Netflix, Inc.: Transforming Entertainment through Technological Innovation

Established in 1997 by Reed Hastings and Marc Randolph, Netflix, Inc. has become a leading American technology and media service provider, headquartered in California. From its origins delivering DVDs via mail, the company strategically pivoted to internet streaming, positioning itself as a pioneer in on-demand, accessible content. Offering a comprehensive subscription service, Netflix boasts a vast library featuring third-party content alongside an extensive collection of original productions. Its global influence is evidenced by a user-friendly website, facilitating widespread popularity and making it an integral part of households worldwide. Recognized for creating compelling original shows and movies, Netflix is acknowledged as a key innovator in the entertainment sector. Beyond geographical boundaries, Netflix has reshaped viewing habits, redefining how people experience entertainment at home. The company's unwavering commitment to technological advancements and content creation solidifies its role as a transformative force shaping the future of the entertainment industry.

Data Analysis Project Objectives:

Content Distribution Analysis: Investigate the distribution of content types (movies, TV shows) on the Netflix platform.

Genre Exploration: Examine the prevalence of various genres in Netflix's content catalog.

Temporal Trends Examination: Analyze the evolution of Netflix content over different time periods to identify trends.

Audience and Rating Understanding: Gain insights into the target audience of Netflix by analyzing viewer demographics and content ratings.

Regional Preferences Investigation: Investigate whether specific countries or regions exhibit distinct content preferences or contribute unique types of content to the Netflix platform.

1. Defining Problem Statement and Analysing basic metrics

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import plotly.express as px
```

Loading the dataset

In [2]:
```
import requests

# URL of the dataset
url = 'https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/original

# Download the file
response = requests.get(url)
with open('netflix.csv', 'wb') as file:
    file.write(response.content)

# Read the CSV file
df = pd.read_csv('netflix.csv')

# Display the first few rows to check if the download and upload were successful
df.head()
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duratio
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 mi
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	Seasor
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Seaso
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Seaso
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	Seasor

In [3]:
```
df
```

Out[3]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	du
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	S
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	S
...	...	...	...	...	...	...	...	...	...	
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2007	R	1
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2018	TV-Y7	S
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	R	

	show_id	type	title	director	cast	country	date_added	release_year	rating	du
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2006	PG	
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2015	TV-14	1

The dataset comprises more than 8,807 titles and includes 12 descriptions. Upon an initial glance at the data frames, it appears to be a standard movie and TV shows dataset, lacking ratings information. Additionally, it's observed that some columns contain NaN values, indicating missing data.

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 [4]:
```
df.columns
```

Out[4]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

Attributes of the columns present in the dataset.

In [5]:
```
df.ndim    #Shape of data
```

Out[5]:
```
2
```

Data types of all the attributes available

In [6]:
```
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
```

Statistical Summary Before Data Cleaning

In [7]:
```
df.describe()
```

Out[7]:

	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

Missing Value Detection

Data Profiling & Cleaning

Data cleaning is the fundamental process in Data Science that involves identifying and addressing incorrect, incomplete, inaccurate, irrelevant, or missing pieces of data. This crucial step aims to modify, replace, or delete problematic data to enhance the overall quality and reliability of the dataset.

In [8]:
```
print('\nColumns with missing value:')
print(df.isnull().any())
```

```
Columns with missing value:
show_id         False
type            False
title           False
director         True
cast             True
country          True
date_added       True
release_year    False
rating           True
duration         True
listed_in       False
description     False
dtype: bool
```

Based on the information, we have a dataset with 8807 entries and 12 columns for our exploratory data analysis (EDA). Several columns, including "director," "cast," "country," "date_added," and "rating," have some missing values.

In [9]:
```
df.T.apply(lambda x: x.isnull().sum(), axis = 1)
```

Out[9]:
```
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 [10]:
```
df.isnull().sum().sum()
```

Out[10]:
```
4307
```

There are a total of 4307 null values across the entire dataset, with 2634 missing points under "director," 825 under "cast," 831 under "country," 11 under "date_added," 4 under "rating," and 3 under “duration.” We need to handle all null data points before delving into exploratory data analysis (EDA) and modeling.

Imputation is a treatment method for missing values, involving the filling in of gaps using certain techniques.

One can use mean, mode, or predictive modeling for this purpose. In this case study, we will discuss the use of the fillna function from Pandas for imputation. Alternatively, dropping rows containing missing values is another option, and the dropna function from Pandas can be employed for this approach.

In [11]:
```
df.director.fillna("No Director", inplace=True)
df.cast.fillna("No Cast", inplace=True)
df.country.fillna("Country Unavailable", inplace=True)
df.dropna(subset=["date_added", "rating"], inplace=True)
df.dropna(subset=['duration'], inplace=True)
```

Check Missing Value

In [12]:
```
df.isnull().any()
```

Out[12]:
```
show_id         False
type            False
title           False
director        False
cast            False
country         False
date_added      False
release_year    False
rating          False
duration        False
listed_in       False
description     False
dtype: bool
```

Handling missing values involves making decisions on how to deal with the gaps in the data. The easiest approach would be to eliminate rows with missing data, but this could result in a loss of valuable information for our Exploratory Data Analysis (EDA). Given that "director," "cast," and "country" have a significant number of null values, we have chosen to consider each missing value as unavailable. The other two columns, namely "date_added" and "rating," have an insignificant proportion of missing data, so they are dropped from the dataset. After these steps, we observe that there are no more missing values in the data frame.

Statistical Summary After Data Cleaning:

In [13]:
```
df.describe()
```

Out[13]:

	release_year
count	8790.000000
mean	2014.183163
std	8.825466
min	1925.000000
25%	2013.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

3. Non-Graphical Analysis: Value counts and unique attributes

Non-graphical analysis entails computing summary statistics without relying on visual representations. The Pandas library provides three main functions for this purpose, and I will discuss each of them.

1. info()
2. isna().sum() or isnull().sum()
3. describe()

Checking the data using .head()

In [14]:
```
df.head()
```

Out[14]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	dura
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	No Cast	United States	September 25, 2021	2020	PG-13	90
1	s2	TV Show	Blood & Water	No Director	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	Sea
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	Country Unavailable	September 24, 2021	2021	TV-MA	1 Se
3	s4	TV Show	Jailbirds New Orleans	No Director	No Cast	Country Unavailable	September 24, 2021	2021	TV-MA	1 Se
4	s5	TV Show	Kota Factory	No Director	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	Sea

1. info()

Primarily, this function provides information on the number of features, non-null count, and data type for each feature. Additionally, it displays the count of features present in each data type, aiding in the understanding of the numerical and categorical features in the dataset.

In [15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8790 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8790 non-null   object
 1   type          8790 non-null   object
 2   title         8790 non-null   object
 3   director      8790 non-null   object
 4   cast          8790 non-null   object
 5   country       8790 non-null   object
 6   date_added    8790 non-null   object
 7   release_year  8790 non-null   int64 
 8   rating        8790 non-null   object
 9   duration      8790 non-null   object
 10  listed_in     8790 non-null   object
 11  description   8790 non-null   object
dtypes: int64(1), object(11)
memory usage: 892.7+ KB
```

2. Read the Description of the data

In [16]:
```
df.describe()
```

Out[16]:

	release_year
count	8790.000000
mean	2014.183163
std	8.825466
min	1925.000000
25%	2013.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

3. isna().sum() or isnull().sum()

In [17]:
```
df.T.apply(lambda x: x.isnull().sum(), axis = 1)
```

Out[17]:
```
show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

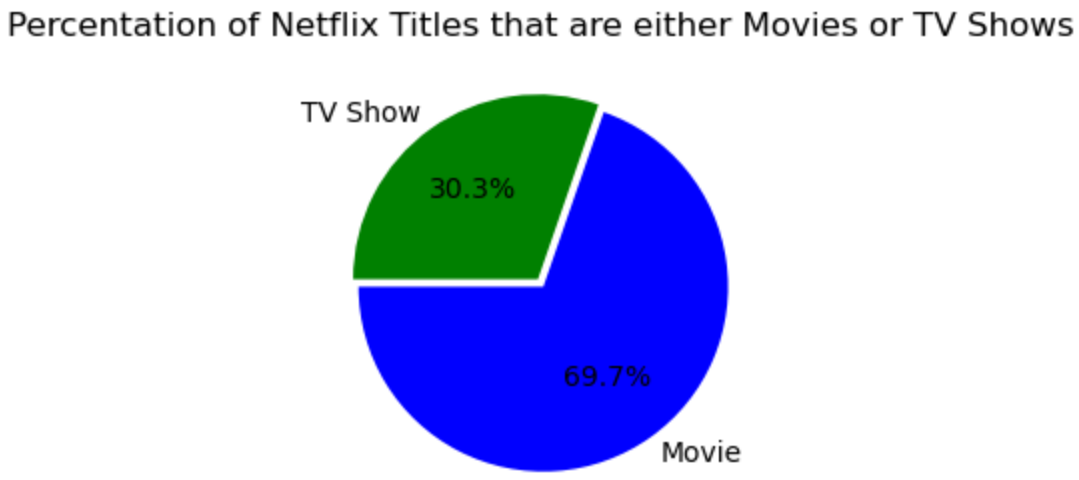
Visual Analysis - Univariate, Bivariate after pre-processing of the data

Note: Pre-processing includes expanding nested data in columns like Actor, Director, Country.

Univariate analysis involves the examination of a single variable. In this graphical analysis, we will not delve into the mathematical concepts behind these analyses for now; let's observe them through graphs. Please have a basic understanding of these concepts, especially if you find them challenging to grasp through visualizations.

A==>Pie plot: Netflix Content By Type This analysis encompasses the entire Netflix dataset, including both movies and shows. Let's compare the total number of movies and shows in this dataset to determine the majority.

In [18]:
```
plt.figure(figsize=(6,3))
plt.title("Percentation of Netflix Titles that are either Movies or TV Shows")
g=plt.pie(df.type.value_counts(),explode=(0.025,0.025),
labels=df.type.value_counts().index, colors=['blue','green'],autopct='%1.1f%%',
startangle=180)
plt.show()
```



There are far more movie titles (69.7%) that TV shows titles (30.3%) in terms of title.

2. Amount of Content Over Time: Distplot

In this analysis, we will investigate the volume of content that Netflix has added over the years. Since our focus is on the timeline of when Netflix introduced titles to its platform, we will create a "year_added" column to extract the year from the "date_added" column.

In [20]:
```
df["year_added"] = pd.to_datetime(df.date_added).dt.year
netflix_movies_df = df[df['type'] == 'Movie'].copy()
netflix_shows_df = df[df['type'] == 'TV Show'].copy()
netflix_movies_df.loc[:, "year_added"] = pd.to_datetime(netflix_movies_df.date_added).
netflix_shows_df.loc[:, "year_added"] = pd.to_datetime(netflix_shows_df.date_added).dt
netflix_year_df = pd.concat([netflix_movies_df, netflix_shows_df])
netflix_year_df = netflix_year_df["year_added"].value_counts().to_frame().reset_index(
netflix_year_df = netflix_year_df[netflix_year_df.year != 2020]
print(netflix_year_df)
```

```
    year  count
0   2019   2016
2   2018   1648
3   2021   1498
4   2017   1185
5   2016    426
6   2015     82
7   2014     24
8   2011     13
9   2013     11
10  2012      3
11  2009      2
12  2008      2
13  2010      1
```

In [21]:
```
movies_year_df = netflix_movies_df.year_added.value_counts().to_frame().reset_index().
"year", "year_added":"count"})
movies_year_df = movies_year_df[movies_year_df != 2020]
movies_year_df
```

Out[21]:

	year	count
0	2019.0	1424
1	NaN	1284
2	2018.0	1237
3	2021.0	993
4	2017.0	836
5	2016.0	251
6	2015.0	56
7	2014.0	19
8	2011.0	13
9	2013.0	6
10	2012.0	3
11	2009.0	2
12	2008.0	1
13	2010.0	1

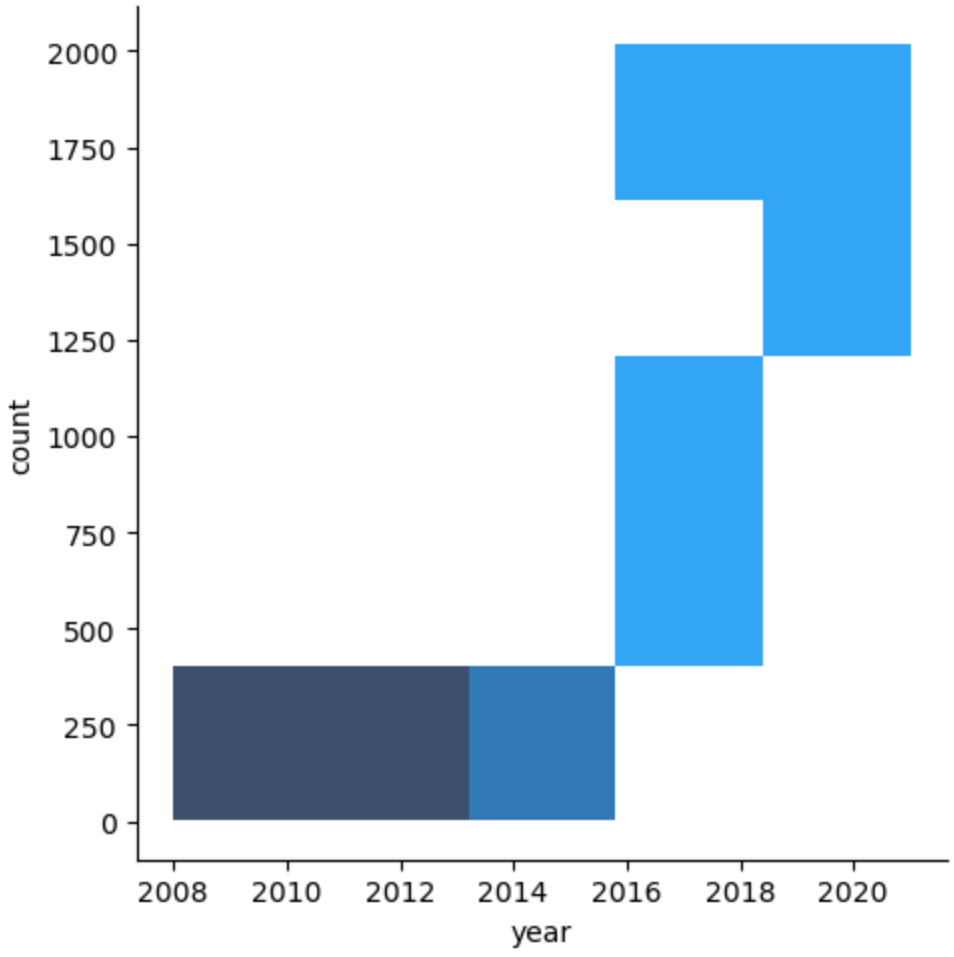
In [22]:
```
shows_year_df = netflix_shows_df["year_added"].value_counts().to_frame().reset_index()
shows_year_df = shows_year_df[shows_year_df["year"] != 2020]
shows_year_df
```

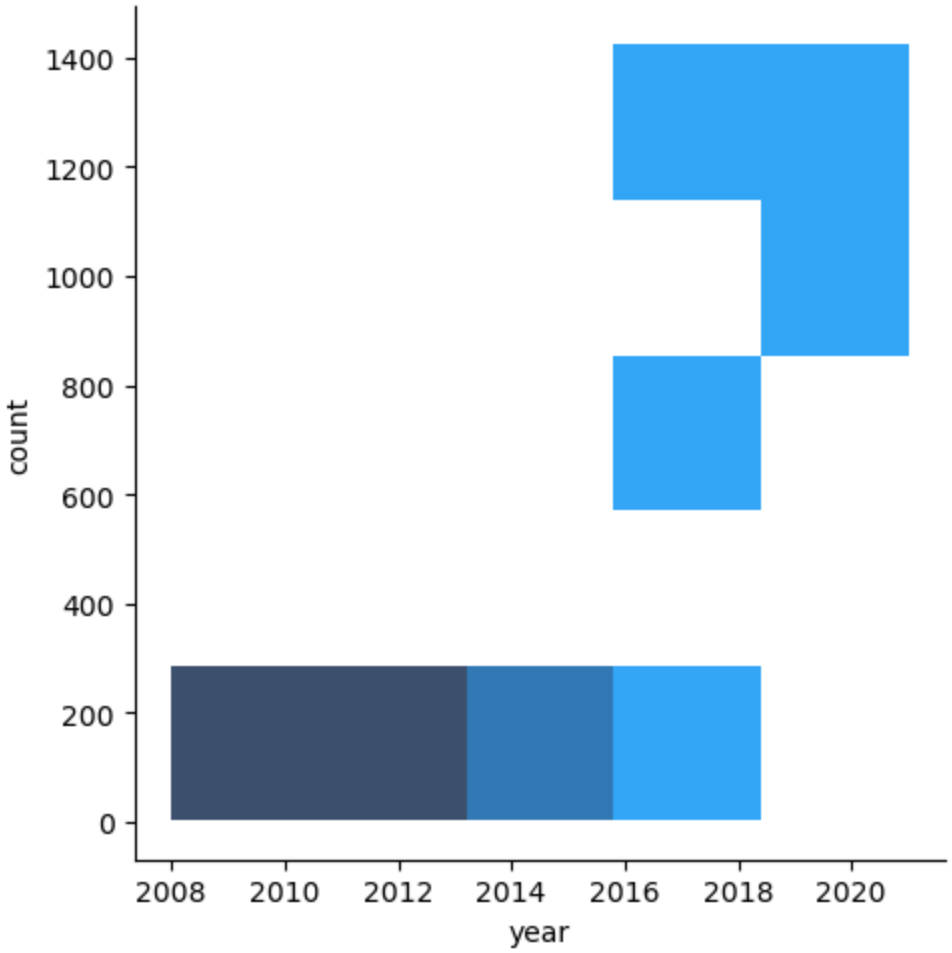
Out[22]:

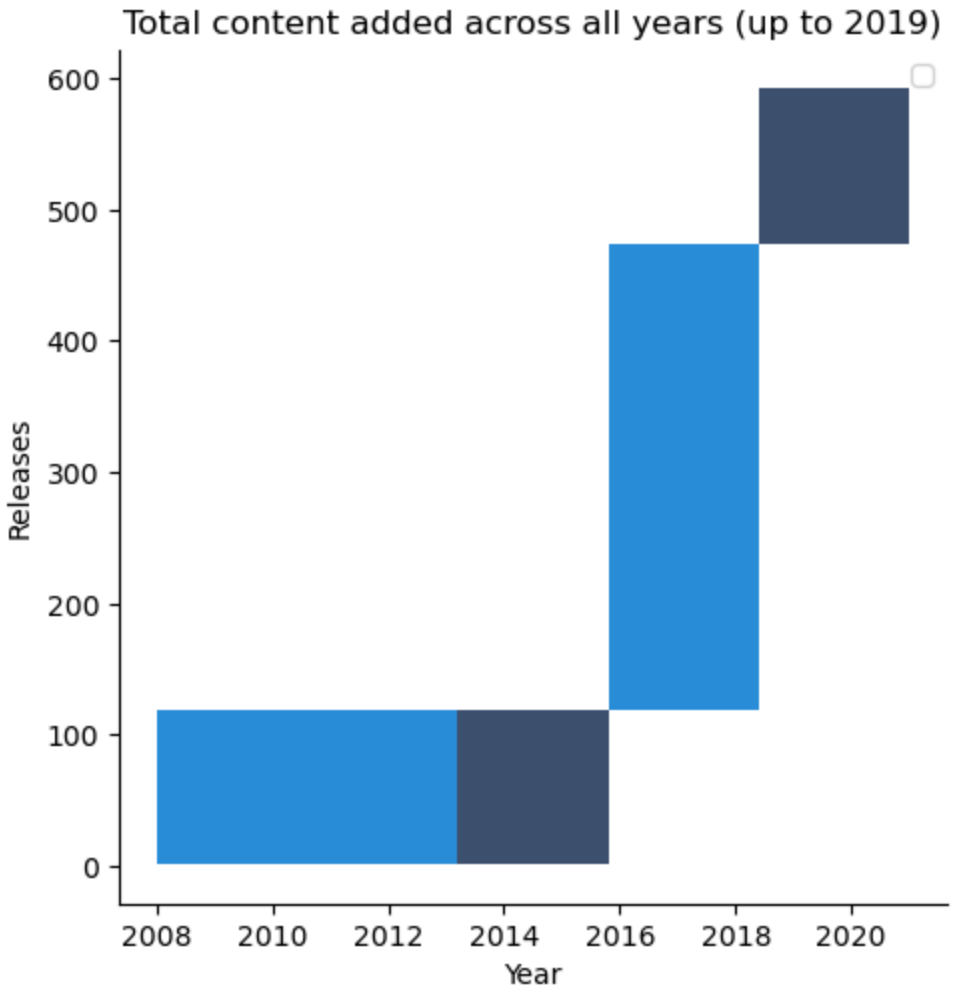
	year	count
1	2019	592
2	2021	505
3	2018	411
4	2017	349
5	2016	175
6	2015	26
7	2014	5
8	2013	5
9	2008	1

In [23]:
```
fig, ax = plt.subplots(figsize=(7, 5))
sns.displot(data=netflix_year_df, x='year', y='count')
sns.displot(data=movies_year_df, x='year', y='count')
sns.displot (data=shows_year_df, x='year', y='count')
ax.set_xticks(np.arange(2008, 2020, 1))
plt.title("Total content added across all years (up to 2019)")
plt.legend(['Total','Movie','TV Show'])
plt.ylabel("Releases")
plt.xlabel("Year")
plt.show()
```


1.0
0.8
0.6
0.4
0.2
0.0


2000
1750
1500
1250
1000
750
500
250
0
count
2008
2010
2012
2014
2016
2018
2020
year


1400
1200
1000
800
600
400
200
0
count
2008
2010
2012
2014
2016
2018
2020
year



Considering the timeline presented above, it is evident that the popular streaming platform began gaining significant traction after 2013. Since then, there has been a substantial increase in the amount of content added. The growth rate for movies on Netflix is notably higher compared to TV shows. For instance, approximately 1,300 new movies were added in both 2018 and 2019. Additionally, it is apparent that Netflix has progressively shifted its focus towards movies rather than TV shows in recent years.

3. Exploring the contribution of countries with the most content on Netflix.

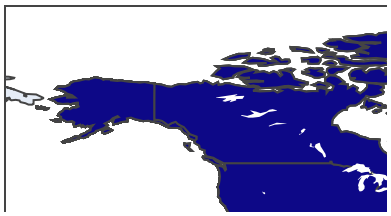
Next, we will analyze the countries based on the volume of content produced for Netflix. To achieve this, we need to extract and separate all countries associated with a film before conducting the analysis. Subsequently, we will remove titles with no specified countries.

Importing Libraries

In [24]:
```
import plotly.graph_objects as go
from plotly.offline import init_notebook_mode, iplot
```

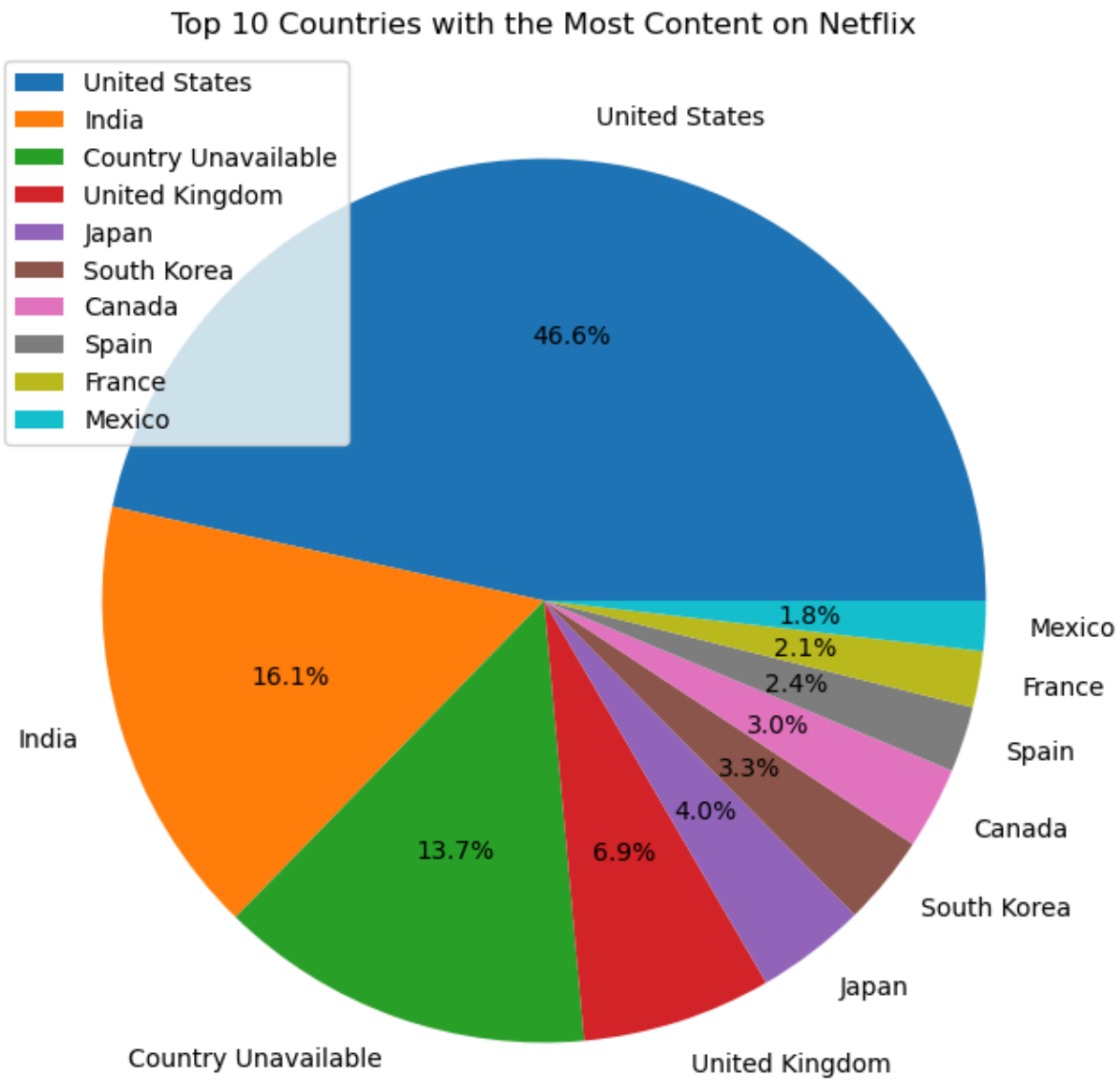
```
import plotly.express as px
```

In [25]:
```
filtered_countries = df.set_index('title').country.str.split(', ',
expand=True).stack().reset_index(level=1, drop=True);
filtered_countries = filtered_countries[filtered_countries != 'Country Unavailable']
iplot([go.Choropleth(
locationmode='country names',
locations=filtered_countries,
z=filtered_countries.value_counts()
)])
```



We should extract individual countries associated with each film before conducting the analysis and subsequently remove titles with no specified countries.

In [26]:
```
plt.figure(figsize=(8, 8))
top_countries = df['country'].value_counts().head(10)
plt.pie(top_countries, labels=top_countries.index, autopct='%1.1f%%')
plt.title('Top 10 Countries with the Most Content on Netflix')
plt.legend(loc='upper left')
plt.show()
```



4. Top Directors and Actors on Netflix

To identify the most popular directors, we can visualize the data using word clouds.from wordcloud import WordCloud, ImageColorGenerator

In [22]:
```
pip install wordcloud
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: wordcloud in c:\users\aa\appdata\roaming\python\python
311\site-packages (1.9.3)
Requirement already satisfied: numpy>=1.6.1 in c:\programdata\anaconda3\lib\site-pack
ages (from wordcloud) (1.24.3)
Requirement already satisfied: pillow in c:\programdata\anaconda3\lib\site-packages
(from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packag
es (from wordcloud) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\lib\site-
packages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-pack
ages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib\site
-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site
-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\site-p
ackages (from matplotlib->wordcloud) (23.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\anaconda3\lib\site-
packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anaconda3\lib\s
ite-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages
(from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [24]:

```
from wordcloud import WordCloud
text = " ".join(str(each) for each in df.director)
# Create and generate a word cloud image:
wordcloud = WordCloud(max_words=200, background_color="gray").generate(text)
plt.figure(figsize=(10,6))
plt.figure(figsize=(15,10))
# Display the generated image:
plt.imshow(wordcloud, interpolation='Bilinear')
plt.title('Most Popular Directors',fontsize = 30)
plt.axis("off")
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```

Most Popular Directors

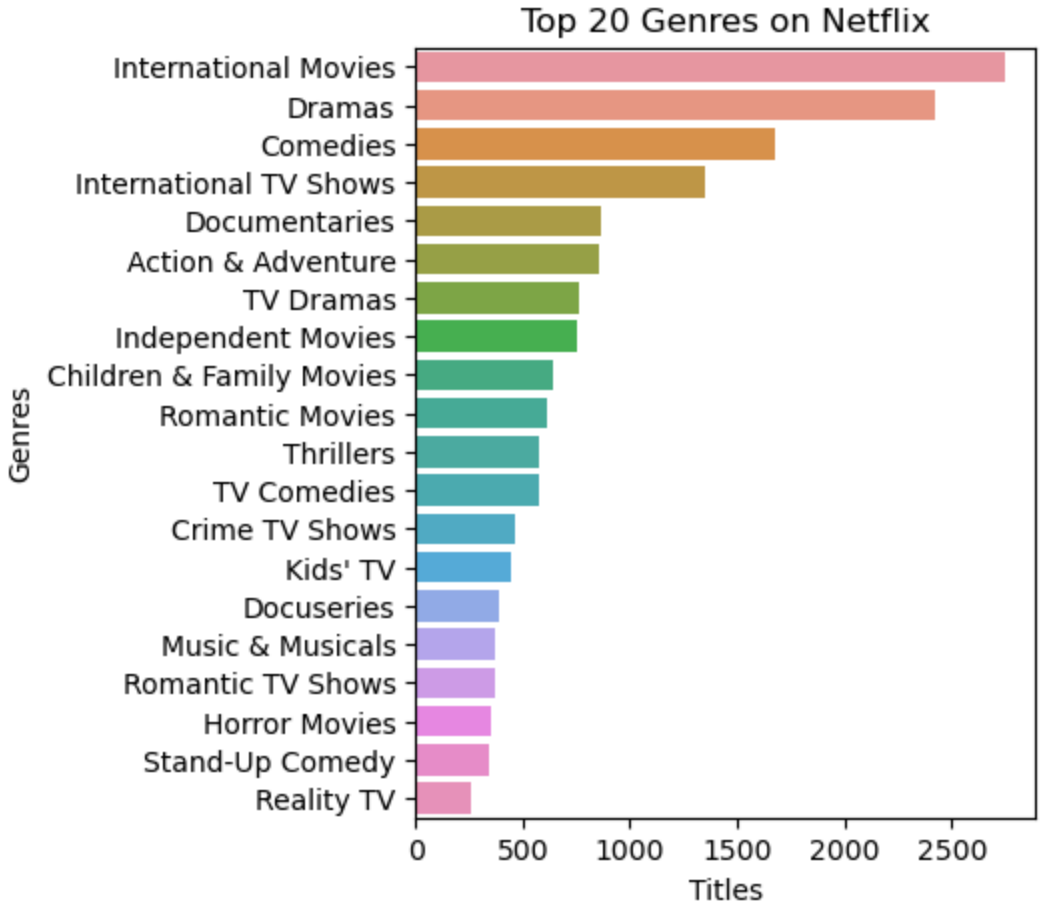
In [28]:
```
actor_text = " ".join(str(each) for each in df.cast)
actor_wordcloud = WordCloud(max_words=200, background_color="gray").generate(actor_tex
plt.figure(figsize=(15, 10))
plt.subplot(1, 2, 2)
plt.imshow(actor_wordcloud, interpolation='bilinear')
plt.title('Most Popular Actors', fontsize=20)
plt.axis("off")
plt.show()
```

Most Popular Actors

5. Top 20 Genres on Netflix: Count Plot

In [30]:
```
filtered_genres = df.set_index('title').listed_in.str.split(', ',
expand=True).stack().reset_index(level=1, drop=True);
plt.figure(figsize=(4,5))
g = sns.countplot(y = filtered_genres,
order=filtered_genres.value_counts().index[:20])
```

```
plt.title('Top 20 Genres on Netflix')
plt.xlabel('Titles')
plt.ylabel('Genres')
plt.show()
```



Based on the graph, we can observe that International Movies are the most prevalent, followed by dramas and comedies.

Bivariate Analysis:

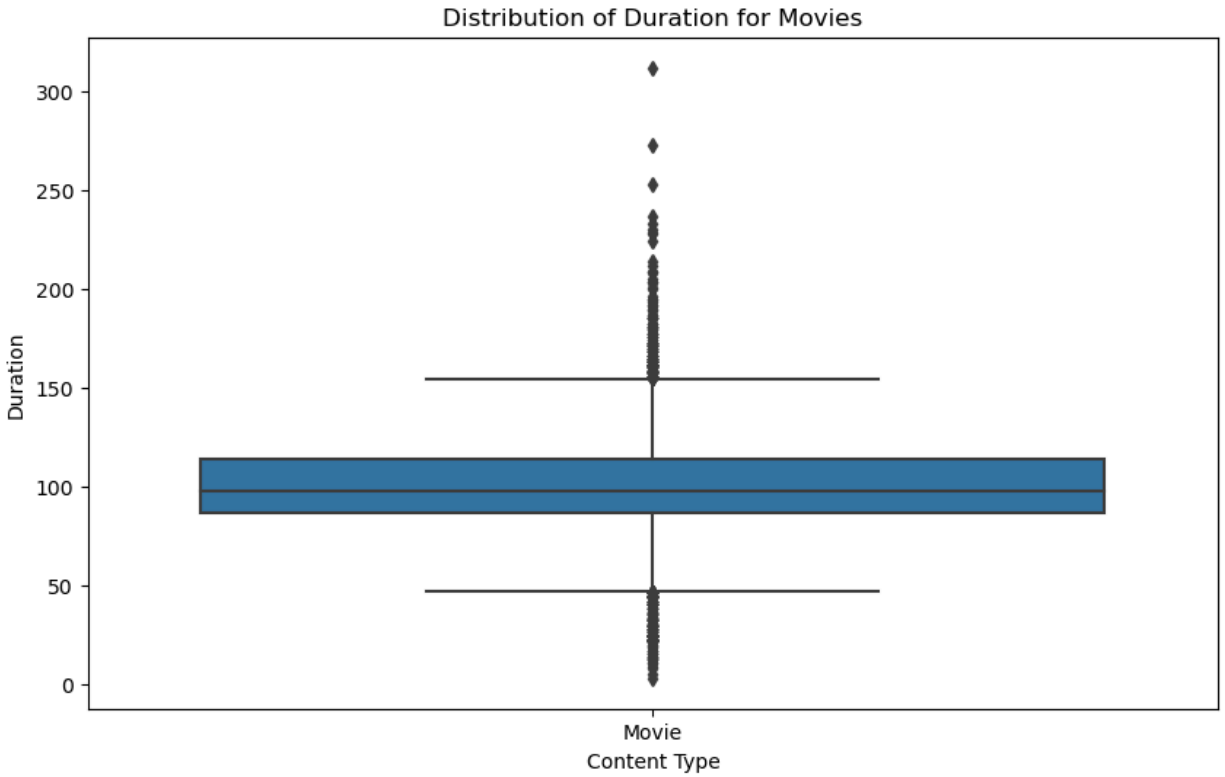
The prefix "bi" indicates two, and "variate" refers to variables, signifying an examination involving two variables. This analysis is concerned with understanding the causation and relationship between the two variables. There are three types of bivariate analysis. A Bivariate Analysis of two Numerical Variables (Numerical-Numerical)

4.2 For categorical variable(s): Boxplot

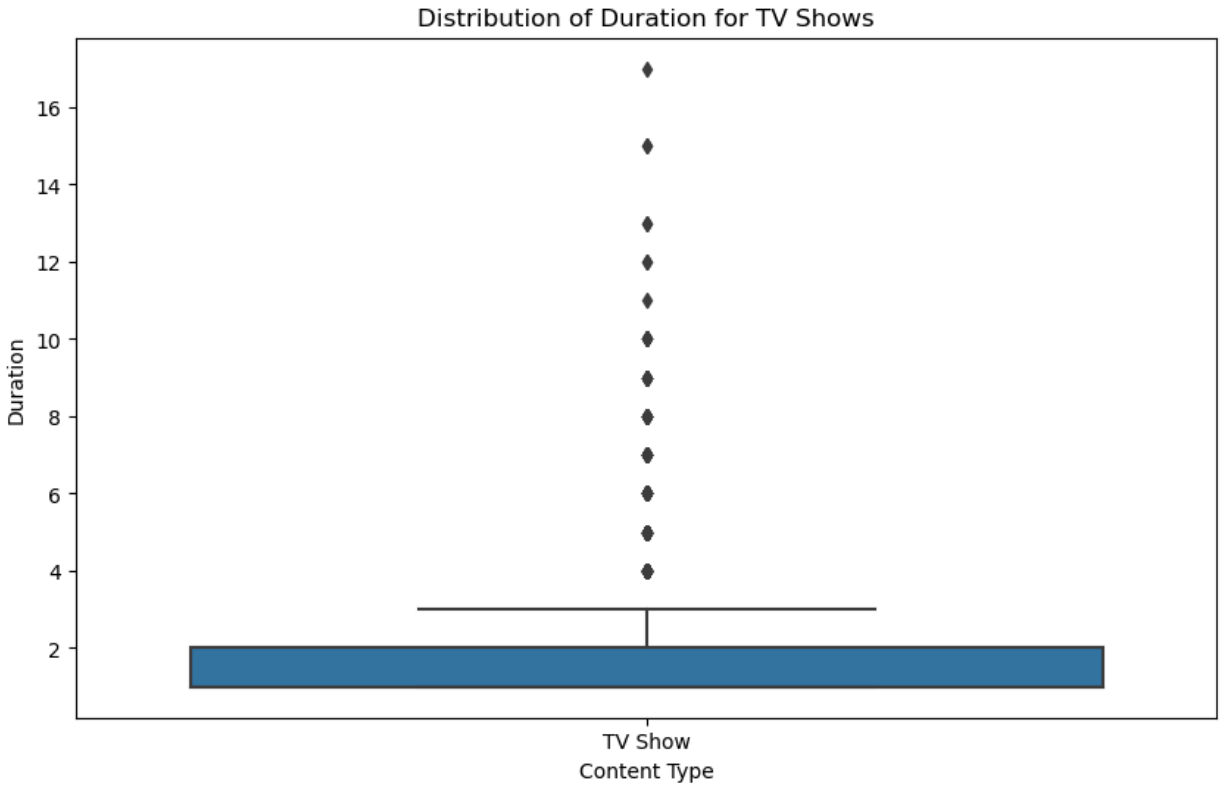
Duration Distribution for Movies and TV Shows

Analyzing the distribution of durations for both movies and TV shows provides insights into the typical length of content on Netflix. Utilizing box plots, we can visually represent these distributions, aiding in the identification of outliers and standard duration patterns.

In [31]:
```
netflix_movies_df = df[df['type'].str.contains("Movie")].copy()
netflix_movies_df['duration'] = netflix_movies_df['duration'].str.extract('(\d+)', exp
plt.figure(figsize=(10, 6))
sns.boxplot(data=netflix_movies_df, x='type', y='duration')
plt.xlabel('Content Type')
plt.ylabel('Duration')
plt.title('Distribution of Duration for Movies')
plt.show()
```



In [32]:
```
netflix_shows_df = df[df['type'].str.contains("TV Show")].copy()
netflix_shows_df['duration'] = netflix_shows_df['duration'].str.extract('(\d+)', expar
plt.figure(figsize=(10, 6))
sns.boxplot(data=netflix_shows_df, x='type', y='duration')
plt.xlabel('Content Type')
plt.ylabel('Duration')
plt.title('Distribution of Duration for TV Shows')
plt.show()
```



Upon analyzing the box plot for movies, it is evident that the majority of movies maintain a reasonable duration range, with few outliers extending to approximately 2.5 hours. This implies that Netflix predominantly offers movies designed to adhere to standard viewing times. Regarding TV shows, the box plot indicates that most shows span one to four seasons, with minimal outliers featuring longer durations. This observation aligns with previous trends, emphasizing Netflix's emphasis on shorter series formats.

4.3 For correlation: Heatmaps, Pairplots

Genre Correlation Heatmap:

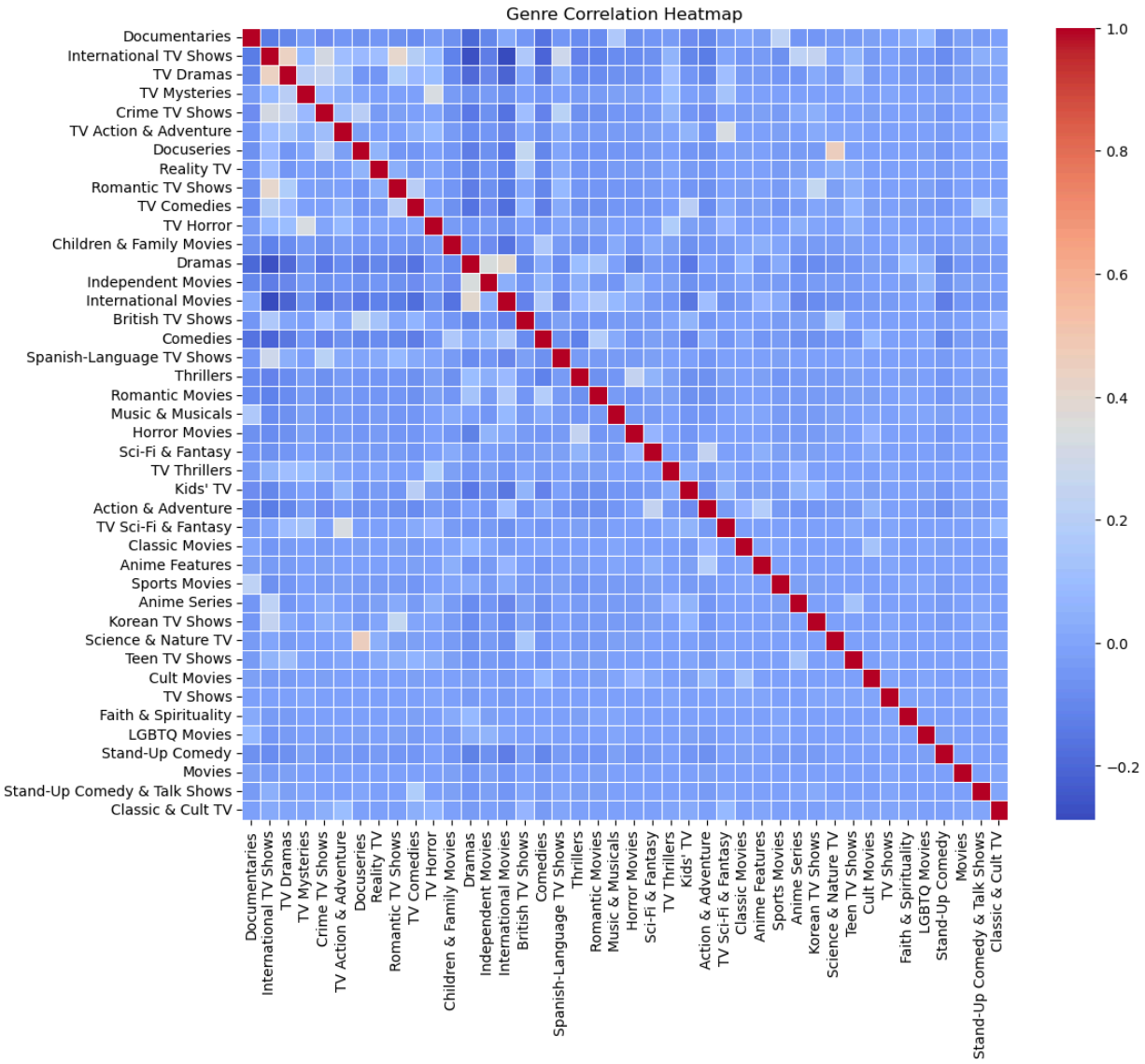
Genres play a crucial role in classifying and structuring content on Netflix. Examining the correlation between genres can unveil intriguing relationships among various content types. To explore genre correlation, we generate a genre data DataFrame initialized with zeros. Iterating over each row in the original DataFrame, we update the genre data DataFrame according to the listed genres. Subsequently, we create a correlation matrix using this genre data and present it as a heatmap.

In [35]:
```
# Creating a genre data DataFrame
genre_data = pd.DataFrame(0, index=df.index, columns=filtered_genres.unique())

# Updating genre data based on listed genres
for idx, genres in df[['title', 'listed_in']].iterrows():
    genre_data.loc[idx, genres['listed_in'].split(', ')] = 1

# Calculating genre correlation matrix
genre_corr_matrix = genre_data.corr()
```

```
# Creating a heatmap to visualize genre correlation
plt.figure(figsize=(12, 10))
sns.heatmap(genre_corr_matrix, cmap="coolwarm", annot=False, linewidths=.5)
plt.title('Genre Correlation Heatmap')
plt.show()
```



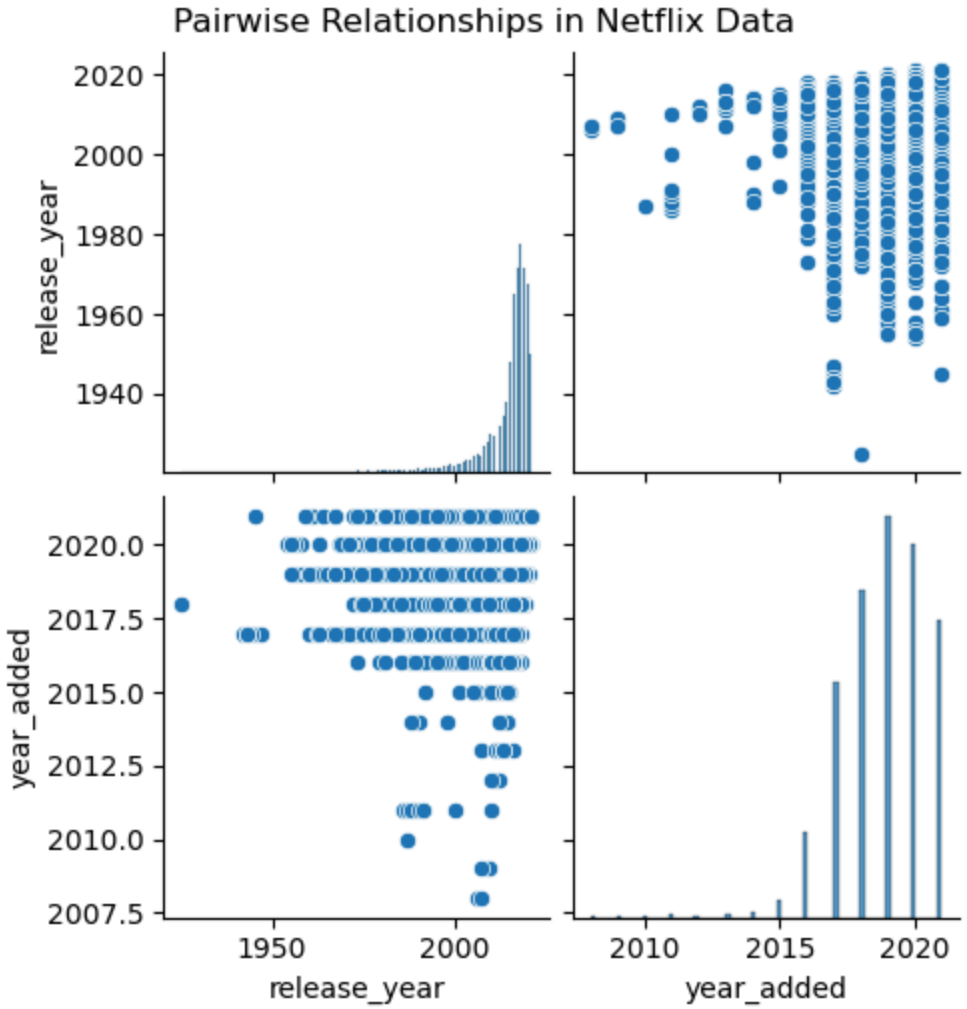
The heatmap illustrates the correlation between various genres. Upon analyzing the heatmap, we can pinpoint strong positive correlations between specific genres, such as TV Dramas and International TV Shows, Romantic TV Shows, and International TV Shows.

Pairplots

A pairplot visualizes pairwise relationships in a dataset. The function creates a grid of axes, with each variable in the data shared on the y-axis across a single row and on the x-axis across a single column.

In [37]:
```
nf_numerical_df = df.select_dtypes(include=['int64', 'float64'])
```

```
# Creating pairplots
sns.pairplot(nf_numerical_df)
plt.suptitle("Pairwise Relationships in Netflix Data", y=1.02)
plt.show()
```



5. Missing Value & Outlier check (Treatment optional)

Definition of Outlier: An outlier in a random sample from a population refers to an observation that significantly deviates from the standard dataset. In simpler terms, it represents a data point considerably distant from the typical values in a dataset. Outliers are identified as data values that fall outside the general distribution.

Identification of Outliers: Outliers, or out-of-line data, can be recognized through various methods. For instance, considering the dataset [10,15,22,330,30,45,60], the value 330 is notably distant from the rest, making it an outlier. Detecting outliers becomes more challenging in larger datasets, necessitating diverse techniques for accurate identification.

Significance of Outlier Treatment: Outliers can adversely impact the accuracy of predictions in machine learning models. Models such as linear regression, logistic regression, and support

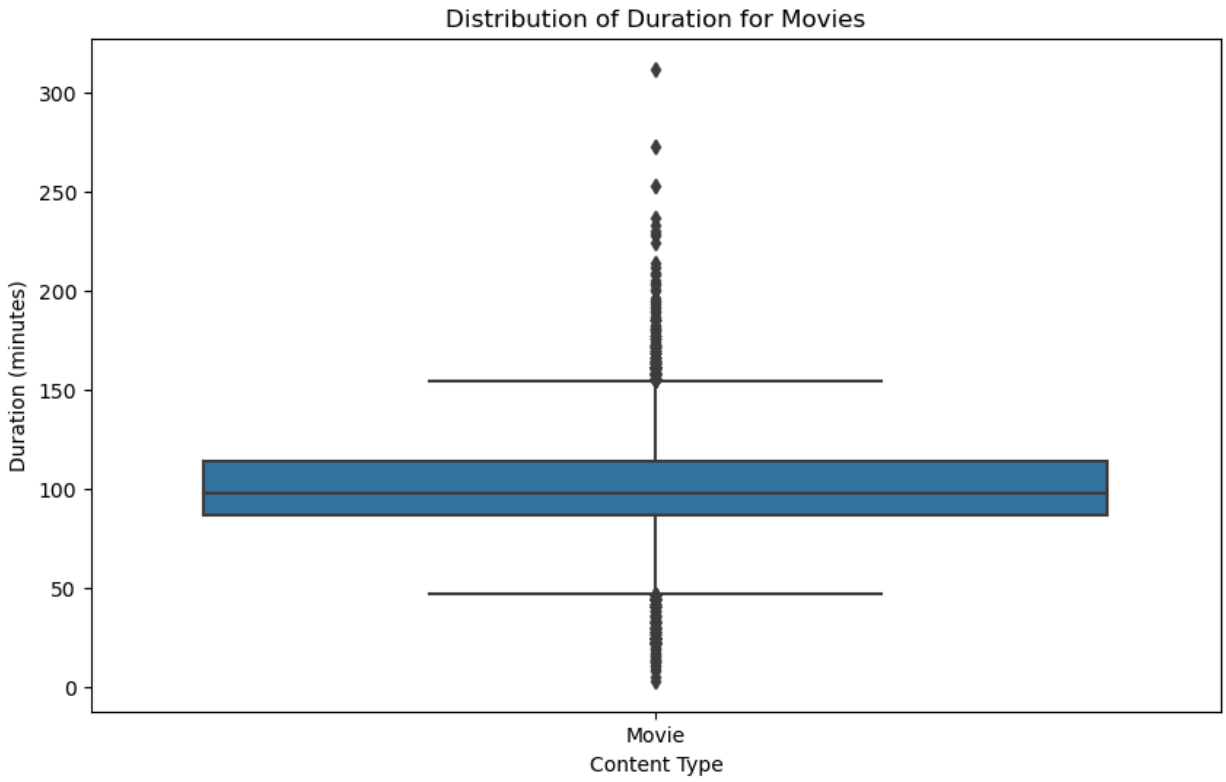
vector machines are particularly sensitive to outliers. The presence of outliers diminishes the robustness of these models, rendering their outputs unreliable. However, the treatment of outliers is contingent on the characteristics of the dataset, as some outliers may signify substantial changes in the data.

Visual Detection with Box Plots: Box plots offer a straightforward method for visualizing data distribution through quantiles and effectively detecting outliers. The Interquartile Range (IQR) forms the foundational statistical concept behind boxplots. The top and bottom whiskers of the boxplot serve as data boundaries, and any data points lying beyond these boundaries are considered outliers.

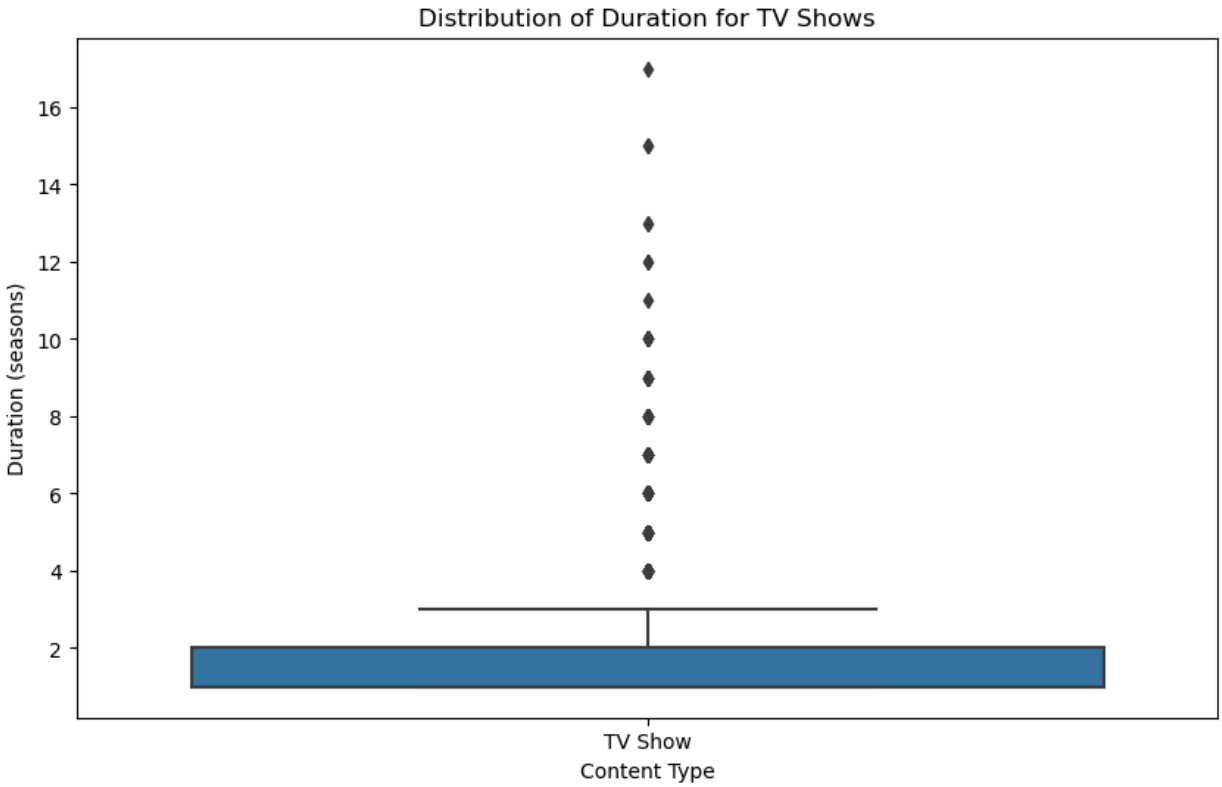
Boxplot for Categorical Variables: Duration Distribution for Movies and TV Shows

Analyzing the duration distribution for movies and TV shows provides insights into the typical length of content available on Netflix. To visually represent these distributions and identify outliers or standard durations, we can utilize box plots.

In [38]:
```
netflix_movies_df = df[df['type'].str.contains("Movie")].copy()
netflix_movies_df['duration'] = netflix_movies_df['duration'].str.extract('(\d+)', exp
plt.figure(figsize=(10, 6))
sns.boxplot(data=netflix_movies_df, x='type', y='duration')
plt.xlabel('Content Type')
plt.ylabel('Duration (minutes)')  # Adjusted ylabel for clarity
plt.title('Distribution of Duration for Movies')
plt.show()
```



In [39]:
```
netflix_shows_df = df[df['type'].str.contains("TV Show")].copy()
netflix_shows_df['duration'] = netflix_shows_df['duration'].str.extract('(\d+)', expar
plt.figure(figsize=(10, 6))  # Adjusted figsize for better visualization
sns.boxplot(data=netflix_shows_df, x='type', y='duration')
plt.xlabel('Content Type')
plt.ylabel('Duration (seasons)')  # Adjusted ylabel for clarity
plt.title('Distribution of Duration for TV Shows')
plt.show()
```



Analysis of Duration Distribution:

Analyzing the box plot for movies reveals that the majority of movies fall within a reasonable duration range, with only a few outliers extending to approximately 2.5 hours. This suggests that most movies on Netflix are designed to fit within a standard viewing time.

For TV shows, the box plot illustrates that the majority of shows have one to four seasons, and very few outliers have longer durations. This observation aligns with earlier trends, indicating that Netflix tends to focus on shorter series formats.

Understanding Missing Values:

In datasets, the presence of empty cells, rows, or columns is termed as missing values. These missing values can lead to dataset inconsistency and hinder analytical processes. Many machine learning algorithms encounter errors when working with datasets containing null values. Detecting and addressing missing values is crucial for effective data analysis.

Detecting Missing Values:

There are various methods to detect missing values in Python, and one commonly used function is isnull(). The expression dataframe.isnull().values.any() helps determine whether there are any null values present in the dataframe.

In [40]:
```
print('\nColumns with missing value:')
print(df.isnull().any())
```

```
Columns with missing value:
show_id         False
type            False
title           False
director        False
cast            False
country         False
date_added      False
release_year    False
rating          False
duration        False
listed_in       False
description     False
year_added      False
dtype: bool
```

6. Insights based on Non-Graphical and Visual Analysis:

6.1 Comments on the Range of Attributes:

The dataset consists of 8807 entries and 12 columns. The attributes encompass various aspects of Netflix content, including titles, descriptions, directors, cast, countries, date added, ratings, and duration. The range of attributes reflects the diverse information available for each title, providing a comprehensive view of Netflix's content catalog.

6.2 Comments on the Distribution of Variables and Relationship Between Them:

Content Type Distribution: The analysis revealed that Netflix has a higher proportion of movies (69.7%) compared to TV shows (30.3%) in its content library.

Content Addition Over Time: By exploring the amount of content added throughout the years, we observed a significant increase after 2013. The growth in the number of movies is notably higher than that of TV shows.

Country Contribution: Exploring the countries' contribution to Netflix content, we observed the distribution of produced content across different regions.

Director Popularity: Utilizing word clouds, we identified the most popular directors on Netflix based on the number of titles associated with each director.

Top Genres: The count plot of the top 20 genres on Netflix indicated that international movies, dramas, and comedies are among the most prevalent genres.

6.3 Comments for Each Univariate and Bivariate Plot:

Pie Plot for Content Type: The pie plot visually represented the distribution of Netflix titles, showcasing the dominance of movies over TV shows.

Distplot for Content Addition Over Time: The distribution plot provided insights into the growth of content added by Netflix over the years, highlighting peaks in 2018 and 2019.

Choropleth Map for Country Contribution: The choropleth map visualized the contribution of different countries to Netflix content, giving a geographical perspective on content production.

Word Cloud for Director Popularity: The word cloud illustrated the popularity of directors based on the frequency of their titles on Netflix.

Count Plot for Top Genres: The count plot effectively conveyed the prevalence of different genres, with international movies taking the lead.

Boxplots for Duration Distribution: The boxplots for movie and TV show durations provided insights into the typical length of content on Netflix. Most movies fell within a standard duration range, while TV shows predominantly had one to four seasons.

7. Business Insights

1. Content Dominance: Netflix’s content library is skewed towards movies, comprising a significant majority (69.7%) compared to TV shows (30.3%). This aligns with industry trends emphasizing the popularity of movies on streaming platforms.
2. Seasonal Content Strategy: A strategic content addition approach is evident, with July and December being peak months for new releases. This aligns with holidays and festive seasons, suggesting a deliberate effort to attract viewers during these periods.
3. Genre Correlations: Strong positive associations between genres, such as TV dramas and international shows, reveal viewer preferences for interconnected content. Understanding these correlations can guide content curation and recommendation algorithms.

4. Movie Length Trends: The historical peak in movie durations around the 1960s, followed by stabilization at around 100 minutes, reflects evolving viewer preferences. This insight can guide content creators in tailoring movie lengths to match audience expectations.

5. TV Show Episode Patterns: The preference for shorter TV series is evident, with most shows having only one season. This emphasizes the popularity of concise storytelling and suggests a potential focus on producing shorter series.

6. Common Themes: Recurring words like love, life, family, and adventure in titles and descriptions indicate prevalent themes in Netflix content. Recognizing and emphasizing these themes can enhance content resonance with viewers.

7. Rating Distribution Evolution: The distribution of ratings over the years provides insights into evolving audience preferences. Continuous monitoring of these changes is essential to ensure content quality aligns with viewers' expectations.

8. Data-Driven Decision Making: The data analysis journey underscores the importance of data-driven insights in navigating Netflix's content landscape. This approach empowers decision-makers to make informed choices regarding content creation and curation.

9. Continued Industry Relevance: As the streaming industry evolves, understanding content consumption patterns becomes crucial for Netflix's continued relevance. Staying adaptable to emerging trends is vital for sustained success.

10. Viewer Engagement: By strategically releasing content during peak months and aligning with common themes, Netflix can enhance viewer engagement. This involves leveraging data to guide the creation and promotion of content.

8. Recommendations

1. Diversify Content Offerings: To cater to diverse viewer preferences, Netflix should actively focus on expanding its TV show library to complement the dominance of movies.

2. Director Collaborations: Strengthen collaborations with top directors to enhance content quality. Additionally, explore partnerships with lesser-known directors who have high ratings, potentially uncovering hidden gems.

3. Explore Untapped Genres: While international movies are popular, there is an opportunity to explore genres like horror and comedy. This diversification can attract a broader audience and keep content fresh.

4. Strategic OTT Releases: Increase the number of direct-to-OTT releases, especially during holidays, year-end, and weekends. This can boost subscriptions by emphasizing the value of timely and exclusive content.

5. Thriller TV Shows: Given the preference for shorter series, focus on producing thriller genres for TV shows. This aligns with viewer expectations and can lead to increased viewership.

6. Continuous Content Addition: Increase the frequency of content releases throughout the year. This constant addition keeps the content library dynamic, attracting subscribers looking for a variety of options.

7. Positive Movie Promotion: Directly release movies with positive reviews into the OTT platform. This strategy leverages positive word-of-mouth, attracting subscribers looking for high-quality content.

8. Leverage Popular Actors: Collaborate with actors who have a significant following. Feature them in TV shows or web series to capitalize on their fan base, boosting viewership.

9. Country-Specific Advertising: Increase advertising efforts in countries with fewer released movies. Additionally, produce native TV shows tailored to the preferences of audiences in those regions.

10. Monitor Industry Trends: Establish a system for continuous monitoring of industry trends, viewer feedback, and emerging content preferences. This ongoing analysis ensures Netflix remains agile in responding to evolving viewer expectations.