Netflix - Data Exploration and Visualisation

Netflix is a leading global streaming platform that offers a wide range of movies, TV shows, and original content across various genres and languages. Founded in 1997, it has grown to serve over 230 million subscribers worldwide, with a strong focus on digital innovation, content creation, and global expansion.

In [1]:
```
!gdown 1E_DZC4yXvoAuEO8mI5BAXlzgdhyUq3R_
```

```
Downloading...
From: https://drive.google.com/uc?id=1E_DZC4yXvoAuEO8mI5BAXlzgdhyUq3R_
To: /content/Netfilx.csv

  0% 0.00/3.40M [00:00<?, ?B/s]
100% 3.40M/3.40M [00:00<00:00, 92.6MB/s]
```

Importing the libraries which are used for Data Visulisation, displaying graphs and charts,statistical plots

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv("Netfilx.csv")
```

In [3]:
```
data.head()
```

Out[3]:

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

Identification of variables and data types

- Numerical - Discrete and Continous
- Categorical - Ordinal and Nominal

Representing various columns with its variable:

- release_year - Numerical discrete variable
- Type ,Director,Cast,Country,listed_in - Nominal Categorical variable
- Ratings - Ordinal Categorical variable
- date_added - Date-time variable

In [4]:
```
data.dtypes
```

Out[4]:

	0
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]:
```
data.info()
```

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

In [6]:
```
data.shape
```

Out[6]: (8807, 12)

In [7]:
```
print("Number of rows : ",data.shape[0])
print("Number of columns : ",data.shape[1])
```

```
Number of rows :  8807
Number of columns :  12
```

1.Analysing the basic metrics and observation of Netflix data

In [8]:
```
# including all columns in a Dataframe description
data.describe(include='all')
```

Out[8]:

	show_id	type	title	director	cast	country	date_added	rel
count	8807	8807	8807	6173	7982	7976	8797	88
unique	8807	2	8807	4528	7692	748	1767	
top	s8807	Movie	Zubaan	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	
freq	1	6131	1	19	19	2818	109	
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	20
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	19
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	20
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	20
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	20
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	20

In [9]:
```
# including only numeric columns
data.describe(include=[np.number])
```

Out[9]:

	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 [10]:
```
# including only string columns
data.describe(include=object)
```

Out[10]:

	show_id	type	title	director	cast	country	date_added	rat
count	8807	8807	8807	6173	7982	7976	8797	8
unique	8807	2	8807	4528	7692	748	1767	
top	s8807	Movie	Zubaan	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	TV
freq	1	6131	1	19	19	2818	109	3

In [11]:
```
#Analysing null values
data.isna().sum()/len(data) * 100
```

Out[11]:

	0
show_id	0.000000
type	0.000000
title	0.000000
director	29.908028
cast	9.367549
country	9.435676
date_added	0.113546
release_year	0.000000
rating	0.045418
duration	0.034064
listed_in	0.000000
description	0.000000

dtype: float64

Conversion of date_added column to datetime(dtype) :

In [12]:
```
data['date_added'] = (data['date_added']).str.strip()
data['date_added'] = pd.to_datetime(data['date_added'])
```

1.1 Defining a clear problem statement ?

:*The objective of this analysis is to explore the Netflix dataset to generate meaningful insights, including content distribution, growth trends over the years, genre diversity, and regional representation. The analysis will also focus on identifying key data quality issues such as missing values and duplicate records, and applying appropriate data cleaning techniques. These steps aim to enhance the dataset's usability and support data-driven decisions that can improve user experience and inform Netflix's content strategy.*

- **Which type of content netflix offers more - Movies or TV shows** ? (Content Analysis)

In [13]:
```
data["type"].value_counts()
```

Out[13]:

type	count
Movie	6131
TV Show	2676

dtype: int64

- **Which region(Top 5) has most of the netflix content** ? (Region Analysis)

In [14]:
```
data["country"].value_counts().sort_values(ascending=False).head()
```

Out[14]:

country	count
United States	2818
India	972
United Kingdom	419
Japan	245
South Korea	199

dtype: int64

- **Which director directed more number of movies and TV shows** ? (Director Analysis)

In [15]:
```
data[data["type"]=="Movie"]["director"].value_counts().sort_values(ascending=F
```

Out[15]:

director	count
Rajiv Chilaka	19
Raúl Campos, Jan Suter	18
Suhas Kadav	16
Marcus Raboy	15
Jay Karas	14

dtype: int64

In [16]:
```
data[data["type"]=="TV Show"]["director"].value_counts().sort_values(ascending
```

Out[16]:

director	count
Alastair Fothergill	3
Rob Seidenglanz	2
Ken Burns	2
Stan Lathan	2
Shin Won-ho	2

dtype: int64

Which genres are most common in Netflix's library? (Genre Analysis)

In [17]:
```
data[data["type"]=='Movie']["listed_in"].value_counts().sort_values(ascending=
```

Out[17]:

listed_in	count
Dramas, International Movies	362
Documentaries	359
Stand-Up Comedy	334
Comedies, Dramas, International Movies	274
Dramas, Independent Movies, International Movies	252

dtype: int64

In [18]:
```
data[data["type"]=='TV Show']["listed_in"].value_counts().sort_values(ascendir
```

Out[18]:

listed_in	count
Kids' TV	220
International TV Shows, TV Dramas	121
Crime TV Shows, International TV Shows, TV Dramas	110
Kids' TV, TV Comedies	99
Reality TV	95

dtype: int64

DATA CLEANING

Major concerns

- Nested columns are present in dataframe
- We have observed the discrepancy in rating column which is filled with duration
- Handling the Nan values
- Cleaning the duplicated data

rating	count
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	80
G	41
TV-Y7-FV	6
NC-17	3
UR	3
74 min	1
84 min	1
66 min	1

dtype: int64

1. **Cleaning the rating column**

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	74 min	NaN	Movies	Louis C.K. muses on religion, eternal love, gi...
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	2010	84 min	NaN	Movies	Emmy-winning comedy writer Louis C.K. brings h...
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	NaN	Movies	The comic puts his trademark hilarious/thought...

In [19]:
```
data.loc[(data["rating"]=="74 min") | (data["rating"]=="84 min") | (data["rati
data.loc[[5541, 5794, 5813], "duration"] = ["74 min", "84 min", "66 min"]
```

Unnesting columns are present in dataframe

In [20]:
```
# Unnesting the director column
d = data["director"].apply(lambda x: str(x).split(", ")).to_list()
df_director = pd.DataFrame(d,index=data['title']).stack().reset_index().drop(c
df_director
```

Out[20]:

	title	Director
0	Dick Johnson Is Dead	Kirsten Johnson
1	Blood & Water	nan
2	Ganglands	Julien Leclercq
3	Jailbirds New Orleans	nan
4	Kota Factory	nan
...	...	...
9607	Zodiac	David Fincher
9608	Zombie Dumb	nan
9609	Zombieland	Ruben Fleischer
9610	Zoom	Peter Hewitt
9611	Zubaan	Mozez Singh

9612 rows × 2 columns

In [21]:
```
# Unnesting the cast column
c = data['cast'].apply(lambda x:str(x).split(", ")).tolist()
df_cast=pd.DataFrame(c,index=data['title']).stack().reset_index().drop(columns
df_cast
```

Out[21]:

	title	Cast
0	Dick Johnson Is Dead	nan
1	Blood & Water	Ama Qamata
2	Blood & Water	Khosi Ngema
3	Blood & Water	Gail Mabalane
4	Blood & Water	Thabang Molaba
...	...	...
64946	Zubaan	Manish Chaudhary
64947	Zubaan	Meghna Malik
64948	Zubaan	Malkeet Rauni
64949	Zubaan	Anita Shabdish
64950	Zubaan	Chittaranjan Tripathy

64951 rows × 2 columns

In [22]:
```
# Unnesting the listed_in(genre) column
l = data['listed_in'].str.split(', ').to_list()
df_listed_in=pd.DataFrame(l,index=data['title']).stack().reset_index().drop(cc
df_listed_in
```

Out[22]:

	title	Listed_in
0	Dick Johnson Is Dead	Documentaries
1	Blood & Water	International TV Shows
2	Blood & Water	TV Dramas
3	Blood & Water	TV Mysteries
4	Ganglands	Crime TV Shows
...	...	...
19318	Zoom	Children & Family Movies
19319	Zoom	Comedies
19320	Zubaan	Dramas
19321	Zubaan	International Movies
19322	Zubaan	Music & Musicals

19323 rows × 2 columns

In [23]:
```
# unnesting the country column
```

```
country = data["country"].apply(lambda x:str(x).split(", ")).tolist()
df_country=pd.DataFrame(country,index=data['title']).stack().reset_index().drc
df_country
```

Out[23]:

	title	Country
0	Dick Johnson Is Dead	United States
1	Blood & Water	South Africa
2	Ganglands	nan
3	Jailbirds New Orleans	nan
4	Kota Factory	India
...	...	...
10840	Zodiac	United States
10841	Zombie Dumb	nan
10842	Zombieland	United States
10843	Zoom	United States
10844	Zubaan	India

10845 rows × 2 columns

In [24]:

```
# Merging all the unnested columns into a Single Dataframe

df_merge1 = pd.merge(df_director,df_cast,on="title")
df_merge2 = pd.merge(df_listed_in,df_country,on="title")
df_merge3 = pd.merge(df_merge1,df_merge2,on="title")
df_merge3
```

Out[24]:

	title	Director	Cast	Listed_in	Country
0	Dick Johnson Is Dead	Kirsten Johnson	nan	Documentaries	United States
1	Blood & Water	nan	Ama Qamata	International TV Shows	South Africa
2	Blood & Water	nan	Ama Qamata	TV Dramas	South Africa
3	Blood & Water	nan	Ama Qamata	TV Mysteries	South Africa
4	Blood & Water	nan	Khosi Ngema	International TV Shows	South Africa
...	...	...	...	...	...
201986	Zubaan	Mozez Singh	Anita Shabdish	International Movies	India
201987	Zubaan	Mozez Singh	Anita Shabdish	Music & Musicals	India
201988	Zubaan	Mozez Singh	Chittaranjan Tripathy	Dramas	India
201989	Zubaan	Mozez Singh	Chittaranjan Tripathy	International Movies	India
201990	Zubaan	Mozez Singh	Chittaranjan Tripathy	Music & Musicals	India

201991 rows × 5 columns

In [25]:
```
# Finally merging with original dataframe and dropping the nested columns
data_clean = pd.merge(data, df_merge3, on="title", how="left")
data_clean.drop(columns=["director","cast","listed_in","country"],inplace=True
data_clean
```

Out[25]:

	show_id	type	title	date_added	release_year	rating	duration	de
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	90 min	fa
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	C
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	C
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	C
4	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	C
...	...	...	...	...	...	...	...	
201986	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	111 min	b hi
201987	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	111 min	b hi
201988	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	111 min	b hi
201989	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	111 min	b hi

	show_id	type	title	date_added	release_year	rating	duration	de
201990	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	111 min	b hi

201991 rows × 12 columns

In [26]:
```
data_clean.duplicated().sum()
```

Out[26]:
```
np.int64(55)
```

In [27]:
```
data_clean.drop_duplicates()
```

Out[27]:

	show_id	type	title	date_added	release_year	rating	duration	de
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	90 min	fa
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	C
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	C
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	C
4	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	2 Seasons	C
...	...	...	...	...	...	...	...	
201986	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	111 min	b hi
201987	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	111 min	b hi
201988	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	111 min	b hi
201989	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	111 min	b hi

	show_id	type	title	date_added	release_year	rating	duration	de
201990	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	111 min	b hi

201936 rows × 12 columns

NaN in Pandas (np.nan) is a special float value representing "Not a Number". When you use str(x) on np.nan, it converts the NaN to the string "nan"

we cant fill any values at nan, so i will replace it to np.Nan

In [28]:
```
data_clean.replace("nan",np.nan,inplace=True)
```

In [29]:
```
data_clean["Movie minutes"] = data_clean[data_clean["type"]=="Movie"]["duratic
data_clean["No_of_Seasons"] = data_clean[data_clean["type"]=="TV Show"]["durat
data_clean.drop(columns="duration",inplace=True)
data_clean.head()
```

Out[29]:

	show_id	type	title	date_added	release_year	rating	description	Direct
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	As her father nears the end of his life, filmm...	Kirst Johns
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...	N
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...	N
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...	N
4	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...	N

In [30]:
```
# fetching out number of nan values in each row
data_clean.isna().sum()
```

Out[30]:

	0
show_id	0
type	0
title	0
date_added	158
release_year	0
rating	70
description	0
Director	50643
Cast	2146
Listed_in	0
Country	11897
Movie minutes	56148
No_of_Seasons	145843

dtype: int64

Treatment of Missing values

Filling the mode at the place of "Nan" in **Director column**

In [31]:
```
d_mode = data_clean.groupby('Listed_in')['Director'].apply(lambda x: x.mode().
data_clean = pd.merge(data_clean,d_mode,on='Listed_in',how='left')
data_clean['Director'] = data_clean['Director'].fillna(data_clean['Director_mc
```

In [32]:
```
data_clean.drop(columns="Director_mode",inplace=True)
```

In [33]:
```
data_clean.drop_duplicates()
```

Out[33]:

	show_id	type	title	date_added	release_year	rating	description
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
4	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
...	...	...	...	...	...	...	...
201986	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
201987	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
201988	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
201989	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms

	show_id	type	title	date_added	release_year	rating	description
							his way into a ty...
201990	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...

201936 rows × 13 columns

In [34]:
```
data_clean.isna().sum()
```

Out[34]:

	0
show_id	0
type	0
title	0
date_added	158
release_year	0
rating	70
description	0
Director	0
Cast	2146
Listed_in	0
Country	11897
Movie minutes	56148
No_of_Seasons	145843

dtype: int64

Filling the mode at the place of "Nan" in **date_added column**

In [35]:
```
date_mode = data_clean.groupby('Listed_in')['date_added'].apply(lambda x: x.mo
data_clean = pd.merge(data_clean,date_mode,on='Listed_in',how='left')
data_clean['date_added'] = data_clean['date_added'].fillna(data_clean['date_ad
data_clean
```

Out[35]:

	show_id	type	title	date_added	release_year	rating	description
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
4	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
...	...	...	...	...	...	...	...
202831	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
202832	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
202833	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
202834	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms

	show_id	type	title	date_added	release_year	rating	description
							his way into a ty...
202835	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...

202836 rows × 14 columns

In [36]:
```
data_clean.drop(columns="date_added_mode",inplace=True)
```

In [37]:
```
data_clean.duplicated().sum()
```

Out[37]:
```
np.int64(899)
```

In [38]:
```
data_clean.drop_duplicates(inplace=True)
```

Filling the mode at the place of "Nan" in **Rating column**

In [39]:
```
rat_mode = data_clean.groupby('Listed_in')['rating'].apply(lambda x: x.mode().
data_clean = pd.merge(data_clean,rat_mode,on='Listed_in',how='left')
data_clean['rating'] = data_clean['rating'].fillna(data_clean['rating_mode'])
data_clean
```

Out[39]:

	show_id	type	title	date_added	release_year	rating	description
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
4	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
...	...	...	...	...	...	...	...
201932	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
201933	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
201934	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
201935	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms

	show_id	type	title	date_added	release_year	rating	description
							his way into a ty...
201936	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...

201937 rows × 14 columns

In [40]:
```
data_clean.drop(columns="rating_mode",inplace=True)
```

Practically for **Genre - Documentaries** --> No cast is need & Filling the mode at the place of "Nan" in **Cast column**

In [41]:
```
data_clean.loc[data_clean["Listed_in"] == "Documentaries", "Cast"] = "No Cast
```

In [42]:
```
cast_mode = data_clean.groupby('Listed_in')['Cast'].apply(lambda x: x.mode().i
data_clean = pd.merge(data_clean,cast_mode,on='Listed_in',how='left')
data_clean['Cast'] = data_clean['Cast'].fillna(data_clean['Cast_mode'])
data_clean
```

Out[42]:

	show_id	type	title	date_added	release_year	rating	description
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
4	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
...	...	...	...	...	...	...	...
201932	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
201933	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
201934	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
201935	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms

	show_id	type	title	date_added	release_year	rating	description
							his way into a ty...
201936	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...

201937 rows × 14 columns

In [43]:
```
data_clean.duplicated().sum()
```

Out[43]:
```
np.int64(1063)
```

In [44]:
```
data_clean.drop_duplicates(inplace=True)
```

In [45]:
```
data_clean.drop(columns="Cast_mode",inplace=True)
```

In [46]:
```
data_clean.reset_index(drop=True,inplace=True)
```

Filling the mode at the place of "Nan" in **Country column**

In [47]:
```
country_mode = data_clean.groupby('Listed_in')['Country'].apply(lambda x: x.mc
data_clean = pd.merge(data_clean,country_mode,on='Listed_in',how='left')
data_clean['Country'] = data_clean['Country'].fillna(data_clean['Country_mode'
data_clean
```

Out[47]:

	show_id	type	title	date_added	release_year	rating	description
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
4	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
...	...	...	...	...	...	...	...
200869	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
200870	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
200871	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
200872	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms

	show_id	type	title	date_added	release_year	rating	description
							his way into a ty...
200873	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...

200874 rows × 14 columns

In [48]:
```
data_clean.drop(columns="Country_mode",inplace=True)
```

In [49]:
```
data_clean["Movie minutes"] = data_clean["Movie minutes"].fillna("Not a Movie"
data_clean["No_of_Seasons"] = data_clean["No_of_Seasons"].fillna("Not a TV Sho
```

In [50]:
```
#Final Clean Dataset
data_clean.head()
```

Out[50]:

	show_id	type	title	date_added	release_year	rating	description	Dir
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	As her father nears the end of his life, filmm...	K Jol
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...	
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...	Alar
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...	Seider
4	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...	

In [51]:
```
# # The resultant had no null values(non -null values)
data_clean.isna().sum()
```

Out[51]:

	0
show_id	0
type	0
title	0
date_added	0
release_year	0
rating	0
description	0
Director	0
Cast	0
Listed_in	0
Country	0
Movie minutes	0
No_of_Seasons	0

dtype: int64

In [52]:
```
# number of duplicates present in dataframe
data_clean.duplicated().sum()
```

Out[52]:
```
np.int64(0)
```

In [53]:
```
#reset the index
data_clean = data_clean.reset_index(drop=True)
data_clean
```

Out[53]:

	show_id	type	title	date_added	release_year	rating	description
0	s1	Movie	Dick Johnson Is Dead	2021-09-25	2020	PG-13	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
2	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
3	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
4	s2	TV Show	Blood & Water	2021-09-24	2021	TV-MA	After crossing paths at a party, a Cape Town t...
...	...	...	...	...	...	...	...
200869	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
200870	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
200871	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...
200872	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms

	show_id	type	title	date_added	release_year	rating	description
							his way into a ty...
200873	s8807	Movie	Zubaan	2019-03-02	2015	TV-14	A scrappy but poor boy worms his way into a ty...

200874 rows × 13 columns

In [54]:
```
data_clean.duplicated().sum()
```

Out[54]:
```
np.int64(0)
```

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

In [55]:
```
count1 = data_clean['title'].value_counts()
print(count1)

print("**********************************************************")

count2 = data_clean['Director'].value_counts()
print(count2)

print("**********************************************************")

count3 = data_clean['Country'].value_counts()
print(count3)
```

```
title
Kahlil Gibran's The Prophet                                      700
Holidays                                                         504
Movie 43                                                         468
The Eddy                                                         416
Narcos                                                           378
                                                                 ...
Whindersson Nunes: Adult                                           1
Maynard                                                            1
WWII: Report from the Aleutians                                    1
World's Weirdest Homes                                             1
Tiffany Haddish: She Ready! From the Hood To Hollywood!            1
Name: count, Length: 8807, dtype: int64
**********************************************************
Director
Noam Murro          12860
Alan Poul            7831
Thomas Astruc        6558
Alejandro Lozano     6088
Guy Vasilovich       4592
                    ...
Matthew Cooke           1
Ellen Brown             1
Rob Zombie              1
Mario Briongos          1
Maïa Sandoz             1
Name: count, Length: 4993, dtype: int64
**********************************************************
Country
United States     66062
India             24454
United Kingdom    12875
Japan             10652
France             8193
                  ...
Botswana              2
United States,        1
Nicaragua             1
Kazakhstan            1
Uganda                1
Name: count, Length: 127, dtype: int64
```

Observation :

◈ Content Title Frequency

The most common title in the dataset is “Kahlil Gibran's The Prophet” (700 occurrences).

Other frequently repeated titles:

Holidays – 504

Movie 43 – 468

The Eddy – 416

Narcos – 378

Insight: Certain shows/movies are listed multiple times, possibly due to regional availability, multiple seasons, or duplicate entries.

◈ Director Frequency

Top Directors:

Noam Murro – 12,860 entries (extremely high count → possibly data duplication or heavy involvement in a long-running series).

Alan Poul – 7,831

Thomas Astruc – 6,558

Alejandro Lozano – 6,088

Guy Vasilovich – 4,592

Insight: A small number of directors dominate the dataset. This may indicate highly syndicated content or misclassified duplication.

◈ Country Distribution

United States leads with 66,062 titles.

India is next with 24,454 titles → very strong representation.

Other top contributors:

UK – 12,875

Japan – 10,652

France – 8,193

Low representation: countries like Botswana, Nicaragua, Kazakhstan, Uganda with just 1–2 titles.

Insight: The dataset is heavily skewed toward U.S. content, followed by India. This shows dominance of Hollywood and Bollywood in global streaming platforms.

◈ Ratings

Available ratings: ['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R', 'TV-G', 'G', 'NC-17', 'NR', 'TV-Y7-FV', 'UR'].

Covers all age categories: from children (TV-Y, TV-Y7) to adult (R, NC-17, TV-MA).

Insight: Wide range of audience targeting. TV-MA, R, PG-13 likely dominate (though counts aren't shown).

◈ Content Type

Only 2 categories:

Movies

TV Shows

Insight: Dataset is clean in terms of type; no extra categories like "Documentary" or "Short".

◈ Release Years

Range spans from 1925 to 2021.

Covers nearly a century of film/TV content.

Classic films (1940s–1960s) are included, but bulk likely belongs to 2000 onwards.

Insight: Strong representation of modern content but also archival data from the early 20th century.

In [56]:
```
unique1 = data_clean["rating"].unique()
print(unique1)
print("************************************************************")
unique2 = data_clean["type"].unique()
print(unique2)
print("************************************************************")
unique3 = data_clean["release_year"].unique()
print(unique3)
```

```
['PG-13' 'TV-MA' 'PG' 'TV-14' 'TV-PG' 'TV-Y' 'TV-Y7' 'R' 'TV-G' 'G'
 'NC-17' 'NR' 'TV-Y7-FV' 'UR']
**********************************************************
['Movie' 'TV Show']
**********************************************************
[2020 2021 1993 2018 1996 1998 1997 2010 2013 2017 1975 1978 1983 1987
 2012 2001 2014 2002 2003 2004 2011 2008 2009 2007 2005 2006 1994 2015
 2019 2016 1982 1989 1990 1991 1999 1986 1992 1984 1980 1961 2000 1995
 1985 1976 1959 1988 1981 1972 1964 1945 1954 1979 1958 1956 1963 1970
 1973 1925 1974 1960 1966 1971 1962 1969 1977 1967 1968 1965 1946 1942
 1955 1944 1947 1943]
```

In [57]:
```
data_clean.dtypes
```

Out[57]:

	0
show_id	object
type	object
title	object
date_added	datetime64[ns]
release_year	int64
rating	object
description	object
Director	object
Cast	object
Listed_in	object
Country	object
Movie minutes	object
No_of_Seasons	object

dtype: object

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

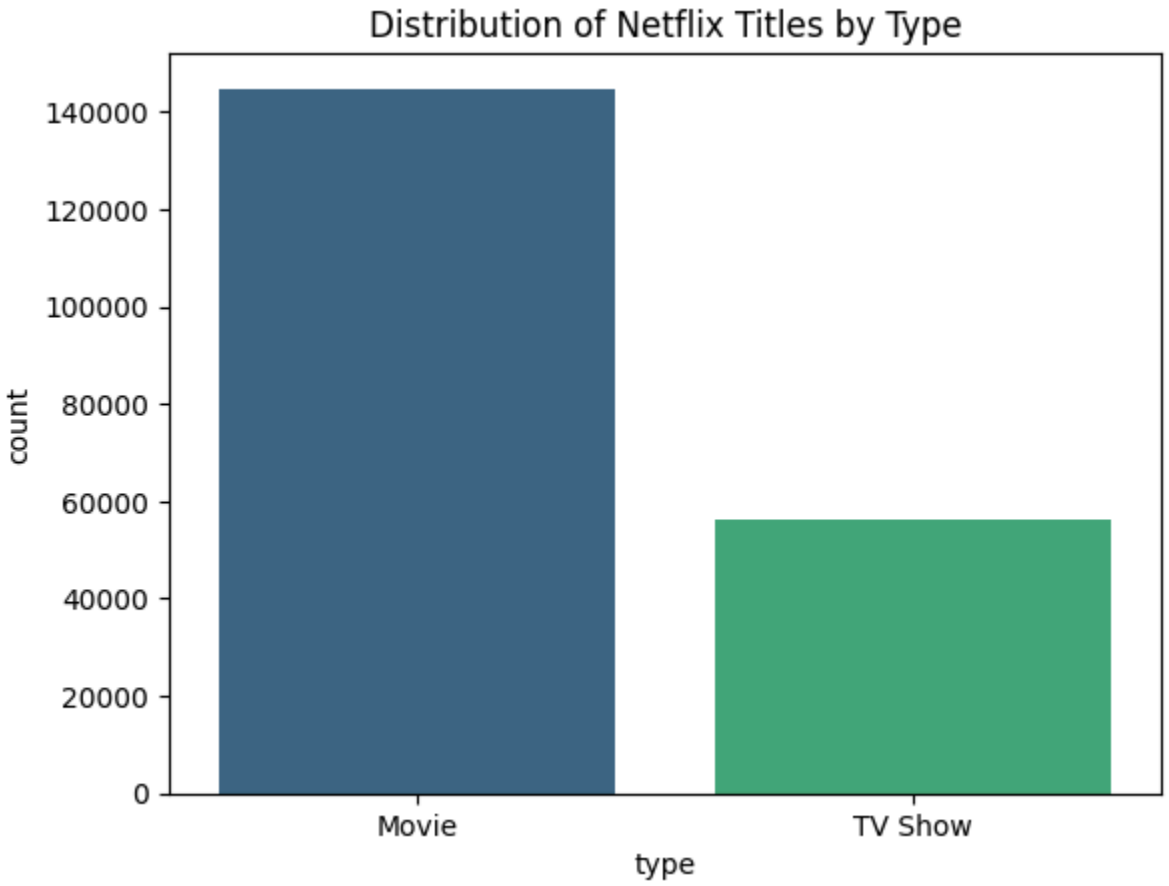
1. Find the counts of each categorical variable both using graphical analysis.

In [58]:
```
category = data_clean.copy()
category = category.select_dtypes(include=object)
```

In [59]:
```
#Countplot for the Type of Show
```

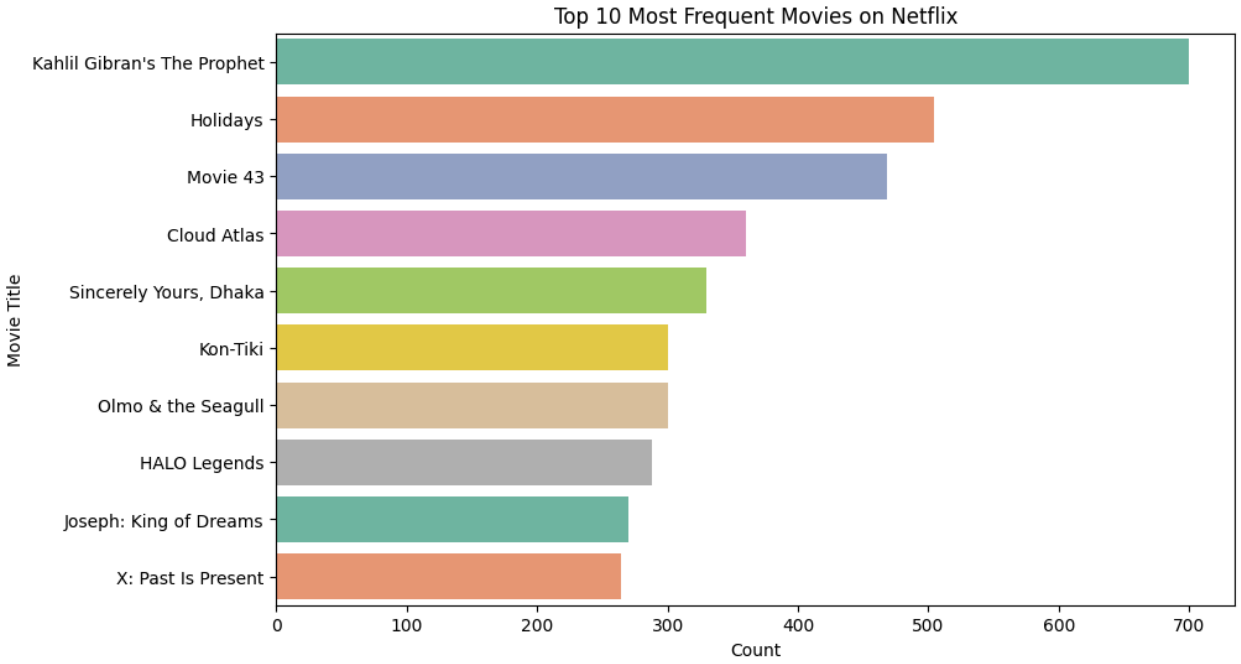
```
sns.countplot(data=category, x="type", palette="viridis")
plt.title("Distribution of Netflix Titles by Type")
plt.show()
```

```
/tmp/ipython-input-312364722.py:2: FutureWarning:

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

  sns.countplot(data=category, x="type", palette="viridis")
```



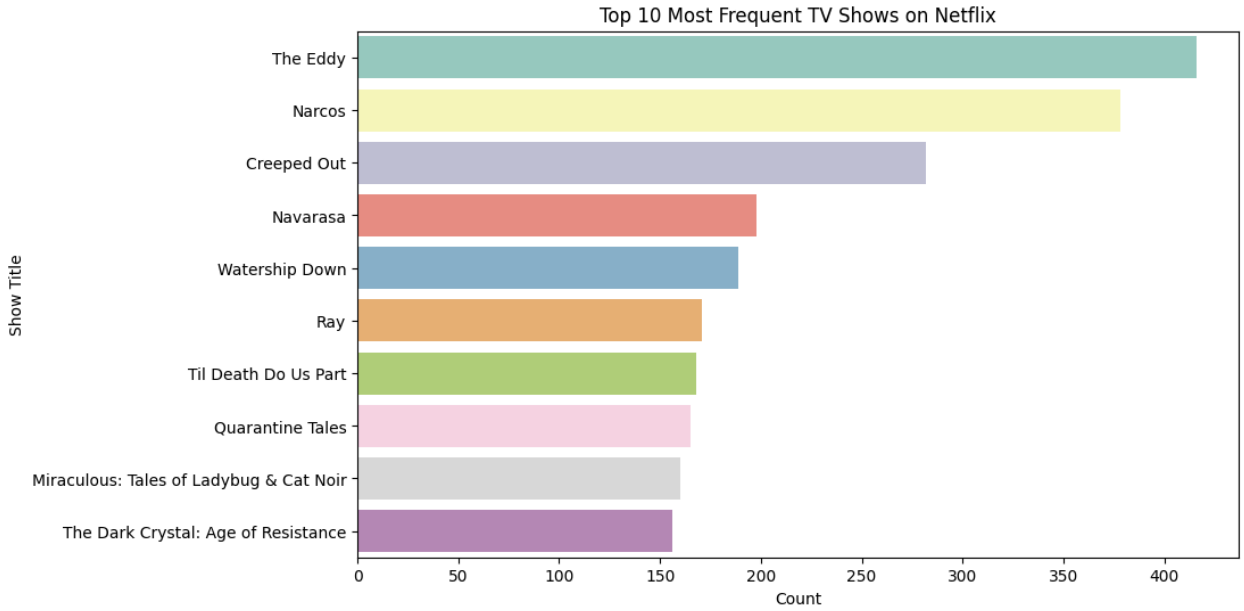
In [60]:
```
# Top 10 Movie title
movies = category[category["type"]=="Movie"]

top10_movies = movies["title"].value_counts().head(10).reset_index()

plt.figure(figsize=(10,6))
sns.barplot(data=top10_movies, y="title", x="count", palette="Set2")

plt.title("Top 10 Most Frequent Movies on Netflix")
plt.xlabel("Count")
plt.ylabel("Movie Title")
plt.show()
```

```
/tmp/ipython-input-2668008792.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.

  sns.barplot(data=top10_movies, y="title", x="count", palette="Set2")
```



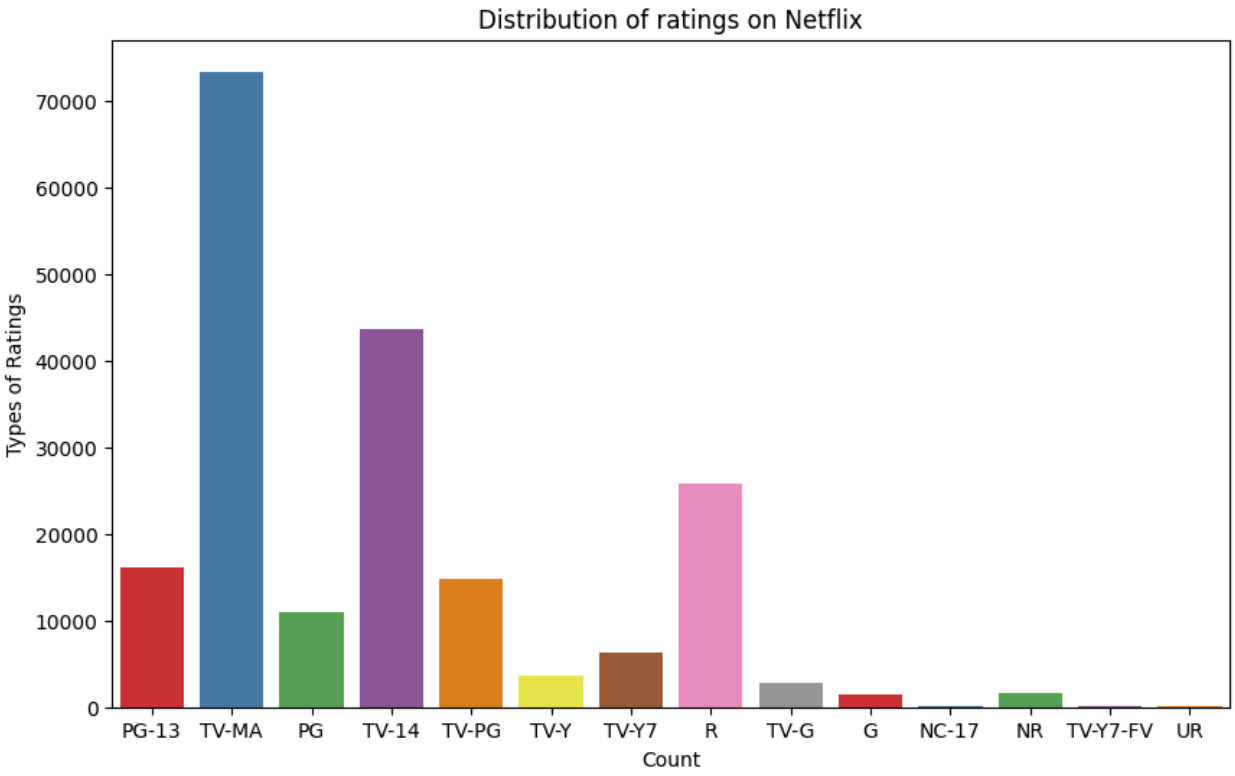
In [61]:
```
# Top 10 TV SHOW title
TV_show = category[category["type"]=="TV Show"]

Top10_tvshows = TV_show["title"].value_counts().head(10).reset_index()

plt.figure(figsize=(10,6))
sns.barplot(data=Top10_tvshows, y="title", x="count", palette="Set3")

plt.title("Top 10 Most Frequent TV Shows on Netflix")
plt.xlabel("Count")
plt.ylabel("Show Title")
plt.show()
```

```
/tmp/ipython-input-72692519.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.

  sns.barplot(data=Top10_tvshows, y="title", x="count", palette="Set3")
```



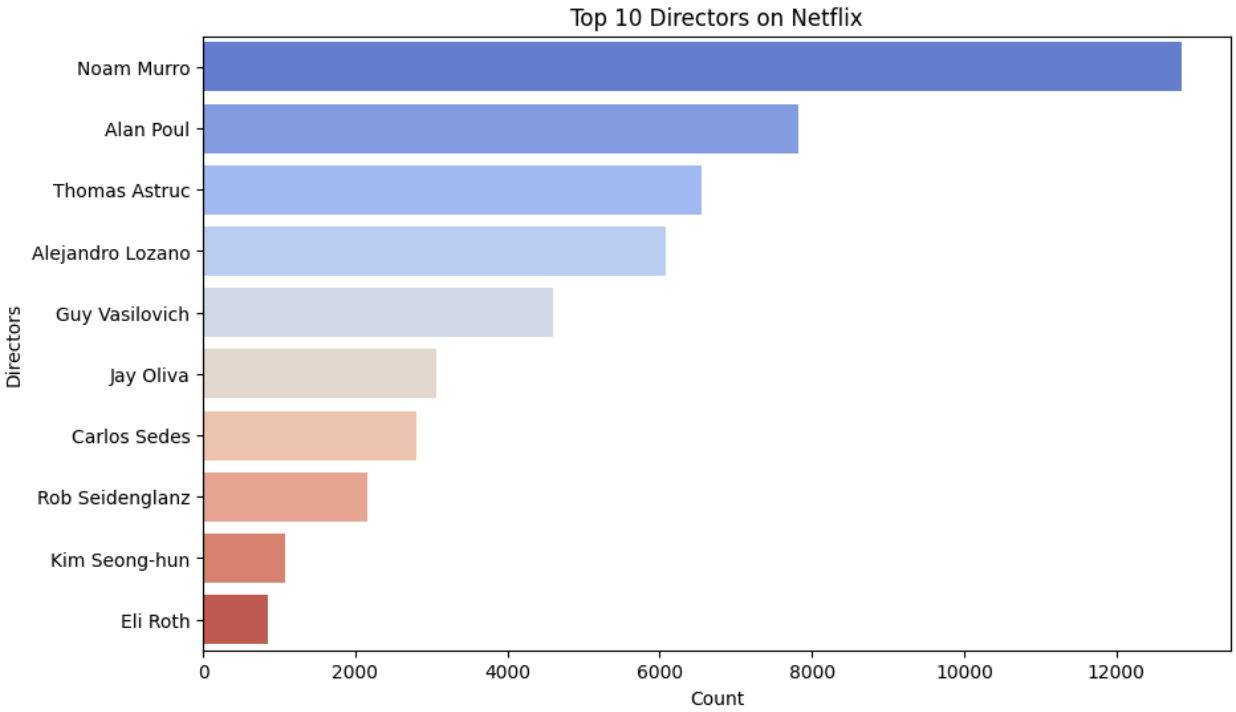
In [62]:
```
#Distribution of ratings on Netflix
plt.figure(figsize=(10,6))
sns.countplot(data=category, x="rating", palette="Set1")

plt.title("Distribution of ratings on Netflix")
plt.xlabel("Count")
plt.ylabel("Types of Ratings")
plt.show()
```

```
/tmp/ipython-input-3767132112.py:3: FutureWarning:

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

  sns.countplot(data=category, x="rating", palette="Set1")
```



In [63]:
```
#Top 10 Directors on Netflix
directors = category["Director"].value_counts().reset_index().head(10)

plt.figure(figsize=(10,6))
sns.barplot(data=directors,y="Director",x="count",palette="coolwarm")
plt.title("Top 10 Directors on Netflix")
plt.xlabel("Count")
plt.ylabel("Directors")
plt.show()
```

```
/tmp/ipython-input-2125654577.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.

  sns.barplot(data=directors,y="Director",x="count",palette="coolwarm")
```



Observations :

⚠ Business Insights from Netflix Dataset

1. **Movies Dominate the Platform**
    - Over **145,000 movies vs ~56,000 TV shows**.
    - This indicates Netflix has a stronger focus on **one-time content consumption (movies)** compared to **long-term engagement (TV shows)**.

---

2. **Content Saturation Around Few Titles & Directors**
    - Certain movies (*Kahlil Gibran's The Prophet*, *Holidays*) and shows (*The Eddy*, *Narcos*) appear excessively often.
    - Similarly, **top directors like Noam Murro (12k+)** dominate.
    - Insight: Either **duplicate records** or **over-representation of certain content** suggests the catalog is not as diverse as raw counts show.

---

3. **Audience Skews Heavily Toward Mature Content**
    - **TV-MA (~74k) and TV-14 (~44k)** dominate the ratings

distribution.
- Lower ratings (kids/family like TV-Y, TV-Y7, G) are much fewer.

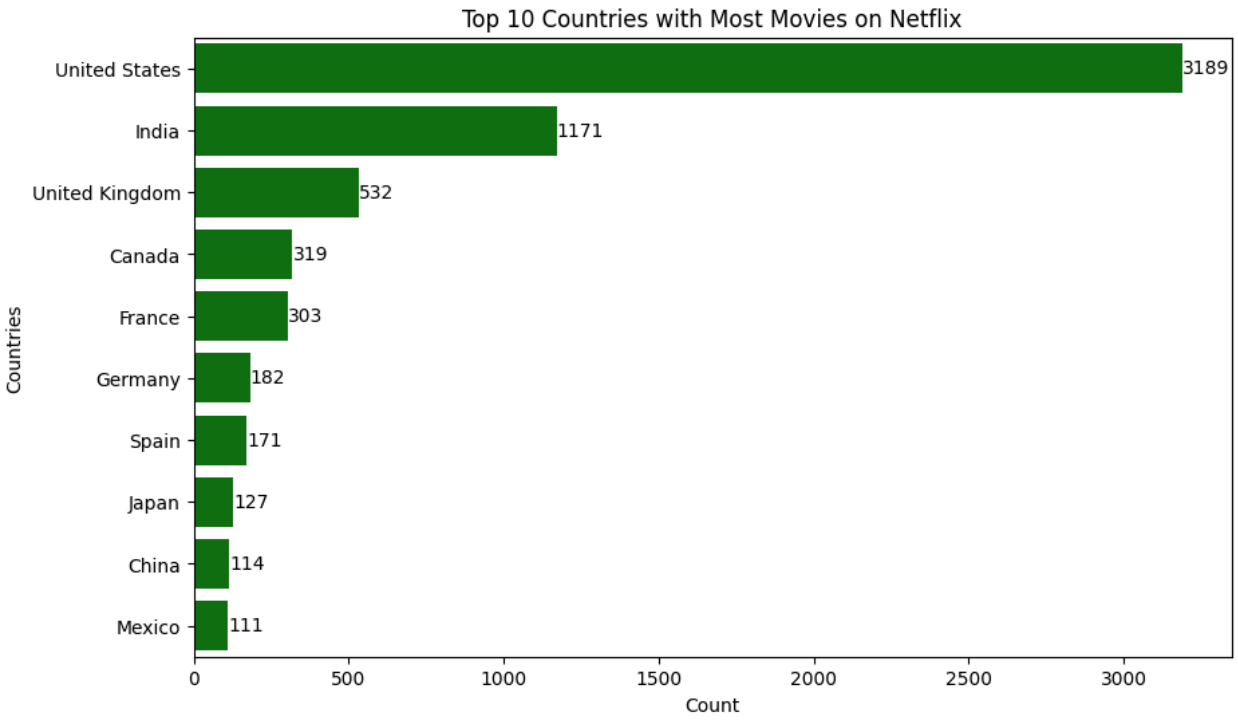
---

4. **Global Content Supply is Uneven**
    - U.S. dominates the dataset (~66k titles), followed by India (~24k) and UK (~12k).
    - Countries like **Botswana, Kazakhstan, Uganda** have only 1–2 entries.

2.Comparison of tv shows vs movies

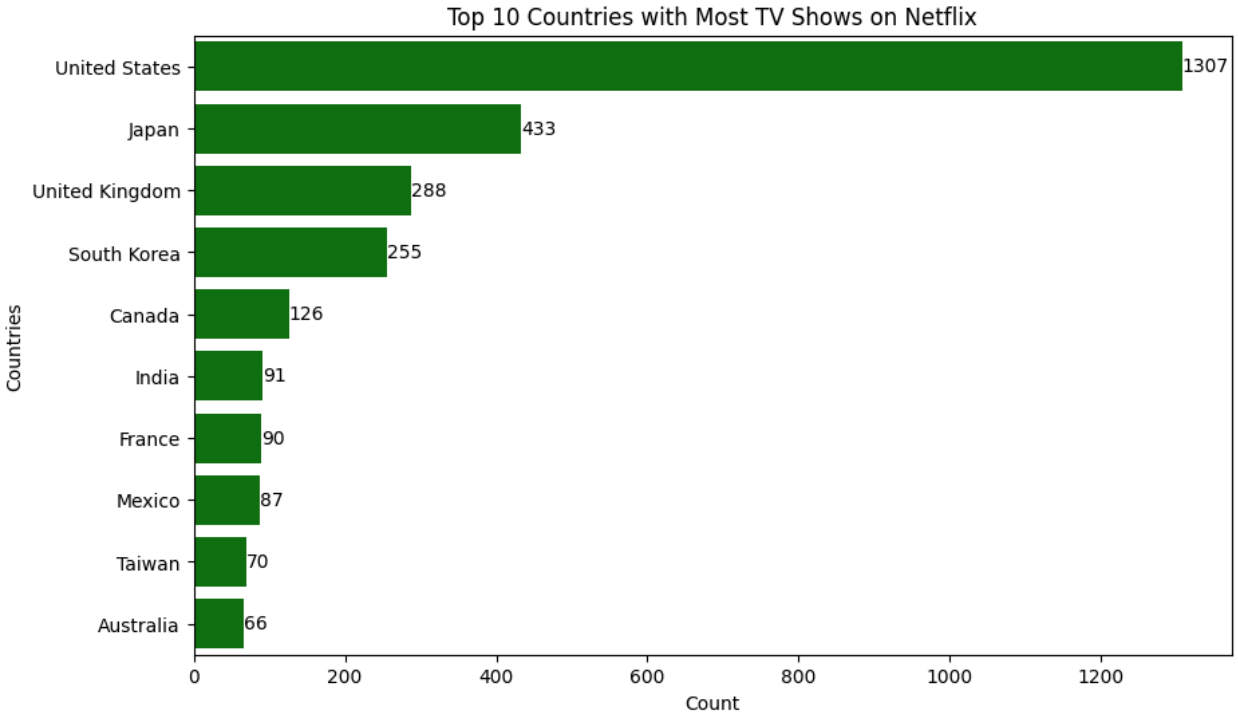
- **Find the number of movies produced in each country and pick the top 10 countries.**

In [64]:
```
country_movies = category[category["type"]=="Movie"].groupby("Country")["title


plt.figure(figsize=(10,6))
ax = sns.barplot(data=country_movies,y="Country",x="Count",color="green")
plt.title("Top 10 Countries with Most Movies on Netflix")
plt.xlabel("Count")
plt.ylabel("Countries")

plt.bar_label(ax.containers[0], fmt='%d')
plt.show()
```



- **Find the number of Tv-Shows produced in each country and pick the top 10 countries.**

In [65]:
```
country_tvshows = category[category["type"]=="TV Show"].groupby("Country")["ti



plt.figure(figsize=(10,6))
ax = sns.barplot(data=country_tvshows,y="Country",x="Count",color="green")
plt.title("Top 10 Countries with Most TV Shows on Netflix")
plt.xlabel("Count")
plt.ylabel("Countries")

plt.bar_label(ax.containers[0], fmt='%d')
plt.show()
```



⍰ Insights from Country Distribution

1. **U.S. Dominates Both Movies & TV Shows**
    - Movies: **3,189 titles**
    - TV Shows: **1,307 titles**
    - The U.S. contributes more than **2–3x compared to the second-ranked country** in both categories.

---

2. **India is Strong in Movies but Weak in TV Shows**
    - India ranks **#2 in movies (1,171)** but drops significantly in TV shows (**only 91**).

---

3. **East Asian Countries Excel in TV Shows**
    - Japan (**433 TV shows**) and South Korea (**255**) are top contributors.
    - These two countries are **global leaders in anime and K-drama content**.

---

4. **Regional Imbalances in Content Mix**
    - Countries like **France, Germany, Spain, Canada** have a

balanced presence across both movies and TV shows.

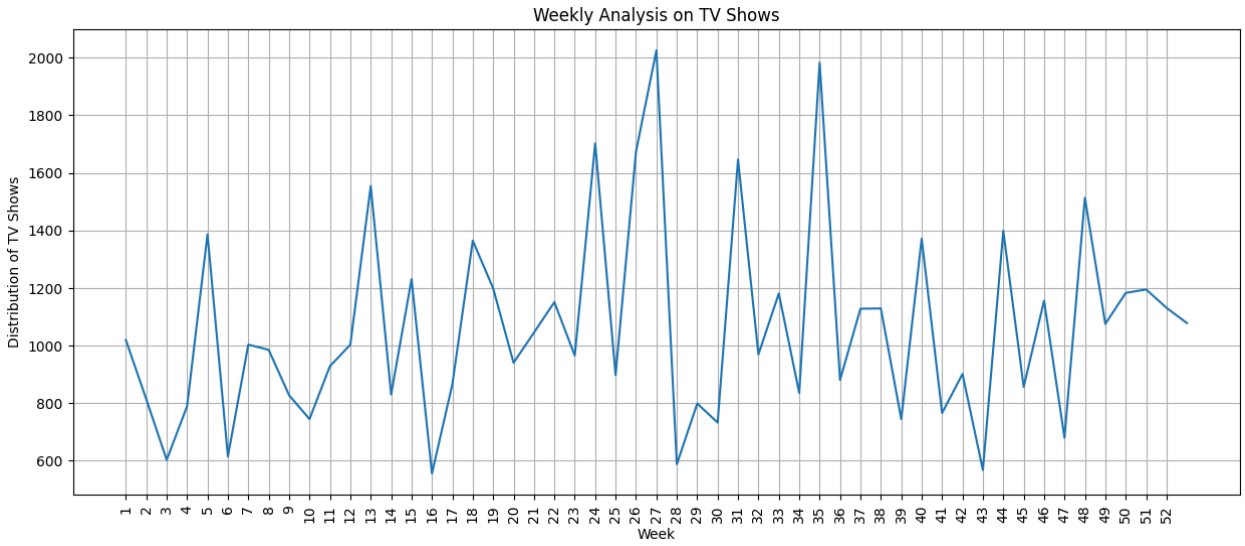
- On the other hand, **China and Mexico** show limited content overall, despite large local markets.

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

- **Find which is the best week to release the Tv-show or the movie. Do the analysis separately for Tv-shows and Movies**

In [66]:
```
data_clean["Week"] = pd.to_datetime(data_clean["date_added"]).dt.isocalendar()
```

In [67]:
```
week_tvshows = data_clean[data_clean["type"]=="TV Show"].groupby("Week")["titl

plt.figure(figsize=(15,6))

sns.lineplot(data=week_tvshows,x="Week",y="count")
plt.xticks(range(1, 53),rotation = 90)
plt.title("Weekly Analysis on TV Shows")
plt.xlabel("Week")
plt.ylabel("Distribution of TV Shows")
plt.grid(True)
plt.show()
```



Observations :

- Q1 (Weeks 1–13) – Average: ~942

  Lowest-performing quarter overall.

Possible reason: fewer big premieres early in the year.

- Q2 (Weeks 14–26) – Average: ~1105

  Strong growth with higher peaks (max 1700).

  Suggests this is a good time for mid-year show launches.

- Q3 (Weeks 27–39) – Average: ~1132 (Highest)

  Strongest quarter with peaks up to 2020.

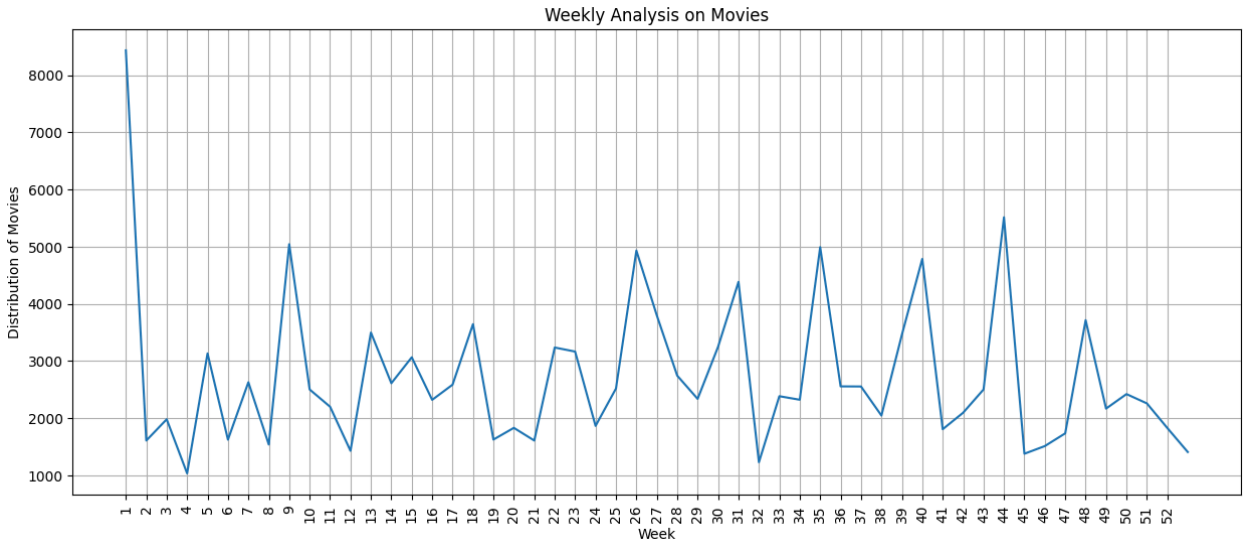
  Indicates high viewer engagement during mid-year breaks/summer.

- Q4 (Weeks 40–52) – Average: ~1062

  Stable with moderate peaks, likely boosted by holiday specials.

- **Weekly Analysis on Movies**

In [68]:
```
week_movies = data_clean[data_clean["type"]=="Movie"].groupby("Week")["title"]

plt.figure(figsize=(15,6))

sns.lineplot(data=week_movies,x="Week",y="count")
plt.xticks(range(1, 53),rotation = 90)
plt.title("Weekly Analysis on Movies")
plt.xlabel("Week")
plt.ylabel("Distribution of Movies")
plt.grid(True)
plt.show()
```



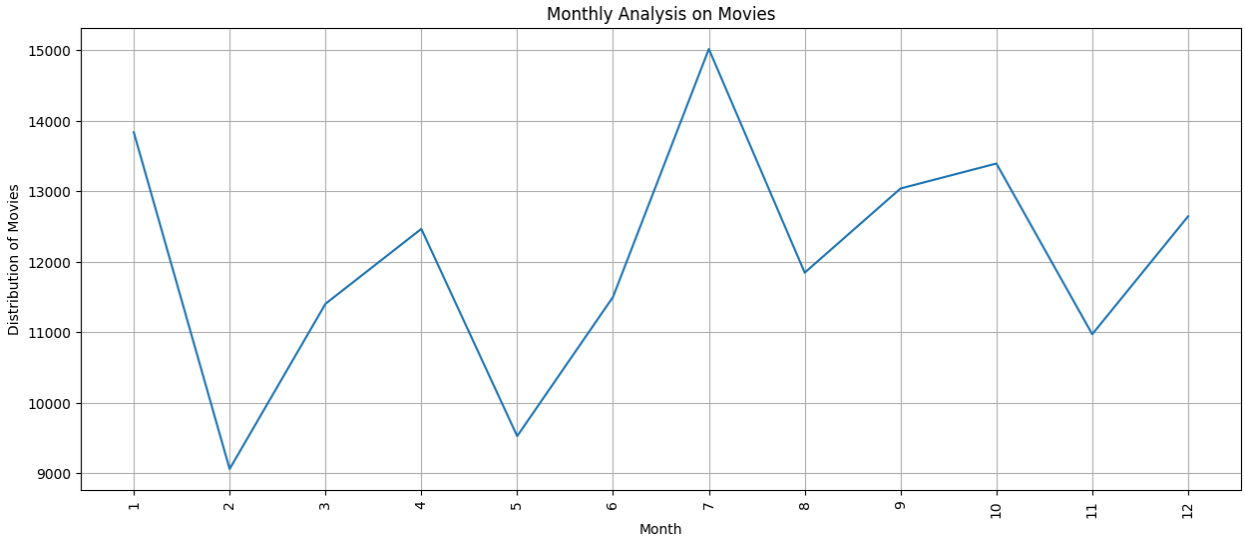
3 business insights from the weekly movie distribution chart:

1. **Massive Spike in Week 1**
    - The first week shows an exceptionally high distribution (~8500), far above all other weeks.
    - Likely due to **holiday season releases, New Year specials, or bulk launches**.
    - Business implication: Capitalize on early-year hype with blockbuster movie releases or subscription promotions.
2. **Regular Spikes Every Few Weeks**
    - Noticeable peaks around weeks **9–10, 26–27, 35–36, 43–44**, suggesting periodic high-demand cycles.
    - Indicates strong opportunities for **quarterly big releases** to sustain momentum and revenue.
3. **Steady Mid-Level Engagement**
    - Outside of spikes, the baseline distribution stabilizes between **1500–3000**.
    - Shows that while major releases bring surges, there's a **consistent audience base** for weekly movie consumption.
    - This baseline can be leveraged for testing **niche genres or regional movies** without major risk.

- **Find which is the best month to release the Tv-show or the movie. Do the analysis separately for Tv-shows and Movies**

In [69]:
```
data_clean["Month"] = pd.to_datetime(data_clean["date_added"]).dt.month
```

In [70]:
```
Monthly_movies = data_clean[data_clean["type"]=="Movie"].groupby("Month")["tit

plt.figure(figsize=(15,6))

sns.lineplot(data=Monthly_movies,x="Month",y="count")
plt.xticks(range(1, 13),rotation = 90)
plt.title("Monthly Analysis on Movies")
plt.xlabel("Month")
plt.ylabel("Distribution of Movies")
plt.grid(True)
plt.show()
```

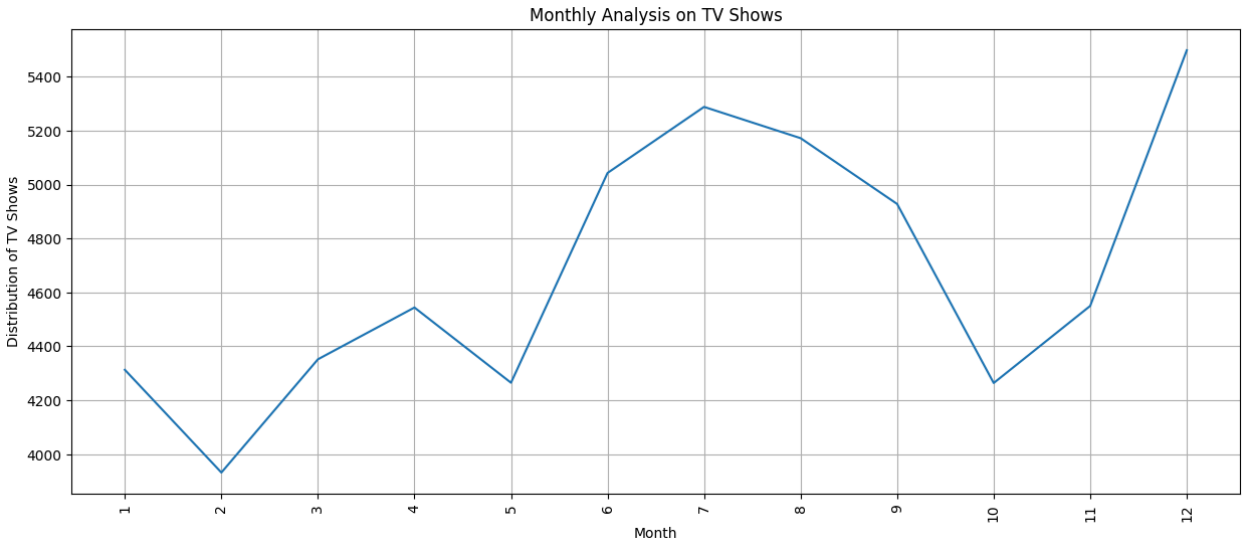


Monthly Analysis on Movies chart, here are the insights:

1. **Peak Months for Movie Distribution**
    - **July (Month 7)** is the strongest with distribution around **15,000**.
    - Other high-performing months: **January (~13,800), September (~13,000), October (~13,400)**.
    - These months are ideal for **big blockbuster releases** since audience engagement is highest.
2. **Low-Performing Months**
    - **February (~9,000) and May (~9,500)** show the weakest demand.
    - Likely due to fewer holidays/events during these months.
    - Business implication: Avoid major launches here, or use this time to release **niche or experimental films**.
3. **Strong Year-End Demand**
    - From **September to December**, distribution is consistently above **12,000**, showing a strong year-end push.
    - Perfect for **holiday season releases, family-oriented movies, and award-contender films**.

In [71]:
```
Monthly_tvshows= data_clean[data_clean["type"]=="TV Show"].groupby("Month")["t

plt.figure(figsize=(15,6))

sns.lineplot(data=Monthly_tvshows,x="Month",y="count")
plt.xticks(range(1, 13),rotation = 90)
plt.title("Monthly Analysis on TV Shows")
```

```
plt.xlabel("Month")
plt.ylabel("Distribution of TV Shows")
plt.grid(True)
plt.show()
```



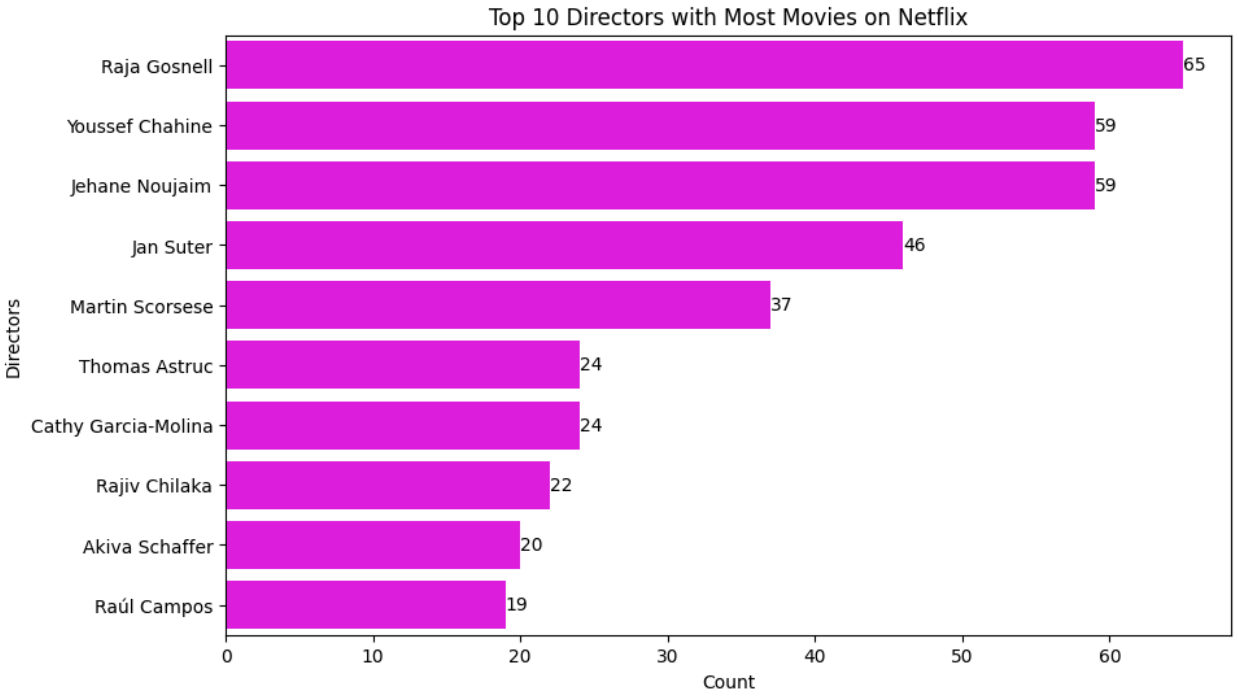
Monthly Analysis on TV Shows chart, here are the insights:

1. **Peak Months for TV Shows**
    - **July (~5300) and December (~5500)** are the strongest months.
    - These align with **mid-year breaks (summer holidays)** and **year-end holidays**, when people binge-watch more content.
    - Ideal for **new season launches and high-budget shows**.
2. **Mid-Year Surge**
    - From **June to September**, distribution is consistently high (5000+).
    - This period is best for **series premieres and long-running shows**, as engagement remains steady.
3. **Low Points**
    - **February (~3900) and October (~4250)** are the weakest months.
    - Business implication: Avoid high-profile launches here; instead, test **experimental or niche shows**.

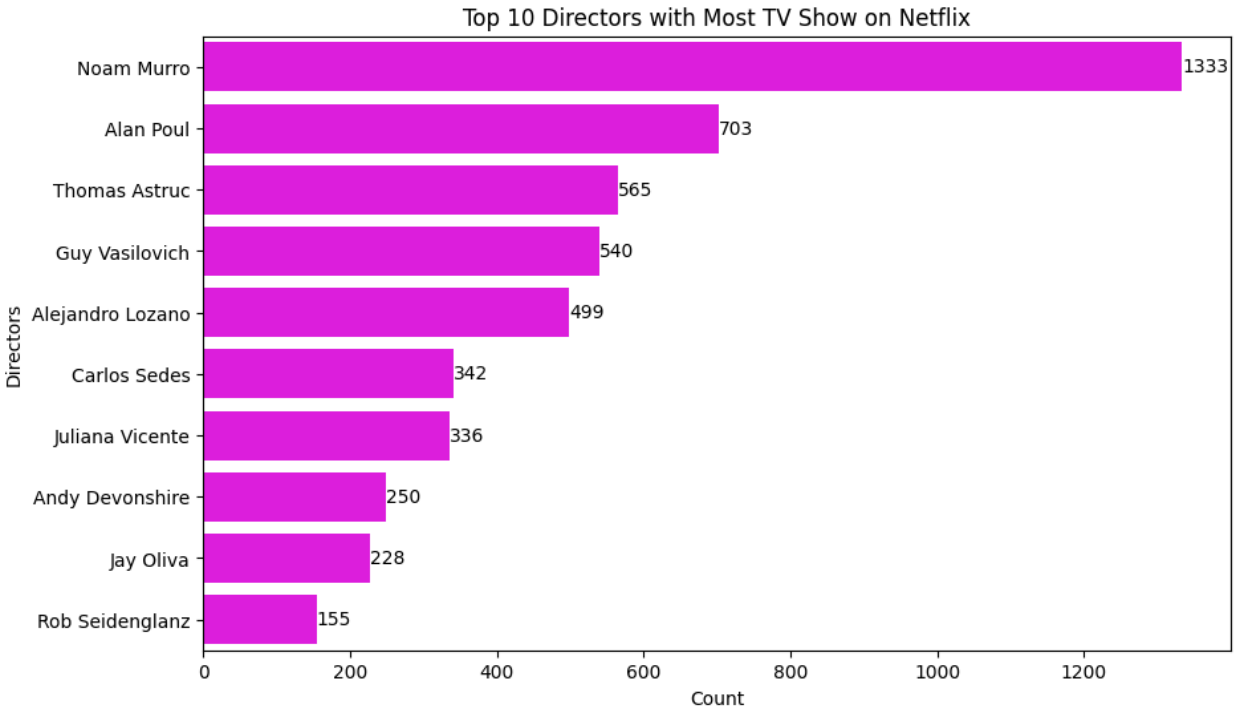
4. Analysis of actors/directors of different types of shows/movies.

- **Identify the top 10 directors who have appeared in most movies or TV shows.**

In [72]:
```
directors_movies = data_clean[data_clean["type"]=="Movie"].groupby("Director")

plt.figure(figsize=(10,6))
ax=sns.barplot(data=directors_movies,y="Director",x="Count",color ="magenta")
plt.title("Top 10 Directors with Most Movies on Netflix")
plt.xlabel("Count")
plt.ylabel("Directors")

plt.bar_label(ax.containers[0], fmt='%d')
plt.show()
```

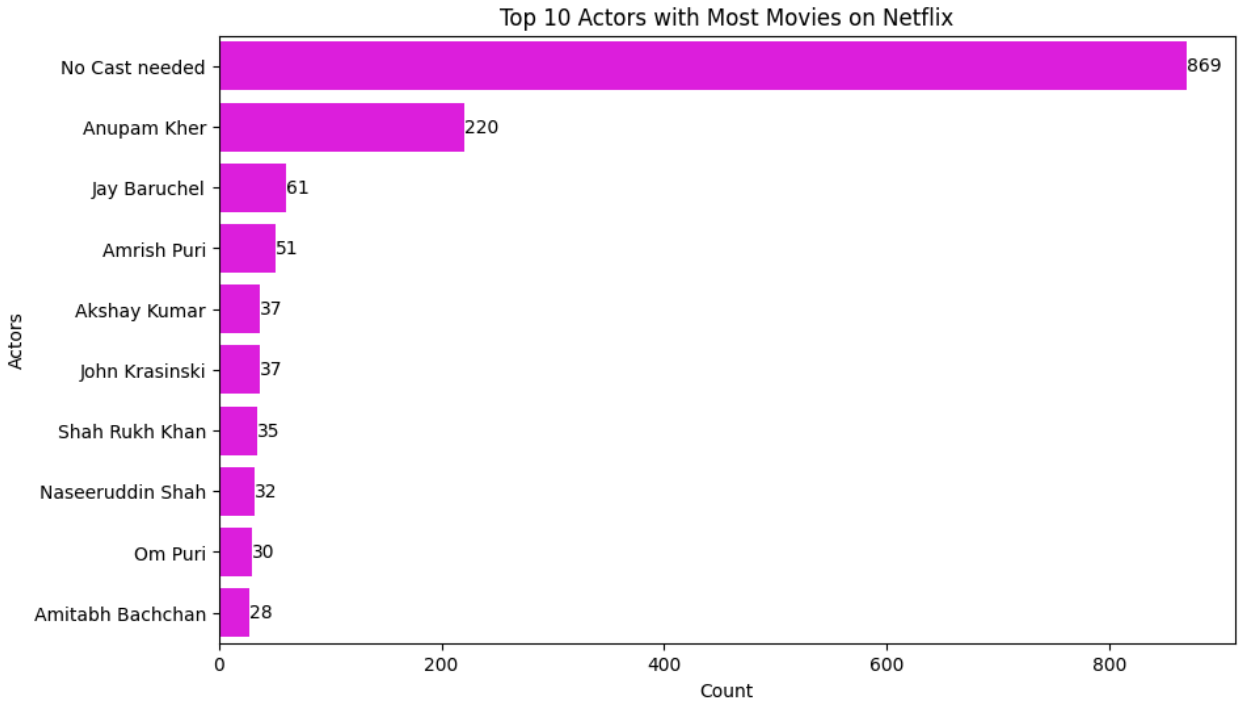


In [73]:
```
directors_TV_Show = data_clean[data_clean["type"]=="TV Show"].groupby("Directo

plt.figure(figsize=(10,6))
ax = sns.barplot(data=directors_TV_Show,y="Director",x="Count",color ="magenta
plt.title("Top 10 Directors with Most TV Show on Netflix")
plt.xlabel("Count")
plt.ylabel("Directors")

plt.bar_label(ax.containers[0], fmt='%d')
plt.show()
```

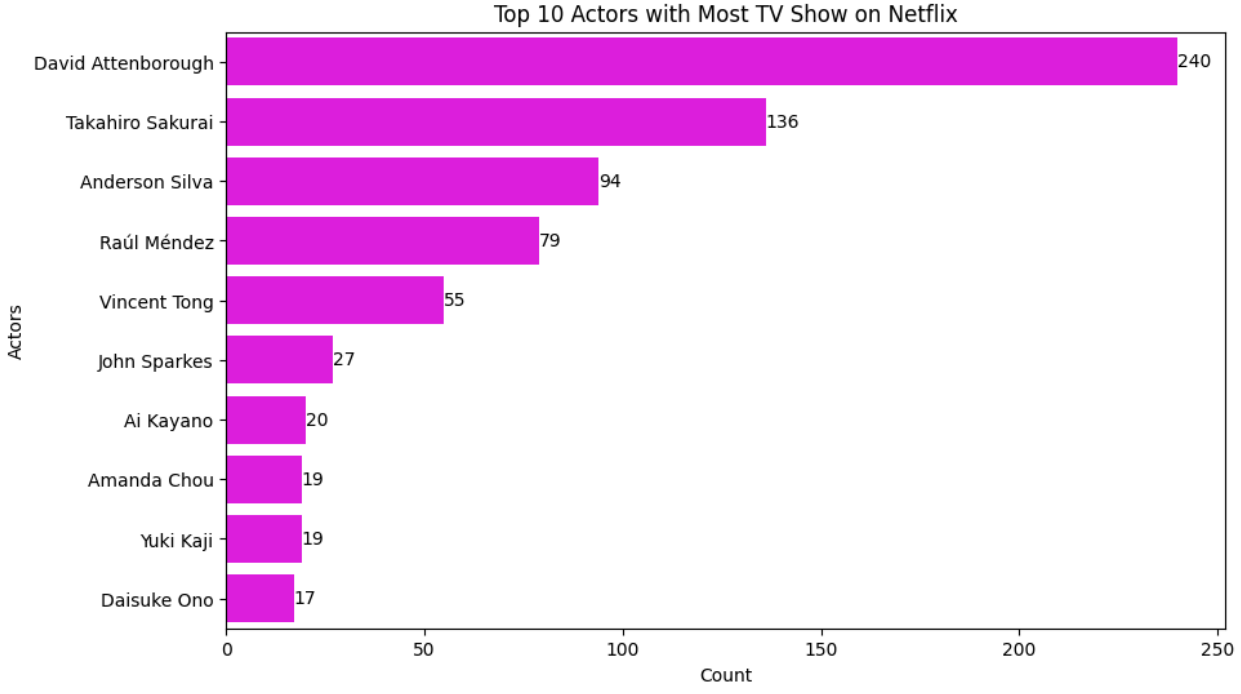


- **Identify the top 10 Actors who have appeared in most movies or TV shows.**

In [74]:
```
Cast_movies = data_clean[data_clean["type"]=="Movie"].groupby("Cast")["title"]

plt.figure(figsize=(10,6))
ax = sns.barplot(data=Cast_movies,y="Cast",x="title",color ="magenta")
plt.title("Top 10 Actors with Most Movies on Netflix")
plt.xlabel("Count")
plt.ylabel("Actors")

plt.bar_label(ax.containers[0], fmt='%d')
plt.show()
```



In [75]:
```
cast_tvshow = data_clean[data_clean["type"]=="TV Show"].groupby("Cast")["title

plt.figure(figsize=(10,6))
ax = sns.barplot(data=cast_tvshow,y="Cast",x="title",color ="magenta")
plt.title("Top 10 Actors with Most TV Show on Netflix")
plt.xlabel("Count")
plt.ylabel("Actors")

plt.bar_label(ax.containers[0],fmt="%d")
plt.show()
```



◈ Top Directors (Movies & TV Shows)

Movies: Raja Gosnell, Youssef Chahine, and Jehane Noujaim dominate Netflix movies.

TV Shows: Noam Murro and Alan Poul lead by a wide margin (over 1000+ shows). ◈ If your content aligns with their genres/styles, it has higher chance of discovery due to audience overlap.

◈ Top Actors (Movies & TV Shows)

Movies: Anupam Kher (220 movies), Akshay Kumar, Shah Rukh Khan, Amitabh Bachchan — strong Indian presence.

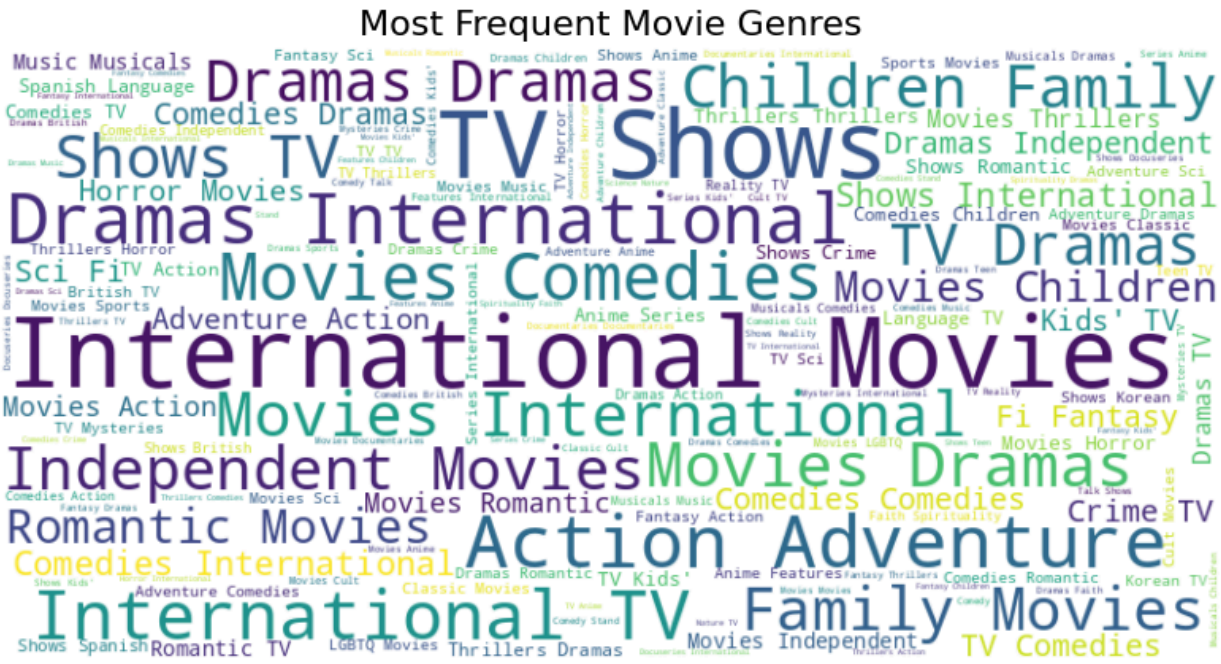
TV Shows: David Attenborough leads (240 shows), followed by Japanese & Latin American actors. ◈ This suggests regional targeting works well — Indian stars for movies, international/niche audiences for TV.

5. Which genre movies are more popular or produced more ?

In [76]:
```
from wordcloud import WordCloud
```

In [77]:
```
text = " ".join(data_clean["Listed_in"].dropna().astype(str))
wordcloud = WordCloud(width=800, height=400, background_color="white",colormap
```

In [78]:
```
plt.figure(figsize=(10,5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Most Frequent Movie Genres", fontsize=16)
plt.show()
```



◈ Dominant Genres

International Movies (biggest presence) → Netflix heavily invests in global, non-Hollywood films.

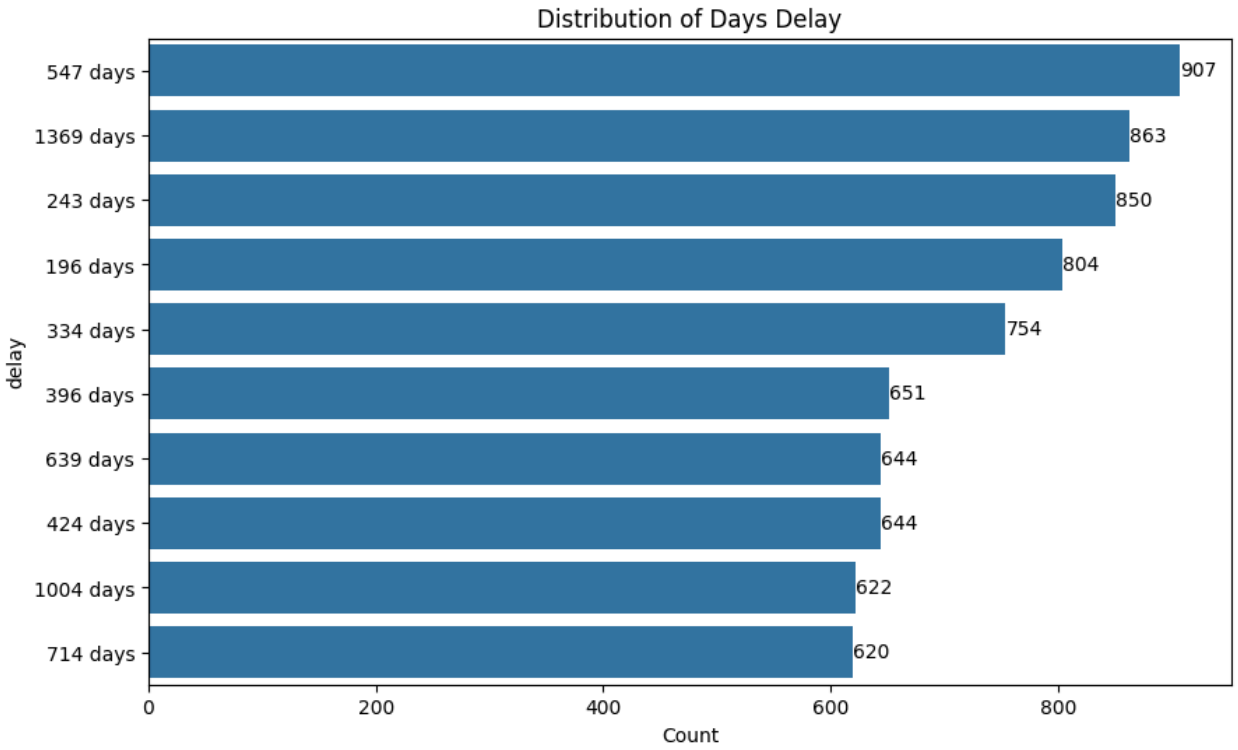
Dramas & TV Shows → Constant demand, especially binge-worthy content.

Children & Family / Kids' TV → A strong evergreen segment.

6 .Find After how many days the movie will be added to Netflix after the release of the movie (you can consider the recent past data)

In [79]:
```
data_clean["date_added"] = pd.to_datetime(data_clean["date_added"])

data_clean["release_year"] = pd.to_datetime(data_clean["release_year"],format=

data_clean["delay"] = data_clean["date_added"] - data_clean["release_year"]

days_delay = data_clean["delay"].value_counts().reset_index().head(10)

plt.figure(figsize=(10,6))
ax = sns.barplot(data=days_delay,y="delay",x="count")

plt.bar_label(ax.containers[0],fmt="%d")
plt.title("Distribution of Days Delay")
```

```
plt.xlabel("Count")
plt.show()
```



Most Common Delay Window

The highest count (907 movies) were added after approx.547 days (~1.5 years).

Another peak at approx.1369 days (~3.7 years, 863 movies) → represents older catalog content entering Netflix much later.

Other Frequent Delay Points:

243 days (~8 months) → 850 movies.

196 days (~6.5 months) → 804 movies.

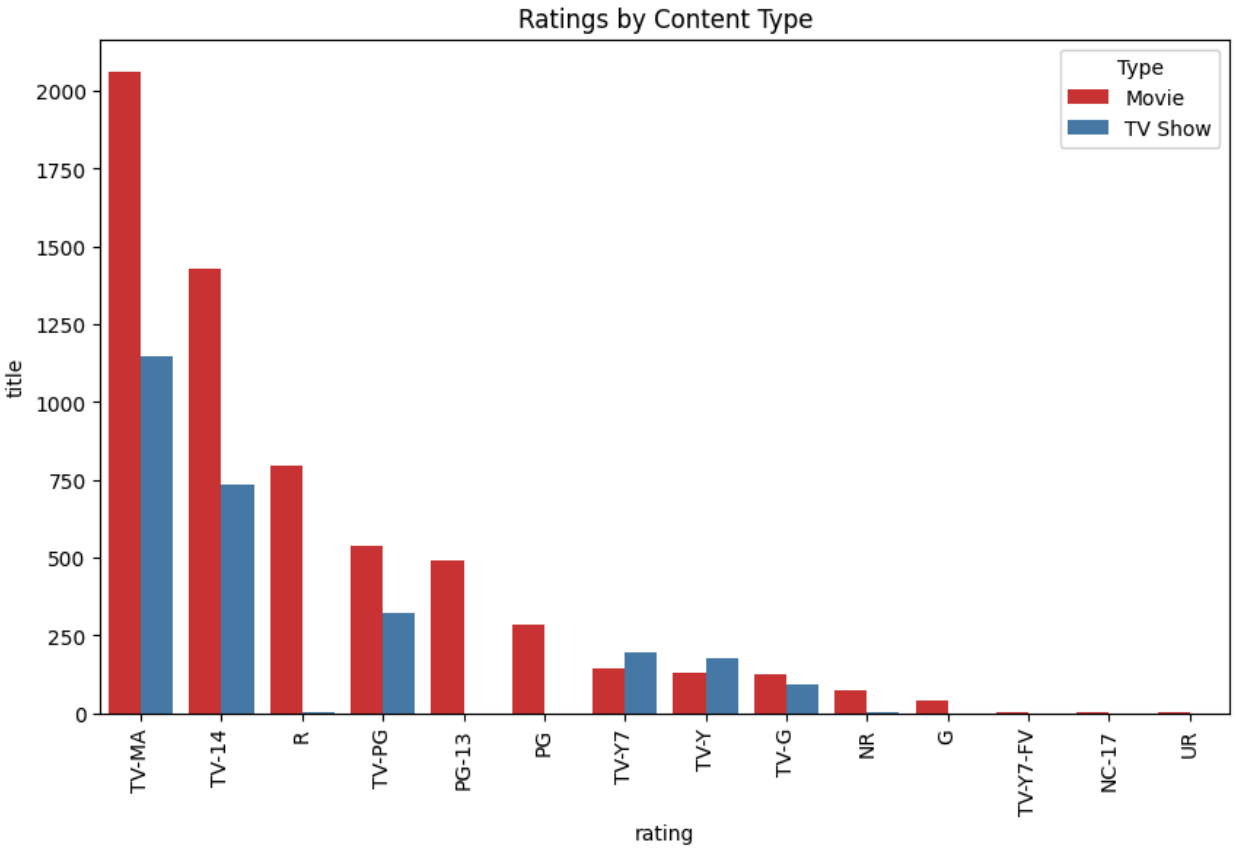
334 days (~11 months) → 754 movies.

Bi-variate Analysis

In [81]:
```
ratings_data = data_clean.groupby(["rating","type"])["title"].nunique().reset_
ratings_data

plt.figure(figsize=(10,6))
sns.barplot(data=ratings_data, x='rating',y="title",hue="type",palette ="Set1"
plt.title("Ratings by Content Type")
plt.xticks(rotation=90)
```

```
plt.legend(title='Type')
```

Out[81]: <matplotlib.legend.Legend at 0x7901088fa4b0>

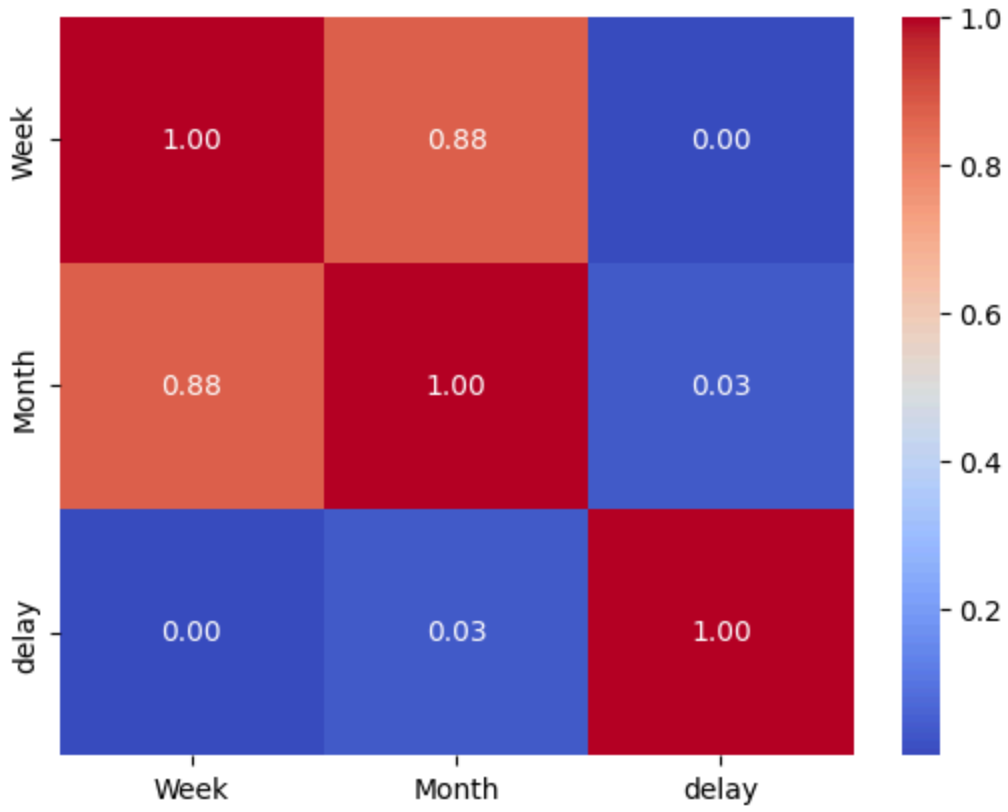


HEATMAP & PAIRPLOT

In [82]:
```
heat_map = data_clean.copy()
heat_map = heat_map.select_dtypes(include=np.number)
```

In [83]:
```
sns.heatmap(heat_map.corr(),annot=True, cmap='coolwarm', fmt='.2f')
```

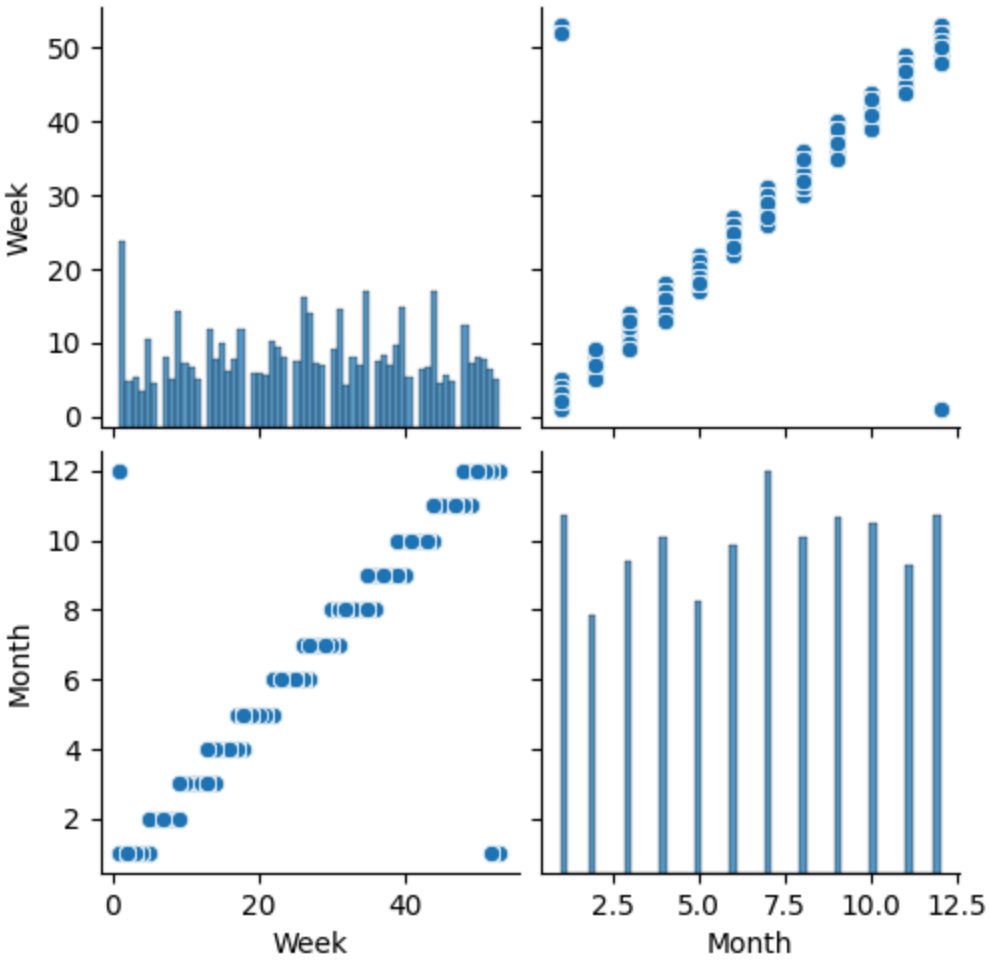
Out[83]: <Axes: >



PAIRPLOT

In [85]:
```
sns.pairplot(data=data_clean)
```

Out[85]: <seaborn.axisgrid.PairGrid at 0x7901089d3440>



Business Insights & Recommendations

Movies

Volume & mix

- **Movies dominate** (~145k vs ~56k TV). **Why:** One-and-done consumption outweighs long-arc engagement. **Action:** Keep a **high-velocity movie pipeline**; use movies to drive acquisitions and pair with TV to lift retention.

Seasonality (weekly/monthly)

- **Mega spike Week 1**; recurring peaks around **Weeks 9–10, 26–27, 35–36, 43–44**.
- **Best months: Jan, Jul, Sep–Oct, Dec**. **Weak months: Feb, May**.

Action:

- Schedule **blockbusters** for **Jan, Jul, Sep–Oct, Dec**.
- Use **Feb/May** for **niche/regional/experimental** titles and audience testing.

Licensing delay to Netflix

- Most common windows: **~6–8 months**, **~11–18 months (peak ~547 days)**; long tail **3–4 years**. **Action:**
- Target **≤ 12-month pay-1 window** for priority titles; negotiate **accelerated (≤ 8 months)** for tentpoles.
- Build a **catalog catch-up program** for 3–4-year titles to fill offseason gaps.

TV Shows

Volume & seasonality

- Smaller base than movies, but **mid-year & year-end peaks**: **Jun–Sep** (steady >5k) and **Dec**.
- **Low points: Feb, Oct**. **Action:**
- Launch **new seasons/premieres in Jun–Jul**; anchor **finales/event drops in Dec**.
- Use **Feb/Oct** for **low-risk pilots**, reality, or minis.

Countries & localization

Distribution

- **US leads** (both movies & TV).
- **India** strong in **movies** but **weak in TV**.
- **Japan & South Korea** over-index in **TV** (anime/K-drama). **Action:**
- **Scale Indian Originals (TV)** to close the TV gap; leverage known stars.
- Expand **anime & K-drama co-pros** and localized marketing.
- Keep a **balanced slate** from France/Germany/Spain/Canada to serve pan-EU audiences.

Genres

Dominant

- **International Movies**, **Dramas**, **Kids/Family**, **Comedies**, **Action/Adventure**. **Action (calendar by genre):**
- **Kids/Family: Summer + Dec**.
- **Romance: Feb** (Valentine's) + shoulder weekends.
- **Action/Adventure: Jun–Jul, Dec**.
- **Dramas/International: year-round**, time with local holidays & festivals.

Talent (Directors & Actors)

Directors

- Heavy concentration (e.g., Noam Murro, Alan Poul for TV; Raja Gosnell, Youssef Chahine, Jehane Noujaim for movies). **Action:**
- Create **director-driven collections** and **row merchandising** around top names.
- Diversify by commissioning **emerging-market directors** to reduce catalog concentration.

Actors

- **Movies:** strong Indian star power (Anupam Kher, Akshay Kumar, Shah Rukh Khan, Amitabh Bachchan).
- **TV:** leaders include **David Attenborough** and notable JP/LatAm voices. **Action:**
- **Geo-target campaigns**: Bollywood stars for India/global diaspora; JP/KR talent for anime/K-drama hubs.
- Use **star-led rows** to improve discovery and CTR.

Titles & Catalog Health

Over-representation / possible duplicates

- Repeated titles (e.g., *Kahlil Gibran's The Prophet*, *Holidays*) and very

high counts for some creators suggest **duplication or variant records**. **Action:**

- Build a **dedup pipeline** (title+year+country+duration, fuzzy title match, ISAN/IMDB id if available).
- Track a **Catalog Diversity Index (HHI)** to avoid over-weighted franchises/creators.

Ratings mix

- **TV-MA & TV-14 dominate**; kids ratings are sparse. **Action:**
- **Grow Kids/Family** to balance the slate and reduce churn among households; time releases to **school holidays**.

Release Playbook (one-glance)

- **Blockbusters (Movies): Jan, Jul, Sep–Oct, Dec**; priority weeks **1, 26–27, 35–36, 43–44**.
- **Series Premieres/Returns (TV): Jun–Jul**; **event finales** in **Dec**.
- **Niche/Experimental: Feb & May (movies)**, **Feb & Oct (TV)**.
- **Kids/Family: Jun–Jul, Dec**.
- **Romance: Feb**.
- **Action/Adventure: Jun–Jul, Dec**.

In [ ]: