merged dataset
flattened_netflix_df = pd.merge(filterd_netflix_df, merged_df, on='show_id', how='left'
flattened_netflix_df.head(2)
```

Out[43]:

	show_id	date_added	duration	rating	release_year	title	type	director	cast	cou
0	s1	2021-09-25	90.0	PG-13	2020	Dick Johnson Is Dead	Movie	Kirsten Johnson	NaN	Ui S
1	s2	2021-09-24	2.0	TV-MA	2021	Blood & Water	TV Show	NaN	Ama Qamata	S A

Inference: All the Nested Values are Flattened and created a new dataset

In [44]:
```
# Filling missing values in the 'Duration' column with the median value, grouped by the
flattened_netflix_df['duration'] = flattened_netflix_df.groupby(['type',"listed_in", "r

# Filling missing values in the 'Rating' column with the mode value, grouped by the 'ty
flattened_netflix_df['rating'] = flattened_netflix_df.groupby(['type', "listed_in", "re

# Filling missing values in the "Date Added" column with the median value, grouped by t
flattened_netflix_df['date_added'] = flattened_netflix_df.groupby(['type', "listed_in",
flattened_netflix_df['date_added'] = flattened_netflix_df.groupby(['type', "listed_in",
flattened_netflix_df['date_added'] = flattened_netflix_df.groupby(['type', "listed_in"]

# Filling missing values in the "Country" column with the mode value, grouped by the 't
flattened_netflix_df['country'] = flattened_netflix_df.groupby(['type', "listed_in", "r
# Still some Values are missing and filling those values in the "Country" column with t
flattened_netflix_df['country'] = flattened_netflix_df.groupby(['type', "listed_in"])['


# Filling missing values in the "Cast" column with the mode value, grouped by the 'type
flattened_netflix_df["cast"] = flattened_netflix_df.groupby(['type', "listed_in", "cour
# Still some Values are missing and filling missing values in the "Cast" column with th
flattened_netflix_df["cast"] = flattened_netflix_df.groupby(['type', "listed_in", "cour
# Still some Values are missing and filling missing values in the "Cast" column with th
flattened_netflix_df["cast"] = flattened_netflix_df.groupby(['type', "listed_in"])['cas


# Filling missing values in the "director" column with the mode value, grouped by the
flattened_netflix_df["director"] = flattened_netflix_df.groupby(['type', "listed_in", '
flattened_netflix_df["director"] = flattened_netflix_df.groupby(['type', "listed_in", '
flattened_netflix_df['director'] = flattened_netflix_df.groupby(['type', "listed_in", '
flattened_netflix_df['director'] = flattened_netflix_df.groupby(['type', "listed_in", '
flattened_netflix_df['director'] = flattened_netflix_df.groupby(['type', "listed_in"])[
flattened_netflix_df['director'] = flattened_netflix_df.groupby(['type'])['director'].t

flattened_netflix_df.isna().sum()
```

```
c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:
```

```
Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice

c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice
```

```
c:\Pthon3117\Lib\site-packages\numpy\lib\nanfunctions.py:1215: RuntimeWarning:

Mean of empty slice
```

Out[44]:
```
show_id         0
date_added      0
duration        0
rating          0
release_year    0
title           0
type            0
director        0
cast            0
country         0
listed_in       0
dtype: int64
```

Non-Graphical Analysis

In [45]:
```
#  Describe the Numerical Columns in dataset
movies_df = flattened_netflix_df[flattened_netflix_df['type'] == 'Movie']
shows_df = flattened_netflix_df[flattened_netflix_df['type'] == 'TV Show']
```

In [46]:
```
stack_1 = movies_df[movies_df.select_dtypes(['int', 'float', 'datetime']).columns].desc
stack_1.columns = [f'{col}_Movies' for col in stack_1.columns]
stack_2 = shows_df[shows_df.select_dtypes(['int', 'float', 'datetime']).columns].descri
stack_2.columns = [f'{col}_Shows' for col in stack_2.columns]

# stack horizontally
pd.concat([stack_1, stack_2], axis=1)
```

Out[46]:

	date_added_Movies	duration_Movies	release_year_Movies	date_added_Shows	duration_
count	145910	145910.000000	145910.000000	56148	56148.0
mean	2019-06-14 21:48:07.747241984	106.839792	2012.131574	2019-07-01 00:10:27.826458368	1.9
min	2008-01-01 00:00:00	3.000000	1942.000000	2008-02-04 00:00:00	1.0
25%	2018-07-01 00:00:00	93.000000	2010.000000	2018-05-04 00:00:00	1.0
50%	2019-08-29 00:00:00	104.000000	2016.000000	2019-10-04 00:00:00	1.0
75%	2020-08-28 00:00:00	119.000000	2018.000000	2020-10-15 00:00:00	2.0
max	2021-09-25 00:00:00	312.000000	2021.000000	2021-09-24 00:00:00	17.0
std	NaN	24.711015	9.815637	NaN	1.8

Inference for Movies

- Entire Data is in the range of 01-01-2008 and 25-09-2021
- On an average, Movie will take 106 minutes

- 50% of the movies were added on 2019 which means it took 11 years(2008-2019) in starting stage to include in netflix. In other words, more number of movies were included in the recent years (2 Years)

Inference for TV Shows

- Same like Movies, TV shows also included in the same range (2008-2021). If we compare dates, Movies were included first.
- On an average, all the TV Shows have 2 seasons
- 50% of the Shows were included with the last 3 years

In [47]:
```
# Describe the Categorical Columns in dataset
stack_1 = movies_df[movies_df.select_dtypes(['object']).columns].describe()
stack_1.columns = [f'{col}_Movies' for col in stack_1.columns]
stack_2 = shows_df[shows_df.select_dtypes(['object']).columns].describe()
stack_2.columns = [f'{col}_Shows' for col in stack_2.columns]

# stack horizontally
pd.concat([stack_1, stack_2], axis=1)
```

Out[47]:

	show_id_Movies	rating_Movies	title_Movies	type_Movies	director_Movies	cast_Movie
count	145910	145910	145910	145910	145910	14591
unique	6131	17	6131	1	4886	2787
top	s7165	TV-MA	Kahlil Gibran's The Prophet	Movie	Martin Scorsese	Russe Simmor
freq	700	44016	700	145910	449	18

Inference for Movies

- There are 17 unique categories for the movies
- There are 4886 directors and 27879 Crew Member, who's movies were added into the Netflix
- There are 37 Genre of movies
- Top Director was Martin Scorsese and Top Country was US. Likewise, other top rated categories were listed in the table.

Inference for Shows

- There are 9 unique categories for the shoes
- There are 300 directors and 15501 Crew Member, who's shoes were added into the Netflix
- There are 36 Genre of Shoes
- Top Director was Danny Cannon and Top Country was US. Likewise, other top rated categories were listed in the table.

In [48]:
```
# Extracting the Unique Values in the dataset which is a part of describe function

movies_unique = movies_df.nunique()
shows_unique = shows_df.nunique()
```

```
# stack horizontally
unique_df = pd.concat([movies_unique, shows_unique], axis=1)
unique_df.columns = ['Movies', 'Shows']
unique_df
```

Out[48]:

	Movies	Shows
show_id	6131	2676
date_added	1533	1012
duration	205	15
rating	17	9
release_year	73	46
title	6131	2676
type	1	1
director	4886	300
cast	27879	15501
country	187	102
listed_in	37	36

Inference

- Showing the unique count for all the fields
- There are 6131 Movies and 2676 TV Shows available in Netflix
- There are 4886 Movie Directors and 300 TV Show Directors who's contents were included in Netflix

In [49]:
```
# Extracting the Highest number of Contents year wise
no_of_movies_df = netflix_df.loc[netflix_df["type"]=="Movie"]["release_year"].value_cou
print(f'The year with the most number of movies released is {no_of_movies_df.index[0]}

no_of_shows_df = netflix_df.loc[netflix_df["type"]=="TV Show"]["release_year"].value_co
print(f'The year with the most number of TV Shows released is {no_of_shows_df.index[0]


# Extracting the most available Genre in Netflix
no_movies_genre = movies_df["listed_in"].value_counts().head(1)
print(f'The most available genre in Movies is "{no_movies_genre.index[0].strip()}" with

no_shows_genre = shows_df["listed_in"].value_counts().head(1)
print(f'The most available genre in TV Shows is "{no_shows_genre.index[0].strip()}" wit
```

```
The year with the most number of movies released is 2017 with 767 movies
The year with the most number of TV Shows released is 2020 with 436 TV Shows
The most available genre in Movies is "International Movies" with 27138 movies
The most available genre in TV Shows is "TV Dramas" with 7956 TV Shows
```

Inference

- Extracted few information like Maximum number of movies release and their corresponding year
- In Movies type, 767 movies were included on 2017

- In TV Shows type, 436 shows were included on 2020
- In Movies type, "International Movies" is the major Genre with 27138 Movies
- In TV Shows type, "TV Dramas" is the major Genre with 7956 Movies

Visual Analysis

Univariate Analysis

In [50]:
```
# Top 3 Directors, Genre, Country
top3_movies_directors = movies_df.groupby("director")["show_id"].nunique().sort_values(
top3_movies_genre = movies_df.groupby("listed_in")["show_id"].nunique().sort_values(asc
top3_movies_country = movies_df.groupby("country")["show_id"].nunique().sort_values(asc

top3_movies_data = movies_df[(movies_df["director"].isin(top3_movies_directors)) & (mov
top3_movies_data.size


top3_shows_directors = shows_df.groupby("director")["show_id"].nunique().sort_values(as
top3_shows_genre = shows_df.groupby("listed_in")["show_id"].nunique().sort_values(ascen
top3_shows_country = shows_df.groupby("country")["show_id"].nunique().sort_values(ascen

top3_shows_data = shows_df[(shows_df["director"].isin(top3_shows_directors)) & (shows_d
top3_shows_data.size

# stack both the dataframes
top3_data = pd.concat([top3_movies_data, top3_shows_data], axis=0)
top3_data.head()
```

Out[50]:

	show_id	date_added	duration	rating	release_year	title	type	director	cast	co
1018	s42	2021-09-16	124.0	PG	1975	Jaws	Movie	Steven Spielberg	Roy Scheider	
1020	s42	2021-09-16	124.0	PG	1975	Jaws	Movie	Steven Spielberg	Roy Scheider	
1021	s42	2021-09-16	124.0	PG	1975	Jaws	Movie	Steven Spielberg	Robert Shaw	
1023	s42	2021-09-16	124.0	PG	1975	Jaws	Movie	Steven Spielberg	Robert Shaw	
1024	s42	2021-09-16	124.0	PG	1975	Jaws	Movie	Steven Spielberg	Richard Dreyfuss	

In [51]:
```
movies_grp_df = movies_df.groupby(['title', 'date_added']).size().reset_index(name='cou
fig = px.histogram(movies_grp_df, x="date_added", barmode='group')
fig.update_layout(title='Date_added Distribution in Movies', xaxis_title='Date Added',
fig.show()

shows_grp_df = shows_df.groupby(['title', 'date_added']).size().reset_index(name='count
fig = px.histogram(shows_grp_df, x="date_added", barmode='group')
fig.update_layout(title='Date_added Distribution in TV Shows', xaxis_title='Date Added'
fig.show()
```

Histogram Inference

- From the Movies Graph, more number of movies were added on November 2019 - December 2019 which was around 355
- From the TV Shows Graph, more number of shows were added on July 2021 - Augest 2021 which was around 149
- After the end of 2015, so many number of Movies and TV Shows were added drastically.
- There was a drop in adding a new contents in the Mid of 2018. But, after that, Netflix managed to increase the number of contents.

In [52]:
```
shows_grp_df.head(10)
```

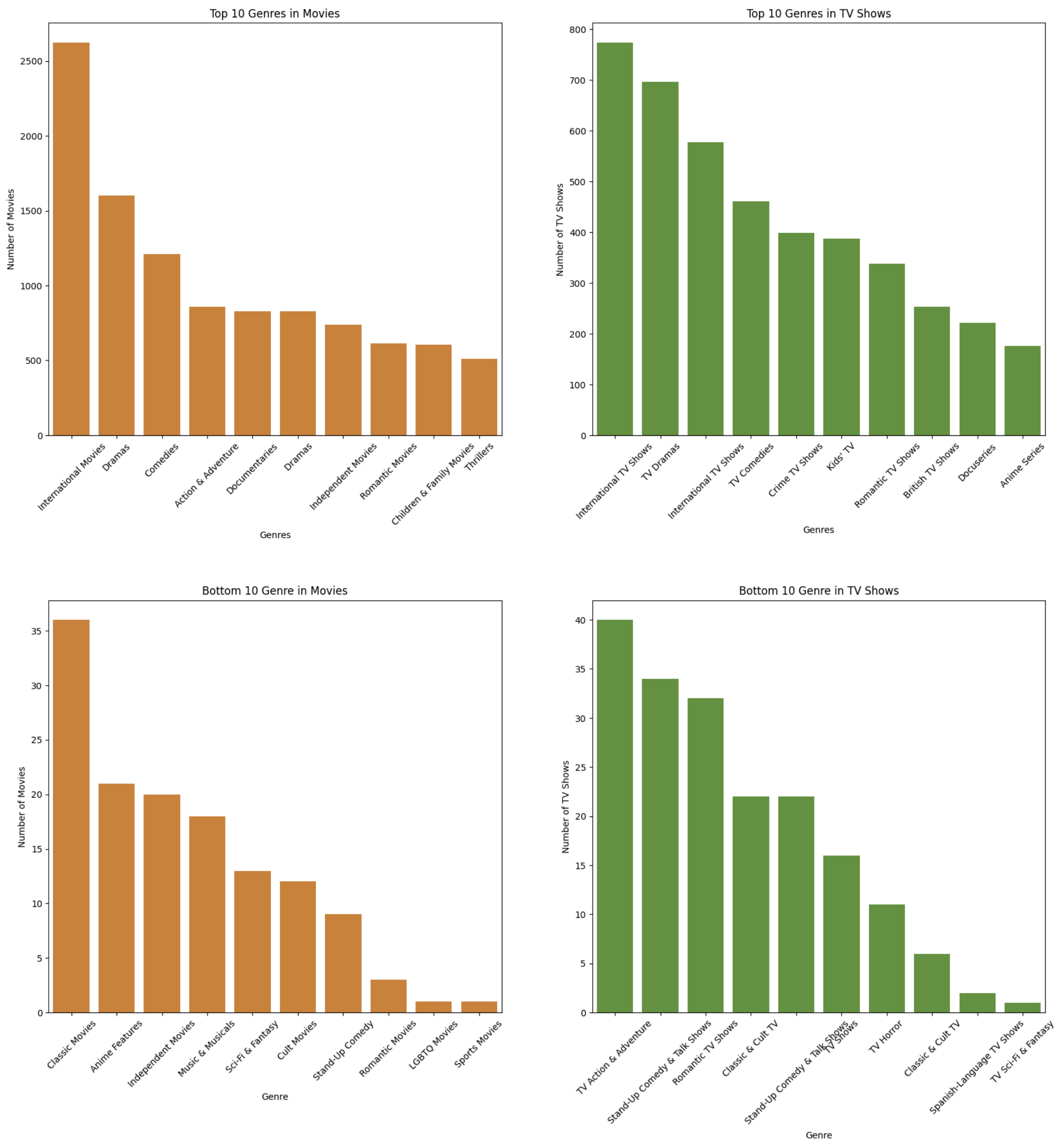
Out[52]:

	title	date_added	count
0	#blackAF	2020-04-17	7
1	(Un)Well	2020-08-12	1
2	100 Days My Prince	2020-12-07	18
3	100 Humans	2020-03-13	6
4	100% Hotter	2019-11-01	12
5	12 Years Promise	2017-05-22	18
6	13 Reasons Why	2020-06-05	39
7	13 Reasons Why: Beyond the Reasons	2019-08-23	56
8	1983	2018-11-30	72
9	1994	2019-05-17	3

In [53]:
```
# shows_grp_df["listed_in"].value_counts().index[-11:-1:]
```

In [54]:
```
def draw_counterplot(data, x, title, xlabel, ylabel, color, start, end=None, step=None)
    sns.countplot(data=data, x=x, order=data[x].value_counts().index[start:end:step], c
    plt.title(title)
    plt.xticks(rotation=45)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)

plt.figure(figsize=(20, 20))
movies_grp_df = movies_df.groupby(['title', 'listed_in']).size().reset_index(name='cour
movies_grp_df.sort_values(by='count', ascending=False, inplace=True)
plt.subplot(2,2,1)
draw_counterplot(movies_grp_df, 'listed_in', 'Top 10 Genres in Movies', 'Genres', 'Numb

shows_grp_df = shows_df.groupby(['title', 'listed_in']).size().reset_index(name='count'
shows_grp_df.sort_values(by='count', ascending=False, inplace=True)
plt.subplot(2,2,2)
draw_counterplot(shows_grp_df, 'listed_in', 'Top 10 Genres in TV Shows', 'Genres', 'Nun

plt.subplots_adjust(hspace=0.4)  # Adjusts the height between subplots

plt.subplot(2,2,3)
draw_counterplot(movies_grp_df, 'listed_in', 'Bottom 10 Genre in Movies', 'Genre', 'Nun
```

```python
plt.subplot(2,2,4)
draw_counterplot(shows_grp_df, 'listed_in', 'Bottom 10 Genre in TV Shows', 'Genre', 'Nu

plt.show()
```



Count Plot Inference

- Graph shows the top and bottom 10 Genre of Movies and TV Shows
- International Movies Genre is the top most one available in Movies Category
- International TV Shows Genre is the top most one available in TV Shows Category
- Sports Movies is the least available one in Movies Category
- TV Sci-Fi & Fantasy is the least available one in the TV Show Category

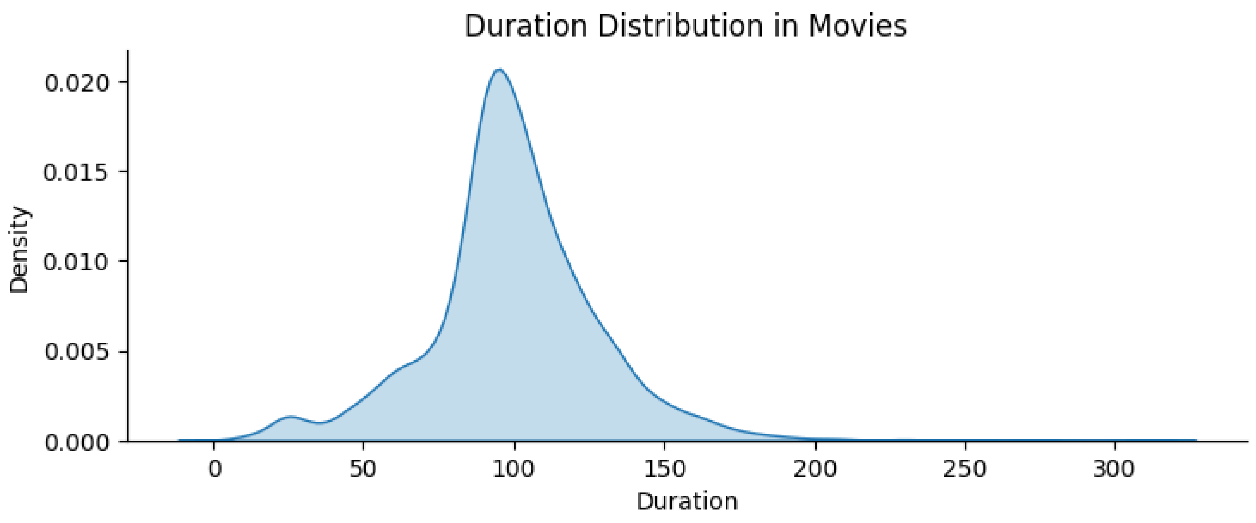
In [55]:
```python
flattened_netflix_df.head()
```

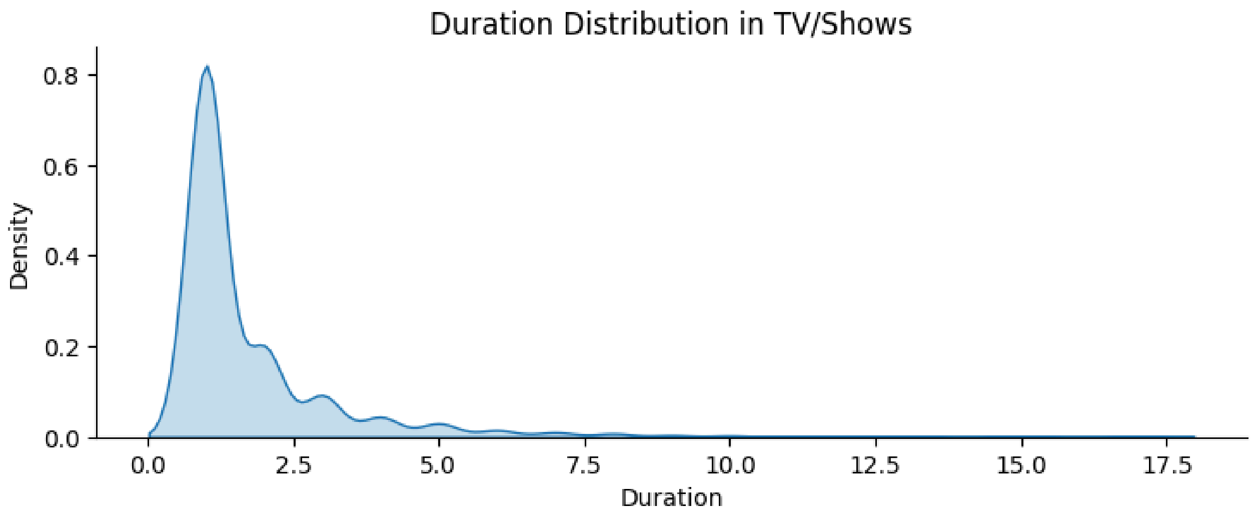
Out[55]:

	show_id	date_added	duration	rating	release_year	title	type	director	cast
0	s1	2021-09-25	90.0	PG-13	2020	Dick Johnson Is Dead	Movie	Kirsten Johnson	Aaron Guy
1	s2	2021-09-24	2.0	TV-MA	2021	Blood & Water	TV Show	Lee Yoon-jung	Ama Qamata
2	s2	2021-09-24	2.0	TV-MA	2021	Blood & Water	TV Show	Danny Cannon	Ama Qamata
3	s2	2021-09-24	2.0	TV-MA	2021	Blood & Water	TV Show	Rob Seidenglanz	Ama Qamata
4	s2	2021-09-24	2.0	TV-MA	2021	Blood & Water	TV Show	Lee Yoon-jung	Khosi Ngema

In [56]:

```
movies_grp_df = movies_df.groupby(['title', 'duration']).size().reset_index(name='count
fig = sns.displot(data=movies_grp_df, x='duration', kind='kde', fill=True, height=3, as
fig.set(title='Duration Distribution in Movies', xlabel='Duration', ylabel='Density')
plt.show()

shows_grp_df = shows_df.groupby(['title', 'duration']).size().reset_index(name='count')
fig = sns.displot(data=shows_grp_df, x='duration', kind='kde', fill=True, height=3, asp
fig.set(title='Duration Distribution in TV/Shows', xlabel='Duration', ylabel='Density')
plt.show()
```





Dist Plot Inference

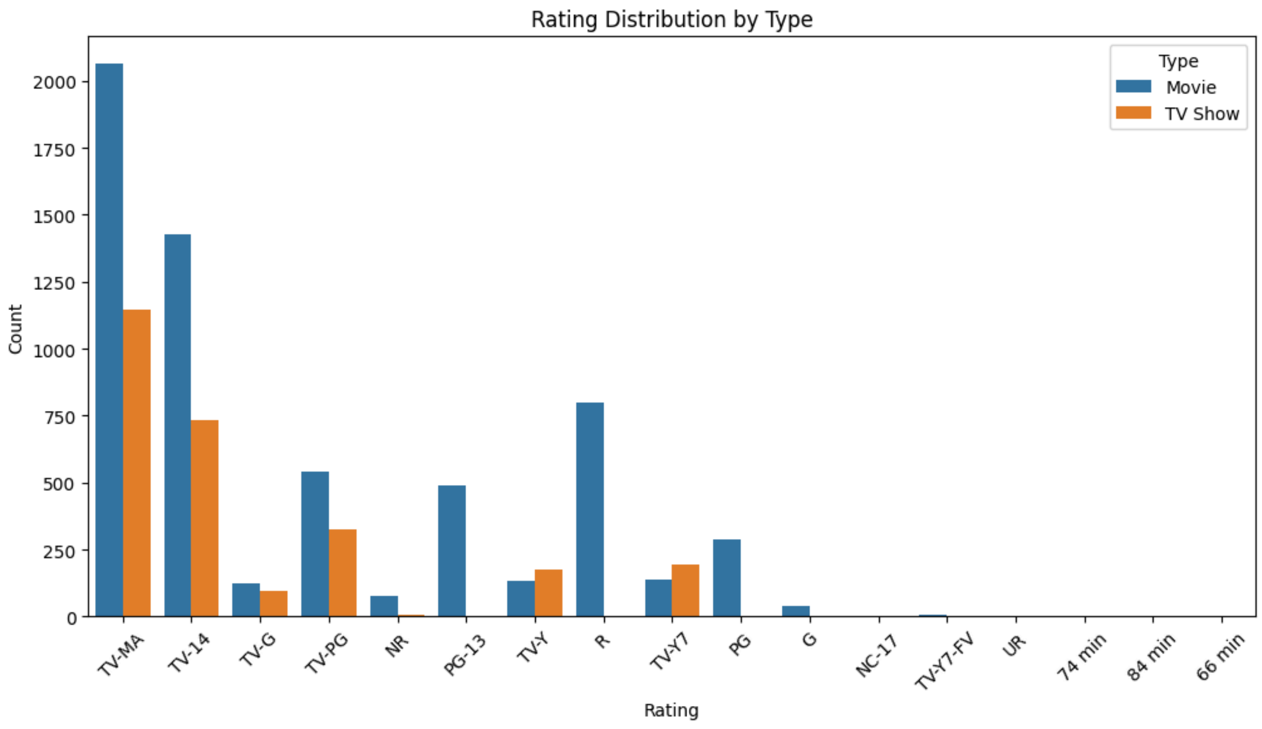
- Graph shows the Distribution of Duration for Movies and TV Shows
- For movies, we can conclude that, most of the movies have the duration of around 100 minutes and only fewer movies have very high and very low duration
- For TV Shows, we can conclude that, most of the shows have 1-2 seasons and only fewer shows have more seasons.

In [57]:
```
grp_df = flattened_netflix_df.groupby(['title', 'date_added', "type"]).size().reset_inc
px.box(grp_df, y='date_added', color="type", title='Added Year Distribution in Movies a
```

Box Plot Inference

- Graph shows the Distribution of Added Year of Movies and TV Shows
- For movies, even thought the data is available from 2008, everything below October, 2014 is considered as outliers
- 50% of the movies were added within June 2019 and 75% of the movies were added by July 2020
- For shows, even thought the data is available from 2008, everything below November, 2014 is considered as outliers
- 50% of the shows were added within August 2019 and 75% of the movies were added by October 2020

In [58]:
```
plt.figure(figsize=(12, 6))
grp_df = flattened_netflix_df.groupby(['title', 'rating', "type"]).size().reset_index(r
sns.countplot(data=grp_df, x='rating', hue='type')
plt.title('Rating Distribution by Type')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Type')
plt.show()
```



Count Plot Inference

- Graph shows the Distribution of Rating by Type
- TV-MA rated movies are included more in Netflix
- Same like Movies, TV-MA rated TV Shows are included more in Netflix

Action Item

- Some rating, shows are not there. Need to include as a future goal
- In some ratings, there is no movie at all. Need to include as a future goal

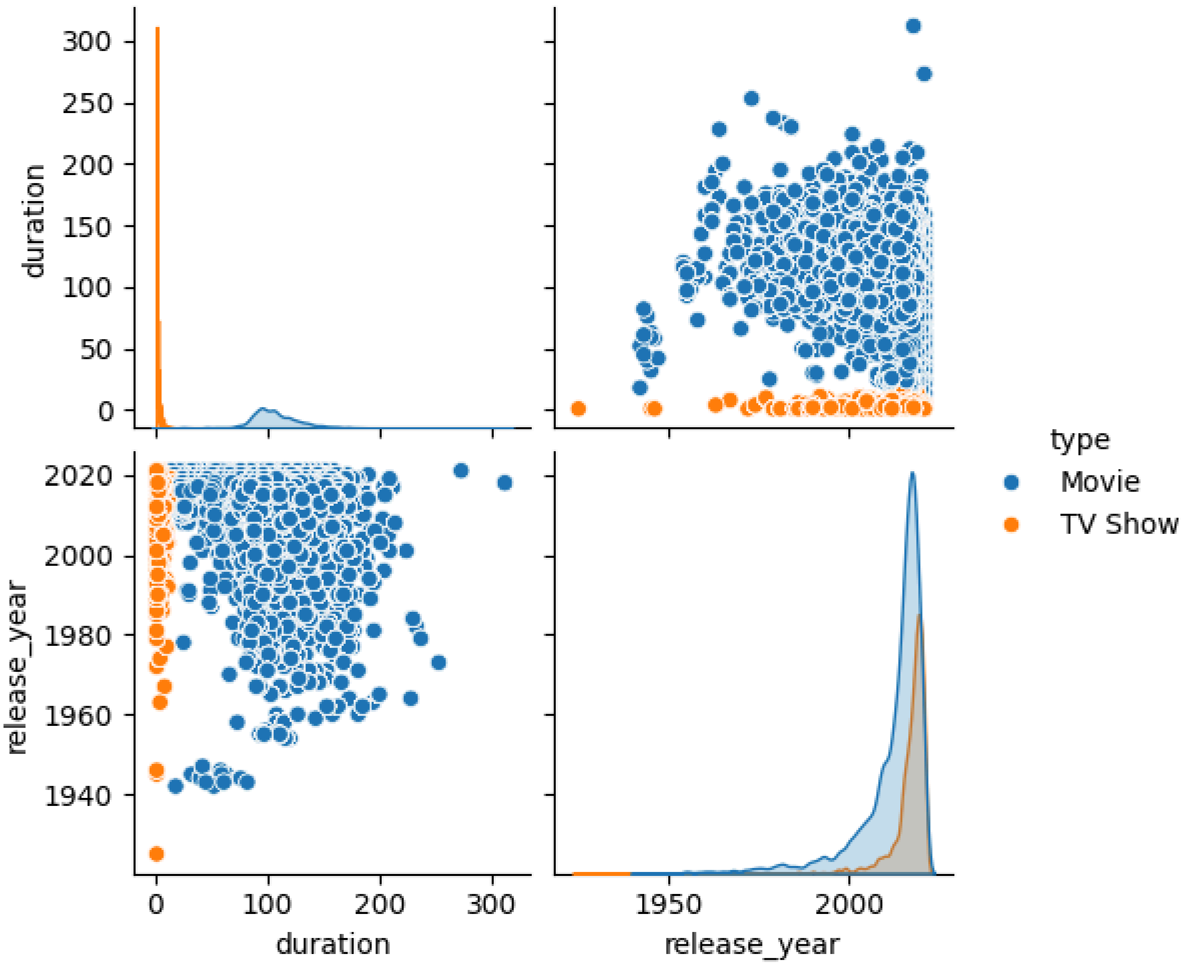
```
In [59]: grp_df = flattened_netflix_df.groupby(['title', "type"]).size().reset_index(name='count

         fig = px.pie(grp_df, names='type', title='Type Distribution in Netflix')
         fig.update_layout(width=800)
         fig.show()
```

Pie Chart Inference

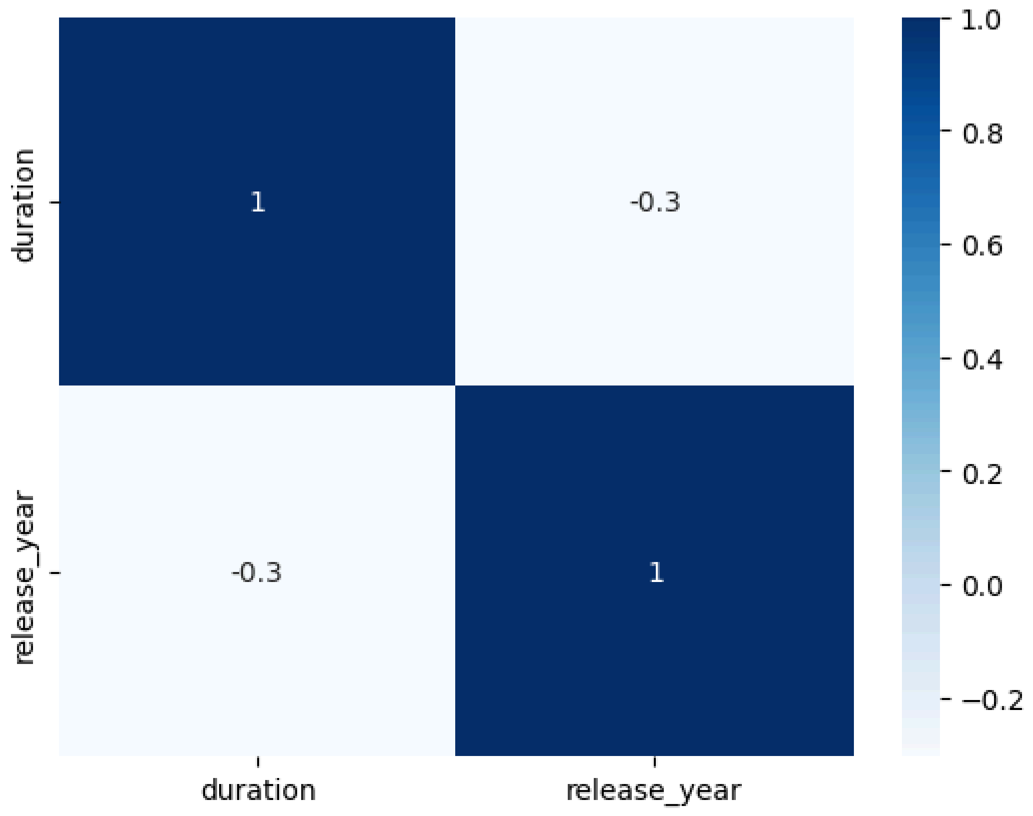
- Out of 100%, approximately 70% of the contents were Movies and the remaining contents were the TV Shoes

```
In [60]: sns.pairplot(data=flattened_netflix_df, hue='type')
```

```
Out[60]: <seaborn.axisgrid.PairGrid at 0x1fd6118da90>
```



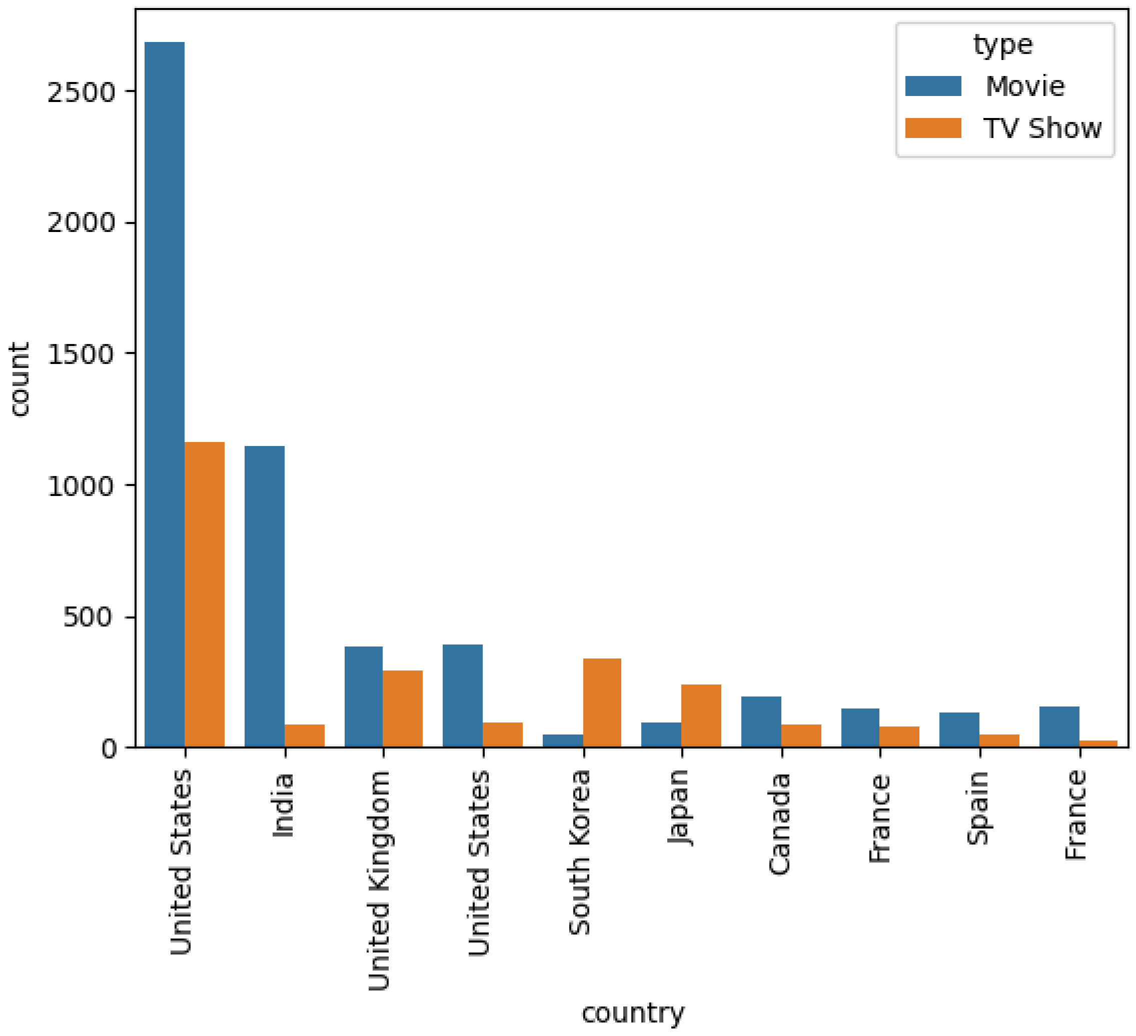
```
In [61]: sns.heatmap(flattened_netflix_df[flattened_netflix_df.select_dtypes(['int', 'float']).c
Out[61]: <Axes: >
```



Heat Map Inference

- Correlation between the release year and duration is negative

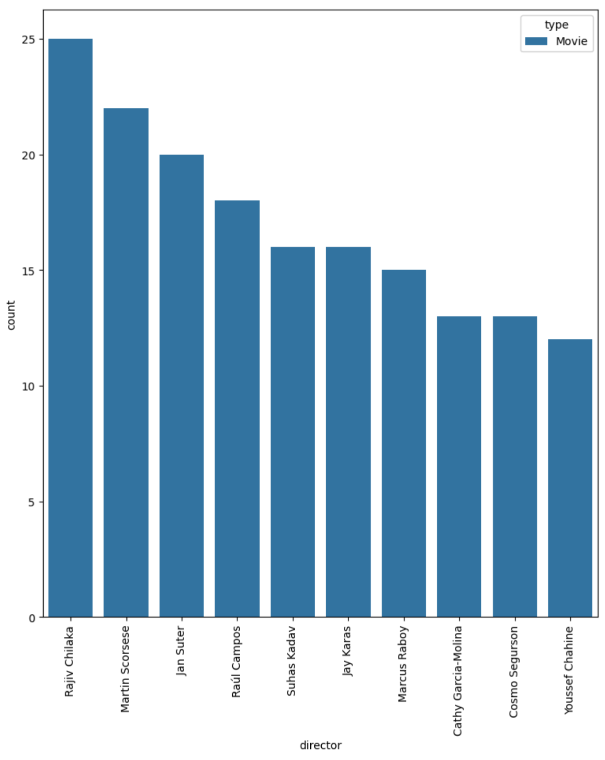
```
In [62]: grp_df = flattened_netflix_df.groupby(['title', "country", "type"]).size().reset_index(
sns.countplot(data=grp_df, x='country', hue="type", order=grp_df['country'].value_count
plt.xticks(rotation=90)
plt.show()
```

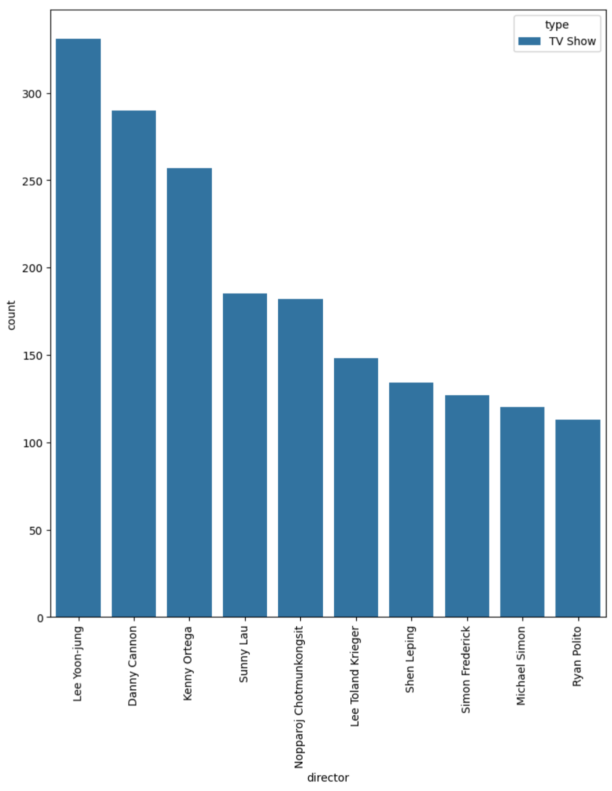


Count Plot Inference

- Graph shows the Country wise Contents available in Netflix
- More number of Movies and TV Shows are from US
- Next to US, more number of movies were came from India
- Likewise, next to US, more number of Shows are coming from Korea

In [63]:
```
plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
movies_grp_df = movies_df.groupby(['title', "director", "type"]).size().reset_index(nar
sns.countplot(data=movies_grp_df, x='director', hue="type", order=movies_grp_df['direct
plt.xticks(rotation=90)
# plt.show()

plt.subplot(1,2,2)
shows_grp_df = shows_df.groupby(['title', "director", "type"]).size().reset_index(name=
sns.countplot(data=shows_grp_df, x='director', hue="type", order=shows_grp_df['director
plt.xticks(rotation=90)
plt.show()
```





Count Plot Inference

- Graph shows the Top 10 Directors and their number of contents
- On Movies section, nearly 25 movies from Rajiv Chilaka were included.
- On TV Shows section, nearly 350 Shows from Lee Yoon-jung were included

In [64]:
```
end_time = datetime.datetime.now()
total_time = end_time - start_time
minutes, seconds = divmod(total_time.total_seconds(), 60)
print(f"Total execution time: {int(minutes)} minutes and {int(seconds)} seconds")
```

```
Total execution time: 4 minutes and 40 seconds
```