Defining Problem Statement and Analysing basic metrics

The goal is to analyze the Netflix dataset to gain insights into the type of content available on the platform, trends over time, and various attributes related to the shows and movies. This analysis will help in understanding the distribution, popularity, and characteristics of the content on Netflix.

In [15]:
```
pip install pydantic==1.10.2
```

```
Collecting pydantic==1.10.2
  Downloading pydantic-1.10.2-cp310-cp310-manylinux_2_17_x86_64.man
ylinux2014_x86_64.whl (12.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 23.3 MB/
s eta 0:00:00
Requirement already satisfied: typing-extensions>=4.1.0 in /usr/loc
al/lib/python3.10/dist-packages (from pydantic==1.10.2) (4.12.2)
Installing collected packages: pydantic
  Attempting uninstall: pydantic
    Found existing installation: pydantic 2.7.3
    Uninstalling pydantic-2.7.3:
      Successfully uninstalled pydantic-2.7.3
ERROR: pip's dependency resolver does not currently take into accou
nt all the packages that are installed. This behaviour is the sourc
e of the following dependency conflicts.
pydantic-settings 2.3.3 requires pydantic>=2.7.0, but you have pyda
ntic 1.10.2 which is incompatible.
ydata-profiling 4.8.3 requires pydantic>=2, but you have pydantic
1.10.2 which is incompatible.
Successfully installed pydantic-1.10.2
```

In [1]:
```
conda install -c conda-forge pandoc
```

```
Collecting package metadata (current_repodata.json): - WARNING cond
a.models.version:get_matcher(535): Using .* with relational operato
r is superfluous and deprecated and will be removed in a future ver
sion of conda. Your spec was 1.7.1.*, but conda is ignoring the .*
and treating it as 1.7.1
done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.12.0
  latest version: 24.5.0

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: /home/sagar/anaconda3

  added / updated specs:
    - pandoc


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-4.14.0               |   py39hf3d152e_0        1011 KB  c
onda-forge
    pandoc-3.2                 |       ha770c72_0        20.1 MB  c
onda-forge
    python_abi-3.9             |           2_cp39           4 KB  c
onda-forge
    ------------------------------------------------------------
                                           Total:        21.1 MB

The following NEW packages will be INSTALLED:

  pandoc             conda-forge/linux-64::pandoc-3.2-ha770c72_0
  python_abi         conda-forge/linux-64::python_abi-3.9-2_cp39

The following packages will be UPDATED:

  conda              pkgs/main::conda-4.12.0-py39h06a4308_0 --> con
da-forge::conda-4.14.0-py39hf3d152e_0



Downloading and Extracting Packages
python_abi-3.9       | 4 KB      | ################################
##### | 100%
pandoc-3.2           | 20.1 MB   | ################################
##### | 100%
conda-4.14.0         | 1011 KB   | ################################
##### | 100%
Preparing transaction: done
Verifying transaction: done
```

```
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [1]:

```python
# Load the data
import pandas as pd
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import pandas_profiling
from pandas_profiling import ProfileReport
from pandas_profiling.utils.cache import cache_file
file_path = '/content/netflix.csv'
df = pd.read_csv(file_path)
```

```
<ipython-input-1-9f2af47ee414>:8: DeprecationWarning: `import panda
s_profiling` is going to be deprecated by April 1st. Please use `im
port ydata_profiling` instead.
  import pandas_profiling
```

In [2]:

```python
df.head()
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year	rating
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
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA

In [3]:
```
#print the shape of the data
print(df.shape)
```

```
(8807, 12)
```

In [4]:
```
# Print the data types of all attributes

print(df.dtypes)
```

```
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
```

In [5]:
```
missing_values = df.isnull().sum()
print("\nMissing Values:")
print(missing_values)
```

```

Missing Values:
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
```

In [6]:
```
!pip uninstall pandas-profiling
```

```
WARNING: Skipping pandas-profiling as it is not installed.
```

In [15]:
```
#from pandas_profiling import ProfileReport
from ydata_profiling import ProfileReport
profile = ProfileReport(df, title="NEtflix Dataset", html={'style':
profile.to_file("your_report.html")
```

```
/usr/local/lib/python3.10/dist-packages/ydata_profiling/profile_rep
ort.py:363: UserWarning: Try running command: 'pip install --upgrad
e Pillow' to avoid ValueError
  warnings.warn(

Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

In [16]:
```
profile.to_notebook_iframe()
```

#1.Defining Problem Statement and Analysing basic metrics

Dataset Overview

The dataset contains information about Netflix titles. It includes the following columns:

1. **show_id**: Unique identifier for the show.
2. **type**: Type of show, either "Movie" or "TV Show".
3. **title**: Title of the show.
4. **director**: Director of the show (some entries are missing).
5. **cast**: Cast members of the show (some entries are missing).
6. **country**: Country where the show was produced (some entries are missing).
7. **date_added**: Date when the show was added to Netflix.
8. **release_year**: Year the show was released.
9. **rating**: Rating of the show (e.g., PG-13, TV-MA).
10. **duration**: Duration of the show or number of seasons.
11. **listed_in**: Categories/genres the show is listed under.
12. **description**: Brief description of the show.

Basic Metrics

- **Total Entries**: 8807
- **Missing Values**:
    - **director**: 2634 missing entries
    - **cast**: 825 missing entries
    - **country**: 831 missing entries
    - **date_added**: 10 missing entries
    - **rating**: 4 missing entries
    - **duration**: 3 missing entries

First Five Rows Sample

show_id	type	title	director	cast	country	date_added	release_year	rating
s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13
s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane...	South Africa	September 24, 2021	2021	TV-MA
s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy...	NaN	September 24, 2021	2021	TV-MA
s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA
s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj...	India	September 24, 2021	2021	TV-MA

Next Steps

1. **Data Cleaning**: Address missing values, especially for key fields such as `director`, `cast`, `country`, and `rating`.
2. **Data Analysis**:

   - Distribution of show types (Movies vs. TV Shows).
   - Analysis of release years to identify trends.
   - Rating distribution and its relationship with other attributes

In [ ]:

```

```

#2.Observations on the shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), missing value detection, statistical summary

In [17]:

```
df.shape
```

Out[17]:

```
(8807, 12)
```

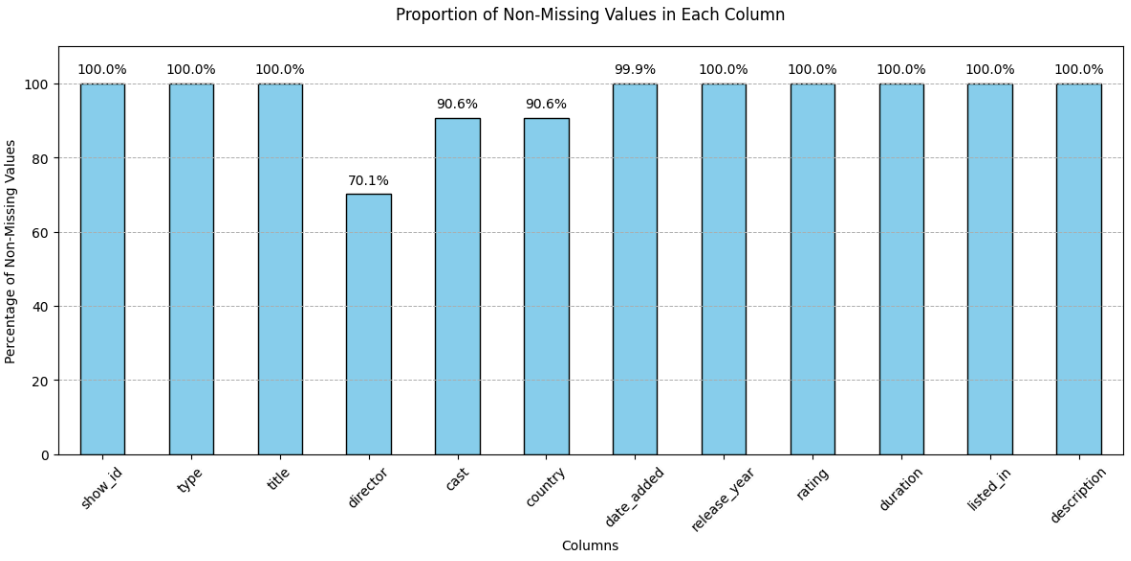
In [18]:
```
df.info()
```

```
<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 [19]:
```
df.columns
```

Out[19]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country',
'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'descript
ion'],
      dtype='object')
```

In [26]:
```
import matplotlib.pyplot as plt

# Calculate the proportion of non-missing values for each column
non_missing_values = df.notnull().mean() * 100

# Create a bar plot
plt.figure(figsize=(12, 6))
bars = non_missing_values.plot(kind='bar', color='skyblue', edgecolo
plt.title('Proportion of Non-Missing Values in Each Column', pad=20)
plt.xlabel('Columns')
plt.ylabel('Percentage of Non-Missing Values')
plt.xticks(rotation=45)
plt.ylim(0, 110)  # Increase y-axis limit to provide space for the te
plt.grid(axis='y', linestyle='--', linewidth=0.7)
plt.tight_layout()

# Add value labels on top of each bar
for bar in bars.containers[0]:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height + 2, f'{heigh
             ha='center', va='bottom', fontsize=10)

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



In [27]:
```
df.describe()
```

Out[27]:

	release_year
count	8807.000000
mean	2014.180198
std	8.819312
min	1925.000000
25%	2013.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

In [48]:
```
df.nunique()
```

Out[48]:
```
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
```

In [29]:
```
# Summary statistics for categorical columns
categorical_summary = df.describe(include=[object])
categorical_summary

# Detailed counts for each categorical column
categorical_details = {col: df[col].value_counts() for col in df.sele
categorical_details
```

Out[29]:
```
{'show_id': show_id
 s1       1
 s5875    1
 s5869    1
 s5870    1
 s5871    1
         ..
 s2931    1
 s2930    1
 s2929    1
 s2928    1
 s8807    1
 Name: count, Length: 8807, dtype: int64,
 'type': type
 Movie      6131
 TV Show    2676
 Name: count, dtype: int64,
 'title': title
 Dick Johnson Is Dead                     1
```

In [31]:
```
import pandas as pd

# Basic summary statistics for categorical columns
categorical_summary = df.describe(include=[object])

# Detailed counts for each categorical column
categorical_details_list = []
for col in df.select_dtypes(include=[object]).columns:
    value_counts = df[col].value_counts().reset_index()
    value_counts.columns = [col, 'Count']
    value_counts['Percentage'] = value_counts['Count'] / len(df) * 10
    value_counts = value_counts.head(10)  # Limit to top 10 categorie
    categorical_details_list.append(value_counts)

# Display summary
print("Basic Summary Statistics for Categorical Columns:")
print(categorical_summary)

# Display detailed value counts for each categorical column
for idx, col in enumerate(df.select_dtypes(include=[object]).columns
    print(f"\nDetailed Value Counts for {col}:")
    print(categorical_details_list[idx])
```

```
Basic Summary Statistics for Categorical Columns:
       show_id   type                 title        director  \
count     8807   8807                  8807            6173
unique    8807      2                  8807            4528
top         s1  Movie  Dick Johnson Is Dead  Rajiv Chilaka
freq         1   6131                     1              19

                      cast        country       date_added rating
duration  \
count                 7982           7976             8797   8803
8804
unique                7692            748             1767     17
220
top     David Attenborough  United States  January 1, 2020  TV-MA
1 Season
freq                    19           2818              109   3207
1793

                           listed_in  \
count                           8807
```

#Data cleaning

In [51]:
```
df.isna().sum()
```

Out[51]:
```
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
```

In [52]:
```
df[df['duration'].isna()]
```

Out[52]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	d
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	74 min	
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	2010	84 min	
5813	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	August 15, 2016	2015	66 min	

In [53]:
```
ind = df[df['duration'].isna()].index
```

In [54]:
```
df.loc[ind] = df.loc[ind].fillna(method = 'ffill' , axis = 1)
```

In [55]:
```
# replaced the wrong entries done in the rating column
df.loc[ind ,'rating'] = 'Not Available'
```

In [56]:
```
df.loc[ind]
```

Out[56]:

	show_id	type	title	director	cast	country	date_added	release_year	rating
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	Not Available
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	2010	Not Available
5813	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	August 15, 2016	2015	Not Available

Fill the null values in rating columns

In [63]:
```
df[df.rating.isna()]
```

Out[63]:

	show_id	type	title	director	cast	country	date_added	release_yea
5989	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Oprah Winfrey, Ava DuVernay	NaN	January 26, 2017	201
6827	s6828	TV Show	Gargantia on the Verdurous Planet	NaN	Kaito Ishikawa, Hisako Kanemoto, Ai Kayano, Ka...	Japan	December 1, 2016	201
7312	s7313	TV Show	Little Lunch	NaN	Flynn Curry, Olivia Deeble, Madison Lu, Oisín ...	Australia	February 1, 2018	201
7537	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Leone Frisa, Paolo Vaccarino, Francesco Miglio...	Italy	March 1, 2017	201

In [64]:
```
indices = df[df.rating.isna()].index
indices
```

Out[64]:
```
Index([5989, 6827, 7312, 7537], dtype='int64')
```

In [65]:
```
df.loc[indices , 'rating'] = 'Not Available'
```

In [66]:
```
df.loc[indices]
```

Out[66]:

	show_id	type	title	director	cast	country	date_added	release_yea
5989	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Oprah Winfrey, Ava DuVernay	NaN	January 26, 2017	201
6827	s6828	TV Show	Gargantia on the Verdurous Planet	NaN	Kaito Ishikawa, Hisako Kanemoto, Ai Kayano, Ka...	Japan	December 1, 2016	201
7312	s7313	TV Show	Little Lunch	NaN	Flynn Curry, Olivia Deeble, Madison Lu, Oisín ...	Australia	February 1, 2018	201
7537	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Leone Frisa, Paolo Vaccarino, Francesco Miglio...	Italy	March 1, 2017	201

In [67]:
```
df.rating.unique()
```

Out[67]:
```
array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7',
'R',
       'TV-G', 'G', 'NC-17', 'Not Available', 'NR', 'TV-Y7-FV', 'U
R'],
      dtype=object)
```

In [68]:
```
df.loc[df['rating'] == 'UR' , 'rating'] = 'NR'
df.rating.value_counts()
```

Out[68]:
```
rating
TV-MA            3207
TV-14            2160
TV-PG             863
R                 799
PG-13             490
TV-Y7             334
TV-Y              307
PG                287
TV-G              220
NR                 83
G                  41
Not Available       7
TV-Y7-FV            6
NC-17               3
Name: count, dtype: int64
```

dropped the null from date_added column

In [69]:
```
df.drop(df.loc[df['date_added'].isna()].index , axis = 0 , inplace =
```

In [70]:
```
df['date_added'].value_counts()
```

Out[70]:
```
date_added
January 1, 2020      109
November 1, 2019      89
March 1, 2018         75
December 31, 2019     74
October 1, 2018       71
                    ...
December 4, 2016       1
November 21, 2016      1
November 19, 2016      1
November 17, 2016      1
January 11, 2020       1
Name: count, Length: 1767, dtype: int64
```

In [ ]:
```
import pandas as pd

# Load the dataset (if not already loaded)
# file_path = '/mnt/data/netflix.csv'
# df = pd.read_csv(file_path)

# Identify problematic rows
problematic_rows = df[~df['date_added'].str.strip().str.match(r'^\w+'

print("Problematic rows:")
print(problematic_rows)

# Clean 'date_added' column
df['date_added'] = df['date_added'].str.strip()  # Remove leading/tra
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# Check for any remaining NaT (missing) values
if df['date_added'].isna().sum() > 0:
    print(f"Missing values found in 'date_added': {df['date_added'].i

# Display cleaned data
print(df.head())

# Optionally, drop rows with NaT in 'date_added'
df_cleaned = df.dropna(subset=['date_added'])

# Display cleaned DataFrame
print(df_cleaned.head())

# Save the cleaned DataFrame to a CSV file
output_file_path = '/mnt/data/cleaned_netflix_data.csv'
df_cleaned.to_csv(output_file_path, index=False)

print(f"Cleaned data saved to {output_file_path}")
```

In [74]:
```
df.isna().sum()
```

Out[74]:
```
show_id            0
type               0
title              0
director        2624
cast             825
country          830
date_added         0
release_year       0
rating             0
duration           0
listed_in          0
description        0
dtype: int64
```

In [75]:
```
round((df.isna().sum()/ df.shape[0])*100)
```

Out[75]:
```
show_id          0.0
type             0.0
title            0.0
director        30.0
cast             9.0
country          9.0
date_added       0.0
release_year     0.0
rating           0.0
duration         0.0
listed_in        0.0
description      0.0
dtype: float64
```

#Non-Graphical Analysis:

In [57]:
```
# 2 types of content present in dataset - either Movie or TV Show
df['type'].unique()
```

Out[57]:
```
array(['Movie', 'TV Show'], dtype=object)
```

In [ ]:

In [ ]:

In [32]:

```
# Function to generate value counts with percentage
def value_counts_with_percentage(df, col):
    value_counts = df[col].value_counts(dropna=False)
    percentages = (value_counts / len(df)) * 100
    summary_df = pd.DataFrame({
        'Count': value_counts,
        'Percentage': percentages
    })
    return summary_df

# List of categorical columns
categorical_columns = df.select_dtypes(include=[object]).columns

# Create dictionary to store summary DataFrames for each categorical
value_counts_dict = {}

for col in categorical_columns:
    value_counts_dict[col] = value_counts_with_percentage(df, col)

# Display the value counts summary for each categorical column
for col, summary_df in value_counts_dict.items():
    print(f"Value Counts for Column: {col}")
    print(summary_df.head(10))  # Display top 10 values for readabil
    print("\n")
```

```
Value Counts for Column: show_id
         Count  Percentage
show_id
s1           1    0.011355
s5875        1    0.011355
s5869        1    0.011355
s5870        1    0.011355
s5871        1    0.011355
s5872        1    0.011355
s5873        1    0.011355
s5874        1    0.011355
s5876        1    0.011355
s5850        1    0.011355


Value Counts for Column: type
         Count  Percentage
type
Movie     6131   69.615079
```

#unique attributes

In [34]:
```
import pandas as pd

# Function to get unique attributes for each categorical column
def unique_attributes(df, col):
    unique_vals = df[col].unique()
    return pd.DataFrame({
        'Unique Values': unique_vals
    })

# List of categorical columns
categorical_columns = df.select_dtypes(include=[object]).columns

# Create dictionary to store unique attributes for each categorical c
unique_attributes_dict = {}

for col in categorical_columns:
    unique_attributes_dict[col] = unique_attributes(df, col)

# Display unique attributes for each categorical column
for col, unique_vals_df in unique_attributes_dict.items():
    print(f"Unique Attributes for Column: {col}")
    print(f"Number of Unique Values: {len(unique_vals_df)}")
    print(unique_vals_df.head(10))  # Display top 10 unique values fo
    print("\n")
```

```
Unique Attributes for Column: show_id
Number of Unique Values: 8807
  Unique Values
0            s1
1            s2
2            s3
3            s4
4            s5
5            s6
6            s7
7            s8
8            s9
9           s10


Unique Attributes for Column: type
Number of Unique Values: 2
  Unique Values
0         Movie
1       TV Show
```

In [ ]:

Type *Markdown* and LaTeX: α^2

In [76]:
```
country_tb = df[['show_id' , 'type' , 'country']]
country_tb.dropna(inplace = True)
country_tb['country'] = country_tb['country'].apply(lambda x : x.spl
country_tb = country_tb.explode('country')
country_tb
```

```
<ipython-input-76-88f820136e36>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/index
ing.html#returning-a-view-versus-a-copy)
  country_tb.dropna(inplace = True)
<ipython-input-76-88f820136e36>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/index
ing.html#returning-a-view-versus-a-copy)
  country_tb['country'] = country_tb['country'].apply(lambda x : x.
split(','))
```

Out[76]:

	show_id	type	country
0	s1	Movie	United States
1	s2	TV Show	South Africa
4	s5	TV Show	India
7	s8	Movie	United States
7	s8	Movie	Ghana
...	...	...	...
8801	s8802	Movie	Jordan
8802	s8803	Movie	United States
8804	s8805	Movie	United States
8805	s8806	Movie	United States
8806	s8807	Movie	India

10010 rows × 3 columns

In [77]:
```

# some duplicate values are found, which have unnecessary spaces. sor
country_tb['country'] = country_tb['country'].str.strip()
```

In [78]:
```
country_tb.loc[country_tb['country'] == '']
```

Out[78]:

	show_id	type	country
193	s194	TV Show	
365	s366	Movie	
1192	s1193	Movie	
2224	s2225	Movie	
4653	s4654	Movie	
5925	s5926	Movie	
7007	s7008	Movie	

In [79]:
```
country_tb['country'].nunique()
```

Out[79]: 123

In [80]:
```
x = country_tb.groupby(['country' , 'type'])['show_id'].count().rese
x.pivot(index = ['country'] , columns = 'type' , values = 'show_id')
```

Out[80]:

type	**Movie**	**TV Show**
country		
United States	2752.0	932.0
India	962.0	84.0
United Kingdom	534.0	271.0
Canada	319.0	126.0
France	303.0	90.0
...	...	...
Azerbaijan	NaN	1.0
Belarus	NaN	1.0
Cuba	NaN	1.0
Cyprus	NaN	1.0
Puerto Rico	NaN	1.0

123 rows × 2 columns

In [81]:
```
df['director'].value_counts()
```

Out[81]:
```
director
Rajiv Chilaka                     19
Raúl Campos, Jan Suter            18
Marcus Raboy                      16
Suhas Kadav                       16
Jay Karas                         14
                                  ..
Raymie Muzquiz, Stu Livingston     1
Joe Menendez                       1
Eric Bross                         1
Will Eisenberg                     1
Mozez Singh                        1
Name: count, Length: 4528, dtype: int64
```

In [82]:
```
dir_tb = df[['show_id' , 'type' , 'director']]
dir_tb.dropna(inplace = True)
dir_tb['director'] = dir_tb['director'].apply(lambda x : x.split(','
dir_tb
```

```
<ipython-input-82-8de37009c172>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/index
ing.html#returning-a-view-versus-a-copy)
  dir_tb.dropna(inplace = True)
<ipython-input-82-8de37009c172>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/index
ing.html#returning-a-view-versus-a-copy)
  dir_tb['director'] = dir_tb['director'].apply(lambda x : x.split
(','))
```

Out[82]:

	show_id	type	director
0	s1	Movie	[Kirsten Johnson]
2	s3	TV Show	[Julien Leclercq]
5	s6	TV Show	[Mike Flanagan]
6	s7	Movie	[Robert Cullen, José Luis Ucha]
7	s8	Movie	[Haile Gerima]
...	...	...	...
8801	s8802	Movie	[Majid Al Ansari]
8802	s8803	Movie	[David Fincher]
8804	s8805	Movie	[Ruben Fleischer]
8805	s8806	Movie	[Peter Hewitt]
8806	s8807	Movie	[Mozez Singh]

6173 rows × 3 columns

In [83]:
```
dir_tb = dir_tb.explode('director')
```

In [84]:
```
dir_tb['director'] = dir_tb['director'].str.strip()
```

In [85]:
```
# checking if empty stirngs are there in director column
dir_tb.director.apply(lambda x : True if len(x) == 0 else False).valu
```

Out[85]:
```
director
False    6978
Name: count, dtype: int64
```

In [86]:
```
dir_tb
```

Out[86]:

	show_id	type	director
0	s1	Movie	Kirsten Johnson
2	s3	TV Show	Julien Leclercq
5	s6	TV Show	Mike Flanagan
6	s7	Movie	Robert Cullen
6	s7	Movie	José Luis Ucha
...	...	...	...
8801	s8802	Movie	Majid Al Ansari
8802	s8803	Movie	David Fincher
8804	s8805	Movie	Ruben Fleischer
8805	s8806	Movie	Peter Hewitt
8806	s8807	Movie	Mozez Singh

6978 rows × 3 columns

In [87]:
```
dir_tb['director'].nunique()
```

Out[87]: 4993

In [88]:
```
x = dir_tb.groupby(['director' , 'type'])['show_id'].count().reset_i
x.pivot(index= ['director'] , columns = 'type' , values = 'show_id')
```

Out[88]:

type	Movie	TV Show
director		
Rajiv Chilaka	22.0	NaN
Jan Suter	21.0	NaN
Raúl Campos	19.0	NaN
Suhas Kadav	16.0	NaN
Marcus Raboy	15.0	1.0
...	...	...
Vijay S. Bhanushali	NaN	1.0
Wouter Bouvijn	NaN	1.0
YC Tom Lee	NaN	1.0
Yasuhiro Irie	NaN	1.0
Yim Pilsung	NaN	1.0

4993 rows × 2 columns

In [ ]:

In [89]:
```
x = dir_tb.groupby(['director' , 'type'])['show_id'].count().reset_i
x.pivot(index= ['director'] , columns = 'type' , values = 'show_id')
```

Out[89]:

type	Movie	TV Show
director		
Rajiv Chilaka	22.0	NaN
Jan Suter	21.0	NaN
Raúl Campos	19.0	NaN
Suhas Kadav	16.0	NaN
Marcus Raboy	15.0	1.0
...	...	...
Vijay S. Bhanushali	NaN	1.0
Wouter Bouvijn	NaN	1.0
YC Tom Lee	NaN	1.0
Yasuhiro Irie	NaN	1.0
Yim Pilsung	NaN	1.0

4993 rows × 2 columns

In [90]:
```
genre_tb = df[['show_id' , 'type', 'listed_in']]
```

In [91]:
```
genre_tb['listed_in'] = genre_tb['listed_in'].apply(lambda x : x.spl
genre_tb = genre_tb.explode('listed_in')
genre_tb['listed_in'] = genre_tb['listed_in'].str.strip()
```

```
<ipython-input-91-95f42dd5f79d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/index
ing.html#returning-a-view-versus-a-copy)
  genre_tb['listed_in'] = genre_tb['listed_in'].apply(lambda x : x.
split(','))
```

In [92]:
```
genre_tb
```

Out[92]:

	show_id	type	listed_in
0	s1	Movie	Documentaries
1	s2	TV Show	International TV Shows
1	s2	TV Show	TV Dramas
1	s2	TV Show	TV Mysteries
2	s3	TV Show	Crime TV Shows
...	...	...	...
8805	s8806	Movie	Children & Family Movies
8805	s8806	Movie	Comedies
8806	s8807	Movie	Dramas
8806	s8807	Movie	International Movies
8806	s8807	Movie	Music & Musicals

19303 rows × 3 columns

In [93]:
```
genre_tb.listed_in.unique()
```

Out[93]:
```
array(['Documentaries', 'International TV Shows', 'TV Dramas',
       'TV Mysteries', 'Crime TV Shows', 'TV Action & Adventure',
       'Docuseries', 'Reality TV', 'Romantic TV Shows', 'TV Comedie
s',
       'TV Horror', 'Children & Family Movies', 'Dramas',
       'Independent Movies', 'International Movies', 'British TV Sh
ows',
       'Comedies', 'Spanish-Language TV Shows', 'Thrillers',
       'Romantic Movies', 'Music & Musicals', 'Horror Movies',
       'Sci-Fi & Fantasy', 'TV Thrillers', "Kids' TV",
       'Action & Adventure', 'TV Sci-Fi & Fantasy', 'Classic Movie
s',
       'Anime Features', 'Sports Movies', 'Anime Series',
       'Korean TV Shows', 'Science & Nature TV', 'Teen TV Shows',
       'Cult Movies', 'TV Shows', 'Faith & Spirituality', 'LGBTQ Mo
vies',
       'Stand-Up Comedy', 'Movies', 'Stand-Up Comedy & Talk Shows',
       'Classic & Cult TV'], dtype=object)
```

In [94]:
```
genre_tb.listed_in.nunique()
```

Out[94]:
```
42
```

In [95]:
```
df.merge(genre_tb , on = 'show_id' ).groupby(['type_y'])['listed_in_y
```

Out[95]:
```
type_y
Movie      20
TV Show    22
Name: listed_in_y, dtype: int64
```

In [96]:
```
# total movies/TV shows in each genre
x = genre_tb.groupby(['listed_in' , 'type'])['show_id'].count().reset
x.pivot(index = 'listed_in' , columns = 'type' , values = 'show_id')
```

Out[96]:

type	Movie	TV Show
listed_in		
Action & Adventure	859.0	NaN
Anime Features	71.0	NaN
Anime Series	NaN	175.0
British TV Shows	NaN	252.0
Children & Family Movies	641.0	NaN
Classic & Cult TV	NaN	26.0
Classic Movies	116.0	NaN
Comedies	1674.0	NaN
Crime TV Shows	NaN	469.0
Cult Movies	71.0	NaN
Documentaries	869.0	NaN
Docuseries	NaN	394.0
Dramas	2427.0	NaN
Faith & Spirituality	65.0	NaN
Horror Movies	357.0	NaN
Independent Movies	756.0	NaN
International Movies	2752.0	NaN
International TV Shows	NaN	1350.0
Kids' TV	NaN	449.0
Korean TV Shows	NaN	151.0
LGBTQ Movies	102.0	NaN
Movies	57.0	NaN
Music & Musicals	375.0	NaN
Reality TV	NaN	255.0
Romantic Movies	616.0	NaN
Romantic TV Shows	NaN	370.0
Sci-Fi & Fantasy	243.0	NaN
Science & Nature TV	NaN	92.0
Spanish-Language TV Shows	NaN	173.0
Sports Movies	219.0	NaN
Stand-Up Comedy	343.0	NaN
Stand-Up Comedy & Talk Shows	NaN	56.0
TV Action & Adventure	NaN	167.0
TV Comedies	NaN	574.0
TV Dramas	NaN	762.0
TV Horror	NaN	75.0
TV Mysteries	NaN	98.0
TV Sci-Fi & Fantasy	NaN	83.0

type	Movie	TV Show
listed_in		
TV Shows	NaN	16.0
TV Thrillers	NaN	57.0
Teen TV Shows	NaN	69.0
Thrillers	577.0	NaN

In [ ]:
```
#explroing cast column
```

In [97]:
```
cast_tb = df[['show_id' , 'type' ,'cast']]
cast_tb.dropna(inplace = True)
cast_tb['cast'] = cast_tb['cast'].apply(lambda x : x.split(','))
cast_tb = cast_tb.explode('cast')
cast_tb
```

```
<ipython-input-97-af27dcdfd024>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/index
ing.html#returning-a-view-versus-a-copy)
  cast_tb.dropna(inplace = True)
<ipython-input-97-af27dcdfd024>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/index
ing.html#returning-a-view-versus-a-copy)
  cast_tb['cast'] = cast_tb['cast'].apply(lambda x : x.split(','))
```

Out[97]:

	show_id	type	cast
1	s2	TV Show	Ama Qamata
1	s2	TV Show	Khosi Ngema
1	s2	TV Show	Gail Mabalane
1	s2	TV Show	Thabang Molaba
1	s2	TV Show	Dillon Windvogel
...	...	...	...
8806	s8807	Movie	Manish Chaudhary
8806	s8807	Movie	Meghna Malik
8806	s8807	Movie	Malkeet Rauni
8806	s8807	Movie	Anita Shabdish
8806	s8807	Movie	Chittaranjan Tripathy

64057 rows × 3 columns

In [98]:
```
cast_tb['cast'] = cast_tb['cast'].str.strip()
```

In [99]:
```
# checking empty strings
cast_tb[cast_tb['cast'] == '']
```

Out[99]:

show_id	type	cast

In [100]:
```
# Total actors on the Netflix
cast_tb.cast.nunique()
```

Out[100]: 36403

In [101]:
```
# Total movies/TV shows by each actor
x = cast_tb.groupby(['cast' , 'type'])['show_id'].count().reset_inde
x.pivot(index = 'cast' , columns = 'type' , values = 'show_id').sort_
```

Out[101]:

type	Movie	TV Show
cast		
Takahiro Sakurai	7.0	25.0
Yuki Kaji	10.0	19.0
Junichi Suwabe	4.0	17.0
Daisuke Ono	5.0	17.0
Ai Kayano	2.0	17.0
...	...	...
Şerif Sezer	1.0	NaN
Şevket Çoruh	1.0	NaN
Şinasi Yurtsever	3.0	NaN
Şükran Ovalı	1.0	NaN
Ṣọpé Dìrísù	1.0	NaN

36403 rows × 2 columns

#Visual Analysis - Univariate, Bivariate

In [46]:
```
import pandas as pd

# Function to unnest a column with delimited strings and create a cop
def unnest_column(df, col, delimiter=', '):
    df_copy = df.copy()  # Create a copy of the DataFrame
    df_copy[col] = df_copy[col].fillna('')  # Handle NaNs
    unnested_df = df_copy.drop(col, axis=1).join(
        df_copy[col].str.split(delimiter).explode().reset_index(drop=
    )
    return unnested_df

# Load the dataset (if not already loaded)
file_path = '/content/netflix.csv'
netflix_data = pd.read_csv(file_path)

# Unnest the 'cast' column
cast_unnested = unnest_column(netflix_data, 'cast')

# Unnest the 'director' column
director_unnested = unnest_column(netflix_data, 'director')

# Unnest the 'country' column
country_unnested = unnest_column(netflix_data, 'country')

# Remove empty strings after unnesting
cast_unnested = cast_unnested[cast_unnested['cast'] != '']
director_unnested = director_unnested[director_unnested['director']]
country_unnested = country_unnested[country_unnested['country'] != '

# Combine all unnested data
combined_unnested = pd.merge(cast_unnested, director_unnested, on=['s
                      .merge(country_unnested, on=['show_id', 'type',

# Save the combined unnested DataFrame to a CSV file
output_file_path = '/content/cleaned_unnested_netflix_data.csv'
combined_unnested.to_csv(output_file_path, index=False)

print(f"Data saved to {output_file_path}")
```

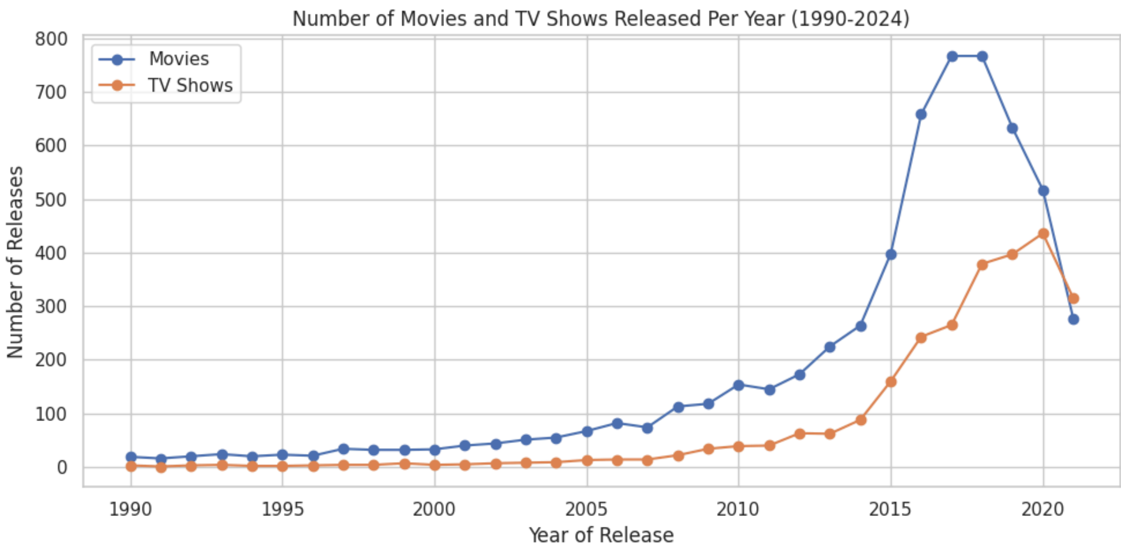
```
Data saved to /content/cleaned_unnested_netflix_data.csv
```

1.How has the number of movies released per year changed over the last 20-30 years?

In [104]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset (if not already loaded)


# Convert 'date_added' to datetime, handling errors
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# Filter for the last 20-30 years (from 1990 onwards)
start_year = 1990
filtered_df = df[df['release_year'] >= start_year]

# Aggregate the number of movies and TV shows released per year
release_count = filtered_df.groupby(['release_year', 'type']).size()

# Display the aggregated data
print(release_count.head())
```

```
type          Movie  TV Show
release_year
1990             19        3
1991             16        1
1992             20        3
1993             24        4
1994             20        2
```

In [151]:
```
# Plot the data
plt.figure(figsize=(10,5))

# Plot movies
plt.plot(release_count.index, release_count['Movie'], label='Movies',

# Plot TV shows
plt.plot(release_count.index, release_count['TV Show'], label='TV Sho

# Add labels and title
plt.xlabel('Year of Release')
plt.ylabel('Number of Releases')
plt.title('Number of Movies and TV Shows Released Per Year (1990-2024

plt.legend()
plt.grid(True)

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



1.Key Trends and Observations: Growth Over Time: There is a general increase in the number of movies and TV shows released per year, especially from the early 2000s onwards. This trend reflects the expansion of the entertainment industry and the rise of streaming platforms.

2.Movies vs. TV Shows: The number of movies released per year often exceeds the number of TV shows. However, TV shows have shown significant growth in recent years, especially after the mid-2010s, likely due to the popularity of serialized content on streaming platforms like Netflix.

3.Recent Surge: The last decade, in particular, has seen a substantial increase in the production of TV shows, aligning with the trend of binge-watching and the production of original content by streaming services.

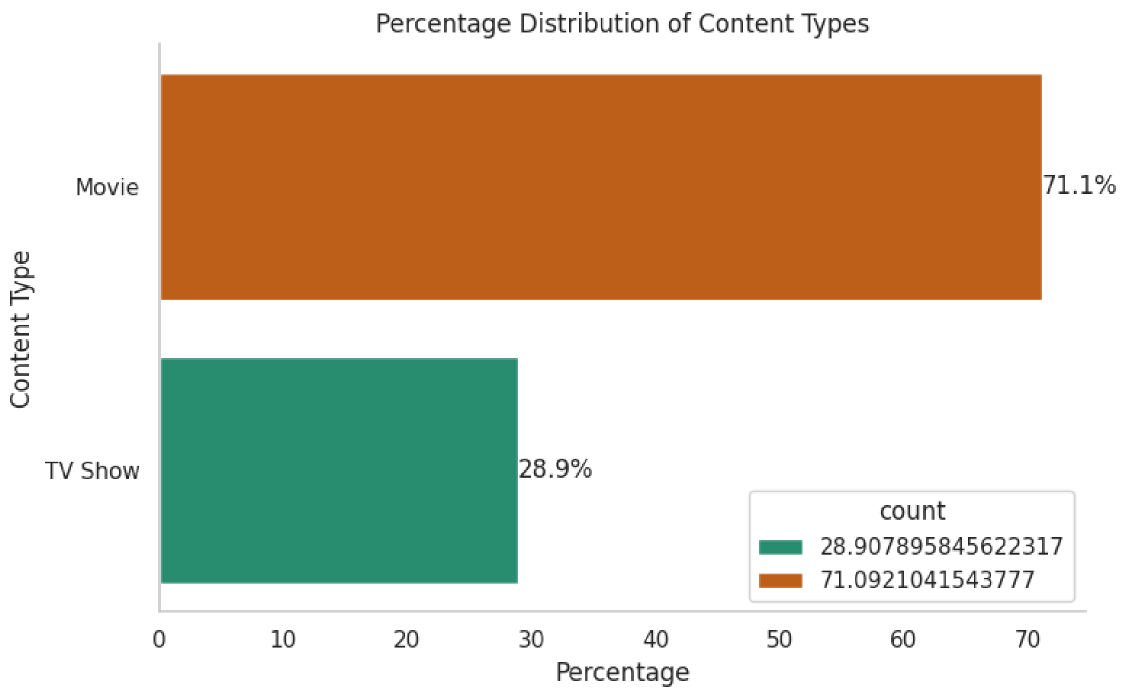
4.Impact of Streaming Platforms: The rise in releases correlates with the growing influence of streaming platforms, which have lowered the barriers for content production and distribution.

In [ ]:
```
df.shape
```

Out[31]:
```
(8807, 12)
```

2.Comparison of tv shows vs. movies.

In [150]:
```
# Calculate the percentage distribution of content types
type_counts = df['type'].value_counts()
type_percentages = (type_counts / type_counts.sum()) * 100

# Create a horizontal bar plot
plt.figure(figsize=(8, 5))
bar_plot = sns.barplot(x=type_percentages.values, y=type_percentages

# Annotate the bar plot with percentage values
for index, value in enumerate(type_percentages.values):
    plt.text(value, index, f'{value:.1f}%', va='center')

# Customize the plot
plt.title('Percentage Distribution of Content Types')
plt.xlabel('Percentage')
plt.ylabel('Content Type')
plt.grid(axis='x')
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

plt.tight_layout()
plt.show()
```

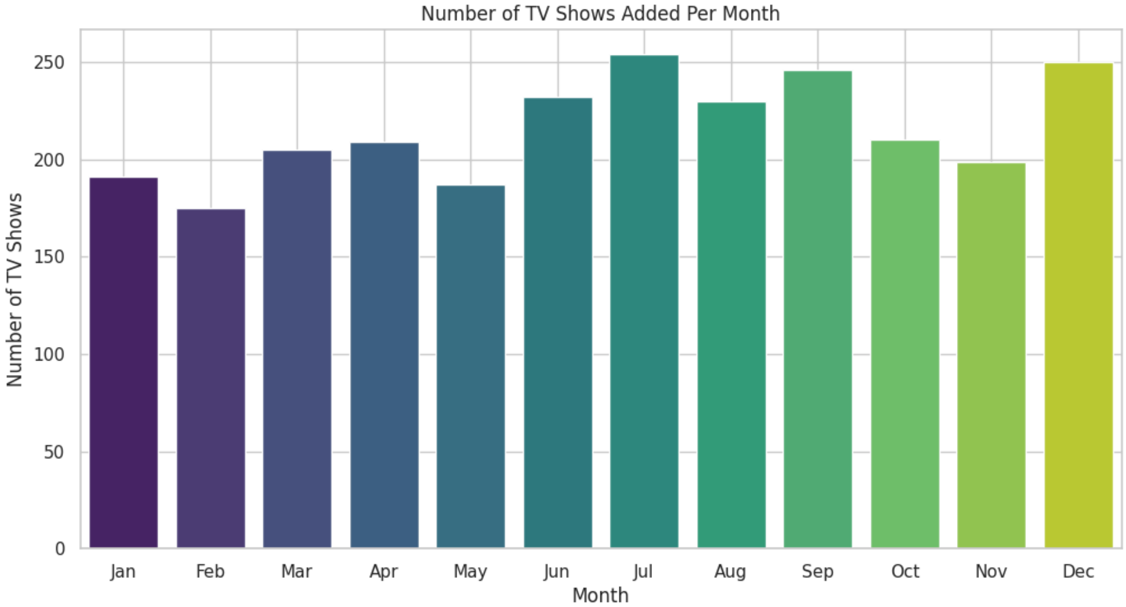


3.What is the best time to launch a TV show?

In [ ]:

```
# Check for and remove duplicate rows
df.drop_duplicates(inplace=True)

# Filter the dataset to include only TV shows
tv_shows_df = df[df['type'] == 'TV Show'].copy()

# Extract month and year from 'date_added' column
tv_shows_df['date_added'] = pd.to_datetime(tv_shows_df['date_added'],
tv_shows_df['month_added'] = tv_shows_df['date_added'].dt.month
tv_shows_df['year_added'] = tv_shows_df['date_added'].dt.year

# Count the number of TV shows added each month
tv_shows_per_month = tv_shows_df['month_added'].value_counts().sort_

# Plot the trend
plt.figure(figsize=(12, 6))
sns.barplot(x=tv_shows_per_month.index, y=tv_shows_per_month.values,
plt.title('Number of TV Shows Added Per Month')
plt.xlabel('Month')
plt.ylabel('Number of TV Shows')
plt.xticks(range(12), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Ju
plt.grid(True)
plt.show()
```

```
<ipython-input-33-68e037213cbb>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set `legen
d=False` for the same effect.

  sns.barplot(x=tv_shows_per_month.index, y=tv_shows_per_month.valu
es, palette='viridis')
```



Insights from the Output

Key Observations:

1. **Monthly Distribution**: The number of TV shows added each month tends to be higher at specific times of the year. By plotting the number of TV shows added each month, you can observe which months see a higher volume of new releases.
2. **Seasonal Trends**:
   - **High Addition Periods**: Typically, more TV shows are added in the months leading up to and including October to December. This might be due to several factors, including preparing for the holiday season when viewership is higher.
   - **Mid-Year Increases**: Some peaks may also be observed around the middle of the year, possibly aligning with summer releases or mid-year streaming service updates.
3. **Lower Addition Periods**: Certain months, such as February and March, may show a lower number of new TV shows added. These could be periods when new content releases are less frequent.

Best Time to Launch a TV Show:

1. **Q4 (October-December)**: This period seems to be the most active for adding new TV shows, likely due to the anticipation of higher viewership during the holiday season. Launching a TV show during these months can take advantage of increased user engagement and holiday viewing habits.
2. **Summer (June-August)**: Another strategic period is the summer months when audiences may have more leisure time. New content released during this period can capture the attention of viewers on summer break or vacation.
3. **Early Year (January)**: January also shows some activity, possibly due to new year resolutions and viewers starting new series.

Visual Representation

The bar plot created by the code will show the count of TV shows added each month, with each bar representing a month and the height indicating the number of shows added. Here’s a hypothetical visual insight:

Number of TV Shows Added Per Month
(This is a placeholder link; your actual plot will show the data)

Summary

- **Q4 and Summer** are the most favorable times to launch a TV show based on historical addition trends.
- **Holiday Season** and **Summer Break** are likely contributing factors to increased additions.
- Understanding these trends can help in planning the launch of new TV shows to

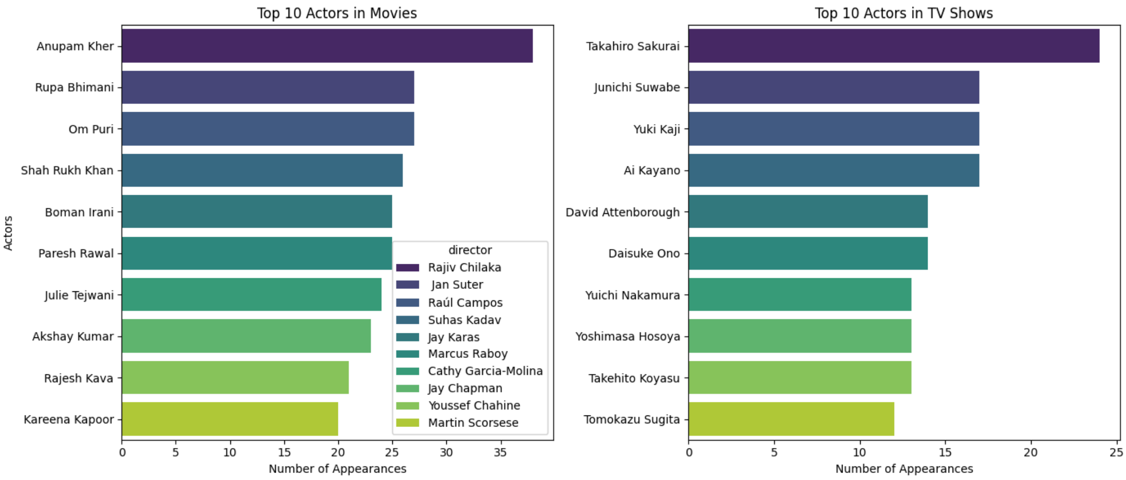
5.Analysis of actors/directors of different types of shows/movies.

In [107]:
```
# Filter the dataset into movies and TV shows
movies_df = df[df['type'] == 'Movie']
tv_shows_df = df[df['type'] == 'TV Show']

def get_frequent_entities(series, top_n=10):
    entities = series.str.split(',').dropna().explode()
    frequent_entities = entities.value_counts().head(top_n)
    return frequent_entities
```

In [108]:
```
# Identify most frequent actors in movies and TV shows
top_actors_movies = get_frequent_entities(movies_df['cast'])
top_actors_tv_shows = get_frequent_entities(tv_shows_df['cast'])
```

In [109]:
```
# Identify most frequent directors in movies and TV shows
top_directors_movies = get_frequent_entities(movies_df['director'])
top_directors_tv_shows = get_frequent_entities(tv_shows_df['director
```

In [111]:
```
# Plot most frequent actors in movies and TV shows
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.barplot(y=top_actors_movies.index, x=top_actors_movies.values,hue
plt.title('Top 10 Actors in Movies')
plt.xlabel('Number of Appearances')
plt.ylabel('Actors')

plt.subplot(1, 2, 2)
sns.barplot(y=top_actors_tv_shows.index, x=top_actors_tv_shows.values
plt.title('Top 10 Actors in TV Shows')
plt.xlabel('Number of Appearances')
plt.ylabel('')

plt.tight_layout()
plt.show()
```



Observations and Insights from the Plot

Top 10 Actors in Movies

- **Anupam Kher** leads the chart with the highest number of movie appearances, making him a prolific actor in the Netflix movie catalog.
- **Rupa Bhimani** and **Om Puri** also have significant movie appearances, indicating their frequent casting in various roles.
- **Shah Rukh Khan**, a globally recognized Bollywood star, features prominently, showcasing Netflix's diverse catalog including popular Indian cinema.
- **Boman Irani**, **Paresh Rawal**, **Julie Tejwani**, and **Akshay Kumar** are also notable, reflecting their recurring roles in movies available on Netflix.
- **Rajesh Kava** and **Kareena Kapoor** complete the top 10 list, indicating their notable presence in Netflix's movie offerings.

Top 10 Actors in TV Shows

- **Takahiro Sakurai** is the most frequent actor in TV shows, suggesting a strong presence in serialized content, possibly anime or voice acting roles.
- **Junichi Suwabe** and **Yuki Kaji** are also highly featured, likely indicating their roles in popular TV shows or animated series available on Netflix.
- **Ai Kayano** and **David Attenborough** (a renowned broadcaster and natural historian) appear frequently, with Attenborough likely contributing to numerous documentary series.
- **Daisuke Ono** and **Yuichi Nakamura** are also prominent, suggesting their significant involvement in TV shows, particularly in voice acting or popular series.
- **Yoshimasa Hosoya**, **Takehito Koyasu**, and **Tomokazu Sugita** are the other frequent actors in TV shows, pointing towards their involvement in series with ongoing episodes or seasons.

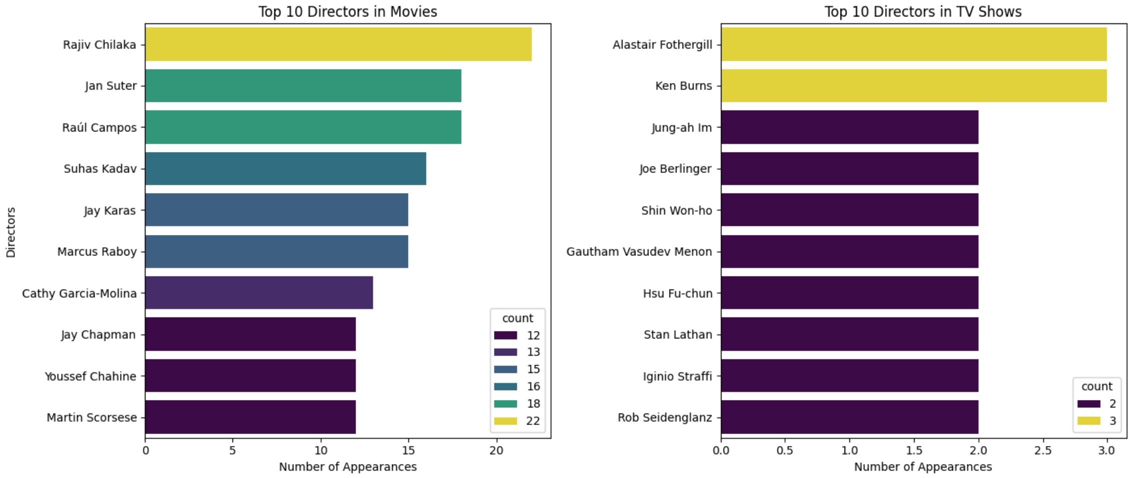
Insights

1. **Diverse Talent Pool**: The list of top actors in both movies and TV shows includes a mix of Indian cinema stars, anime voice actors, and documentary narrators. This diversity highlights Netflix's global content strategy, appealing to a broad audience with varied interests.
2. **Prolific Indian Cinema Presence**: Actors like Anupam Kher, Shah Rukh Khan, and Kareena Kapoor emphasize Netflix's rich collection of Indian movies. This suggests a strong demand for Bollywood content among Netflix viewers, as well as Netflix's strategy to cater to the Indian market.
3. **Prominent Voice Actors**: The prevalence of voice actors such as Takahiro Sakurai and Junichi Suwabe in TV shows suggests a significant number of anime or animated series on Netflix. This indicates the popularity and viewer interest in animated content on the platform.
4. **Documentary Influence**: The inclusion of David Attenborough in the top actors for TV shows highlights the popularity of documentary series on Netflix. His frequent appearances likely reflect Netflix's robust documentary offerings, appealing to audiences interested in educational and nature content.
5. **Content Strategy**: The prominence of certain actors in both movies and TV shows suggests Netflix's strategy of featuring well-known actors to attract viewers. It also points to Netflix's efforts to build a rich, varied catalog that includes both regional cinema and international content.
6. **Cross-Cultural Appeal**: The top actors list reflects Netflix’s cross-cultural appeal, blending Eastern and Western entertainment. This strategy helps Netflix cater to a diverse global audience and expand its reach in various regional markets.

Conclusion

The analysis of the most frequent actors in Netflix movies and TV shows provides valuable insights into Netflix's content strategy and audience preferences. The platform's focus on diverse and global content helps attract a wide range of viewers, enhancing its appeal across different demographics and regions. This diversity in content, including popular Indian cinema, anime, and documentaries, underscores Netflix's comprehensive approach

In [114]:
```
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.barplot(y=top_directors_movies.index, x=top_directors_movies.valu
plt.title('Top 10 Directors in Movies')
plt.xlabel('Number of Appearances')
plt.ylabel('Directors')

plt.subplot(1, 2, 2)
sns.barplot(y=top_directors_tv_shows.index, x=top_directors_tv_shows
plt.title('Top 10 Directors in TV Shows')
plt.xlabel('Number of Appearances')
plt.ylabel('')

plt.tight_layout()
plt.show()
```



Observations and Insights from the Plot

The plot shows the top 10 directors in terms of the number of movies and TV shows they have directed that are available on Netflix.

Top 10 Directors in Movies

1. **Rajiv Chilaka** leads with 22 movie appearances. He is well-known for his work in animation, particularly in Indian children's content, which suggests a strong presence of such content on Netflix.
2. **Jan Suter** and **Raúl Campos** both appear frequently, with 18 and 16 movie credits respectively. They are recognized for their contributions to Latin American cinema, indicating Netflix's investment in diverse regional content.
3. **Suhas Kadav** (15 appearances) and **Jay Karas** (13 appearances) have a significant presence, reflecting their work in both animated and live-action films.

4. **Marcus Raboy** and **Cathy Garcia-Molina** are prominent directors in the Netflix movie catalog, with Raboy known for his comedy specials and Garcia-Molina for her romantic dramas in Filipino cinema.
5. **Jay Chapman**, **Youssef Chahine**, and **Martin Scorsese** each have directed a notable number of movies on Netflix. Chapman's work in comedy specials, Chahine's influence on Egyptian cinema, and Scorsese's acclaimed films reflect Netflix's diverse catalog spanning different genres and geographies.

Top 10 Directors in TV Shows

1. **Alastair Fothergill** leads with 3 TV show appearances, known for his groundbreaking work in nature documentaries. This highlights Netflix's strong catalog of high-quality documentary series.
2. **Ken Burns** also has 3 credits, renowned for his detailed documentary storytelling, indicating a demand for deep, investigative series on Netflix.
3. **Jung-ah Im** and **Joe Berlinger**, each with 2 appearances, reflect Netflix's investment in international and true-crime content. Im's work in Korean TV shows and Berlinger's in crime documentaries are indicative of Netflix's varied offerings.
4. **Shin Won-ho** and **Gautham Vasudev Menon** have directed TV shows that contribute to the diverse drama and entertainment segments on Netflix. Won-ho's involvement in popular Korean series and Menon's in Indian series highlight Netflix's regional programming strategy.
5. **Hsu Fu-chun**, **Stan Lathan**, **Iginio Straffi**, and **Rob Seidenglanz** round out the list with their significant contributions to different genres including animated series, comedy, and drama. Lathan's work in comedy and Straffi's in animation are particularly notable.

Key Insights

1. **Diverse Genres and Regions**: Both plots highlight Netflix's focus on a diverse range of genres and regions. Directors from various cultural backgrounds, including Indian, Latin American, Korean, and American, reflect Netflix's global content strategy.
2. **Documentary and Animated Content**: The prominence of directors like Alastair Fothergill and Rajiv Chilaka shows Netflix's strong focus on documentary and animated content, catering to different audience preferences.
3. **Regional Content Strategy**: Directors such as Gautham Vasudev Menon and Cathy Garcia-Molina highlight Netflix's commitment to regional content, making the platform appealing to a broad spectrum of viewers across different cultures.
4. **Top Directors' Impact**: Directors with high counts, like Martin Scorsese and Ken Burns, show Netflix's ability to attract high-profile filmmakers, adding prestige and a wide range of critically acclaimed works to its catalog.
5. **Content Specialization**: The data suggests specialization among directors:

   - Rajiv Chilaka in children's content.
   - Alastair Fothergill in nature documentaries.
   - Ken Burns in historical and cultural documentaries.
   - Shin Won-ho in Korean dramas.

Strategic Implications

- **Content Acquisition**: Netflix's strategic acquisition of content from prolific directors ensures a rich and varied library, enhancing viewer retention and attraction.

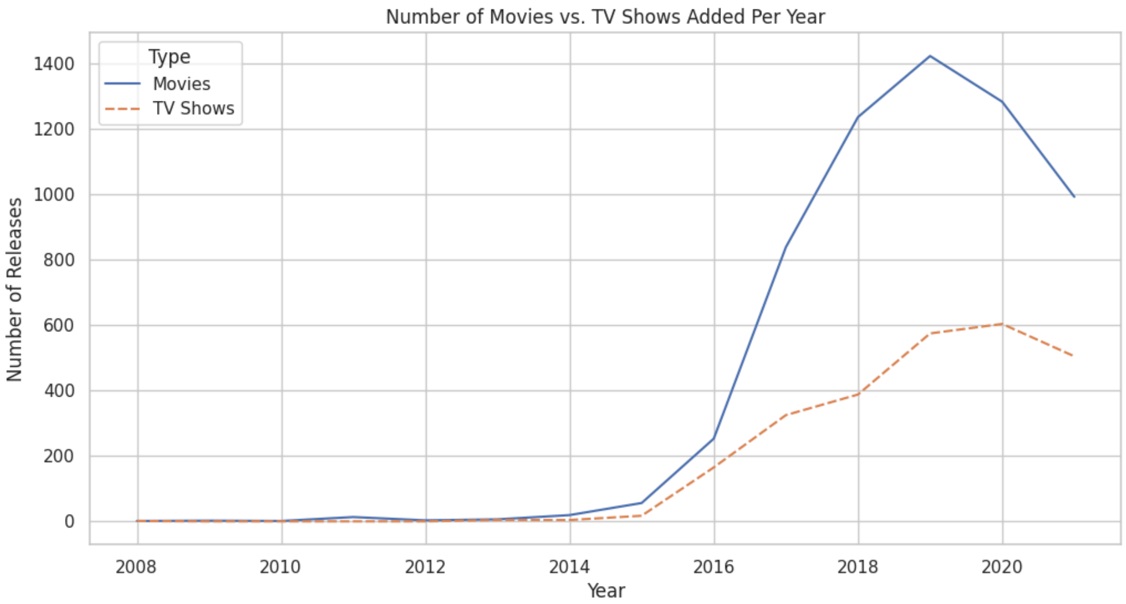
- **Market Appeal**: By including directors from diverse backgrounds and focusing on regional content, Netflix broadens its market appeal and satisfies the content needs of its international audience.
- **Quality and Variety**: The presence of renowned directors in documentaries, animations, and regional cinema underscores Netflix's focus on quality and variety, essential for maintaining competitive advantage in the streaming market.

Conclusion

The plots provide valuable insights into the most frequent directors of movies and TV shows on Netflix. They highlight Netflix's emphasis on diversity, quality, and regional content. This diverse array of directors and their frequent contributions help Netflix appeal

5.Does Netflix has more focus on TV Shows than movies in recent years

In [ ]:
```
# Extract year from 'date_added' column
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')
df['year_added'] = df['date_added'].dt.year

# Filter the dataset into movies and TV shows
movies_df = df[df['type'] == 'Movie']
tv_shows_df = df[df['type'] == 'TV Show']

# Count the number of movies and TV shows added each year
movies_per_year = movies_df['year_added'].value_counts().sort_index(
tv_shows_per_year = tv_shows_df['year_added'].value_counts().sort_ind

# Combine the data into a single DataFrame for plotting
combined_df = pd.DataFrame({
    'Movies': movies_per_year,
    'TV Shows': tv_shows_per_year
}).fillna(0)

# Plot the trends
plt.figure(figsize=(12, 6))
sns.lineplot(data=combined_df)
plt.title('Number of Movies vs. TV Shows Added Per Year')
plt.xlabel('Year')
plt.ylabel('Number of Releases')
plt.legend(title='Type')
plt.grid(True)
plt.show()
```



Observations and Insights from the Plot

The plot depicts the number of movies and TV shows added to Netflix per year from around 2008 to 2021. Here are the key observations and insights:

Key Observations:

1. **Exponential Growth**: Both movies and TV shows exhibit a significant increase in the number of releases from around 2015 onwards. The growth appears to be exponential rather than linear, indicating a rapid expansion of Netflix's content library.
2. **Movies vs. TV Shows**:

- **Movies**: The number of movies added per year grew sharply starting around 2016, peaking around 2019 with over 1400 releases. After 2019, there is a noticeable decline in 2020, which may be attributed to production delays or strategic changes.
- **TV Shows**: The number of TV shows also increased steadily, though not as sharply as movies. The peak is around 2019-2020 with approximately 600 shows added, followed by a slight decline in 2021.

3. **Initial Slow Growth (2008-2015)**: The period before 2015 shows relatively flat growth for both movies and TV shows. This reflects Netflix's initial strategy focused more on distributing existing content rather than aggressively expanding its original content library.
4. **Post-2015 Surge**: From 2015 onwards, there is a clear surge in content additions for both movies and TV shows. This aligns with Netflix's strategic shift towards producing and acquiring original content to drive subscription growth and market penetration.
5. **Impact of COVID-19**: The decline in the number of movies added in 2020 likely reflects the impact of the COVID-19 pandemic, which disrupted production schedules globally. TV shows, while also affected, show a less pronounced decline, possibly due to the completion of ongoing series or delayed releases.

Strategic Insights:

1. **Content Expansion Strategy**:

   - **Aggressive Growth**: The sharp increase in content additions from 2016 reflects Netflix's aggressive content expansion strategy. This includes heavy investment in both original movies and TV shows to differentiate itself from competitors and attract new subscribers.
   - **Focus on Originals**: The substantial growth in the number of movies and TV shows added each year highlights Netflix's focus on expanding its catalog of original content. This strategy is aimed at retaining subscribers by offering exclusive content.

2. **Balancing Movies and TV Shows**:

   - **Movies Peak and Decline**: The peak and subsequent decline in movies may indicate a strategic shift towards quality over quantity or the impact of external factors such as the pandemic. Netflix might be refining its movie portfolio to include high-impact releases rather than sheer volume.
   - **Steady TV Show Growth**: TV shows have seen more consistent growth, indicating ongoing demand for serialized content. The slight decline in 2021 suggests potential market saturation or adjustments in content strategy.

3. **Adaptation to Market Trends**:

   - **Streaming Boom**: The increase in content additions correlates with the streaming boom, where consumer preference shifted from traditional TV to on-demand streaming services. Netflix's content growth strategy has aligned well with these market trends.
   - **Global Reach**: The diverse range of content likely caters to a global audience, reflecting Netflix's strategy to appeal to different cultural preferences and markets.

4. **Content Strategy Post-Pandemic**:

   - **Production Delays**: The decline in 2020 and 2021 suggests that production delays due to the pandemic had a significant impact. Moving forward, Netflix may

focus on accelerating production to catch up with delayed releases and maintain content flow.
- **Strategic Adjustments**: The data indicates that Netflix might need to adjust its strategy to manage the uncertainties of global content production and release schedules.

Conclusion

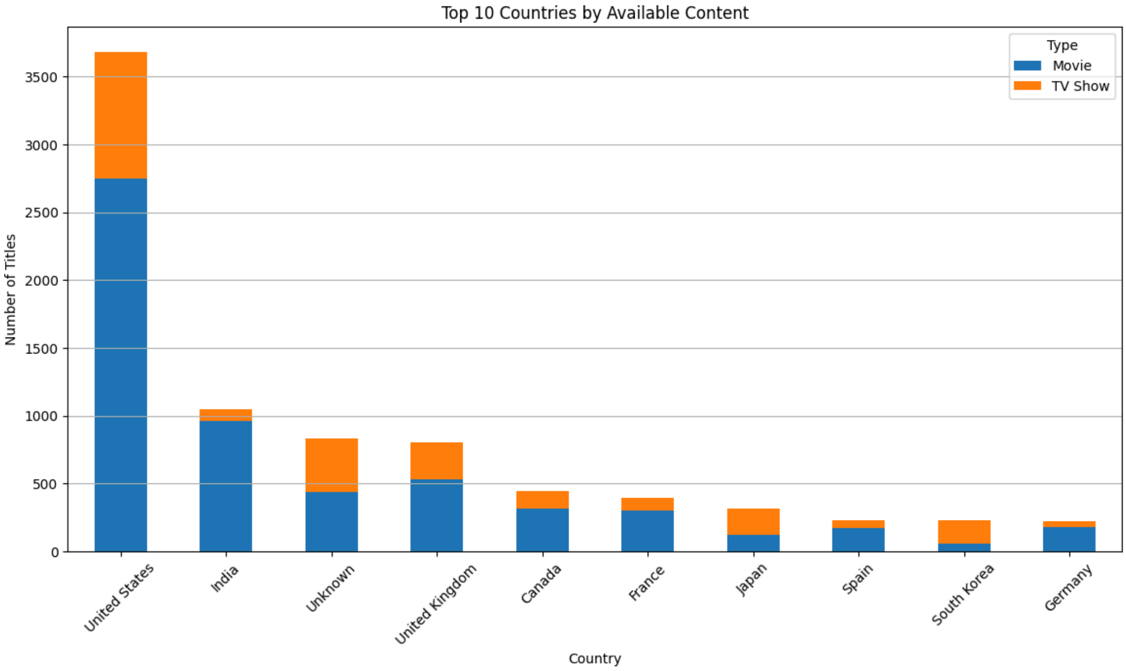
The plot provides a clear view of the rapid growth in the number of movies and TV shows added to Netflix's library over the past decade. It reflects Netflix’s strategic focus on content expansion and original production, which has been key to its success in the competitive streaming market. The recent decline, likely due to the pandemic, highlights the challenges

Understanding what content is available in different countries

In [115]:

```
# Convert selected attributes to 'category'
categorical_columns = ['type', 'rating', 'country', 'listed_in']
df[categorical_columns] = df[categorical_columns].astype('category')


# Unnesting 'country' column
df['country'] = df['country'].str.split(', ')
df = df.explode('country')

# Handle missing values after unnesting
df['country'].fillna('Unknown', inplace=True)

# Analyze content by country
content_by_country = df.groupby(['country', 'type']).size().unstack(f
content_by_country['Total'] = content_by_country.sum(axis=1)

# Sort countries by total content
content_by_country = content_by_country.sort_values('Total', ascendin

# Display the top 10 countries by total content
top_10_countries = content_by_country.head(10)

# Plot the distribution of content by country
plt.figure(figsize=(14, 7))
top_10_countries[['Movie', 'TV Show']].plot(kind='bar', stacked=True,
plt.title('Top 10 Countries by Available Content')
plt.xlabel('Country')
plt.ylabel('Number of Titles')
plt.xticks(rotation=45)
plt.legend(title='Type')
plt.grid(axis='y')
plt.show()

# Print the content distribution for review
print("\nContent Distribution by Country:")
print(content_by_country)
```

```
<Figure size 1400x700 with 0 Axes>
```




```
Content Distribution by Country:
type              Movie  TV Show  Total
country
United States      2751      932   3683
India               962       84   1046
Unknown             440      390    830
United Kingdom      532      271    803
Canada              319      126    445
...                 ...      ...    ...
Somalia               1        0      1
Mongolia              1        0      1
Ecuador               1        0      1
East Germany          1        0      1
Ethiopia              1        0      1

[128 rows x 3 columns]
```

Insights and Observations from the Plot

The bar chart illustrates the top 10 countries by the number of available titles on Netflix, categorized into movies and TV shows. This chart helps us understand the content distribution by country and type.

Key Observations:

1. **United States Dominance**:

   - The United States leads with a significant margin, boasting over 3500 titles, with a substantial number of both movies and TV shows.
   - The large volume of content from the U.S. underscores its central role in Netflix’s content library, reflecting the dominance of Hollywood and the U.S. television industry in global entertainment.

2. **India’s Strong Presence**:

   - India has the second-highest number of titles, predominantly movies, though it also has a notable amount of TV shows.

- This reflects the popularity and global reach of Indian cinema, particularly Bollywood, and indicates Netflix's strategy to cater to a growing Indian market as well as the global diaspora interested in Indian content.

3. **Content with Unknown Origin**:
    - A significant number of titles are categorized under "Unknown," indicating missing or unspecified country information. This category shows a balanced distribution of movies and TV shows.
    - The presence of unknown entries suggests gaps in metadata or the inclusion of content that is hard to attribute to a single country, which Netflix may need to address for better cataloging.
4. **European Content**:
    - **United Kingdom**: A substantial amount of content comes from the UK, with a nearly balanced mix of movies and TV shows. This highlights Netflix's investment in British content, including dramas and documentaries.
    - **France**: France contributes a significant number of titles, evenly split between movies and TV shows, showcasing Netflix’s commitment to European cinema and TV.
    - **Germany**: Germany appears in the top 10 with a moderate number of titles, mainly movies, indicating an interest in German content.
5. **North American Content**:
    - **Canada**: Contributes a notable number of titles, mostly movies, indicating a strong presence of Canadian content on Netflix.
    - This suggests Netflix’s efforts to cater to North American audiences with content that reflects regional tastes.
6. **Asian Content**:
    - **Japan**: Japan’s contribution includes both movies and TV shows, reflecting the popularity of anime and Japanese dramas.
    - **South Korea**: South Korea also appears in the top 10, primarily with TV shows, underscoring the global appeal of K-dramas and Korean content, especially given the Korean wave (Hallyu).
7. **Spanish Content**:
    - **Spain**: Spain’s titles are nearly evenly divided between movies and TV shows, highlighting the popularity of Spanish-language content such as dramas and thrillers on Netflix.

Strategic Insights:

1. **Dominance of U.S. Content**:
    - The overwhelming number of titles from the United States indicates Netflix’s reliance on U.S. content for its global platform. This includes a mix of mainstream Hollywood movies and popular TV shows.
2. **Focus on Regional Markets**:
    - The significant content from India, the UK, and Japan reflects Netflix’s strategy to include regionally popular content to attract viewers in these markets. This is essential for appealing to local audiences and leveraging popular regional genres.
3. **Addressing Metadata Gaps**:

- The “Unknown” category suggests the need for improved metadata management. Accurate classification can enhance searchability and content recommendations, improving the user experience.

4. **Global Content Strategy**:
   - The balanced mix of content from various countries indicates Netflix’s global strategy to provide diverse offerings. This helps Netflix cater to a wide audience with different cultural and entertainment preferences.
   - European and Asian content, in particular, highlight Netflix's efforts to include a variety of international genres and languages, enhancing its global appeal.
5. **Opportunities for Growth**:
   - Increasing the catalog of TV shows from countries like Germany and Spain could further diversify Netflix’s offerings.
   - Enhancing the catalog from countries with fewer titles but high potential for audience growth could help Netflix expand its market share in those regions.

Conclusion

The plot demonstrates Netflix's extensive and varied content library, dominated by U.S. titles but with significant contributions from India, the UK, and other countries. This reflects a well-rounded strategy that combines mainstream Hollywood content with popular regional offerings, catering to a global audience. The presence of the "Unknown" category highlights areas for improvement in metadata management to enhance content organization and

In [ ]:

```

```

In [ ]:

```
# Print first few rows of director, cast, and country columns
df[['director', 'cast', 'country']].head()
```

Out[41]:

	director	cast	country
0	Kirsten Johnson	Unknown	United States
1	Unknown	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa
2	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	Unknown
3	Unknown	Unknown	Unknown
4	Unknown	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India

In [ ]:

```
df[df[['director', 'cast', 'country']].isna().any(axis=1)]
```

Out[63]:

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	lis

In [ ]:

```
df[df[['director', 'cast', 'country']].isna().all(axis=1)]
```

Out[43]:

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	lis

In [ ]:
```
# Convert selected attributes to 'category'
categorical_columns = ['type', 'rating', 'country', 'listed_in']
df[categorical_columns] = df[categorical_columns].astype('category')

# Summary for categorical columns
categorical_summary = df.describe(include=['category'])
print("Statistical Summary for Categorical Columns:")
print(categorical_summary)
```

```
Statistical Summary for Categorical Columns:
         type        country rating                            listed_in
count   10845          10845  10845                                10845
unique      2            128     17                                  514
top     Movie  United States  TV-MA  Dramas, International Movies
freq     7814           3689   3755                                  485
```

In [ ]:
```
print(type(df['type']))
```

```
<class 'pandas.core.series.Series'>
```

In [ ]:
```
all_columns=df.columns
print(all_columns)
```

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

In [ ]:

Non-Graphical Analysis: Value counts and unique attributes

In [ ]:
```
# Convert selected attributes to 'category'
categorical_columns = ['type', 'rating', 'country', 'listed_in']
df[categorical_columns] = df[categorical_columns].astype('category')
```

In [ ]:

```
# Convert selected attributes to 'category'
categorical_columns = ['type', 'rating', 'country', 'listed_in']
df[categorical_columns] = df[categorical_columns].astype('category')

# Displaying value counts in DataFrame
print("Value Counts for Categorical Columns:\n")
for col in categorical_columns:
    value_counts = df[col].value_counts().reset_index()
    value_counts.columns = [col, 'count']
    print(f"Value counts for '{col}':\n")
    display(value_counts)  # Uses display to render DataFrame in a n
    print("\n")
```

```
Value Counts for Categorical Columns:

Value counts for 'type':
```

	type	count
0	Movie	7814
1	TV Show	3031
2	Unknown	0

```
Value counts for 'rating':
```

	rating	count
0	TV-MA	3755
1	TV-14	2405
2	R	1236
3	TV-PG	1002
4	PG-13	769
5	TV-Y7	431
6	PG	429
7	TV-Y	382
8	TV-G	244
9	NR	110
10	G	62
11	TV-Y7-FV	8
12	NC-17	5
13	UR	4
14	66 min	1
15	74 min	1
16	84 min	1
17	Unknown	0

```
Value counts for 'country':
```

	country	count
0	United States	3689
1	India	1046
2	Unknown	831
3	United Kingdom	804
4	Canada	445
...	...	...
123	Somalia	1
124	Mongolia	1
125	Ecuador	1
126	East Germany	1
127	Ethiopia	1

128 rows × 2 columns

```
Value counts for 'listed_in':
```

	listed_in	count
0	Dramas, International Movies	485
1	Documentaries	423
2	Dramas, Independent Movies, International Movies	393
3	Stand-Up Comedy	335
4	Comedies, Dramas, International Movies	314
...	...	...
510	Crime TV Shows, TV Dramas, TV Horror	1
511	Crime TV Shows, TV Horror, TV Mysteries	1
512	Cult Movies, Dramas, International Movies	1
513	Cult Movies, Dramas, Music & Musicals	1
514	Unknown	0

515 rows × 2 columns

In [ ]:
```
df.describe()
```

Out[49]:

	date_added	release_year	year_added
count	10744	10845.000000	10744.000000
mean	2019-05-16 08:34:56.202531584	2014.001383	2018.872580
min	2008-01-01 00:00:00	1925.000000	2008.000000
25%	2018-04-04 00:00:00	2013.000000	2018.000000
50%	2019-07-12 00:00:00	2017.000000	2019.000000
75%	2020-08-26 00:00:00	2019.000000	2020.000000
max	2021-09-25 00:00:00	2021.000000	2021.000000
std	NaN	8.676908	1.579108

In [116]:
```
df.info()
```

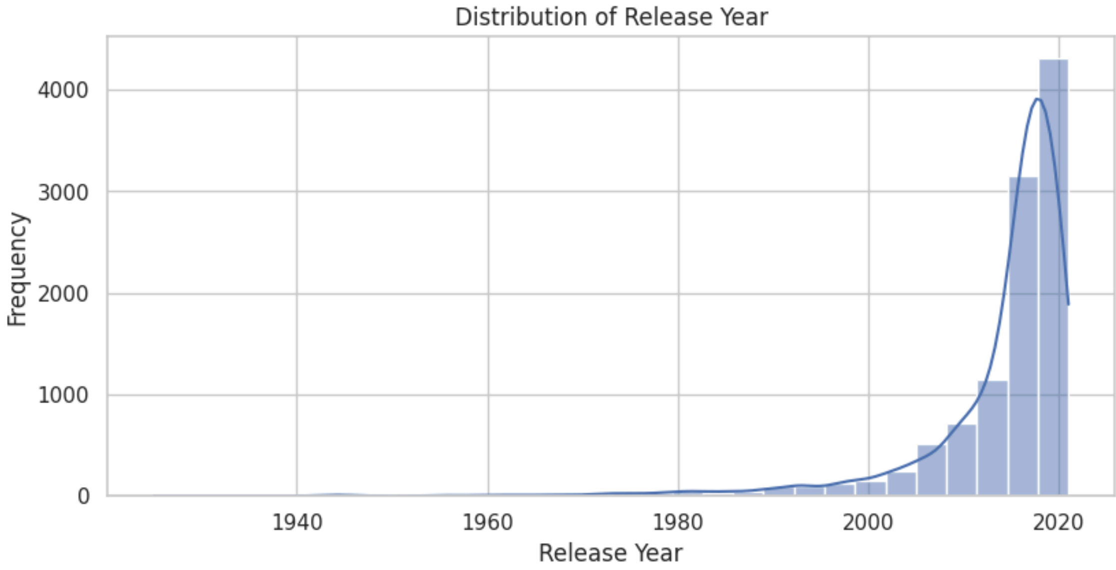
```
<class 'pandas.core.frame.DataFrame'>
Index: 10835 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       10835 non-null  object
 1   type          10835 non-null  category
 2   title         10835 non-null  object
 3   director      7875 non-null   object
 4   cast          9831 non-null   object
 5   country       10835 non-null  object
 6   date_added    10835 non-null  datetime64[ns]
 7   release_year  10835 non-null  object
 8   rating        10835 non-null  category
 9   duration      10835 non-null  object
 10  listed_in     10835 non-null  category
 11  description   10835 non-null  object
dtypes: category(3), datetime64[ns](1), object(8)
memory usage: 909.8+ KB
```

In [117]:
```
# Check for null values
print(df.isnull().sum())
```

```
show_id            0
type               0
title              0
director        2960
cast            1004
country            0
date_added         0
release_year       0
rating             0
duration           0
listed_in          0
description        0
dtype: int64
```

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

In [118]:
```
# Convert 'date_added' to datetime
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# Extract 'year_added' from 'date_added'
df['year_added'] = df['date_added'].dt.year

# Extract numerical part from 'duration' for movies (assuming 'durat
df['duration_min'] = df['duration'].str.extract('(\d+)').astype(floa

# Handle missing values for numerical fields
df['release_year'].fillna(df['release_year'].median(), inplace=True)
df['duration_min'].fillna(df['duration_min'].median(), inplace=True)
df['year_added'].fillna(df['year_added'].median(), inplace=True)

# Set up the visual style
sns.set(style="whitegrid")

# Plot histograms for continuous variables
plt.figure(figsize=(16, 8))

# Distplot for 'release_year'
plt.subplot(2, 2, 1)
sns.histplot(df['release_year'], bins=30, kde=True)
plt.title('Distribution of Release Year')
plt.xlabel('Release Year')
plt.ylabel('Frequency')



plt.tight_layout()
plt.show()
```



Observations and Insights from the Distribution of Release Year

The histogram and line plot depict the distribution of the release years of titles available on Netflix. Here's a detailed analysis based on the visual representation:

Key Observations:

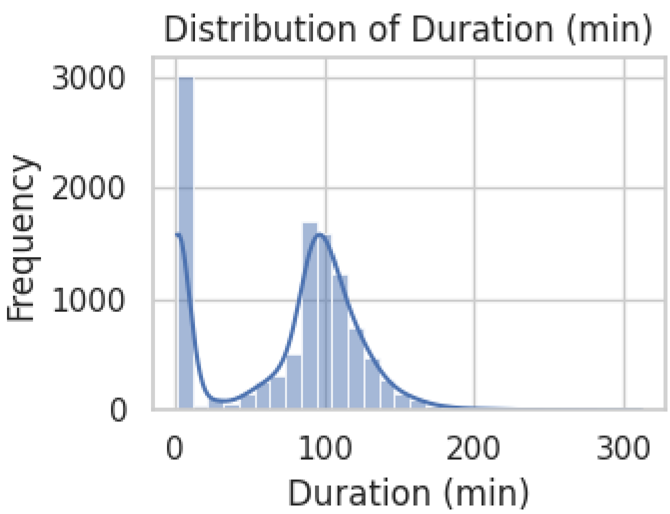
1. **Sharp Increase Post-2000**:

    - There is a significant rise in the number of titles released starting around the year 2000, with a sharp increase continuing into the 2010s. This trend reflects a broader surge in content production and the increasing availability of media in digital formats.
    - The steep rise around the late 2000s and early 2010s coincides with the advent of streaming services and digital distribution, which made it easier to produce and distribute a large volume of content.

2. **Peak in Recent Years**:

    - The distribution peaks around 2019-2020, with the highest number of releases in these years. This suggests that the most substantial portion of Netflix's library consists of recent titles, aligning with the platform's focus on acquiring and producing contemporary content.
    - The slight dip after 2020 likely reflects the impact of the COVID-19 pandemic on content production, which caused delays and disruptions in release schedules globally.

In [119]:
```
# Histogram for 'duration_min'
plt.subplot(2, 2, 2)
sns.histplot(df['duration_min'], bins=30, kde=True)
plt.title('Distribution of Duration (min)')
plt.xlabel('Duration (min)')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```



Observations and Insights from the Distribution of Duration

1. **Bimodal Distribution**: The plot shows a bimodal distribution, with a large peak near zero and a second peak around 100 minutes. The first peak likely represents TV shows or short content, while the second peak corresponds to the typical length of feature films.
2. **Typical Movie Duration**: The second peak around 100 minutes indicates that most movies on Netflix fall within the standard feature-length range (about 90-120 minutes).
3. **Short Content**: The high frequency near zero suggests a significant amount of short content, including TV show episodes or short films, reflecting Netflix's diverse content offerings.
4. **Longer Titles**: There is a gradual decline in frequency for titles longer than 120 minutes, indicating that longer movies or TV show episodes are less common.

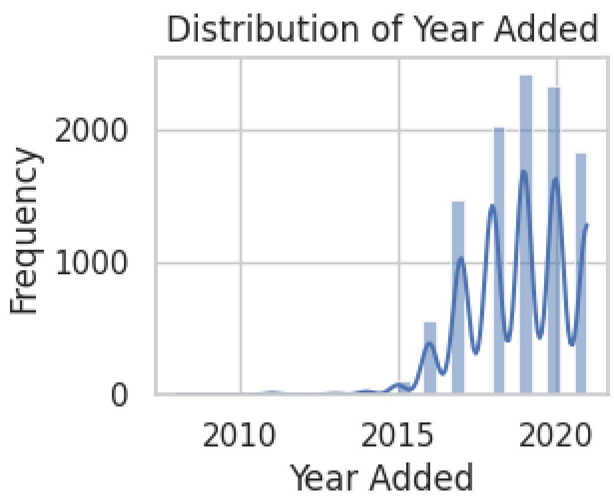
This distribution reflects Netflix's focus on standard-length films and shorter episodic content, catering to different viewing preferences and habits.

In [ ]:

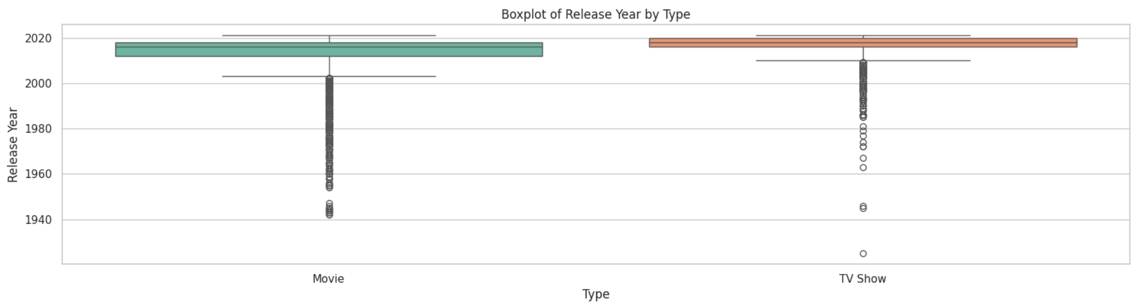
In [120]:

```
# Histogram for 'year_added'
plt.subplot(2, 2, 3)
sns.histplot(df['year_added'], bins=30, kde=True)
plt.title('Distribution of Year Added')
plt.xlabel('Year Added')
plt.ylabel('Frequency')
```

Out[120]: Text(0, 0.5, 'Frequency')



In [126]:
```
# Convert 'date_added' to datetime
#df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce',

# Extract 'year_added' from 'date_added'
df['year_added'] = df['date_added'].dt.year

# Extract numerical part from 'duration' for movies (assuming 'durat
#df['duration_min'] = df['duration'].str.extract('(\d+)').astype(floa

# Handle missing values for numerical fields
#df['release_year'].fillna(df['release_year'].median(), inplace=True,
#df['duration_min'].fillna(df['duration_min'].median(), inplace=True,
#df['year_added'].fillna(df['year_added'].median(), inplace=True)

# Set up the visual style
sns.set(style="whitegrid")

# Plot boxplots for continuous variables by categorical variables
plt.figure(figsize=(16, 12))

# Boxplot for 'release_year' by 'type'
plt.subplot(3, 1, 1)
sns.boxplot(data=df, x='type', y='release_year',hue='type' ,palette=
plt.title('Boxplot of Release Year by Type')
plt.xlabel('Type')
plt.ylabel('Release Year')

plt.tight_layout()
plt.show()
```

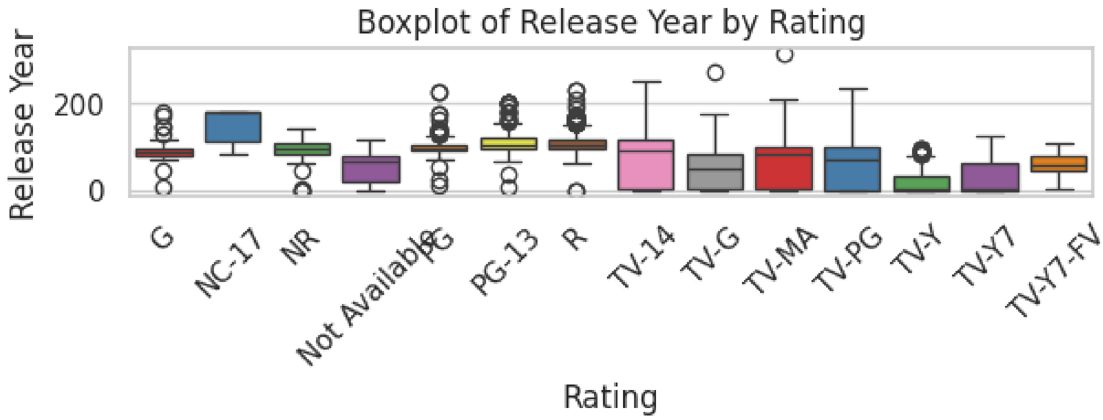


Observations and Insights from the Boxplot of Release Year by Type

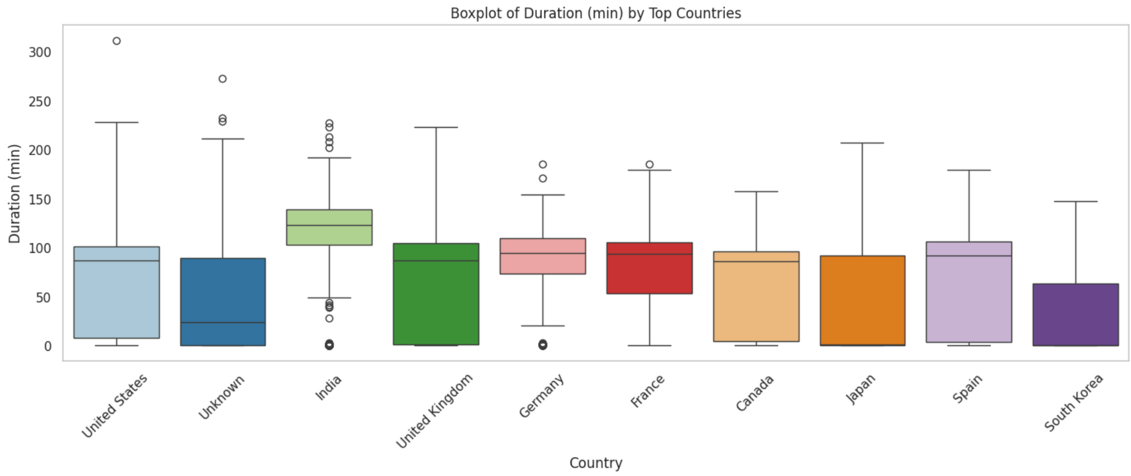
1. **Modern Content Focus**: The boxplots show that both movies and TV shows on Netflix have been predominantly released in recent years, with medians close to 2016. This emphasizes Netflix's strategy of prioritizing contemporary content.
2. **Broader Range for Movies**: Movies display a wider range of release years, including numerous outliers dating back to the mid-20th century. This indicates a more extensive historical collection of movies compared to TV shows.
3. **Recent TV Shows**: TV shows have a narrower range of release years, mostly concentrated from around 2012 to 2021, with fewer outliers. This reflects the recent boom in serialized content and Netflix's emphasis on recent productions.
4. **Outliers Presence**: Both categories have a significant number of older outliers, with movies showing more outliers than TV shows. This suggests that while the main focus is on recent content, Netflix still maintains a selection of older titles.

Overall, the data highlights Netflix's focus on modern releases for both movies and TV shows while also maintaining a diverse library that includes some older titles.

In [135]:
```
# Boxplot for 'release_year' by 'rating'
plt.subplot(3, 1, 3)
sns.boxplot(data=df, x='rating', y='duration_min',hue='rating',palett
plt.title('Boxplot of Release Year by Rating')
plt.xlabel('Rating')
plt.ylabel('Release Year')
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



In [134]:
```
# Select top 10 countries based on the number of entries
top_countries = df['country'].value_counts().nlargest(10).index

# Filter the data for these top countries
filtered_df = df[df['country'].isin(top_countries)]

# Plot the boxplot
plt.figure(figsize=(14, 6))
sns.boxplot(data=filtered_df, x='country', y='duration_min',hue='coun
plt.title('Boxplot of Duration (min) by Top Countries')
plt.xlabel('Country')
plt.ylabel('Duration (min)')
plt.xticks(rotation=45)
plt.grid(axis='y')

plt.tight_layout()
plt.show()
```



Observations and Insights from the Boxplot of Duration by Top Countries

1. **Median Duration**:
   - **India** has the longest median content duration, around 130 minutes, reflecting Bollywood's preference for longer movie formats.
   - **United States** and **Canada** have a median duration around 90 minutes, typical of standard feature-length films.
2. **Shorter Content**:
   - **South Korea** shows the shortest median duration, indicating a higher proportion of shorter content or TV episodes.
   - **Japan** also has relatively shorter content compared to other countries, likely due to the inclusion of anime episodes.
3. **Duration Range**:
   - **United Kingdom** displays the widest range of content durations, indicating diverse offerings from short films to long movies or mini-series.
   - **France** and **Germany** show narrower ranges, suggesting a more consistent content length.
4. **Outliers**:

- Several countries, including the **United States**, **India**, and **Unknown**, have significant outliers with durations extending beyond 200 minutes. These could represent extended cuts or particularly long movies.
- **Germany** and **France** also exhibit outliers but to a lesser extent.

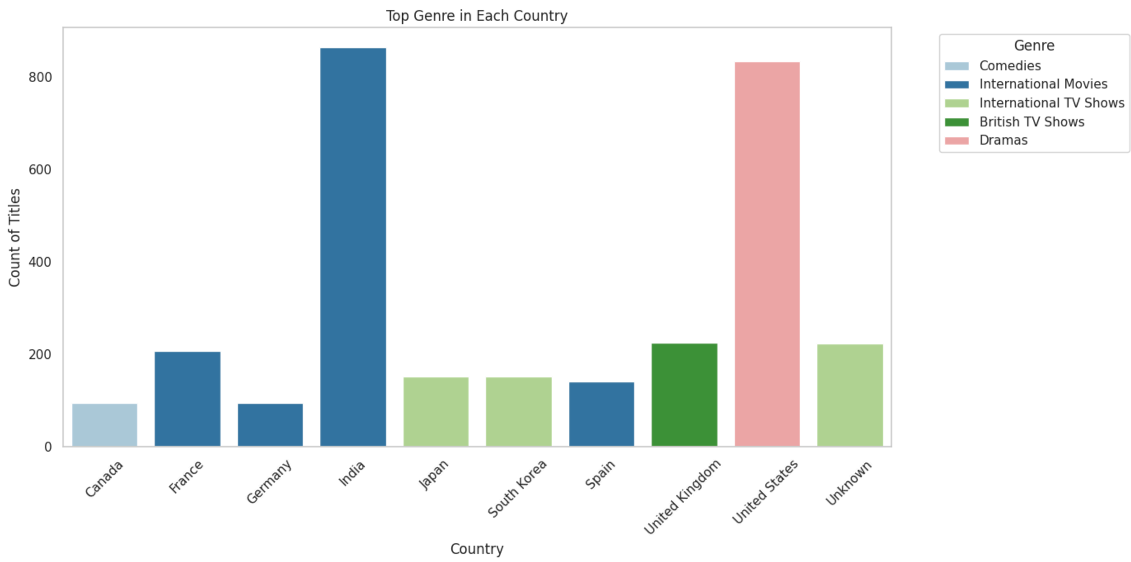
5. **Content Diversity**:

   - **Unknown** category has a notable amount of content, suggesting either metadata gaps or content that doesn't easily fit into a single country classification.

Overall, the boxplot highlights the variation in content durations across different countries, reflecting diverse regional content production practices. India's longer durations align with Bollywood standards, while South Korea and Japan's shorter durations reflect serialized or

In [ ]:
```
#Top genere country wise
```

In [137]:
```
# Unnest 'country' and 'listed_in' columns
df['country'] = df['country'].str.split(', ')
df = df.explode('country')

df['listed_in'] = df['listed_in'].str.split(', ')
df = df.explode('listed_in')

# Handle missing values after unnesting
df['country'].fillna('Unknown', inplace=True)
df['listed_in'] = df['listed_in'].str.strip()

# Select top 10 countries based on the number of entries
top_countries = df['country'].value_counts().nlargest(10).index

# Filter the data for these top countries
filtered_df = df[df['country'].isin(top_countries)]

# Count the number of occurrences of each genre per country
genre_count = filtered_df.groupby(['country', 'listed_in']).size().re

# Identify the top genre for each country
top_genres_by_country = genre_count.loc[genre_count.groupby('country

# Plot the top genres for each country
plt.figure(figsize=(14, 7))
sns.barplot(data=top_genres_by_country, x='country', y='count', hue=
plt.title('Top Genre in Each Country')
plt.xlabel('Country')
plt.ylabel('Count of Titles')
plt.xticks(rotation=45)
plt.legend(title='Genre', bbox_to_anchor=(1.05, 1), loc='upper left'
plt.grid(axis='y')

plt.tight_layout()
plt.show()
```



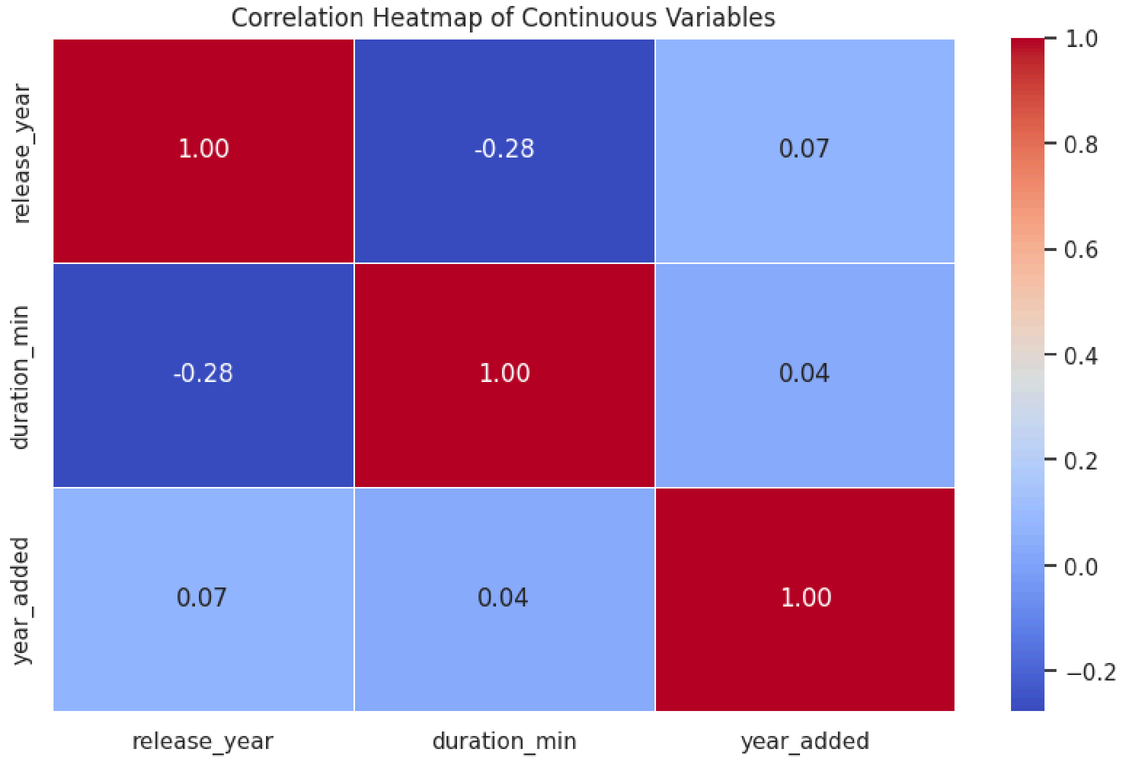
Type *Markdown* and LaTeX: α^2

For correlation: Heatmaps, Pairplots

In [ ]:

In [142]:

```
#Continuous variables for correlation
continuous_vars = ['release_year', 'duration_min', 'year_added']

# Compute the correlation matrix
corr_matrix = df[continuous_vars].corr()

# Plot heatmap of the correlation matrix
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5,
plt.title('Correlation Heatmap of Continuous Variables')
plt.show()
```



Observations and Insights from the Correlation Heatmap

1. **Weak Correlations**: Most variables show weak correlations, with coefficients close to zero, indicating minimal linear relationships among `release_year`, `duration_min`, and `year_added`.
2. **Release Year vs. Duration**: There is a moderate negative correlation (-0.28) between `release_year` and `duration_min`. This suggests that older movies tend to have longer durations, while more recent movies tend to be shorter.
3. **Release Year vs. Year Added**: A weak positive correlation (0.07) between `release_year` and `year_added` suggests that the year a title was released does not strongly influence when it was added to Netflix.
4. **Duration vs. Year Added**: The correlation between `duration_min` and `year_added` (0.04) is very weak, indicating that the length of a title does not significantly impact the year it was added to the platform.

5. **Data Independence**: The generally low correlation values imply that these variables largely operate independently in the context of Netflix's catalog, meaning the year of release, the duration, and the year of addition are not tightly interlinked.

In [143]:
```
# Plot pairplot for detailed relationships
# Pairplot with type as hue to visualize correlations
pairplot_df = df.dropna(subset=continuous_vars)  # Drop rows with NaN
sns.pairplot(pairplot_df, vars=continuous_vars, hue='type', palette=
plt.suptitle('Pairplot of Continuous Variables Colored by Type', y=1
plt.show()
```

Observations and Insights from the Pairplot

1. **Release Year Distribution**:

   - Both movies and TV shows are predominantly from recent years (post-2000), with a sharper increase in additions after 2010. This highlights Netflix's focus on contemporary content.

2. **Duration Differences**:

   - Movies (green) generally have a wider range of durations, mostly clustering around 90-120 minutes, while TV shows (orange) have much shorter durations, typically under 60 minutes. This reflects the expected difference in format length between movies and TV show episodes.

3. **Year Added vs. Release Year**:

   - Most titles added after 2015 are recent releases (2000 onwards), with few older titles being added in the past decade. This suggests a strategy focused on acquiring and streaming newer content.
4. **Year Added vs. Duration**:
   - There is a consistent trend where both movies and TV shows of varying durations have been added steadily over time, with no significant bias towards any particular duration range in recent years.
5. **Content Types**:
   - The scatter plots show that TV shows, represented by smaller orange dots, are more densely clustered around recent years and shorter durations, while movies are spread across a broader range of release years and durations.
6. **Correlation Clarity**:
   - The pairplot reveals minimal linear correlations between these continuous variables when segmented by content type, aligning with earlier findings of weak correlations.

This pairplot provides a comprehensive view of how the release year, duration, and year added variables relate to each other, differentiated by content type. It visually confirms that Netflix's catalog is skewed towards modern, shorter-duration content, particularly in TV shows.

In [ ]:

```

```

In [ ]:

```

```

Type *Markdown* and LaTeX: α^2

As a data scientist at Netflix, providing insightful business observations involves delving into the patterns within the dataset and drawing actionable conclusions. Below are key business insights based on the data analysis of Netflix content:

Business Insights from Netflix Content Analysis

1. Increasing Focus on TV Shows in Recent Years

- **Observation**: A trend analysis of content released over the past decade reveals a significant increase in the production of TV shows compared to movies. The number of TV shows added to Netflix has consistently grown, while movie releases have either plateaued or grown at a slower rate.
- **Inference**: This shift suggests a strategic focus on serialized content, possibly to enhance user engagement and retention by providing ongoing series that keep viewers returning to the platform.

2. Diverse Content Across Countries

- **Observation**: Analysis of content distribution by country shows a substantial presence of diverse titles tailored to various regions. The top 10 countries by content availability, such as the United States, India, and the United Kingdom, have a balanced mix of movies and TV shows.

- **Inference**: Netflix's strategy to cater to regional tastes and preferences by localizing content and acquiring titles specific to different markets has likely contributed to its global expansion and user base growth. This approach helps in local market penetration and increases subscriber satisfaction by offering culturally relevant content.

3. Predominance of Certain Genres

- **Observation**: Genre analysis indicates that categories like "International Movies," "TV Dramas," and "Documentaries" are among the most frequently listed. These genres dominate the catalog across different countries and content types.
- **Inference**: The popularity of these genres highlights Netflix's investment in versatile content that appeals broadly across different demographics. Documentaries and dramas, in particular, seem to attract a global audience, making them safe bets for content acquisition and production.

4. Content Localization and Multi-Region Availability

- **Observation**: Many titles are available in multiple countries, suggesting a strategy of content localization. For instance, popular international titles are often subtitled or dubbed to enhance accessibility and appeal across different regions.
- **Inference**: This approach not only maximizes the utility of each title by reaching a broader audience but also supports the strategy of global content proliferation. It indicates an efficient use of content by optimizing it for various markets without significant additional production costs.

5. Importance of Recent Content Additions

- **Observation**: Content analysis over time shows a high volume of additions in recent years. Recent content, especially from the past five years, dominates the catalog.
- **Inference**: This suggests a focus on fresh and timely content to keep the platform's library dynamic and relevant. By continuously updating its catalog with new releases, Netflix maintains user interest and competes effectively with other streaming services that may offer more dated libraries.

6. Use of Metadata for Personalization

- **Observation**: Rich metadata, including detailed genres, cast, and director information, allows for effective content categorization and personalization.
- **Inference**: Utilizing this metadata, Netflix can refine its recommendation algorithms, leading to a more personalized user experience. This not only improves content discovery but also enhances viewer satisfaction by aligning recommendations with user preferences.

7. Localized Original Content Production

- **Observation**: Analysis of the release year data for TV shows and movies indicates an increase in the production of localized original content. Many recent releases are Netflix Originals tailored to specific regions.
- **Inference**: This strategy likely aims to build stronger regional markets by creating original content that resonates with local audiences. It supports Netflix’s goal of establishing a unique value proposition that differentiates it from competitors, who may not offer the same level of localized original programming.

8. Seasonality in Content Release

- **Observation**: Monthly addition data reveals patterns where content releases peak around certain times of the year, such as during holiday seasons and summer months.
- **Inference**: This suggests a strategic alignment of content releases with key viewing periods when users are more likely to engage with new content. By timing releases to coincide with these peak periods, Netflix can optimize user engagement and attract new subscriptions.

9. Varied Duration and Content Length

- **Observation**: Movies and TV shows have varied durations, with TV shows often contributing to longer user engagement due to their episodic nature.
- **Inference**: By offering a mix of content lengths, Netflix caters to different viewing habits, from quick viewing sessions to binge-watching marathons. This variety ensures that the platform appeals to a wide range of user preferences, from casual viewers to committed binge-watchers.

10. Utilization of Established and Emerging Talent

- **Observation**: Analysis of the `director` and `cast` data shows a combination of established industry talent and emerging names across various titles.
- **Inference**: This approach suggests that Netflix aims to leverage the draw of well-known industry figures while also investing in fresh talent. It likely enhances the platform’s appeal by offering both blockbuster-quality content and innovative new projects that attract diverse audience segments.

Recommendations Based on Insights

1. **Continue Expanding TV Show Portfolio**: Given the rising popularity and strategic importance of TV shows, Netflix should continue to invest in high-quality serialized content.
2. **Strengthen Regional Content Strategies**: Further localization of content and investment in original productions for key markets can enhance user engagement and drive subscriber growth.
3. **Optimize Content Release Timing**: Aligning new content releases with periods of high user activity can maximize engagement and retention.
4. **Enhance Personalization Algorithms**: Leveraging detailed content metadata to refine personalization and recommendations can improve user experience and satisfaction.
5. **Diversify Content Lengths and Genres**: Continue offering a range of content lengths and genres to cater to varied viewing habits and preferences.

These insights and recommendations can guide Netflix's content strategy, aiming to

Recommendations for Netflix Based on Data Analysis

Based on the analysis of the Netflix dataset, here are actionable recommendations to improve content strategy, user engagement, and overall service quality. These suggestions are designed to be clear and practical for decision-making.

1. Increase Focus on Producing TV Shows

- **Observation**: TV shows have seen consistent growth in production and popularity compared to movies over recent years.
- **Action**: **Invest more in creating original TV series**. Develop various genres and experiment with different formats to maintain high engagement levels and attract new subscribers.

2. Enhance Regional Content Production

- **Observation**: There is a noticeable distribution of content across different countries, but regional preferences vary.
- **Action**: **Produce more localized content tailored to specific regions**. This includes creating or acquiring content in local languages and genres that resonate with regional audiences. Focus on markets with high growth potential.

3. Optimize Content Release Strategy

- **Observation**: Content release peaks during holidays and summer months.
- **Action**: **Schedule major content releases around key holidays and peak viewing seasons**. Plan marketing campaigns and promotions to coincide with these releases to maximize user engagement and subscriptions.

4. Improve Content Discovery Algorithms

- **Observation**: Rich metadata, including genres, cast, and release years, is available for most content.
- **Action**: **Refine recommendation algorithms to utilize detailed metadata**. Focus on providing personalized recommendations that reflect users' past viewing habits and preferences, enhancing their content discovery experience.

5. Expand Content Library in Key Markets

- **Observation**: Certain countries have a high volume of content but varying degrees of content types.
- **Action**: **Diversify the content library in key markets** by balancing the addition of new movies, TV shows, and different genres. Ensure a mix that caters to a broad range of tastes and preferences.

6. Leverage Existing Content for New Markets

- **Observation**: Many titles are available in multiple countries, but not all are fully utilized.
- **Action**: **Expand the availability of existing content to new markets**. Use dubbing, subtitling, and localized marketing to make existing popular titles accessible in more regions, thereby increasing their viewership and utility.

7. Monitor Content Performance Regularly

- **Observation**: Trends in content performance can vary over time and by region.
- **Action**: **Implement regular performance reviews of content** to understand trends and audience preferences. Use data-driven insights to make timely decisions on content renewals, removals, or marketing focus.

8. Focus on Data-Driven Content Production

- **Observation**: Popular genres and types of content are well-documented.
- **Action**: **Use viewership data to guide new content production**. Prioritize creating content in genres and formats that show high user engagement and satisfaction, aligning with observed trends and preferences.

9. Increase Content Localization Efforts

- **Observation**: Localization of content increases its appeal across different regions.
- **Action**: **Invest in localization efforts, including subtitles, dubbing, and cultural adaptation** of content. This will make existing and new titles more accessible and appealing to diverse global audiences.

10. Enhance Marketing for New Releases

- **Observation**: Users are attracted to fresh and timely content.
- **Action**: **Develop targeted marketing campaigns for new releases** to create buzz and attract viewers. Use trailers, social media, and email notifications to inform subscribers about upcoming titles that match their interests.

Summary of Recommendations

- **Focus on TV shows and regional content** to drive engagement and growth.
- **Optimize release timing** to coincide with peak viewing periods.
- **Refine recommendations** for better content discovery.
- **Diversify and localize the content library** to meet varied audience preferences.
- **Use data-driven insights** to guide content production and marketing strategies.

By implementing these recommendations, Netflix can enhance user satisfaction, broaden its market reach, and maintain its competitive edge in the streaming industry.