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

```
# In this jupyter notebook we have performed data-analysis on the
netflix dataset.
# We have only two types of content:
#     i) Movie
#     ii) TV-Show



# In this notebook; we have drawn insights for these topics:
#          a) Frequency of each content-type
#              (i.e No. of movies v/s No. of tv-shows)
#          b) No. of releases in each year
#          c) Top 5 Genres generating the most traffic
#          d) Which ratings are most oftenly given to the netflix
content-library?
#          e) Which countries have contributed to the netflix-library
#          f) Who are the top-10 actors by unique content-types?
#          g) Correlation between content's release-year and year (in
which it was added to netflix) or unique titles
#          h) Try to predict the content type (Movie vs. TV Show)
using features like rating, release_year, and country
#          i) Clustering titles by description using K-Means to
identify thematic groups
#          j) Listing down these on seprate word-clouds:
#             j.1) directors
#             j.2) actors
#             j.3) titles




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


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

df.head(10)
```

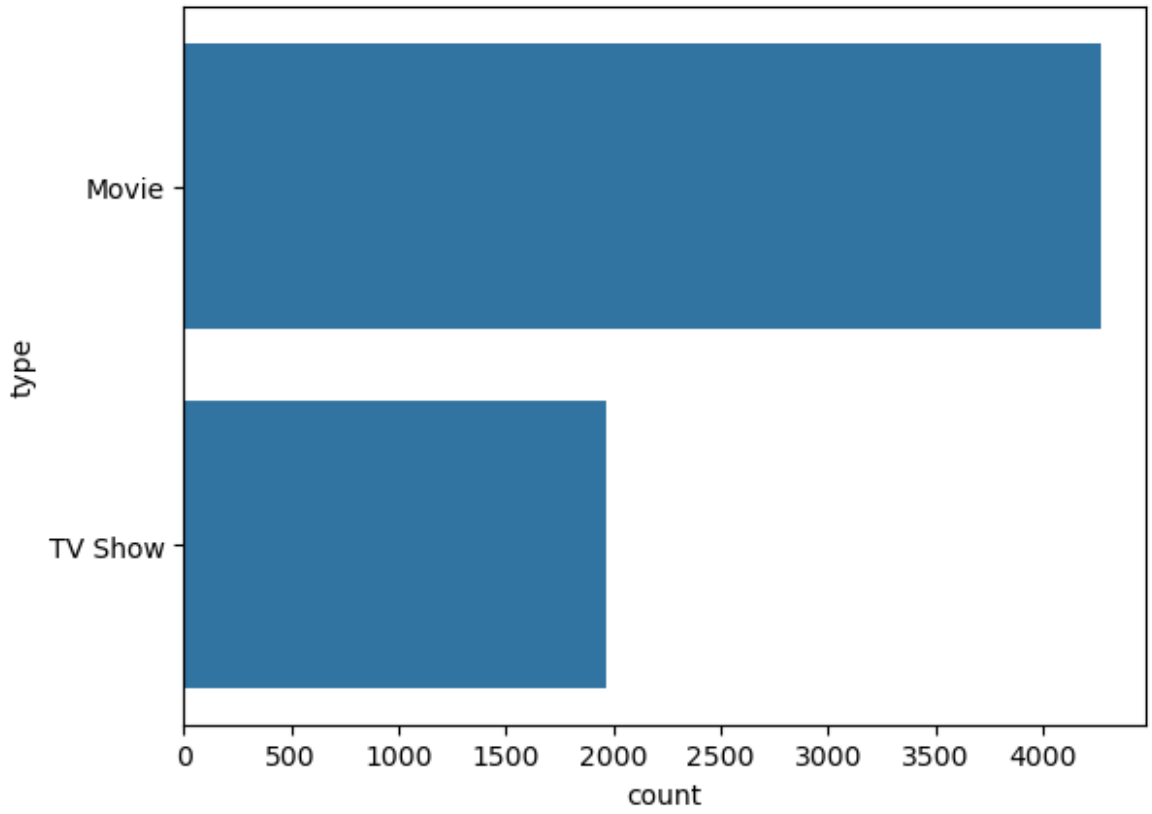
```
    show_id     type                                          title  \
0  81145628    Movie  Norm of the North: King Sized Adventure
```

```
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
```

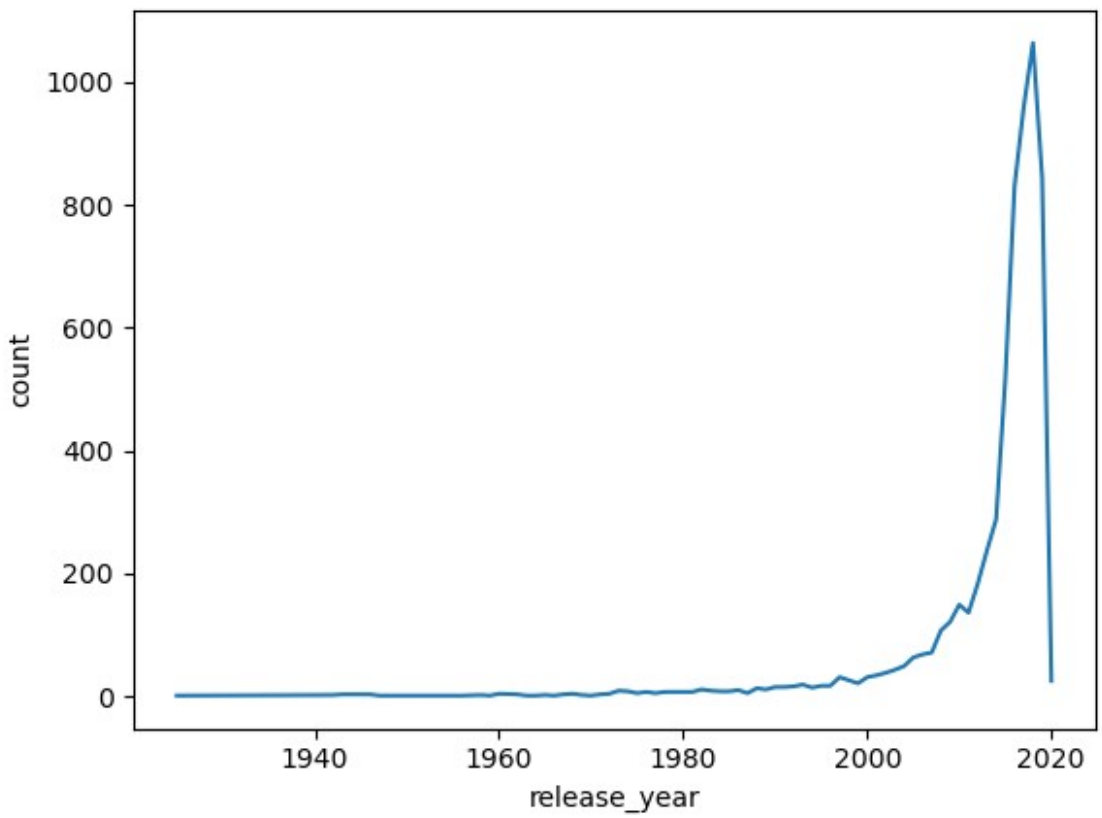
```
7                                                Chile  September 8, 2017

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()
```




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



DATA CLEANING:

```
# NOW UNTIL HERE WE HAVE NOTICED SOME THAT SOME COLUMNS HAVE MULTIPLE
VALUES FOR SOME (or) MOST OF THE ROWS.
# SOMEWHERE WE HAVE MISSING VALUES


# NOW WE WILL START DATA CLEANING BY PERFORMING THE FOLLOWING STEPS:
#        1) Breaking 'cast' column so that every row must have only
one value for the 'cast' column
#        2) Breaking 'country' column so that every row must have only
one value for the 'country' column
#        3) Breaking 'listed_in' (means; GENERES) column so that every
row must have only one value for the 'listed_in' column
#        4) Breaking 'director' column so that every row must have
only one value for the 'director' column

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.dropna(inplace=True)
```

```
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
          ..
139699    15
139700    15
```

```
139701    15
139702    15
139703    15
Name: day_added, Length: 98233, dtype: int64
```

df.info()

NOW EXPORT THIS CLEANED FILE:

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

PART 2 OF DATA VISUALIZATION:

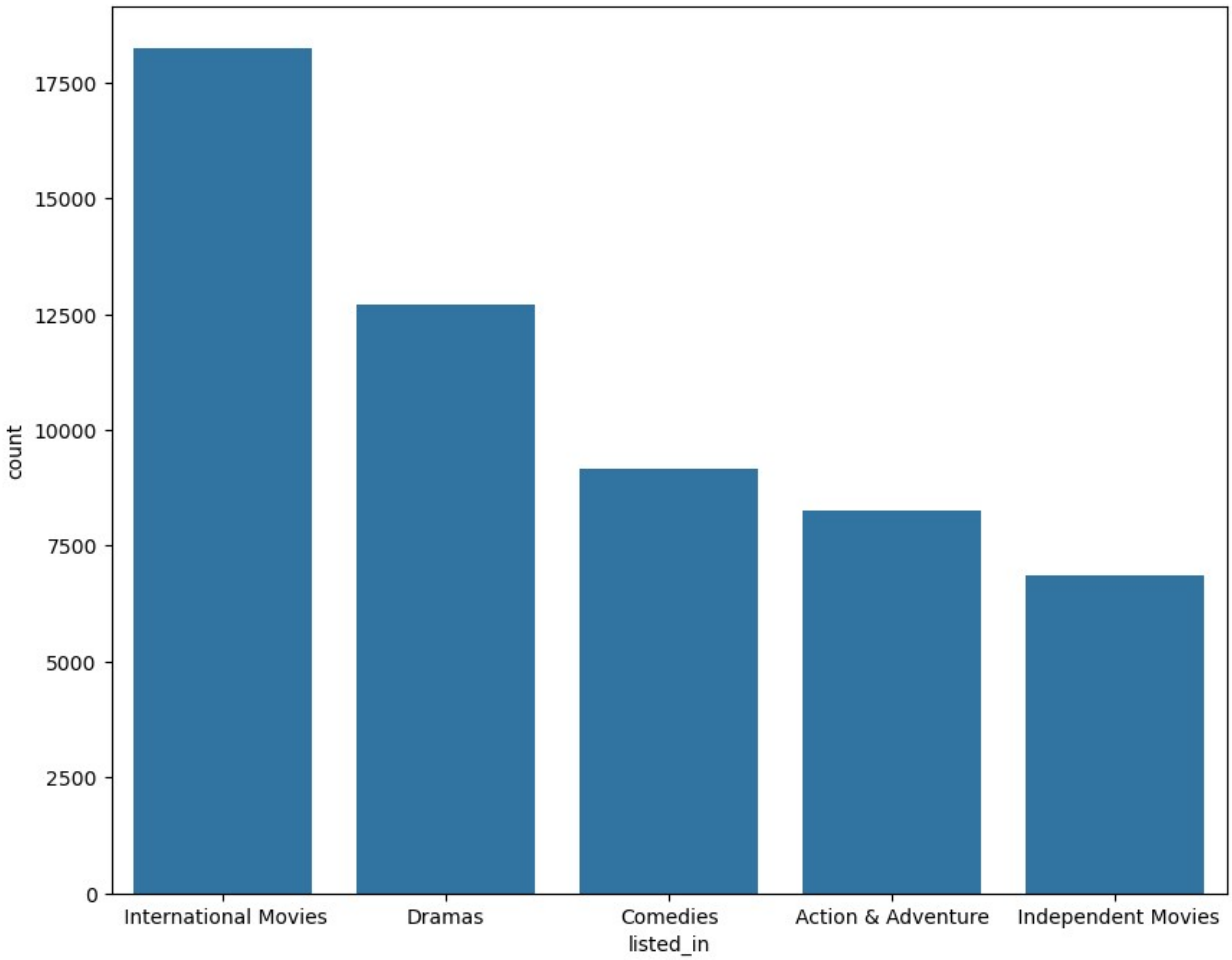
```
#q3) top genres:
data=df["listed_in"].value_counts().reset_index()
data
```

```
                        listed_in  count
0           International Movies  18249
1                         Dramas  12697
2                       Comedies   9167
3             Action & Adventure   8251
4             Independent Movies   6871
..                           ...    ...
62              Classic & Cult TV      9
63           Science & Nature TV       7
64              Classic & Cult TV      7
65  Stand-Up Comedy & Talk Shows       6
66                    Reality TV       4

[67 rows x 2 columns]
```

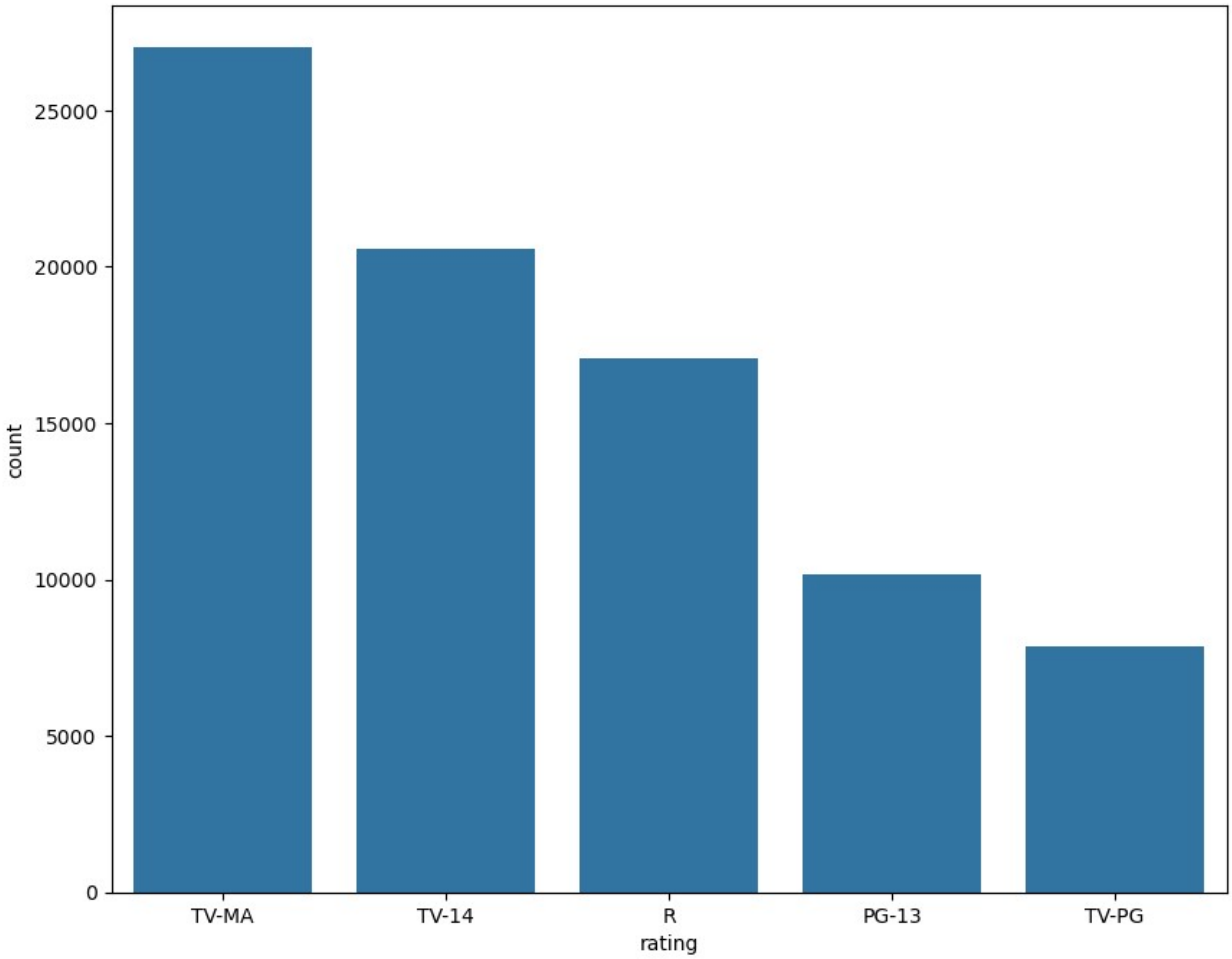
```
data.head(5)
```

```
               listed_in  count
0   International Movies  18249
1                 Dramas  12697
2               Comedies   9167
3     Action & Adventure   8251
4     Independent Movies   6871
```

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




```
#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()
```




```
# 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}
count=%{z}<extra></extra>","locationmode":"country
names","locations":["United States","India"," United States","United
Kingdom"," France","Spain","Canada","France"," United
Kingdom","Japan"," Canada","Hong Kong","Turkey"," Germany","Mexico","
Belgium","South Korea","Indonesia","Philippines","Germany","China","
China","Thailand","Nigeria","Egypt","Argentina","Australia","
Japan","Brazil"," Spain"," India"," Hong Kong","
Australia","Pakistan"," Sweden","Denmark","Italy","
Italy","Bangladesh","Taiwan","Ireland"," Mexico","Chile","South
Africa","Colombia","Poland","United Arab Emirates","
Qatar","Israel","Malaysia"," South Korea"," Argentina","Belgium","
Singapore"," Denmark","Netherlands"," United Arab Emirates","
Netherlands","New Zealand"," Switzerland"," Brazil"," Poland","
Luxembourg"," South Africa"," Norway"," Czech Republic","
Lebanon","Singapore","Switzerland","Norway","
Ireland","Bulgaria","Romania"," Greece"," Morocco"," Jordan"," Soviet
Union"," Portugal"," Iran","Peru"," Malaysia"," Chile","Austria","
Peru","Czech Republic","Vietnam"," New Zealand"," Turkey"," West
Germany","
Austria","Russia","Sweden","Georgia","Paraguay","Hungary","Ghana","Uru
guay"," Taiwan"," Uruguay","Soviet Union"," Nepal","Serbia","
Slovenia","Iceland"," Pakistan"," Iceland"," Serbia","
Hungary","Portugal"," Indonesia"," Bulgaria"," Russia"," East
Germany"," Egypt"," Finland"," Romania"," Liechtenstein","
Bangladesh"," Senegal","","Cambodia"," Malawi","Lebanon"," Colombia","
Croatia"," Latvia"," Cambodia","Finland","Iran"," Cayman Islands","
Slovakia"," Malta"," Montenegro","Croatia"," Israel"," Dominican
Republic"," Thailand","Saudi Arabia"," Ecuador","
Bermuda","Guatemala"," Sudan","Somalia"," Kenya"," Albania","
Iraq","Slovenia"," Vatican City","Venezuela","Dominican Republic","
Panama"," Sri Lanka"," Afghanistan","
Nicaragua"],"name":"","type":"choropleth","z":
{"bdata":"LWdNQtkRVxCcCX8IPwi0B2oHNwd/
Bu4FowVCBYcESwQjBBcEDQTqA6IDWAMuAykDKAPqAuQCuwKzAmACCQLxAesBpgGgAXwBbQ
FhAVkBPAEtASsBKQElAQwBAQH7APUA8ADmAOMA2ADUAM8AzgDKAMUAwwC6ALEArAClAJwA
mwCYAI4AjACLAH4AdAByAGYAYgBaAFcAVgBUAFQATABIAEgARwBHAEEAQAA/
AD8APwA8ADwAOAA3ADYANAAzADMAMgAxADEAMAAwAC8ALwAvAC0ALAAsACwAKgAqACkAIg
AeAB4AHgAeABsAGwAbABsAGAAYABgAGAAXABYAFgAWABYAFAAUABQAFAAUABMAEgAQAA8A
DAAMAAwACgAKAAoACAAEAAMAAwACAAIAAgACAAIAAQA=","dtype":"i2"}}],"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"}}}
```


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
 Anupam Kher          29
 Om Puri              25
 Boman Irani          23
Shah Rukh Khan        23
 Paresh Rawal         22
Akshay Kumar          19
 Naseeruddin Shah     18
 Asrani               15
 Gulshan Grover       15
Amitabh Bachchan      15
Name: show_id, dtype: int64
```

```
# 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']))
```

```
-0.047192615150680024
```

```
# 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'])
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))
```

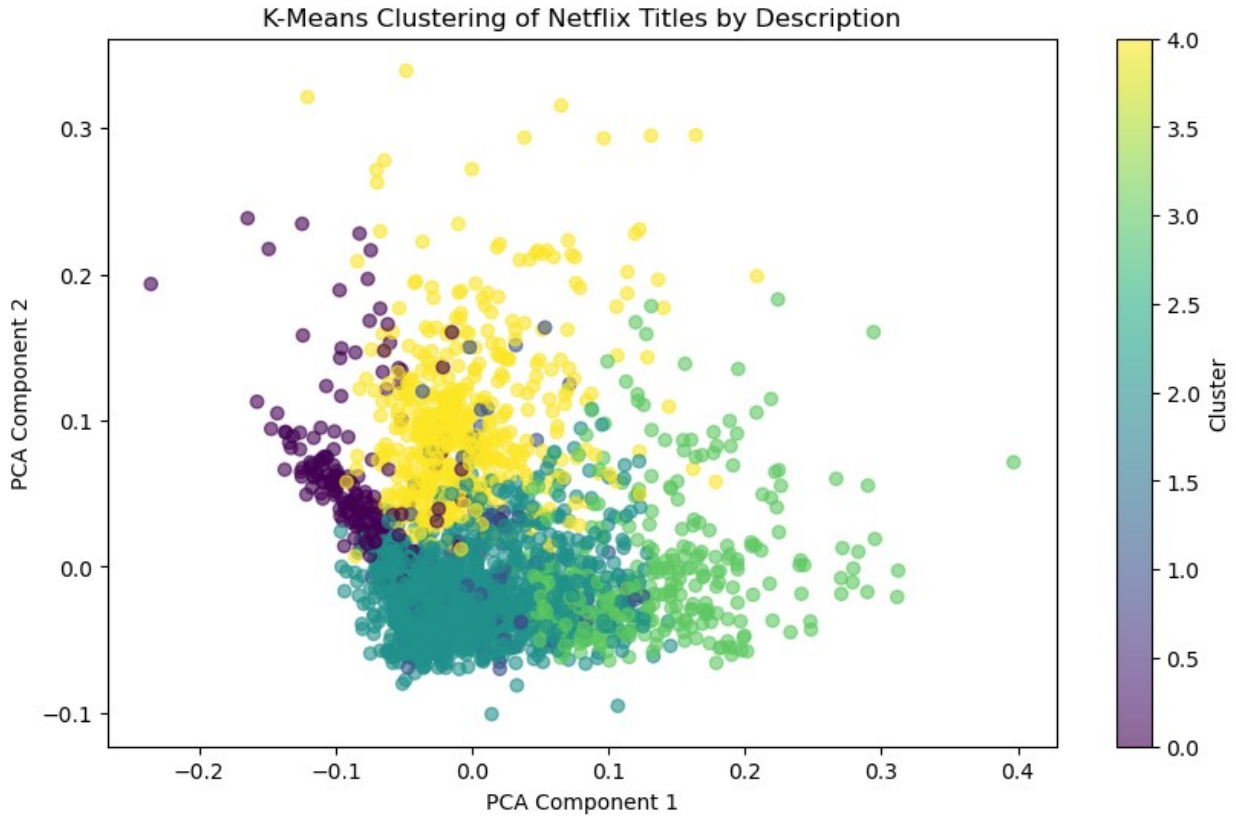
```
Accuracy: 0.8728476821192053
Classification Report:
              precision    recall  f1-score   support

       Movie       0.98      0.89      0.93       739
     TV Show       0.03      0.19      0.06        16

    accuracy                           0.87       755
   macro avg       0.51      0.54      0.50       755
weighted avg       0.96      0.87      0.91       755
```

```
# 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
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()
```

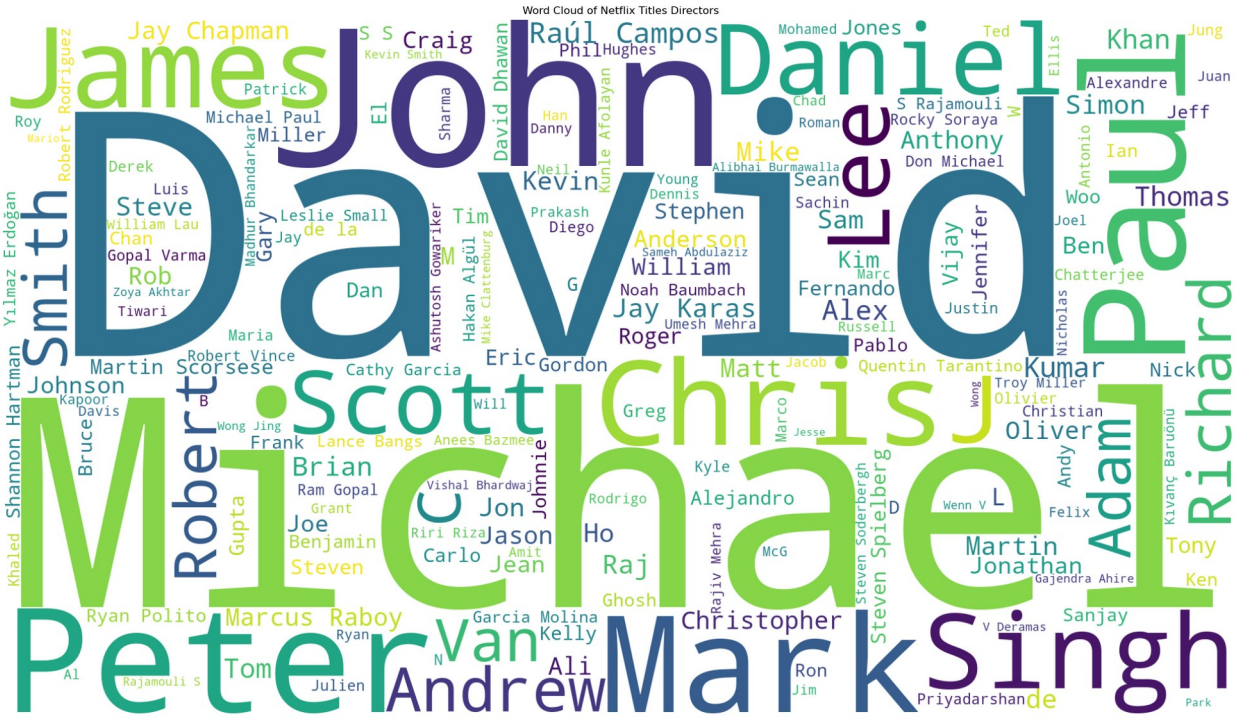
```
Cluster 0:
                                    title  \
316   Fabrizio Copano: Solo pienso en mi
1565                Marc Maron: Too Real
1764              Mo Gilligan: Momentum

                                              description
316   Fabrizio Copano takes audience participation t...
```

```
1565  Battle-scarred stand-up comedian Marc Maron un...
1764  Comedian Mo Gilligan blends smooth moves and s...

Cluster 1:
             title                                              description
1286           ATM  When a broken ATM dishes out a fortune, a coup...
1549  Carrie Pilby  A socially awkward 19-year-old genius makes bi...
1671        Man Up  A single woman seizes an opportunity when, whi...

Cluster 2:
                                        title  \
0    Norm of the North: King Sized Adventure
181                            #realityhigh
208                                Automata

                                             description
0    Before planning an awesome wedding for his gra...
181  When nerdy high schooler Dani finally attracts...
208  In a dystopian future, an insurance adjuster f...

Cluster 3:
                   title
description
665   Stonehearst Asylum  In 1899, a young doctor arrives at an asylum
f...
699              6 Years  As a volatile young couple who have been
toget...
1240           Hard Tide  A drug dealer who's been emulating his
father'...

Cluster 4:
                  title
description
685         The Runner  A New Orleans politician finds his
idealistic ...
1639     Black Panther  T'Challa, the superpowered new leader of the
h...
1896  Act of Vengeance  Two Turkish agents are sent to New York City
o...
```



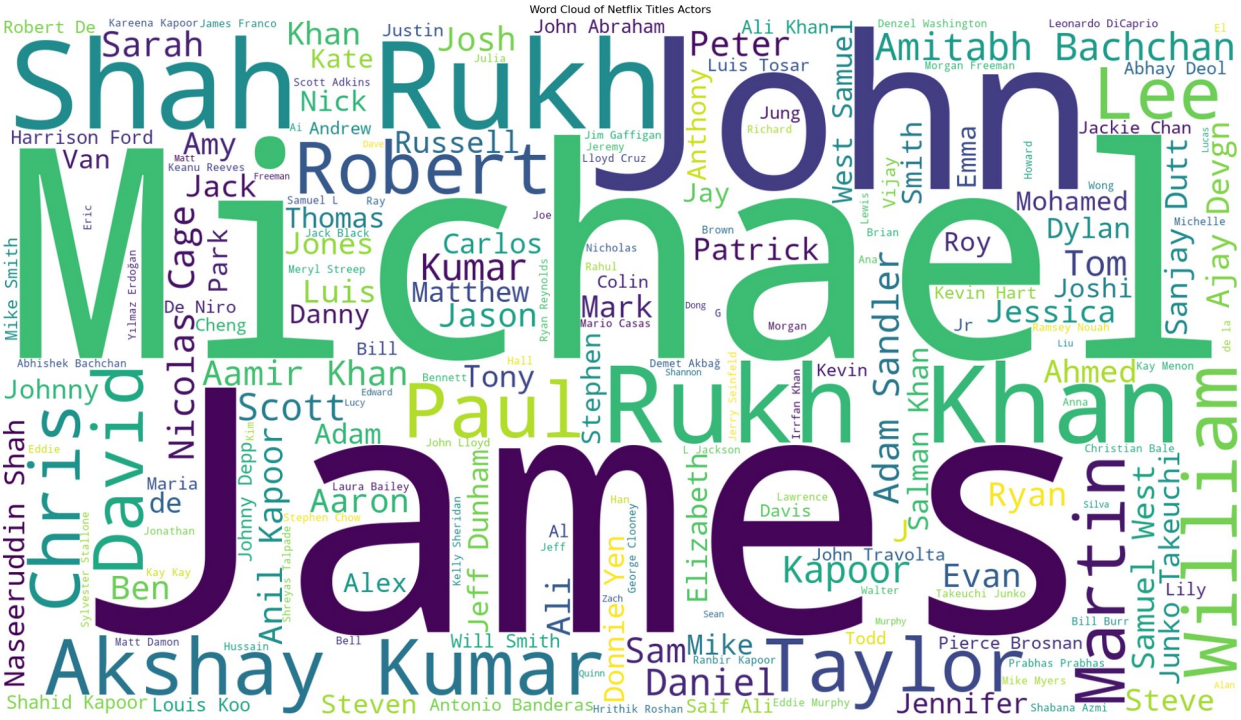

```
# q10) WORD-CLOUD ON DIRECTORS:

from wordcloud import WordCloud, STOPWORDS

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

# 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()
```



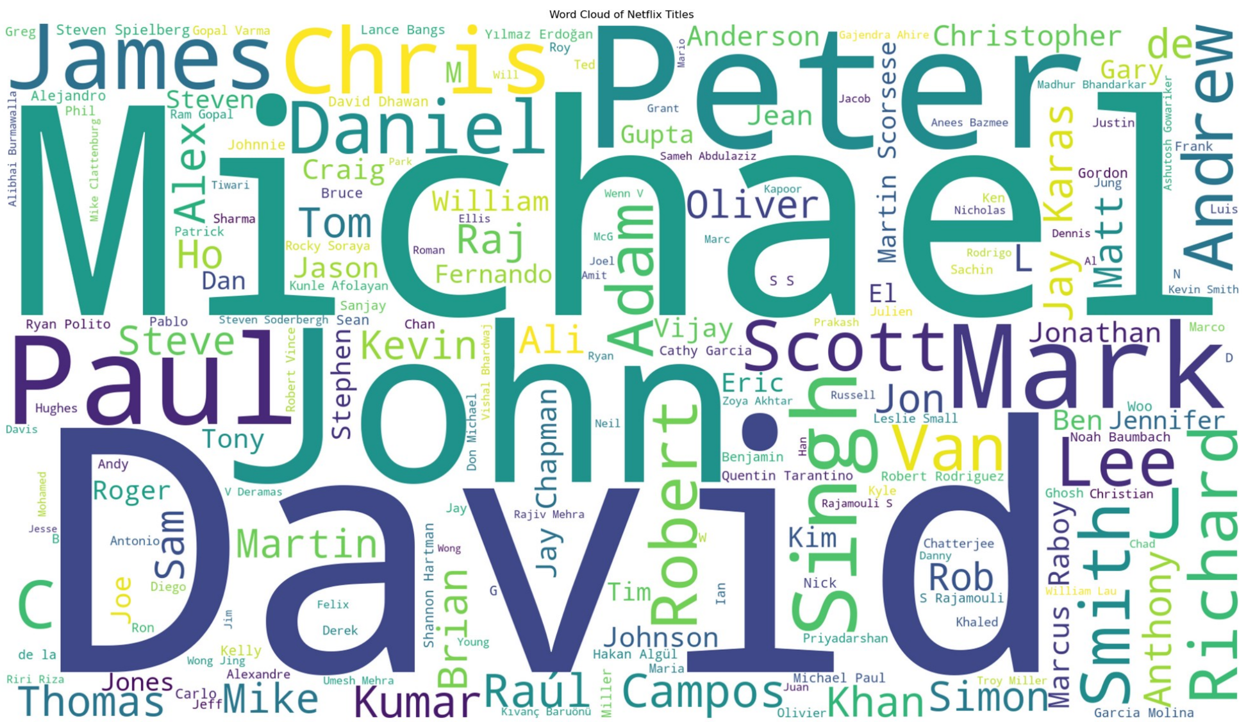

```
# q11) WORDCLOUD ON ACTORS:
from wordcloud import WordCloud, STOPWORDS

# 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()
```




```
#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
```

```python
# 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()
```




```python
# END OF THE NOTEBOOK


# Please feel free to contact me.

# Here are my contact details:
# NAME: Om Satyawan Pathak
# EMAIL: omsatyawanpathakwebdevelopment@gmail.com
```