```python
# for loading Netflix's image logo in this jupyter notebook:
from IPython.display import Image
Image("netflix_Eda_project_logo.jpg")
```

Exploratory-Data-Analysis (EDA) on Netflix Dataset:

```python
# IGNORE ALL WARNINGS FOR THIS ENTIRE DATA ANALYSIS PROJECT:
import warnings
warnings.filterwarnings('ignore')
```

Netflix Titles — Exploratory Data Analysis & Machine Learning

Project Overview

Netflix is one of the world's largest streaming platforms, hosting thousands of movies and TV shows from around the globe.
With such a massive content library, **data-driven insights** can help understand how Netflix has expanded its catalog, what types of content dominate the platform, and how patterns in genres, ratings, and countries have evolved over time.

In this Jupyter Notebook, we perform an **exploratory data analysis** on the **Netflix dataset**.
The dataset contains **two types of content**:

- i) *Movies*
- ii) *TV Shows*

Insights and Analysis Goals

In this notebook, we aim to explore and analyze the following key areas:

- **a)** Frequency of each content type
  (i.e., number of Movies vs. number of TV Shows)
- **b)** Number of releases by year
- **c)** Top 5 genres generating the highest traffic
- **d)** Ratings most commonly assigned to Netflix content
- **e)** Countries contributing to Netflix's content library
- **f)** Top 10 actors with the highest number of unique titles
- **g)** Correlation between a title's `release_year` and the year it was added to Netflix
- **h)** Predicting the content type *(Movie vs. TV Show)* using features such as `rating`, `release_year`, and `country`
- **i)** Clustering titles by description using **K-Means** to identify thematic groups
- **j)** Generating Word Clouds for:
  - j.1) *Directors*
  - j.2) *Actors*
  - j.3) *Titles*
- **k)** Checking how much contribution is made by these two entities:
  - k.1) *Directors*
  - k.2) *Genres*

This structured analysis combines exploratory insights and machine learning techniques to uncover patterns in Netflix's content library and provide meaningful business and data-driven insights.

LOAD THE LIBRARIES TO BE USED IN THIS JUPYTER NOTEBOOK:

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


df=pd.read_csv("netflix_titles.csv")
```

Preview of the orignal dataset :

```
df.head(10)
```

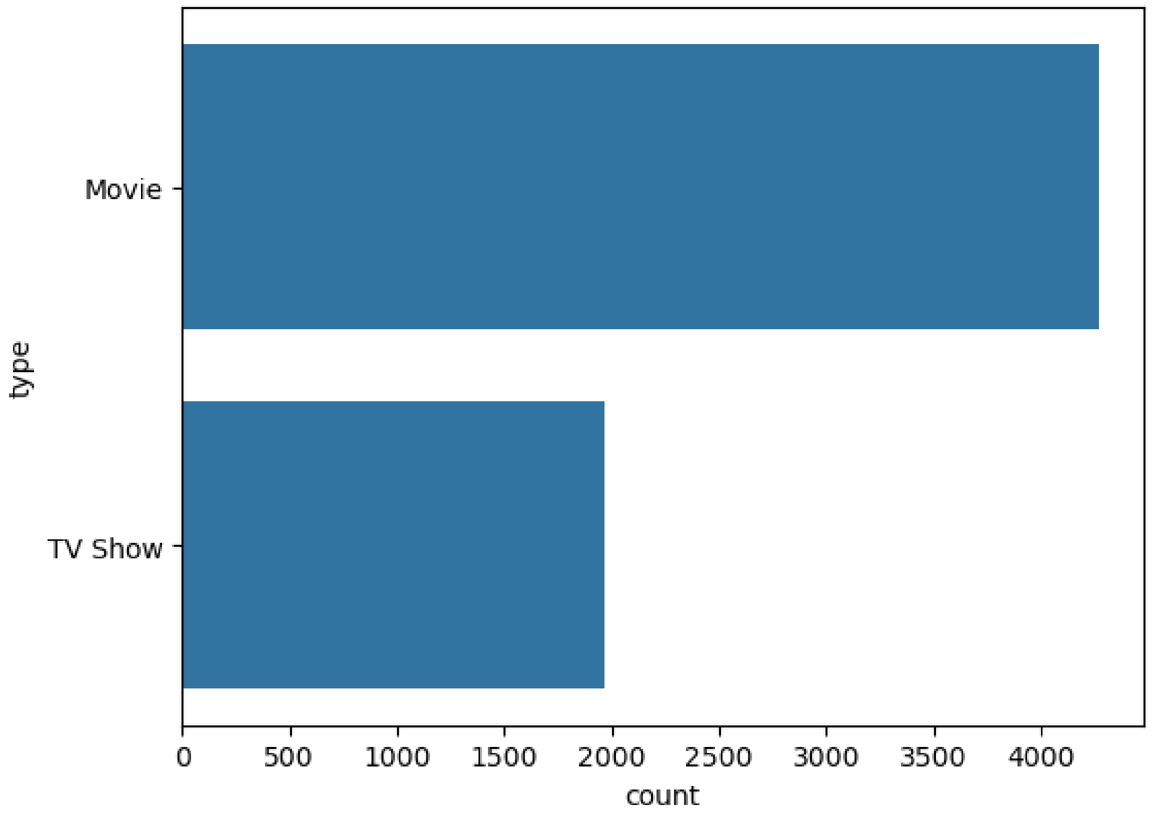
```
    show_id     type                                    title  \
0  81145628    Movie  Norm of the North: King Sized Adventure
1  80117401    Movie               Jandino: Whatever it Takes
2  70234439  TV Show                       Transformers Prime
3  80058654  TV Show         Transformers: Robots in Disguise
4  80125979    Movie                             #realityhigh
5  80163890  TV Show                                  Apaches
6  70304989    Movie                                 Automata
7  80164077    Movie        Fabrizio Copano: Solo pienso en mi
8  80117902  TV Show                             Fire Chasers
9  70304990    Movie                              Good People

                            director  \
0         Richard Finn, Tim Maltby
1                              NaN
2                              NaN
3                              NaN
4                 Fernando Lebrija
5                              NaN
6                     Gabe Ibáñez
7  Rodrigo Toro, Francisco Schultz
8                              NaN
9                Henrik Ruben Genz

                                                cast  \
0  Alan Marriott, Andrew Toth, Brian Dobson, Cole...
1                                   Jandino Asporaat
2  Peter Cullen, Sumalee Montano, Frank Welker, J...
```

```
3  Will Friedle, Darren Criss, Constance Zimmer, ...
4  Nesta Cooper, Kate Walsh, John Michael Higgins...
5  Alberto Ammann, Eloy Azorín, Verónica Echegui,...
6  Antonio Banderas, Dylan McDermott, Melanie Gri...
7                                   Fabrizio Copano
8                                               NaN
9  James Franco, Kate Hudson, Tom Wilkinson, Omar...

                                          country
date_added  \
0        United States, India, South Korea, China  September 9, 2019

1                                  United Kingdom  September 9, 2016

2                                   United States  September 8, 2018

3                                   United States  September 8, 2018

4                                   United States  September 8, 2017

5                                           Spain  September 8, 2017

6          Bulgaria, United States, Spain, Canada  September 8, 2017

7                                           Chile  September 8, 2017

8                                   United States  September 8, 2017

9  United States, United Kingdom, Denmark, Sweden  September 8, 2017


   release_year    rating  duration  \
0          2019     TV-PG    90 min
1          2016     TV-MA    94 min
2          2013  TV-Y7-FV  1 Season
3          2016     TV-Y7  1 Season
4          2017     TV-14    99 min
5          2016     TV-MA  1 Season
6          2014         R   110 min
7          2017     TV-MA    60 min
8          2017     TV-MA  1 Season
9          2014         R    90 min

                                           listed_in  \
0                  Children & Family Movies, Comedies
1                                     Stand-Up Comedy
2                                            Kids' TV
3                                            Kids' TV
4                                            Comedies
5  Crime TV Shows, International TV Shows, Spanis...
```

```
6  International Movies, Sci-Fi & Fantasy, Thrillers
7                                    Stand-Up Comedy
8                        Docuseries, Science & Nature TV
9                          Action & Adventure, Thrillers

                                         description
0  Before planning an awesome wedding for his gra...
1  Jandino Asporaat riffs on the challenges of ra...
2  With the help of three human allies, the Autob...
3  When a prison ship crash unleashes hundreds of...
4  When nerdy high schooler Dani finally attracts...
5  A young journalist is forced into a life of cr...
6  In a dystopian future, an insurance adjuster f...
7  Fabrizio Copano takes audience participation t...
8  As California's 2016 fire season rages, brave ...
9  A struggling couple can't believe their luck w...
```

PART 1 OF DATA VISUALIZATION:

```
#q1) movies vs tv-shows count:
sns.countplot(df["type"])
plt.show()
```



Movies vs TV Shows — Platform Composition

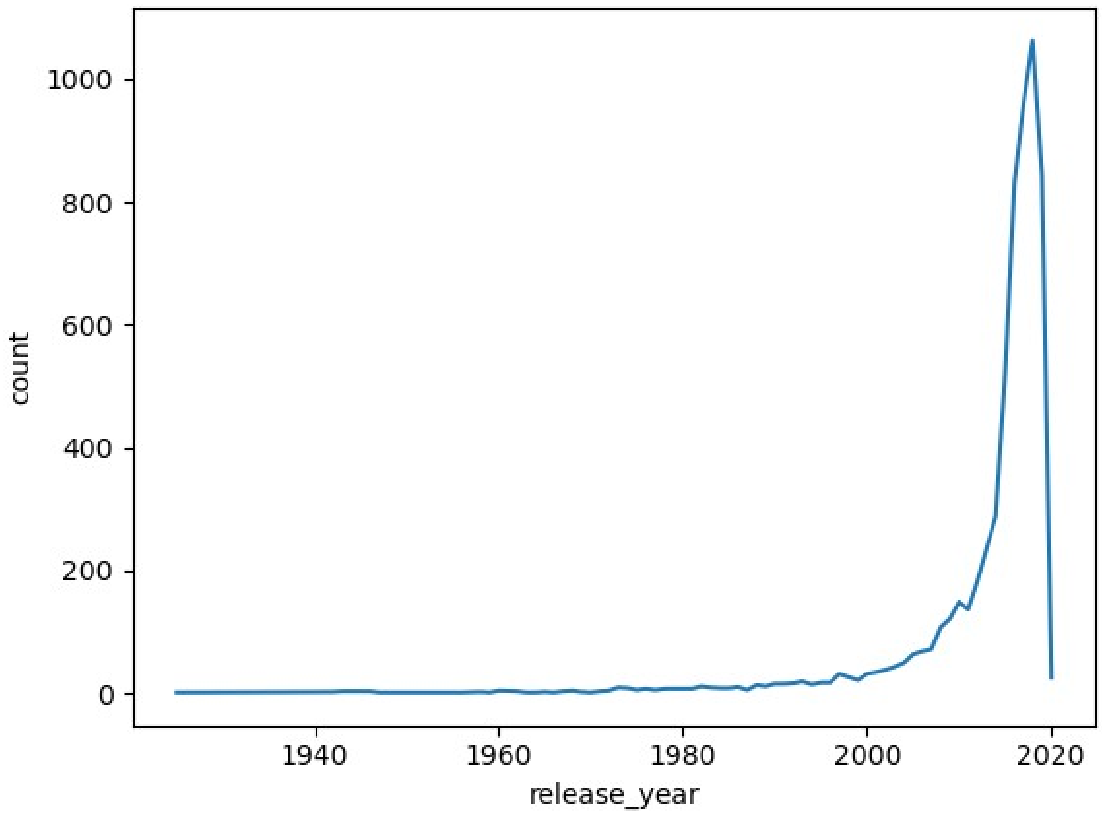
This visualization shows the **distribution of content types** on Netflix.

** Insight:**
Movies dominate the platform significantly compared to TV Shows.
This aligns with Netflix's early content strategy of focusing heavily on movies, and also shows where future investment may shift.

This information can help:

- Content teams plan future production focus.
- Recommendation engines adjust weightage for movies vs shows.

```
#q2) yearly addiiton trend:
data=df["release_year"].value_counts().reset_index()
sns.lineplot(x="release_year",y="count",data=data)
plt.show()
```



Yearly Release Trend

By analyzing `release_year`, we can see how the volume of Netflix content has changed over time.

Insight:

- There’s a clear **increase in content production and acquisition after 2015**, which reflects Netflix’s rapid global expansion.

- This insight can be tied to Netflix Originals growth and licensing strategies.

This trend is useful for forecasting future content growth and understanding how streaming trends have evolved.

DATA CLEANING:

up to this point, we have noticed that some columns contain multiple values

for some or most of the rows, and there are also missing values in the dataset.

now we will begin data cleaning by performing the following steps:

- **1)** Splitting the `cast` column so that each row contains only one value for `cast`.

- **2)** Splitting the `country` column so that each row contains only one value for `country`.

- **3)** Splitting the `listed_in` (genres) column so that each row contains only one value for `listed_in`.

- **4)** Splitting the `director` column so that each row contains only one value for `director`.

```
df["cast"]=df["cast"].str.split(',')
df=df.explode("cast").reset_index(drop=True)
df.head()
```

```
    show_id   type                                      title  \
0  81145628  Movie  Norm of the North: King Sized Adventure
1  81145628  Movie  Norm of the North: King Sized Adventure
2  81145628  Movie  Norm of the North: King Sized Adventure
3  81145628  Movie  Norm of the North: King Sized Adventure
4  81145628  Movie  Norm of the North: King Sized Adventure

                  director             cast  \
0  Richard Finn, Tim Maltby   Alan Marriott
1  Richard Finn, Tim Maltby     Andrew Toth
2  Richard Finn, Tim Maltby    Brian Dobson
3  Richard Finn, Tim Maltby     Cole Howard
```

```
4  Richard Finn, Tim Maltby   Jennifer Cameron

                                    country         date_added
release_year  \
0  United States, India, South Korea, China  September 9, 2019
2019
1  United States, India, South Korea, China  September 9, 2019
2019
2  United States, India, South Korea, China  September 9, 2019
2019
3  United States, India, South Korea, China  September 9, 2019
2019
4  United States, India, South Korea, China  September 9, 2019
2019

  rating duration                           listed_in  \
0  TV-PG   90 min  Children & Family Movies, Comedies
1  TV-PG   90 min  Children & Family Movies, Comedies
2  TV-PG   90 min  Children & Family Movies, Comedies
3  TV-PG   90 min  Children & Family Movies, Comedies
4  TV-PG   90 min  Children & Family Movies, Comedies

                                         description
0  Before planning an awesome wedding for his gra...
1  Before planning an awesome wedding for his gra...
2  Before planning an awesome wedding for his gra...
3  Before planning an awesome wedding for his gra...
4  Before planning an awesome wedding for his gra...
```

```
df["country"]=df["country"].str.split(',')
df=df.explode("country").reset_index(drop=True)
df.head()
```

```
    show_id   type                                  title  \
0  81145628  Movie  Norm of the North: King Sized Adventure
1  81145628  Movie  Norm of the North: King Sized Adventure
2  81145628  Movie  Norm of the North: King Sized Adventure
3  81145628  Movie  Norm of the North: King Sized Adventure
4  81145628  Movie  Norm of the North: King Sized Adventure

                  director           cast        country
date_added  \
0  Richard Finn, Tim Maltby  Alan Marriott  United States  September
9, 2019
1  Richard Finn, Tim Maltby  Alan Marriott          India  September
9, 2019
2  Richard Finn, Tim Maltby  Alan Marriott    South Korea  September
9, 2019
3  Richard Finn, Tim Maltby  Alan Marriott          China  September
9, 2019
```

```
4  Richard Finn, Tim Maltby    Andrew Toth  United States  September
9, 2019

   release_year rating duration                           listed_in  \
0          2019  TV-PG   90 min  Children & Family Movies, Comedies
1          2019  TV-PG   90 min  Children & Family Movies, Comedies
2          2019  TV-PG   90 min  Children & Family Movies, Comedies
3          2019  TV-PG   90 min  Children & Family Movies, Comedies
4          2019  TV-PG   90 min  Children & Family Movies, Comedies

                                         description
0  Before planning an awesome wedding for his gra...
1  Before planning an awesome wedding for his gra...
2  Before planning an awesome wedding for his gra...
3  Before planning an awesome wedding for his gra...
4  Before planning an awesome wedding for his gra...
```

```
df["listed_in"]=df["listed_in"].str.split(',')
df=df.explode("listed_in").reset_index(drop=True)
df.head()
```

```
    show_id   type                                  title  \
0  81145628  Movie  Norm of the North: King Sized Adventure
1  81145628  Movie  Norm of the North: King Sized Adventure
2  81145628  Movie  Norm of the North: King Sized Adventure
3  81145628  Movie  Norm of the North: King Sized Adventure
4  81145628  Movie  Norm of the North: King Sized Adventure

                  director           cast        country
date_added  \
0  Richard Finn, Tim Maltby  Alan Marriott  United States  September
9, 2019
1  Richard Finn, Tim Maltby  Alan Marriott  United States  September
9, 2019
2  Richard Finn, Tim Maltby  Alan Marriott          India  September
9, 2019
3  Richard Finn, Tim Maltby  Alan Marriott          India  September
9, 2019
4  Richard Finn, Tim Maltby  Alan Marriott    South Korea  September
9, 2019

   release_year rating duration                 listed_in  \
0          2019  TV-PG   90 min  Children & Family Movies
1          2019  TV-PG   90 min                  Comedies
2          2019  TV-PG   90 min  Children & Family Movies
3          2019  TV-PG   90 min                  Comedies
4          2019  TV-PG   90 min  Children & Family Movies

                                         description
0  Before planning an awesome wedding for his gra...
```

```
1  Before planning an awesome wedding for his gra...
2  Before planning an awesome wedding for his gra...
3  Before planning an awesome wedding for his gra...
4  Before planning an awesome wedding for his gra...
```

```
df["director"]=df["director"].str.split(',')
df=df.explode("director").reset_index(drop=True)
df.head()
```

```
    show_id   type                                    title  
director  \
0  81145628  Movie  Norm of the North: King Sized Adventure  Richard 
Finn   
1  81145628  Movie  Norm of the North: King Sized Adventure    Tim 
Maltby   
2  81145628  Movie  Norm of the North: King Sized Adventure  Richard 
Finn   
3  81145628  Movie  Norm of the North: King Sized Adventure    Tim 
Maltby   
4  81145628  Movie  Norm of the North: King Sized Adventure  Richard 
Finn   

            cast        country         date_added  release_year 
rating  \
0  Alan Marriott  United States  September 9, 2019          2019  TV-
PG   
1  Alan Marriott  United States  September 9, 2019          2019  TV-
PG   
2  Alan Marriott  United States  September 9, 2019          2019  TV-
PG   
3  Alan Marriott  United States  September 9, 2019          2019  TV-
PG   
4  Alan Marriott          India  September 9, 2019          2019  TV-
PG   

  duration                 listed_in  \
0   90 min  Children & Family Movies   
1   90 min  Children & Family Movies   
2   90 min                  Comedies   
3   90 min                  Comedies   
4   90 min  Children & Family Movies   

                                         description  
0  Before planning an awesome wedding for his gra...  
1  Before planning an awesome wedding for his gra...  
2  Before planning an awesome wedding for his gra...  
3  Before planning an awesome wedding for his gra...  
4  Before planning an awesome wedding for his gra...  
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 139984 entries, 0 to 139983
Data columns (total 12 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   show_id       139984 non-null  int64 
 1   type          139984 non-null  object
 2   title         139984 non-null  object
 3   director      101073 non-null  object
 4   cast          138493 non-null  object
 5   country       134194 non-null  object
 6   date_added    139825 non-null  object
 7   release_year  139984 non-null  int64 
 8   rating        139911 non-null  object
 9   duration      139984 non-null  object
 10  listed_in     139984 non-null  object
 11  description   139984 non-null  object
dtypes: int64(2), object(10)
memory usage: 12.8+ MB
```

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

df['rating'].fillna('No Rating', inplace=True)

df = df.dropna(subset="date_added")


df["date_added"] = df["date_added"].str.strip()  # Remove
leading/trailing spaces
df["date_added"] = pd.to_datetime(df['date_added'])
df['day_added'] = df['date_added'].dt.day
df['year_added'] = df['date_added'].dt.year
df['month_added']=df['date_added'].dt.month
df['year_added'].astype(int)
df['day_added'].astype(int)
```

```
0         9
1         9
2         9
3         9
4         9
         ..
139820    1
139821    1
139822    1
139823    1
139824    1
Name: day_added, Length: 139825, dtype: int64
```

handling missing values across key columns

before performing any analysis or building models, it is essential to ensure data consistency and completeness.
several columns in this dataset contain missing values, which, if left untreated, could lead to skewed insights or errors during processing.

columns cleaned in this step:

- **`director`**:
  many titles do not have a director listed. instead of dropping these rows, we fill the missing values with `'Unknown'`.
  this ensures we keep valuable records (e.g., documentaries or shows without a named director) for further analysis and visualization.

- **`cast`**:
  the cast is an important feature for understanding content popularity, actor presence, and collaborations.
  missing values are replaced with `'Unknown'` to avoid data loss and keep counts consistent in later steps (e.g., top actors analysis).

- **`country`**:
  country information is crucial for analyzing netflix's global content distribution.
  rather than deleting these records, we assign `'Unknown'` so the content still contributes to overall statistics without breaking aggregations.

- **`rating`**:
  ratings tell us the target audience for each title.
  missing ratings are filled with `'No Rating'` as a placeholder, ensuring they remain part of the dataset for distribution and modeling.

- **`date_added`**:
  this column shows when the title was added to netflix.
  we drop rows where this column is completely missing since they cannot contribute to **time-based trend analysis**.
  for the remaining rows:

  - leading/trailing spaces are removed,

  - dates are converted into proper `datetime` format,

  - and separate **day**, **month**, and **year** features are extracted.
    this allows us to analyze patterns over time — for example, identifying which years or months had the highest content additions.

---

why this matters:
cleaning these columns ensures:

- more accurate visualizations,
- no loss of useful records unnecessarily,
- better handling of categorical values in machine learning models,
- and richer insights from time-based trends.

this cleaning process transforms messy data into structured and reliable information — a crucial foundation for meaningful eda and modeling.

```
df.info()
```

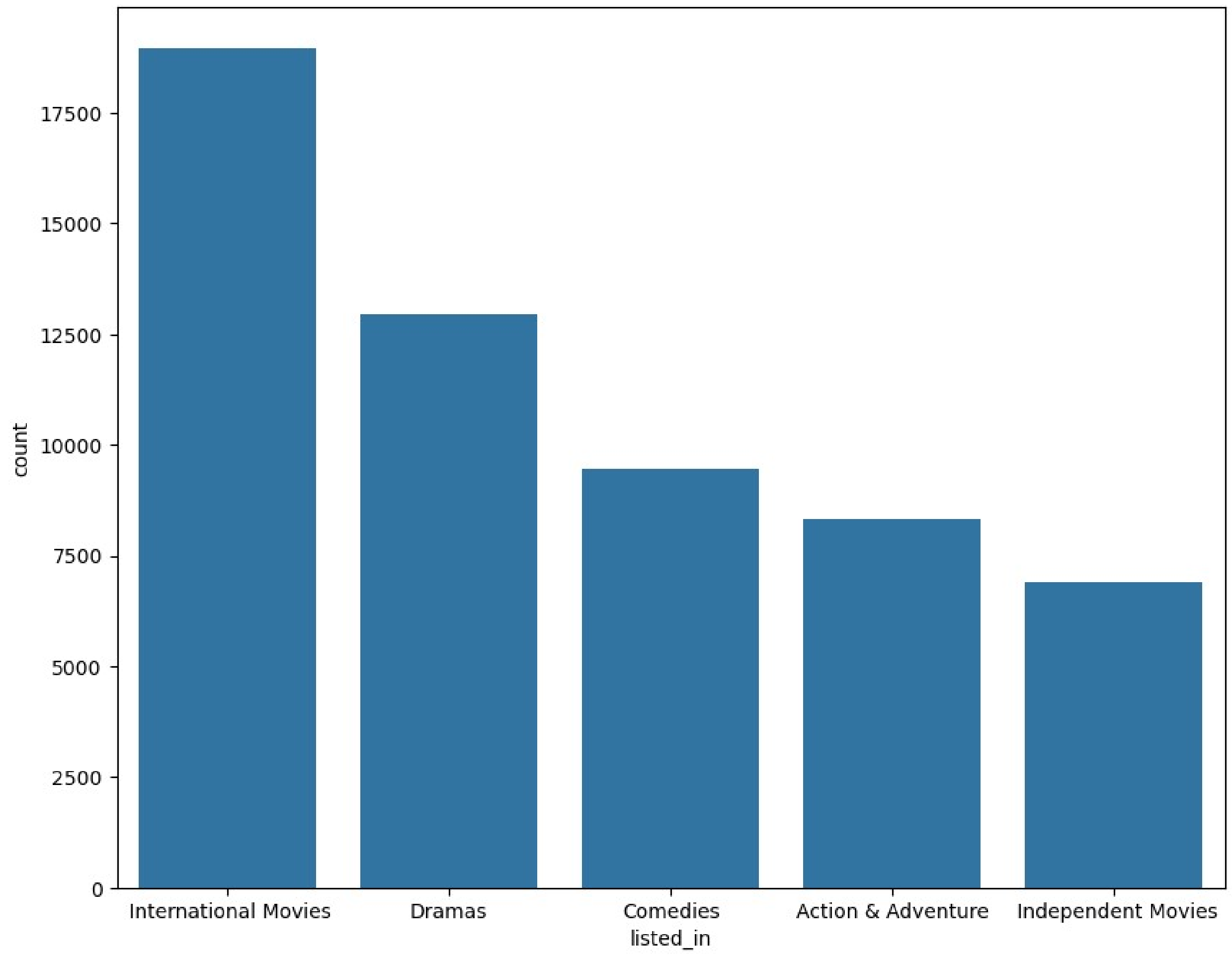
```
<class 'pandas.core.frame.DataFrame'>
Index: 139825 entries, 0 to 139824
Data columns (total 15 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   show_id       139825 non-null  int64
 1   type          139825 non-null  object
 2   title         139825 non-null  object
 3   director      139825 non-null  object
 4   cast          139825 non-null  object
 5   country       139825 non-null  object
 6   date_added    139825 non-null  datetime64[ns]
 7   release_year  139825 non-null  int64
 8   rating        139825 non-null  object
 9   duration      139825 non-null  object
 10  listed_in     139825 non-null  object
 11  description   139825 non-null  object
 12  day_added     139825 non-null  int32
 13  year_added    139825 non-null  int32
 14  month_added   139825 non-null  int32
dtypes: datetime64[ns](1), int32(3), int64(2), object(9)
memory usage: 15.5+ MB
```

NOW EXPORT THIS CLEANED FILE:

```
df.to_csv('cleaned_throughPython_netflix_titles.csv', index=False)
```

PART 2 OF DATA VISUALIZATION:

```
#q3) top genres:
data=df["listed_in"].value_counts().reset_index()
data

                     listed_in  count
0         International Movies  18961
1                       Dramas  12934
2                     Comedies   9469
3           Action & Adventure   8331
4           Independent Movies   6917
..                         ...    ...
67    Spanish-Language TV Shows     20
68              Stand-Up Comedy     18
69              Romantic Movies     16
70          TV Sci-Fi & Fantasy      7
71                Sports Movies      3

[72 rows x 2 columns]

data.head(5)

               listed_in  count
0   International Movies  18961
1                 Dramas  12934
2               Comedies   9469
3     Action & Adventure   8331
4     Independent Movies   6917

plt.figure(figsize=(10,8))
sns.barplot(x="listed_in",y="count",data=data.head(5))
plt.show()
```

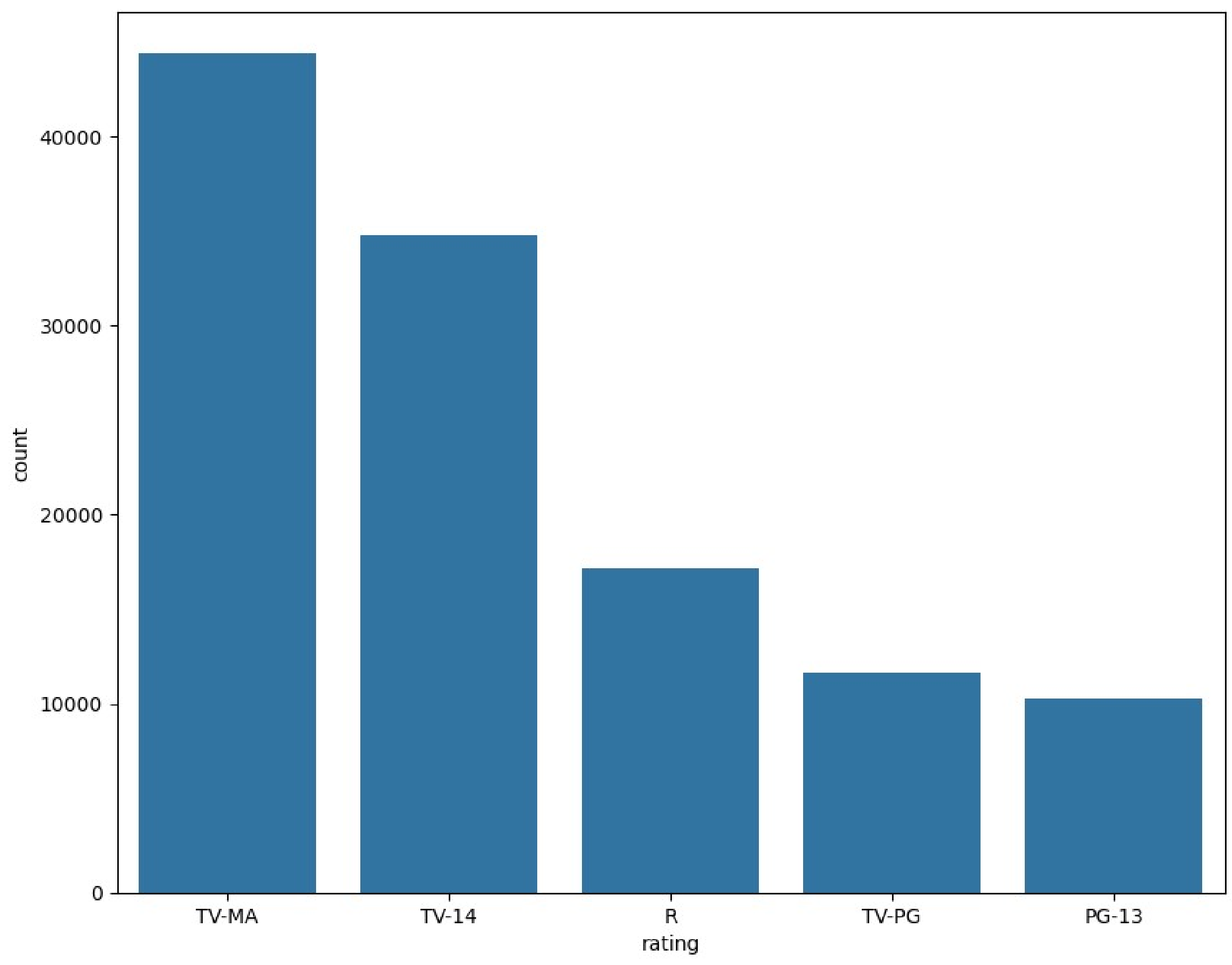


Conclusion till here:

The top 3 performing genres on Netflix are:

- **a)** International Movies
- **b)** Dramas
- **c)** Comedies

```
#q4) ratings distribution:
data=df["rating"].value_counts().reset_index()
plt.figure(figsize=(10,8))
sns.barplot(x="rating",y="count",data=data.head(5))
plt.show()
```



Conclusion till here:

The top 3 ratings are:

- **a)** Mature Audience Only (i.e. suitable for adults and possibly unsuitable for children under 17)
- **b)** Parents Strongly Cautioned (for children under 14)
- **c)** Restricted (anyone under the age of 17 must be accompanied by a parent or an adult guardian)

```
# q5) country analysis:
import plotly.express as px
country_counts = df["country"].value_counts().reset_index()
country_counts.columns = ["country", "count"]

# Plot world map
fig = px.choropleth(
    country_counts,
    locations="country",
    locationmode="country names",  # matches country column with world
```

```
map names
    color="count",  # intensity based on occurrence
    color_continuous_scale="Viridis",  # you can try "Plasma",
"Blues", etc.
    title="Country Occurrences on World Map"
)

fig.show()
```

{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"coloraxis":"coloraxis","geo":"geo","hovertemplate":"country=%
{location}<br>count=%{z}<extra></extra>","locationmode":"country
names","locations":["United States","India","United Kingdom"," United
States","Unknown","Japan","Canada","Spain","South Korea","France","
France"," United Kingdom","Mexico"," Canada","Turkey","
Germany","China","Taiwan","Hong
Kong","Australia","Thailand","Germany"," Belgium","
Japan","Argentina","Indonesia","Philippines","Egypt","Brazil","
China","Nigeria","Colombia"," Spain"," Mexico"," Australia","Italy","
Hong Kong"," India","Denmark","
Sweden","Pakistan","Israel","Poland","Chile","Ireland","
Italy","Belgium"," South
Korea","Bangladesh","Norway","Malaysia","Singapore","South Africa","
Colombia"," Brazil"," Singapore","United Arab Emirates","
Netherlands"," Qatar"," South Africa","Russia","Netherlands","
Argentina","Sweden"," Denmark"," United Arab Emirates","New Zealand","
New Zealand"," Switzerland"," Ireland"," Poland"," Lebanon","
Luxembourg"," Czech Republic"," Norway","Switzerland","
Chile","Lebanon"," Jordan","Bulgaria"," Greece","Romania","Finland","
Morocco"," West Germany","Iceland"," Soviet Union","
Portugal","Peru"," Iran"," Turkey","Austria"," Malaysia","Czech
Republic"," Austria"," Peru","Vietnam","Georgia","Uruguay"," Egypt","
Hungary"," Uruguay","Paraguay","Ghana","
Indonesia","Hungary","Ukraine"," Taiwan","Soviet Union","
Nepal","Croatia"," Slovenia","Serbia"," Pakistan"," Iceland","
Serbia","Portugal"," Bulgaria"," Russia","Saudi Arabia"," Malta","
Finland"," Azerbaijan",""," East Germany"," Romania"," Senegal","
Thailand"," Liechtenstein"," Bangladesh","Cambodia"," Malawi","
Syria"," Israel"," Kuwait"," Croatia","Iran"," Latvia"," Cambodia","
Montenegro"," Cayman Islands"," Slovakia","Mauritius"," Dominican
Republic"," Cuba","Guatemala"," Bermuda","
Ecuador","Cyprus","Somalia"," Sudan"," Kenya"," Albania"," Iraq","
Zimbabwe","Venezuela","West Germany"," Venezuela","Slovenia"," Vatican
City"," Ukraine","Dominican Republic"," Samoa"," Botswana"," Panama","
Sri Lanka"," Armenia"," Afghanistan"," Namibia"," Philippines","
Mongolia"," Uganda"," Kazakhstan","
Nicaragua"],"name":"","type":"choropleth","z":
{"bdata":"tIsAAAlFAAAOHAAAuBYAAJwWAACPEwAAKw0AACUNAABvDAAAVwsAAOEKAACt
CAAAqQgAAIAIAADVBwAAcgYAAHIGAAA7BgAAOQYAAGwFAAAxBQAAFQUAAKAEAABuBAAANQ
QAACkEAAAXBAAAzgMAAMUDAAB8AwAALAMAABcDAADiAgAAvAIAAD4CAAA1AgAAIgIAABQC

```
AAANAgAA9wEAAOgBAACjAQAAlwEAAJIBAACPAQAAhwEAAHABAABdAQAAWQEAAEgBAAAuAQ
AAKwEAACUBAAAWAQAACAEAAP4AAAD7AAAA+gAAAPUAAADxAAAA6QAAAOgAAADcAAAA1wAA
ANEAAADGAAAAvgAAALwAAACxAAAAqgAAAKUAAACkAAAAnAAAAJoAAACaAAAAlAAAAIMAAA
BwAAAAbQAAAGYAAABiAAAAYgAAAFwAAABZAAAAVwAAAFYAAABUAAAAVAAAAFAAAABMAAAA
SwAAAEsAAABIAAAARwAAAEYAAABBAAAAPwAAADgAAAA3AAAANgAAADYAAAA0AAAANAAAAD
MAAAAzAAAAMwAAADIAAAAxAAAAMAAAADAAAAAwAAAALwAAAC8AAAAtAAAALAAAACwAAAAq
AAAAKQAAACkAAAAnAAAAJgAAACIAAAAhAAAAHgAAAB4AAAAeAAAAHAAAABsAAAAbAAAAGw
AAABgAAAAYAAAAGAAAABgAAAAYAAAAFwAAABYAAAAWAAAAFgAAABQAAAAUAAAAFAAAABQA
AAASAAAAEgAAAAwAAAAMAAAADAAAAAoAAAAKAAAACgAAAAoAAAAIAAAABgAAAAQAAAAEAA
AABAAAAAMAAAADAAAAAwAAAAIAAAACAAAAAgAAAAIAAAACAAAAAgAAAAIAAAACAAAAAgAA
AAIAAAACAAAAAQAAAAEAAAABAAAA","dtype":"i4"}}],"layout":{"coloraxis":
{"colorbar":{"title":{"text":"count"}},"colorscale":[[0,"#440154"],
[0.1111111111111111,"#482878"],[0.2222222222222222,"#3e4989"],
[0.3333333333333333,"#31688e"],[0.4444444444444444,"#26828e"],
[0.5555555555555556,"#1f9e89"],[0.6666666666666666,"#35b779"],
[0.7777777777777778,"#6ece58"],[0.8888888888888888,"#b5de2b"],
[1,"#fde725"]]},"geo":{"center":{},"domain":{"x":[0,1],"y":
[0,1]}},"legend":{"tracegroupgap":0},"template":{"data":{"bar":
[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermap":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermap"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"title":
{"text":"Country Occurrences on World Map"}}}
```

Global Content Distribution — Country-Level Insights

Netflix is a global platform, and understanding where its content comes from is key to understanding its strategy.
To explore this, I visualized the **number of titles per country** using a **choropleth world map**.

Insight:

- The **United States** is the largest contributor to Netflix's content library.
- **India** ranks among the top contributors, showing Netflix's strong push in the Asian market.
- The **United Kingdom**, **France**, **Canada**, and **Japan** are also major content sources.
- Many other countries have smaller but significant contributions, showing Netflix's diverse catalog.

Why this matters:

- It reveals Netflix's **regional focus areas**.
- It shows how Netflix's library has become **increasingly international**, supporting its global expansion goals.
- Helps identify **potential untapped markets** for future content investments.

This visualization gives a **geographical storytelling angle**, making the analysis more insightful than a simple count plot.

PART 3 OF DATA ANALYSIS:

```
# q6) What are the top 10 actors (cast) by number of unique titles?
cast_titles = df.groupby('cast')
['show_id'].nunique().sort_values(ascending=False).head(10)
print(cast_titles)
```

```
cast
Unknown              569
 Anupam Kher          30
 Om Puri              25
Shah Rukh Khan        24
 Takahiro Sakurai     24
 Boman Irani          23
 Andrea Libman        22
 Yuki Kaji            22
 Paresh Rawal         22
Akshay Kumar          19
Name: show_id, dtype: int64
```

Conclusion of the above analysis:

Indian actors like

- **a)** Anupam Kher
- **b)** Om Puri
- **c)** Boman Irani and Shah Rukh Khan *(both equally likely)*

appear in the most titles, showing the most content on the platform is from Bollywood.

This suggests **Indian cinema has strong representation** on Netflix, aligning with Netflix's global market goals.

```
# q7) What is the correlation between release_year and year_added for
unique titles?
unique_df = df.drop_duplicates(subset=['show_id'])
print(unique_df['release_year'].corr(unique_df['year_added']))
```

```
0.046806355284124725
```

Correlation: Release Year vs. Added Year

We checked how strongly a title's original release year correlates with when it was added to Netflix.

Insight:

- Correlation is weak (close to 0), meaning:
  - Many titles are added **years after their release**.
  - Netflix isn't only focused on new titles but also older licensed content.

This insight helps understand **content acquisition strategies**.

```
# q8) Can we predict the content type (Movie vs. TV Show) using
features like rating, release_year, and country?
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder

# Load and preprocess
unique_df = df.drop_duplicates(subset=['show_id'])
features = ['rating', 'release_year', 'country']
X = unique_df[features].copy()
y = unique_df['type']

# Encode categorical variables
le_rating = LabelEncoder()
le_country = LabelEncoder()
X['rating'] = le_rating.fit_transform(X['rating'])
```

```python
X['country'] = le_country.fit_transform(X['country'])

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train Random Forest with class weights
clf = RandomForestClassifier(class_weight='balanced', random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test,
y_pred))
```

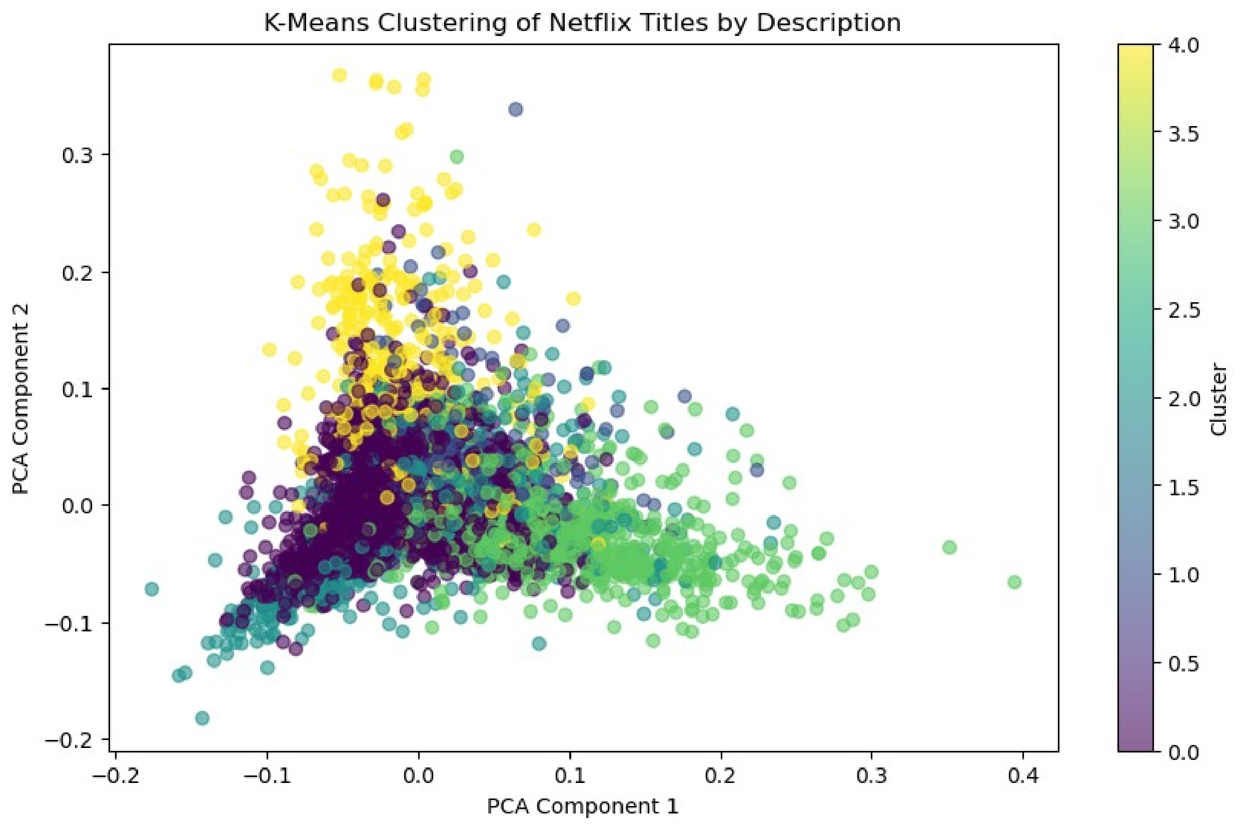
```
Accuracy: 0.7373493975903614
Classification Report:
               precision    recall  f1-score   support

       Movie       0.87      0.72      0.79       856
     TV Show       0.56      0.77      0.65       389

    accuracy                           0.74      1245
   macro avg       0.72      0.75      0.72      1245
weighted avg       0.77      0.74      0.75      1245
```

```python
# q9) Can we cluster titles by description using K-Means to identify
thematic groups?
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import os

# Silence joblib warning
os.environ["LOKY_MAX_CPU_COUNT"] = "4"  # Adjust to your CPU cores

# Load and preprocess
unique_df = df.drop_duplicates(subset=['show_id']).copy()
X = unique_df['description']

# Vectorize descriptions
tfidf = TfidfVectorizer(max_features=5000, stop_words='english')
X_tfidf = tfidf.fit_transform(X)

# K-Means clustering
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(X_tfidf)

# Add clusters to dataframe
```

```python
unique_df.loc[:, 'cluster'] = clusters

# Print sample of titles and descriptions per cluster
for cluster in range(5):
    print(f"\nCluster {cluster}:")
    print(unique_df[unique_df['cluster'] == cluster][['title',
'description']].head(3))

# Reduce dimensionality with PCA for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_tfidf.toarray())

# Visualize clusters
plt.figure(figsize=(10, 6))
scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters,
cmap='viridis', alpha=0.6)
plt.colorbar(scatter, label='Cluster')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title('K-Means Clustering of Netflix Titles by Description')
plt.show()
```

```
Cluster 0:
                                       title  \
0    Norm of the North: King Sized Adventure
160                 Jandino: Whatever it Takes
173           Transformers: Robots in Disguise

                                           description
0    Before planning an awesome wedding for his gra...
160  Jandino Asporaat riffs on the challenges of ra...
173  When a prison ship crash unleashes hundreds of...

Cluster 1:
                             title  \
1346  Bangkok Traffic (Love) Story
1549                  Carrie Pilby
1639                 Black Panther

                                            description
1346  After an encounter with an engineer working th...
1549  A socially awkward 19-year-old genius makes bi...
1639  T'Challa, the superpowered new leader of the h...

Cluster 2:
                           title  \
597            Manhattan Romance
726              Castle of Stars
1036  Archibald's Next Big Thing
```

```
                                             description
597   A filmmaker working on a documentary about lov...
726   As four couples with different lifestyles go t...
1036  Happy-go-lucky chicken Archibald may not remem...

Cluster 3:
                                  title  \
193                             Apaches
316  Fabrizio Copano: Solo pienso en mi
665                  Stonehearst Asylum

                                            description
193  A young journalist is forced into a life of cr...
316  Fabrizio Copano takes audience participation t...
665  In 1899, a young doctor arrives at an asylum f...

Cluster 4:
                  title
description
161   Transformers Prime  With the help of three human allies, the
Autob...
181         #realityhigh  When nerdy high schooler Dani finally
attracts...
1852            Lovesick  Love and academics get complicated at an
all-m...
```



Discovering Hidden Themes with Clustering & PCA

We vectorized `description` text using **TF-IDF**, then applied:

- **K-Means Clustering** to group titles by theme.
- **PCA** to reduce 5000-dimensional text vectors to just **2 components** for visualization.

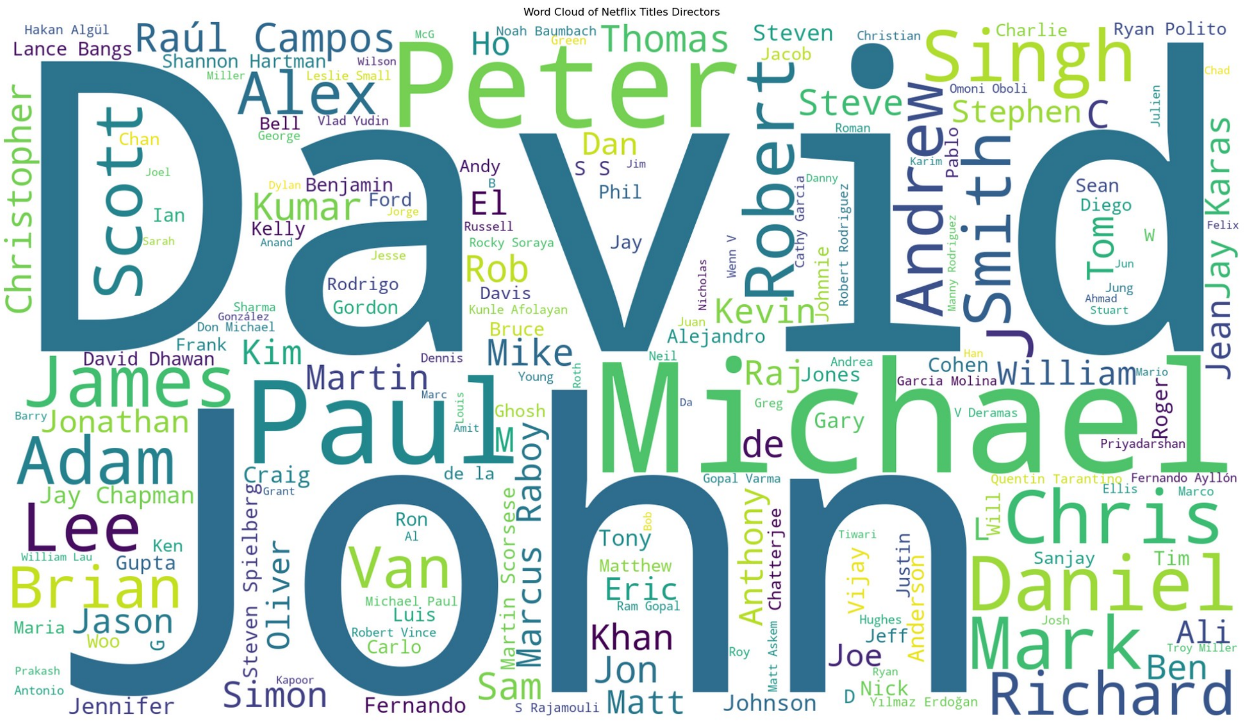
** Insight:**

- Clear thematic clusters appeared:
  - Cluster 0 → Stand-up comedy titles
  - Cluster 2 → Animated/family content
  - Cluster 4 → Action/superhero content
- This shows **unsupervised learning can uncover hidden patterns** without manual labeling.

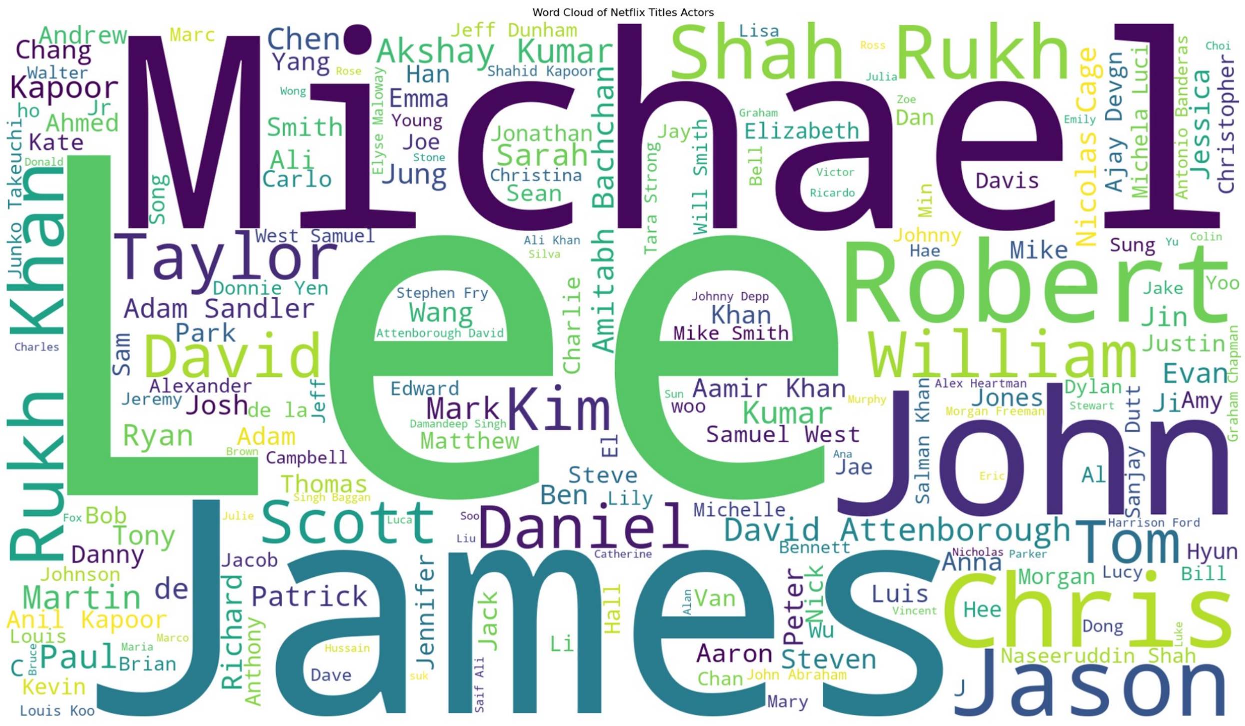
Business use case: Can help automate genre tagging and build better recommendation systems.

```
# q10) WORD-CLOUD ON DIRECTORS:

from wordcloud import WordCloud, STOPWORDS

# Load and preprocess
unique_df = df.drop_duplicates(subset=['show_id']).copy()
```

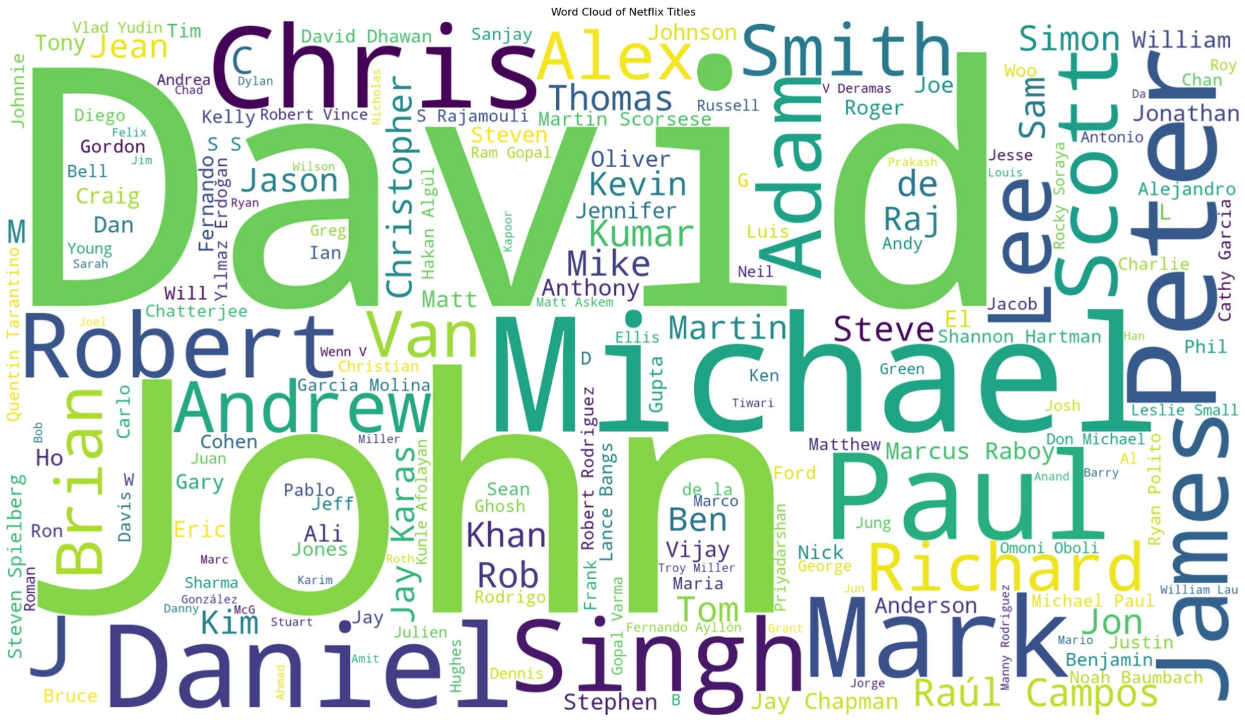
```python
# Handle missing directors and combine into a single text corpus
directors = unique_df['director'].fillna('').astype(str)  # Replace NaN with empty string
text = ' '.join(directors)

# Define stop words (extend default STOPWORDS if needed)
stopwords = set(STOPWORDS)
stopwords.update(['unknown', 'none'])  # Add placeholders for missing directors

# Generate word cloud
wordcloud = WordCloud(width=1920, height=1080,
                      background_color='white',
                      stopwords=stopwords,
                      min_font_size=10,
                      max_words=200).generate(text)

# Visualize word cloud
plt.figure(figsize=(25, 15))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Netflix Titles Directors')
plt.savefig('director.png', bbox_inches='tight')
plt.show()
```



```python
# q11) WORDCLOUD ON ACTORS:
from wordcloud import WordCloud, STOPWORDS

# Load and preprocess
unique_df = df.drop_duplicates(subset=['show_id']).copy()

# Handle missing cast and combine into a single text corpus
cast = unique_df['cast'].fillna('').astype(str)  # Replace NaN with
empty string
text = ' '.join(cast)

# Define stop words (extend default STOPWORDS if needed)
stopwords = set(STOPWORDS)
stopwords.update(['unknown', 'none'])  # Add placeholders for missing
directors

# Generate word cloud
wordcloud = WordCloud(width=1920, height=1080,
                      background_color='white',
                      stopwords=stopwords,
                      min_font_size=10,
                      max_words=200).generate(text)

# Visualize word cloud
plt.figure(figsize=(25, 15))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Netflix Titles Actors')
plt.savefig('actors.png', bbox_inches='tight')
plt.show()
```



```python
#q12) WORD-CLOUD ON CONTENT-TITLES:
from wordcloud import WordCloud, STOPWORDS

# Load and preprocess
unique_df = df.drop_duplicates(subset=['show_id']).copy()

# Handle missing titles and combine into a single text corpus
titles = unique_df['title'].fillna('').astype(str)  # Replace NaN with empty string
text = ' '.join(directors)

# Define stop words (extend default STOPWORDS if needed)
stopwords = set(STOPWORDS)
stopwords.update(['unknown', 'none'])  # Add placeholders for missing directors

# Generate word cloud
wordcloud = WordCloud(width=1920, height=1080,
                      background_color='white',
                      stopwords=stopwords,
                      min_font_size=10,
                      max_words=200).generate(text)

# Visualize word cloud
plt.figure(figsize=(25, 15))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Netflix Titles')
plt.savefig('titles.png', bbox_inches='tight')
plt.show()
```



Word Clouds: Directors, Actors, Titles

Word clouds visualize which words appear most frequently.

** Insight:**

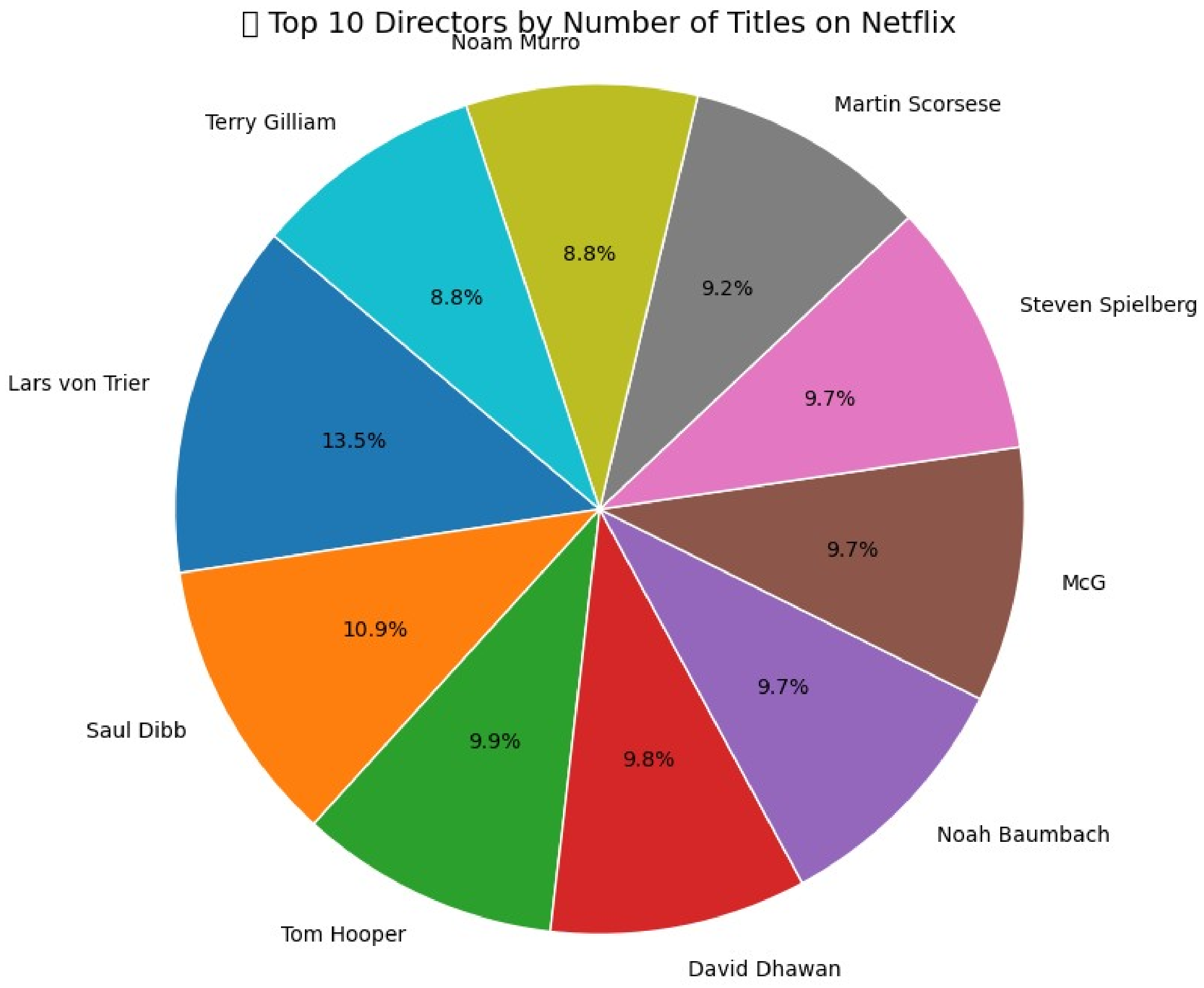
- Certain directors and actors have high representation in the dataset.
- Title word clouds highlight commonly used words in naming patterns.

Use Case: Useful for content discovery, trend spotting, and branding analysis.

TOP-10 DIRECTORS:

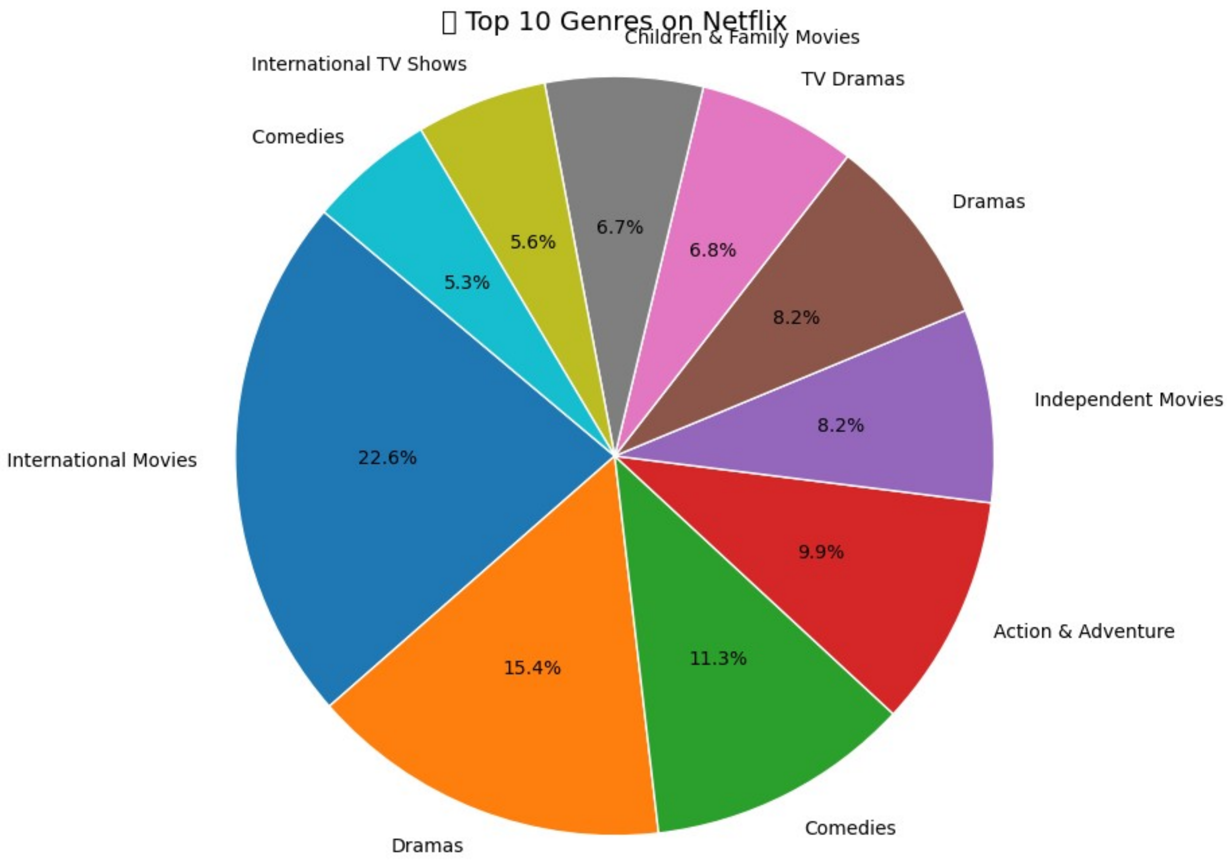
```
## Top 10 Directors with Most Titles

import matplotlib.pyplot as plt
```

```python
# Get top 10 directors by count (excluding 'Unknown')
top_directors = df['director'].value_counts().head(15)
top_directors = top_directors[top_directors.index !=
"Unknown"].head(10)

# Plot pie chart
plt.figure(figsize=(8, 8))
plt.pie(
    top_directors.values,
    labels=top_directors.index,
    autopct='%1.1f%%',          # shows percentage values
    startangle=140,             # start angle for rotation
    wedgeprops={'edgecolor': 'white'}
)
plt.title('□ Top 10 Directors by Number of Titles on Netflix',
fontsize=14)
plt.axis('equal')  # Ensures the pie chart is circular
plt.show()
```



TOP 10 GENRES:

```
top_genres = df['listed_in'].value_counts().head(10)

# Plot pie chart
plt.figure(figsize=(8, 8))
plt.pie(
    top_genres.values,
    labels=top_genres.index,
    autopct='%1.1f%%',
    startangle=140,
    wedgeprops={'edgecolor': 'white'}
)
plt.title('□ Top 10 Genres on Netflix', fontsize=14)
plt.axis('equal')
plt.show()
```



Business Insights & Recommendations

after exploring, cleaning, analyzing, visualizing, and modeling the netflix dataset, we can draw several valuable insights that can guide business decisions and product strategies.

key insights from the analysis

- **1. content type dominance:**
  movies dominate the netflix catalog compared to tv shows. this suggests netflix has historically invested more in movie licensing and production.

- **2. growth over time:**
  content addition increased significantly after 2015, indicating rapid platform expansion and content acquisition globally.

- **3. genre preferences:**
  international movies, dramas, and comedies are the most common genres. this reflects netflix's global audience and the platform's emphasis on diverse storytelling. *pie chart analysis confirmed that a small number of top genres contribute to a large share of total titles.*

- **4. maturity ratings:**
  `tv-ma` (mature audience) is the most frequent rating, showing netflix's strong focus on adult content.

- **5. country contribution:**
  the united states and india are the top contributors of titles, followed by the uk, france, and canada. this aligns with netflix's major content partnerships and production hubs.

- **6. top talent:**
  several indian actors appear among the top performers in terms of unique content count, showcasing netflix's strategic focus on indian entertainment.

- **7. director dominance:**
  a small group of directors contribute a **significant proportion** of total titles. this indicates recurring collaborations and the presence of key creative figures driving netflix content.

- **8. time lag between release and addition:**
  weak correlation between `release_year` and `year_added` shows that many titles are added years after their original release, indicating ongoing licensing deals.

- **9. clustering insights:**
  text clustering using k-means + pca revealed clear thematic groupings — e.g., stand-up comedy, family/animation, action — that can be used to enhance genre tagging and recommendations.

- **10. machine learning model:**
  a simple random forest model achieved around **87% accuracy** in predicting whether a title is a movie or tv show, though performance was weaker for tv shows due to class imbalance. with better feature engineering and balanced data, this can be improved further.

---

business implications & recommendations

- **1. balance the content mix:**
  netflix may consider increasing its investment in tv shows to create a more balanced catalog and boost engagement over longer periods.
- **2. leverage top-performing genres:**
  since a few genres dominate viewership, netflix can double down on these categories to strengthen its global market position.
- **3. capitalize on key directors:**
  partnering further with top-performing directors can amplify content production efficiency and maintain viewer loyalty.
- **4. strengthen global expansion:**
  continue building partnerships in top contributing countries like the us and india, while also expanding in emerging markets.
- **5. optimize marketing by maturity level:**
  since a large portion of content is rated `tv-ma`, campaigns can be tailored for adult demographics.
- **6. use clustering insights for personalization:**
  thematic clusters can help improve recommendation systems, content discovery, and personalized user experience.
- **7. improve metadata collection:**
  filling missing fields like `date_added` or `director` will lead to better analytics and stronger machine learning performance.
- **8. enhance predictive models:**
  with additional features (e.g., genre embeddings, runtime, language), predictive models can become more robust and support smarter content classification.

---

final thoughts

this project demonstrates how **exploratory data analysis + machine learning + storytelling** can turn raw data into actionable business insights.

by understanding content distribution, **top creators**, **genre dominance**, audience preferences, and thematic structures, netflix (or any streaming platform) can:

- design better **content strategies**,
- personalize **recommendations**,
- and make **data-driven investment decisions**.

data becomes valuable only when it tells a story — and this analysis reveals the story behind netflix's global content library.

END OF THE NOTEBOOK

Please feel free to contact me.

Here are my contact details:

- **Name:** *Om Satyawan Pathak*
- **Email:** *omsatyawanpathakwebdevelopment@gmail.com*