Business Case: Netflix - Data Exploration and Visualisation

** Problem Statement**

Netflix wants to understand what kind of shows/movies to produce and how to grow in different countries. Using the dataset of all available titles on Netflix, we aim to derive actionable insights and recommend content strategies using data.

Import Libraries & Load Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Load the Dataset

```
!wget
https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940
/original/netflix.csv

--2025-05-10 15:59:02--
https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940
/original/netflix.csv
Resolving d2beiqkhq929f0.cloudfront.net
(d2beiqkhq929f0.cloudfront.net)... 3.169.117.64, 3.169.117.63,
3.169.117.127, ...
Connecting to d2beiqkhq929f0.cloudfront.net
(d2beiqkhq929f0.cloudfront.net)|3.169.117.64|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3399671 (3.2M) [text/plain]
Saving to: ‘netflix.csv.1’

netflix.csv.1       100%[===================>]   3.24M  --.-KB/s    in
0.04s

2025-05-10 15:59:02 (91.1 MB/s) - ‘netflix.csv.1’ saved
[3399671/3399671]
```

```python
df=pd.read_csv('netflix.csv')
df
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8807,\n  \"fields\":
[\n    {\n      \"column\": \"show_id\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 8807,\n
\"samples\": [\n          \"s4971\",\n          \"s3363\",\n
\"s5495\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"type\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"TV
Show\",\n          \"Movie\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"title\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 8807,\n
\"samples\": [\n          \"Game Over, Man!\",\n          \"Arsenio
Hall: Smart & Classy\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"director\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 4528,\n
\"samples\": [\n          \"Kanwal Sethi\",\n          \"R\\u00e9my
Four, Julien War\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cast\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 7692,\n        \"samples\": [\n          \"Tzi
Ma, Christine Ko, Hong-Chi Lee, Hayden Szeto, Kunjue Li, Fiona Fu,
James Saito, Joan Chen\",\n          \"Priyanshu Painyuli,
Chandrachoor Rai, Shadab Kamal, Rajeev Siddhartha, Sheetal Thakur,
Ninad Kamat, Swati Semwal, Eijaz Khan\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"country\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
748,\n        \"samples\": [\n          \"United States, United
Kingdom, Denmark, Sweden\",\n          \"United Kingdom, Hong Kong\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"date_added\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 1767,\n        \"samples\": [\n
\"October 22, 2018\",\n          \"January 29, 2021\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"release_year\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
8,\n        \"min\": 1925,\n        \"max\": 2021,\n
\"num_unique_values\": 74,\n        \"samples\": [\n          1996,\n
1969\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 17,\n
\"samples\": [\n          \"PG-13\",\n          \"TV-MA\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"duration\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 220,\n        \"samples\": [\n          \"37
min\",\n          \"177 min\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"listed_in\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 514,\n
\"samples\": [\n          \"Crime TV Shows, International TV Shows, TV
Mysteries\",\n          \"Children & Family Movies, Classic Movies,
Dramas\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"description\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 8775,\n
\"samples\": [\n          \"A heedless teen drifter who falls for a
small-town waitress makes the mistake of robbing a drug lord, putting
his life and newfound love in jeopardy.\",\n          \"Twelve-year-
old Calvin manages to join the navy and serves in the battle of
Guadalcanal. But when his age is revealed, the boy is sent to the
brig.\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

Data Overview & Cleaning

```
df.shape
```

```
(8807, 12)
```

```
df.head()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8807,\n  \"fields\":
[\n    {\n      \"column\": \"show_id\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 8807,\n
\"samples\": [\n          \"s4971\",\n          \"s3363\",\n
\"s5495\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"type\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"TV
Show\",\n          \"Movie\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"title\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 8807,\n
\"samples\": [\n          \"Game Over, Man!\",\n          \"Arsenio
Hall: Smart & Classy\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"director\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 4528,\n
\"samples\": [\n          \"Kanwal Sethi\",\n          \"R\\u00e9my
Four, Julien War\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cast\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 7692,\n        \"samples\": [\n          \"Tzi
Ma, Christine Ko, Hong-Chi Lee, Hayden Szeto, Kunjue Li, Fiona Fu,
James Saito, Joan Chen\",\n          \"Priyanshu Painyuli,
Chandrachoor Rai, Shadab Kamal, Rajeev Siddhartha, Sheetal Thakur,
Ninad Kamat, Swati Semwal, Eijaz Khan\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"country\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
748,\n        \"samples\": [\n          \"United States, United
Kingdom, Denmark, Sweden\",\n          \"United Kingdom, Hong Kong\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"date_added\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 1767,\n        \"samples\": [\n
\"October 22, 2018\",\n          \"January 29, 2021\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"release_year\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
8,\n        \"min\": 1925,\n        \"max\": 2021,\n
\"num_unique_values\": 74,\n        \"samples\": [\n          1996,\n
1969\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 17,\n
\"samples\": [\n          \"PG-13\",\n          \"TV-MA\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"duration\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 220,\n        \"samples\": [\n          \"37
min\",\n          \"177 min\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"listed_in\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 514,\n
\"samples\": [\n          \"Crime TV Shows, International TV Shows, TV
Mysteries\",\n          \"Children & Family Movies, Classic Movies,
Dramas\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"description\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 8775,\n
\"samples\": [\n          \"A heedless teen drifter who falls for a
small-town waitress makes the mistake of robbing a drug lord, putting
his life and newfound love in jeopardy.\",\n          \"Twelve-year-
old Calvin manages to join the navy and serves in the battle of
Guadalcanal. But when his age is revealed, the boy is sent to the
brig.\"\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

```
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
```

```
df.isnull().sum()
```

```
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
```

```
df[df.duplicated()]
```

{"repr_error":"Out of range float values are not JSON compliant:
nan","type":"dataframe"}

```
df['director'].fillna("Unknown_Value",inplace=True)
df['cast'].fillna("Unknown_Value",inplace=True)
df['country'].fillna("Unknown_Value",inplace=True)
```

```
<ipython-input-42-e84223180a08>:1: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['director'].fillna("Unknown_Value",inplace=True)
<ipython-input-42-e84223180a08>:2: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['cast'].fillna("Unknown_Value",inplace=True)
<ipython-input-42-e84223180a08>:3: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['country'].fillna("Unknown_Value",inplace=True)
```

```
df.dropna(subset=['date_added'],inplace=True)
df.dropna(subset=['rating'],inplace=True)
df.dropna(subset=['duration'],inplace=True)
```

```
df.nunique()
```

```
show_id         8790
type               2
```

```
title           8790
director        4527
cast            7679
country          749
date_added      1765
release_year      74
rating            14
duration         220
listed_in        513
description     8758
dtype: int64
```

Convert 'date_added' to datetime

```
df['date_added'] = pd.to_datetime(df['date_added'], format='%B %d,
%Y', errors='coerce')
```

Convert 'rating', 'type', etc. to category

```
df['type'] = df['type'].astype('category')

df['rating'] = df['rating'].astype('category')
```

Non-Graphical Analysis

```
df['type'].value_counts()
df['country'].value_counts().head(10)
df['listed_in'].value_counts().head(10)
df['release_year'].value_counts().sort_index().tail(30)

release_year
1992      23
1993      28
1994      22
1995      25
1996      24
1997      38
1998      36
1999      39
2000      37
2001      45
2002      51
2003      59
2004      64
2005      80
2006      96
2007      88
```

```
2008     135
2009     152
2010     192
2011     185
2012     236
2013     286
2014     352
2015     555
2016     901
2017    1030
2018    1146
2019    1030
2020     953
2021     592
Name: count, dtype: int64
```

```
df.nunique()
```

```
show_id         8790
type               2
title           8790
director        4527
cast            7679
country          749
date_added      1697
release_year      74
rating            14
duration         220
listed_in        513
description     8758
dtype: int64
```

Data Analysis and Data Visualization.

```
df.head()
```

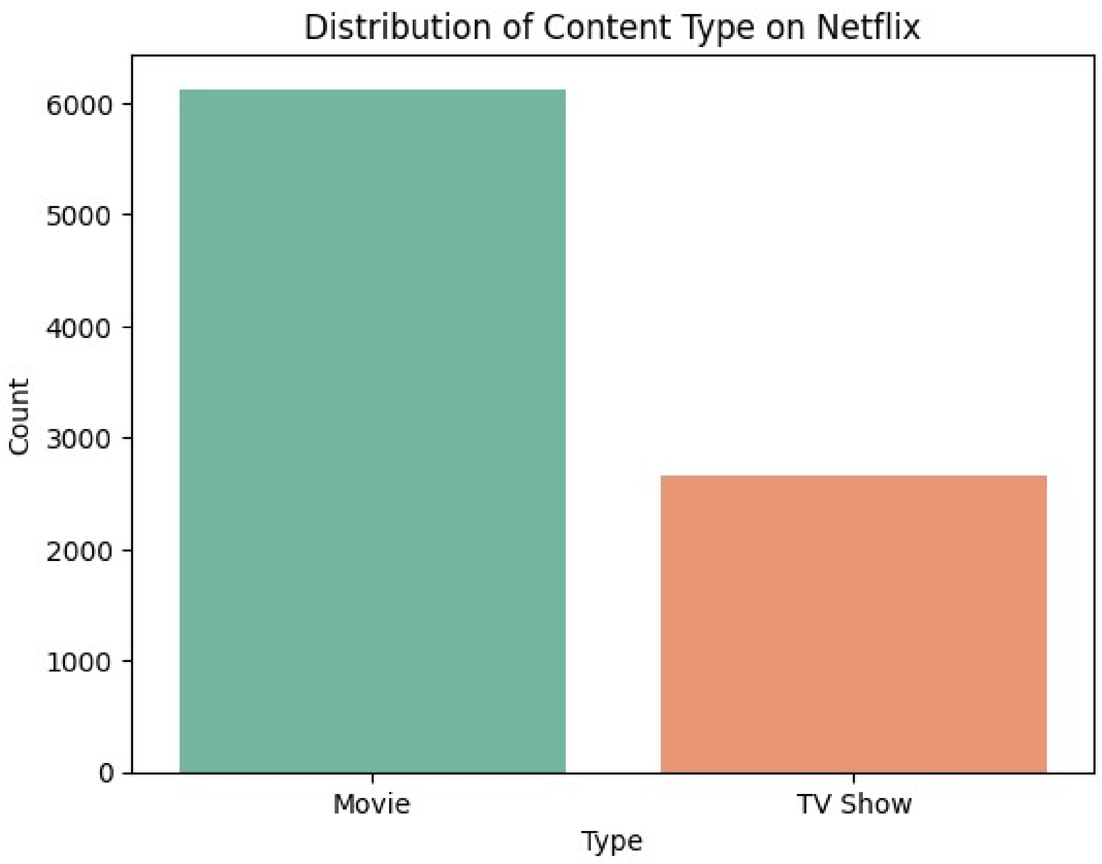
{"summary":"{\n \"name\": \"df\",\n \"rows\": 8790,\n \"fields\": [\n   {\n   \"column\": \"show_id\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 8790,\n   \"samples\": [\n   \"s3943\",\n   \"s8378\",\n   \"s222\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"type\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"TV Show\",\n   \"Movie\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"title\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 8790,\n

```
\"samples\": [\n          \"Roman Empire: Reign of Blood\",\n
\"The King's Speech\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"director\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 4527,\n
\"samples\": [\n          \"Jatla Siddartha\",\n          \"Petra
Costa, Lea Glob\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cast\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 7679,\n        \"samples\": [\n          \"Fred
Ward, Chris Gartin, Helen Shaver, Michael Gross, Marcelo Tubert, Marco
Hernandez, Jos\\u00e9 Ram\\u00f3n Rosario, Thomas Rosales Jr.\",\n
\"Vanessa Paradis\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"country\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 749,\n
\"samples\": [\n          \"United States, United Kingdom, Denmark,
Sweden\",\n          \"Spain, France\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"date_added\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"2008-01-01 00:00:00\",\n        \"max\": \"2021-09-25 00:00:00\",\n
\"num_unique_values\": 1697,\n        \"samples\": [\n
\"2019-10-30 00:00:00\",\n          \"2021-05-19 00:00:00\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"release_year\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 8,\n        \"min\": 1925,\n
\"max\": 2021,\n        \"num_unique_values\": 74,\n
\"samples\": [\n          1996,\n          1969\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"rating\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
14,\n        \"samples\": [\n          \"G\",\n          \"NR\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"duration\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 220,\n        \"samples\": [\n          \"37
min\",\n          \"177 min\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"listed_in\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 513,\n
\"samples\": [\n          \"Crime TV Shows, International TV Shows, TV
Mysteries\",\n          \"Children & Family Movies, Classic Movies,
Dramas\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"description\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 8758,\n
\"samples\": [\n          \"Led by a trio of Korean celebs, a
multinational panel of men engage in \\u2013 usually \\u2013
lighthearted debates on issues that surround Korea and beyond.\",\n
\"Based on the audio diaries of academic and theologian John Hull,
this film offers an impressionistic re-creation of his slow descent
into blindness.\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

Q1. What is the distribution of content type (Movies vs. TV Shows)?

```
sns.countplot(x='type', data=df,  palette='Set2')
plt.title("Distribution of Content Type on Netflix")
plt.xlabel("Type")
plt.ylabel("Count")
plt.show()
```

```
<ipython-input-53-b9caf8cb1748>:1: 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 effect.

  sns.countplot(x='type', data=df,  palette='Set2')
```



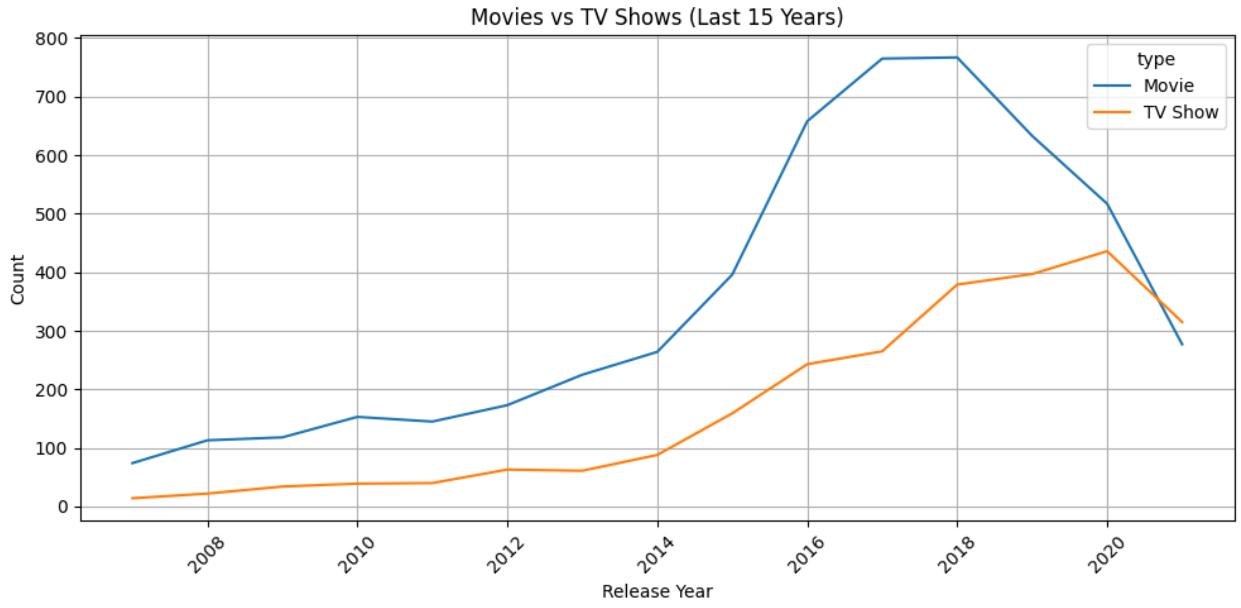
Insight: Netflix has added more movies than TV shows overall. However, if we look at the trend over the years, the number of TV shows being added has been increasing steadily.

**Q2. How has the number of Movies and TV Shows released on Netflix changed over the years? (Bivariate Analysis)

```
df_grouped = df.groupby(['release_year', 'type']).size().unstack()
df_grouped_recent = df_grouped.tail(15)
df_grouped_recent.plot(kind='line', figsize=(10, 5), color=['#1f77b4',
'#ff7f0e'])
plt.title("Movies vs TV Shows (Last 15 Years)")
plt.xlabel("Release Year")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

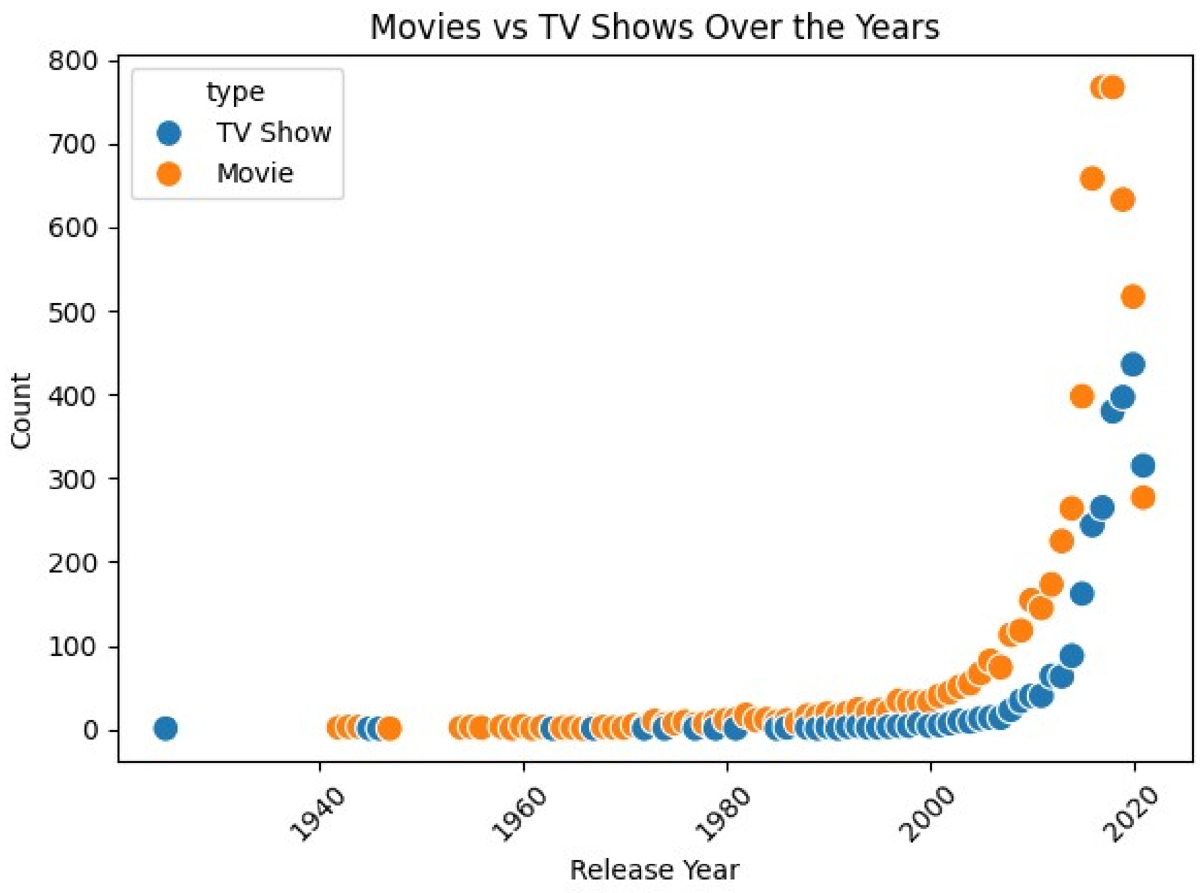
```
<ipython-input-54-917ee76a4d8c>:1: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
```

```
observed=True to adopt the future default and silence this warning.
  df_grouped = df.groupby(['release_year', 'type']).size().unstack()
```



Insights: Netflix initially focused on movies, but has shifted towards TV show production, especially since 2016 to 2018.

```
df_grouped = df.groupby(['release_year',
'type']).size().reset_index(name='count')
sns.scatterplot(data=df_grouped, x='release_year', y='count',
hue='type', palette=['#1f77b4', '#ff7f0e'], s=100)
plt.title("Movies vs TV Shows Over the Years")
plt.xlabel("Release Year")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

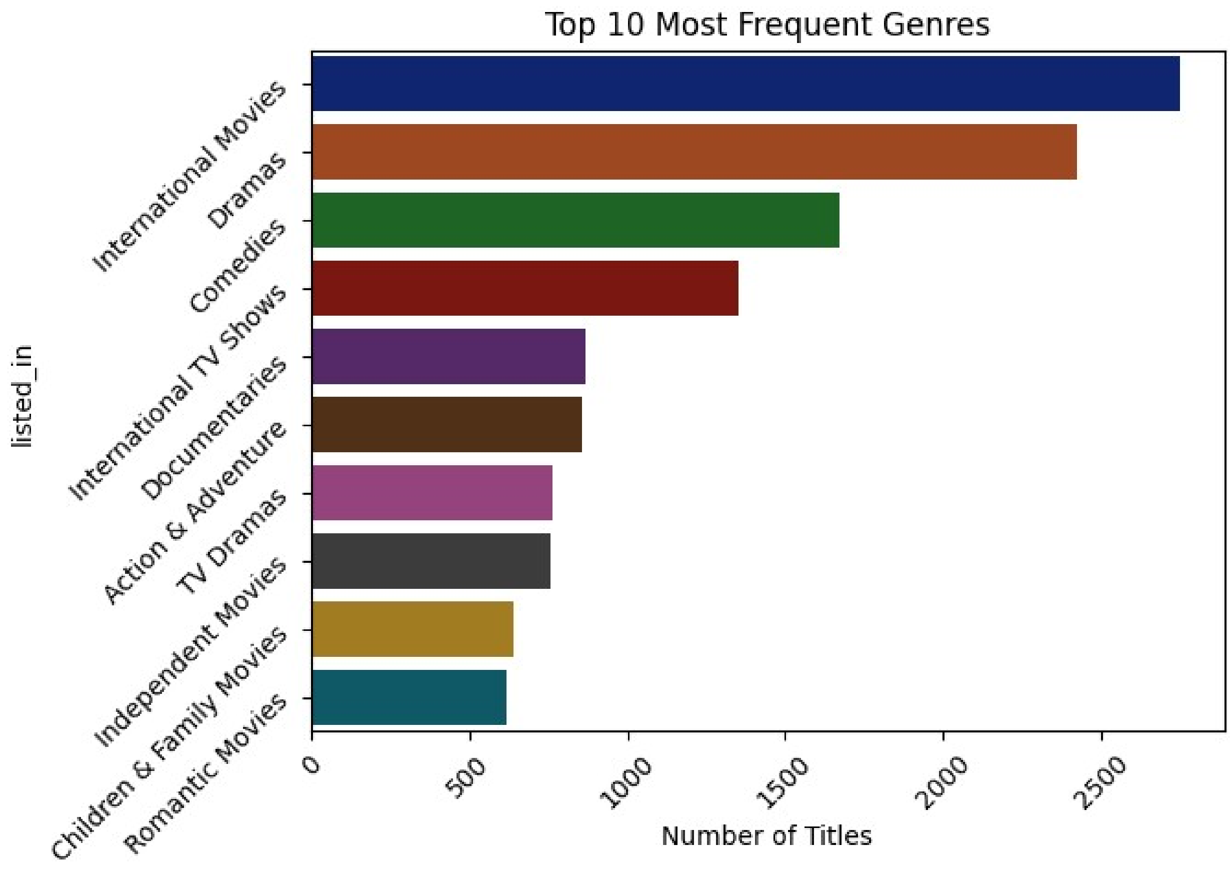


Q3. What are the Most Popular listed_in on Netflix? (Univariate - Categorical)

```
df_exploded_genre = df.assign(genre=df['listed_in'].str.split(',
')).explode('genre')

top_genres = df_exploded_genre['genre'].value_counts().head(10)
sns.barplot(y=top_genres.index, x=top_genres.values, palette='dark')
plt.title("Top 10 Most Frequent Genres")
plt.xlabel("Number of Titles")
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.ylabel("listed_in")
plt.show()
```

```
<ipython-input-56-acfb40ea99f6>:2: 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 effect.
```

```
sns.barplot(y=top_genres.index, x=top_genres.values, palette='dark')
```

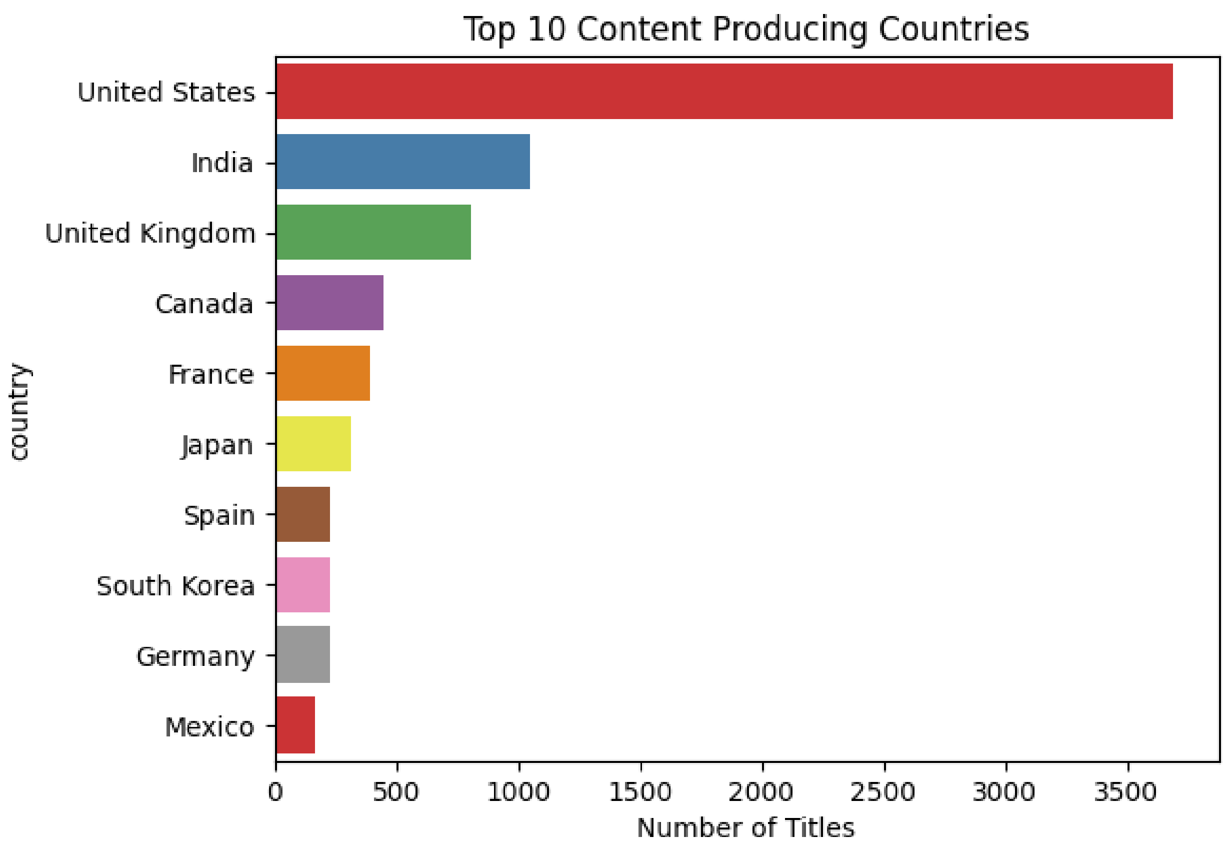


Insight: we can see in above chart Drama, International Movies, and Comedies are most common and popular.

4. Which Countries Produce the Most Content on Netflix?

```
df_exploded_country = df.assign(country=df['country'].str.split(',
')).explode('country')

top_countries = df_exploded_country['country'].value_counts().head(10)
sns.barplot(y=top_countries.index, x=top_countries.values,
palette='Set1')
plt.title("Top 10 Content Producing Countries")
plt.xlabel("Number of Titles")
plt.ylabel("country")
plt.show()
```

```
<ipython-input-57-f89f159c4486>:2: 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 effect.

  sns.barplot(y=top_countries.index, x=top_countries.values,
palette='Set1')
```

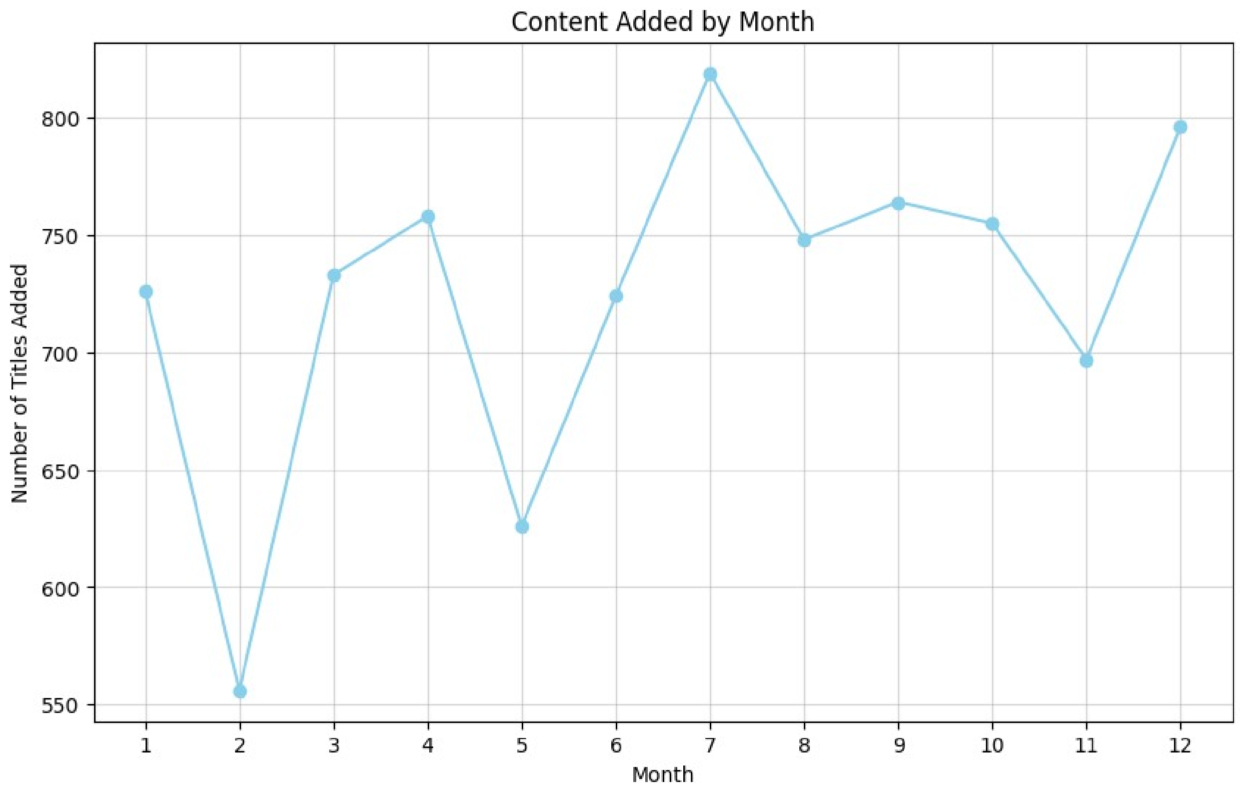


** Insight:** as per Chart and analysis USA, India and UK produced more content on Netflix.

When is the best time to launch content on Netflix?

```
df['month_added'] = df['date_added'].dt.month

month_counts = df['month_added'].value_counts().sort_index()

plt.figure(figsize=(10, 6))
plt.plot(month_counts.index, month_counts.values, marker='o',
linestyle='-', color='skyblue')
```

```
plt.title("Content Added by Month")
plt.xlabel("Month")
plt.ylabel("Number of Titles Added")
plt.xticks(month_counts.index)  # Show all months on x-axis
plt.grid(True, alpha=0.5)
plt.show()
```

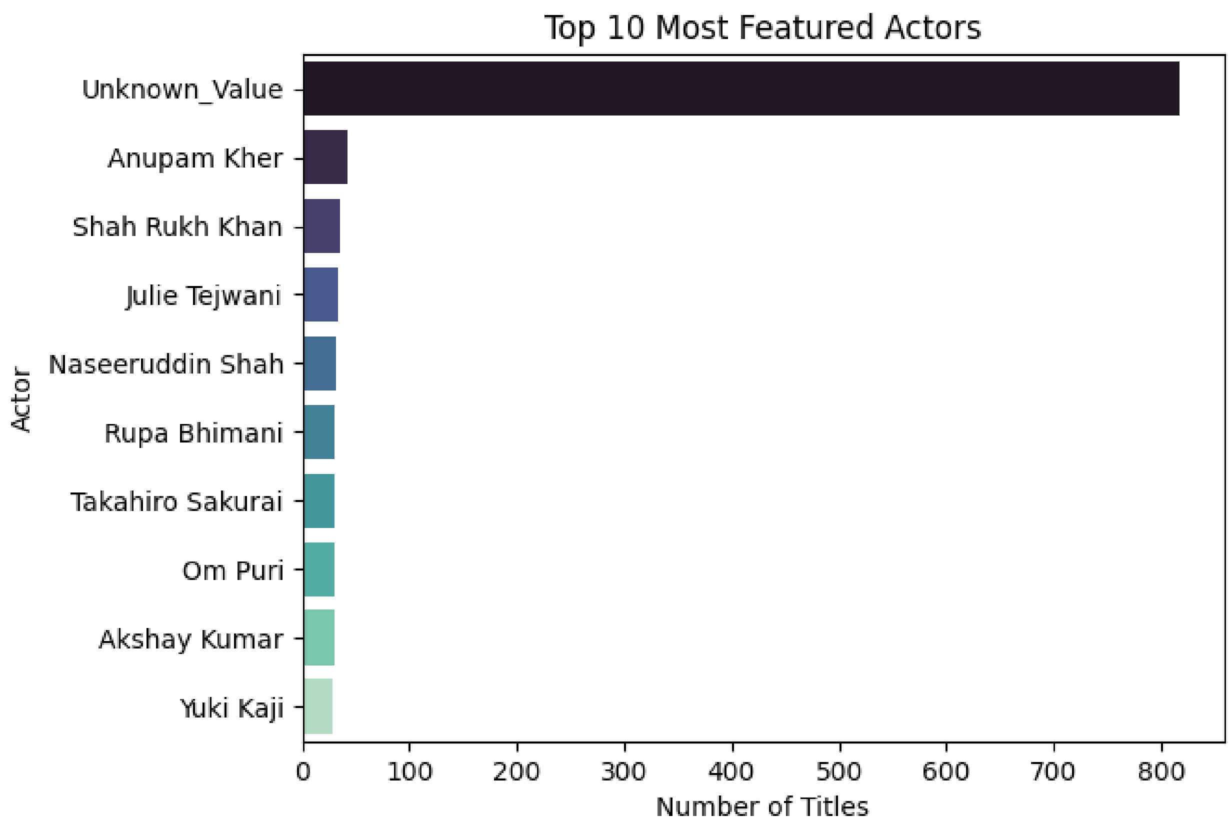


** Insight:** High releases in December, October, and September.**

Who are the most frequently featured actors on Netflix?

```
df_exploded_cast = df.assign(actor=df['cast'].str.split(',
')).explode('actor')

top_actors = df_exploded_cast['actor'].value_counts().head(10)
sns.barplot(y=top_actors.index, x=top_actors.values, palette='mako')
plt.title("Top 10 Most Featured Actors")
plt.xlabel("Number of Titles")
plt.ylabel("Actor")
plt.show()
```

```
<ipython-input-48-1d74befe259b>:2: 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 effect.

  sns.barplot(y=top_actors.index, x=top_actors.values, palette='mako')
```



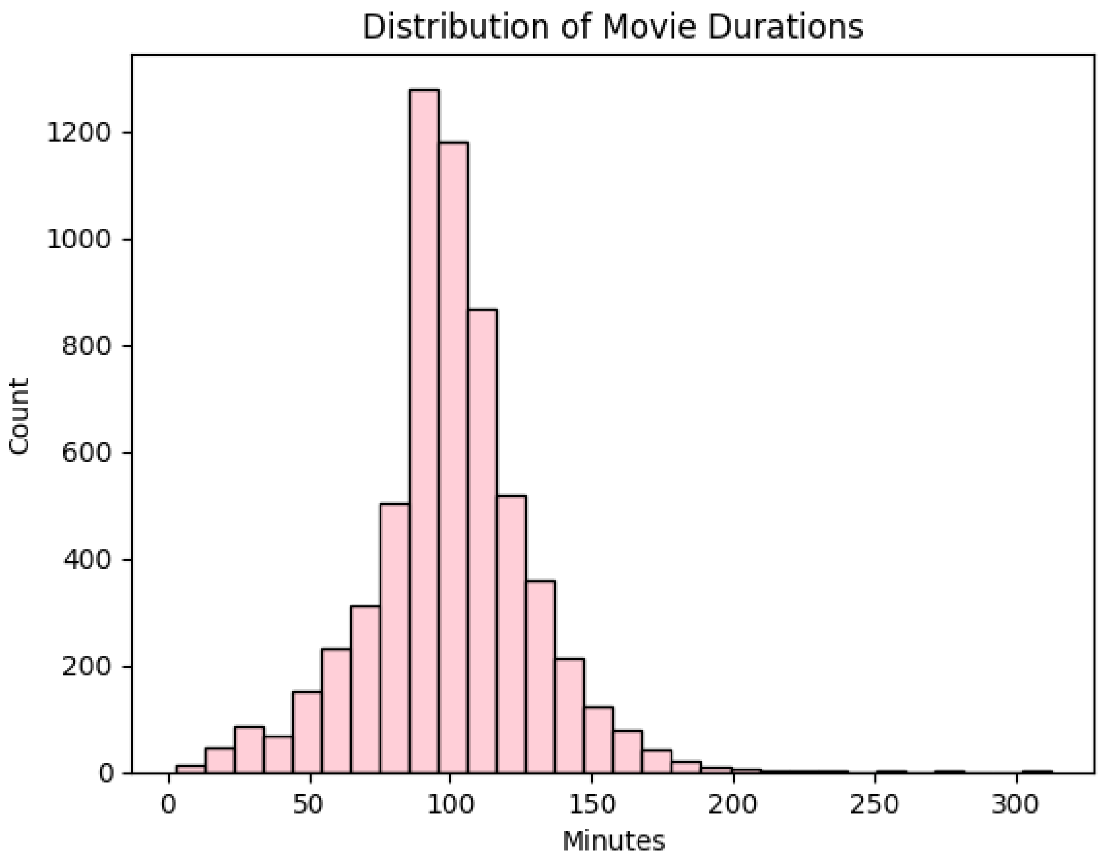
** Insight:** Indian and US actors are frequently cast.

What is the distribution of movie durations on Netflix?

```
df_movies = df[df['type'] == 'Movie']

df_movies['minutes'] = df_movies['duration'].str.extract('(\
d+)').astype(int)
```

```
<ipython-input-51-d32f7c0cf2ab>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_movies['minutes'] = df_movies['duration'].str.extract('(\
d+)').astype(int)

sns.histplot(df_movies['minutes'], bins=30, color='pink')
plt.title("Distribution of Movie Durations")
plt.xlabel("Minutes")
plt.show()
```

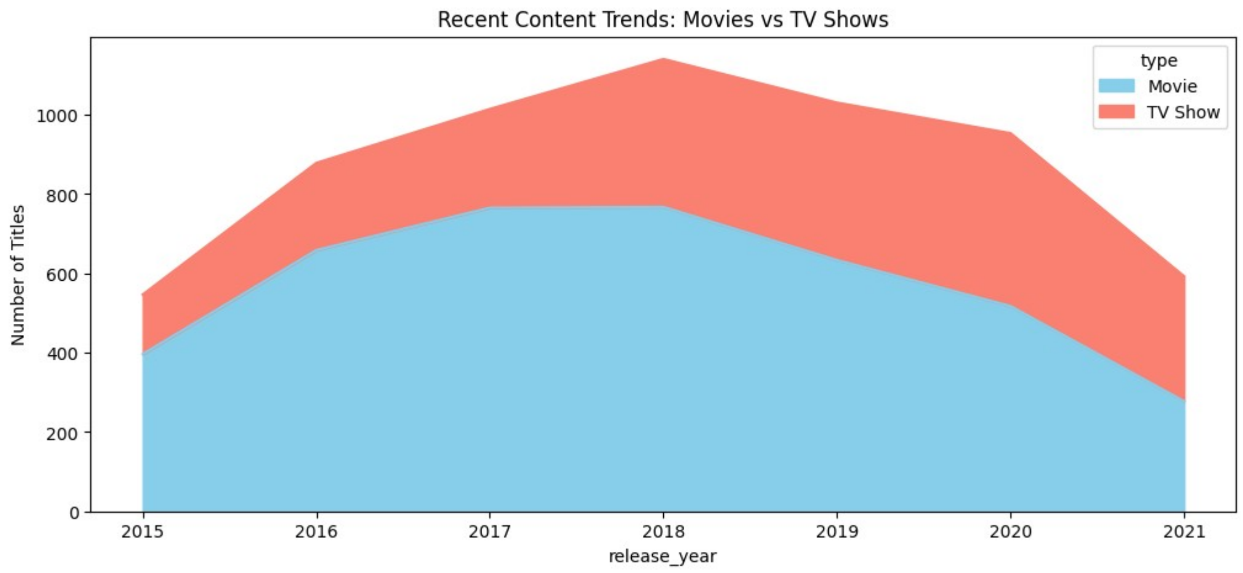


** Insight:** Most movies 80–120 mins. Outliers >200 mins.

Has Netflix shifted its focus towards TV Shows in recent years?

```
recent = df[df['release_year'] >= 2015]
recent_trend = recent.groupby(['release_year',
'type']).size().unstack()
recent_trend.plot(kind='area', figsize=(12,5), color=['skyblue',
'salmon'])
plt.title("Recent Content Trends: Movies vs TV Shows")
plt.ylabel("Number of Titles")
plt.show()
```

```
<ipython-input-59-ee04f122fa4f>:2: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  recent_trend = recent.groupby(['release_year',
'type']).size().unstack()
```



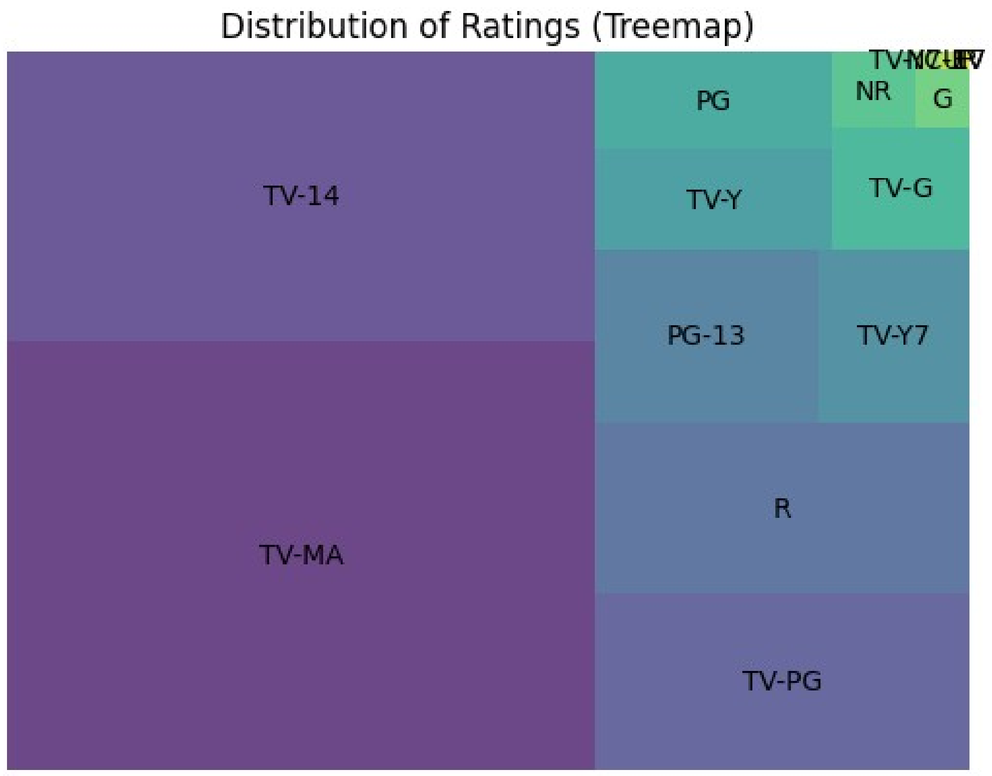
** Insight:** TV Shows surge from 2016. Strategy shift.

What is the distribution of content ratings on Netflix? ***

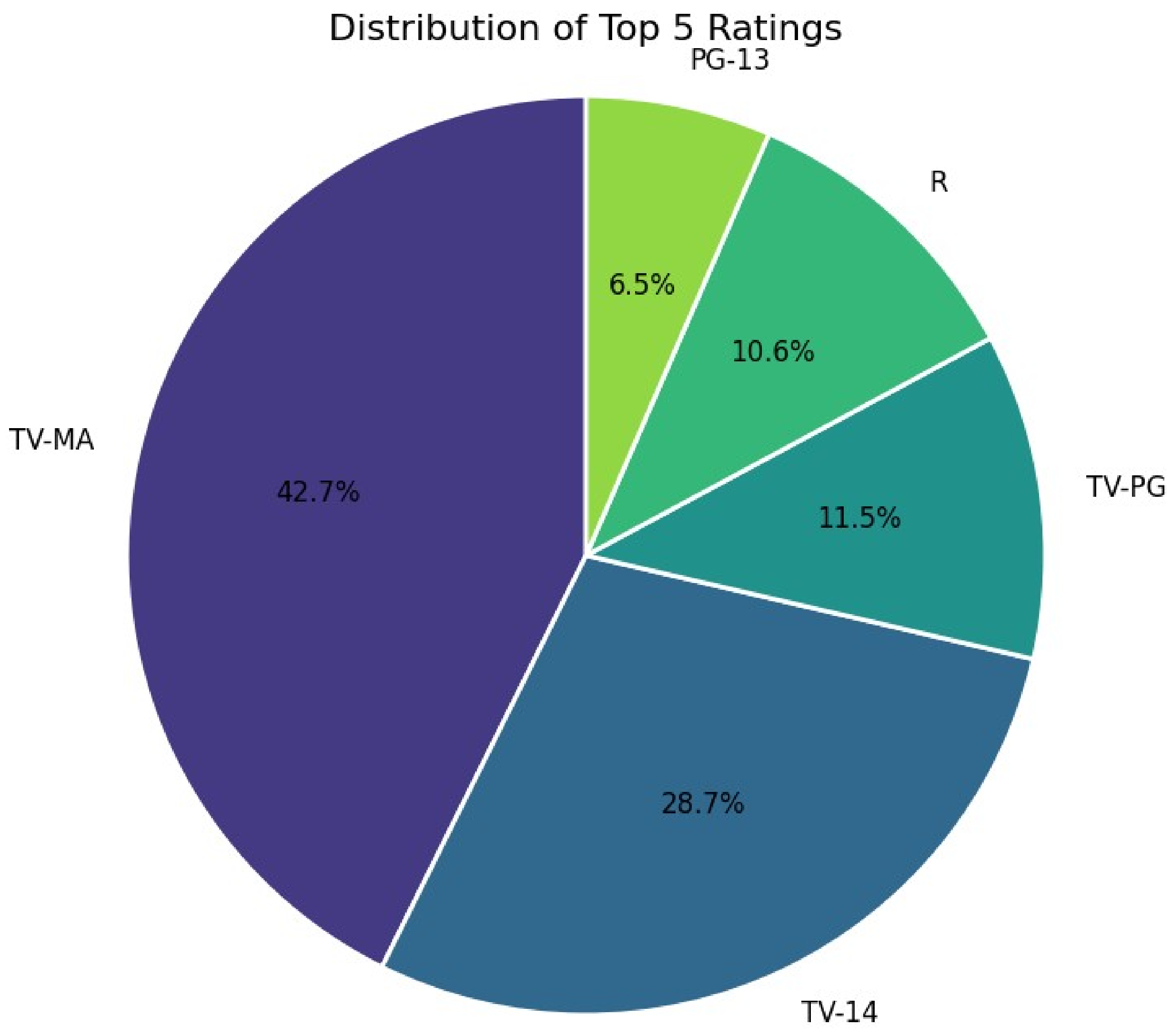
```
!pip install squarify
import squarify
```

```
Collecting squarify
  Downloading squarify-0.4.4-py3-none-any.whl.metadata (600 bytes)
Downloading squarify-0.4.4-py3-none-any.whl (4.1 kB)
Installing collected packages: squarify
Successfully installed squarify-0.4.4
```

```
rating_counts = df['rating'].value_counts()
labels = rating_counts.index.tolist()
sizes = rating_counts.values.tolist()

squarify.plot(sizes=sizes, label=labels, alpha=.8,
color=sns.color_palette('viridis', len(labels)))
plt.title("Distribution of Ratings (Treemap)")
plt.axis('off')
plt.show()
```



```
top_5_ratings = df['rating'].value_counts().head(5)

# Customize plot appearance
plt.figure(figsize=(8, 8))
plt.pie(top_5_ratings,
        labels=top_5_ratings.index,
        autopct='%1.1f%%',
        colors=sns.color_palette('viridis', len(top_5_ratings)),
        startangle=90,
        textprops={'fontsize': 12},
        wedgeprops={'linewidth': 2, 'edgecolor': 'white'})

plt.title("Distribution of Top 5 Ratings", fontsize=16)
plt.axis('equal')

plt.show()
```



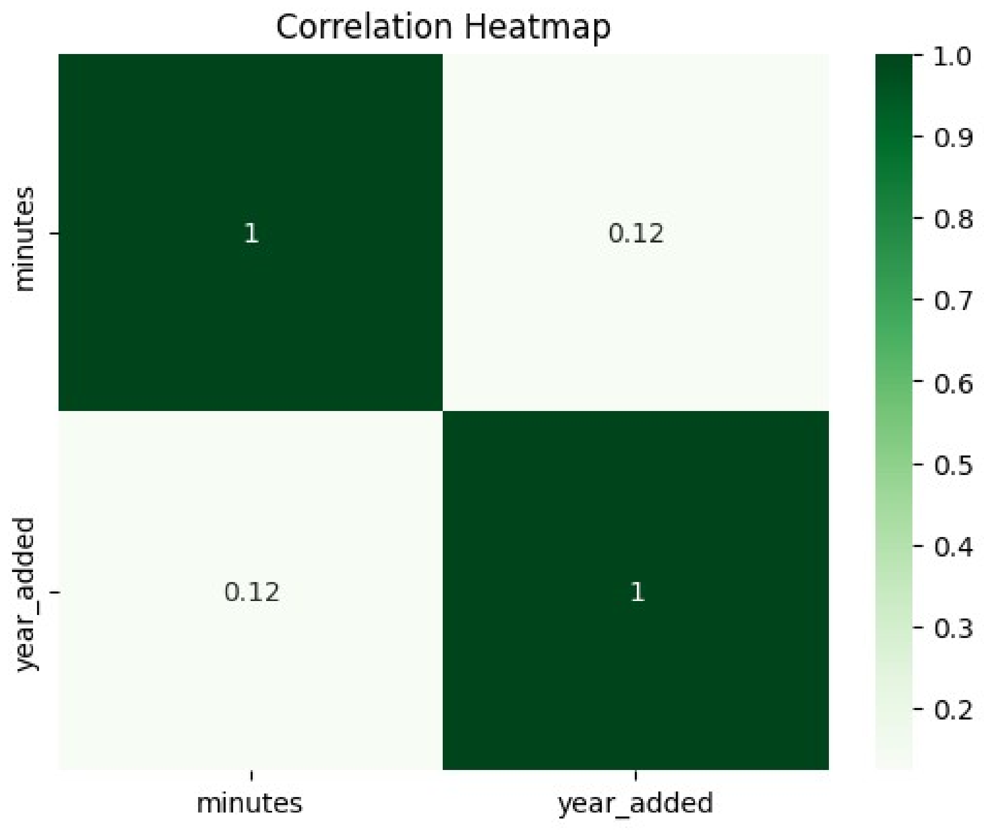
Insight:TV-MA and TV-14 are the most frequent ratings. These ratings target teens and adults. This suggests Netflix's content strategy focuses on these audiences.

What is the correlation between the duration of movies (in minutes) and the year they were added to Netflix?

```
df_movies = df[df['type'] == 'Movie']

df_movies['year_added'] = df_movies['date_added'].dt.year
```

```
<ipython-input-79-144c2e1a0b86>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_movies['year_added'] = df_movies['date_added'].dt.year

df_movies['minutes'] = df_movies['duration'].str.extract(r'(\
d+)').astype(float)

<ipython-input-86-33618ac21afd>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_movies['minutes'] = df_movies['duration'].str.extract(r'(\
d+)').astype(float)

sns.heatmap(df_movies[['minutes', 'year_added']].corr(), annot=True,
cmap='Greens')
plt.title("Correlation Heatmap")
plt.show()
```



** Insight:** No strong correlation. Feature engineering may be needed.

Director, cast, and country fields had missing values.

Business Insights**

=>Movies are still the most common, but TV Shows are growing fast and becoming more popular.

=>Drama and Comedy are the most popular genres worldwide, making them a good choice for future content.

=>December is a peak month for adding new content—possibly due to holidays and more viewers.

=>Most content comes from the US, India, and the UK, showing strong production from these countries and chances to grow through local partnerships.

=>A lot of content is made for adults, which means there's a good opportunity to create more shows and movies for kids.

Recommendations**

1. Increase production of TV Shows, especially in drama/comedy listed_in.
2. Launch major titles in Q4 (Sept–Dec) to capitalize on viewership peaks.
3. Promote actors with proven Netflix presence to attract loyal viewers.