```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
!pip install wordcloud
from wordcloud import WordCloud
```

```
Requirement already satisfied: wordcloud in
/usr/local/lib/python3.10/dist-packages (1.9.3)
Requirement already satisfied: numpy>=1.6.1 in
/usr/local/lib/python3.10/dist-packages (from wordcloud) (1.25.2)
Requirement already satisfied: pillow in
/usr/local/lib/python3.10/dist-packages (from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (from wordcloud) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud)
(1.2.0)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud)
(4.48.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud)
(1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud)
(23.2)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud)
(3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud)
(2.8.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib->wordcloud) (1.16.0)
```

```
df = pd.read_csv("/content/netflix.csv")
```

```
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8807,\n  \"fields\":
[\n    {\n      \"column\": \"show_id\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"samples\": [\n          \"s4971\",\n
\"s3363\",\n          \"s5495\"\n        ],\n
\"num_unique_values\": 8807,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"type\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"samples\": [\n          \"TV Show\",\n          \"Movie\"\
n        ],\n        \"num_unique_values\": 2,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"title\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"samples\": [\n
\"Game Over, Man!\",\n          \"Arsenio Hall: Smart & Classy\"\n
],\n        \"num_unique_values\": 8807,\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"director\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"samples\": [\n          \"Kanwal Sethi\",\n
\"R\\u00e9my Four, Julien War\"\n        ],\n
\"num_unique_values\": 4528,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cast\",\n      \"properties\": {\n        \"dtype\": \"string\",\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        \"num_unique_values\": 7692,\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"country\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"United States,
United Kingdom, Denmark, Sweden\",\n          \"United Kingdom, Hong
Kong\"\n        ],\n        \"num_unique_values\": 748,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"date_added\",\n
\"properties\": {\n        \"dtype\": \"object\",\n        \"min\":
\"2008-01-01 00:00:00\",\n        \"max\": \"2021-09-25 00:00:00\",\n
\"samples\": [\n          \"October 22, 2018\",\n          \"January
29, 2021\"\n        ],\n        \"num_unique_values\": 1767,\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
\"samples\": [\n          1996,\n          1969\n        ],\n
\"num_unique_values\": 74,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"PG-13\",\n
\"TV-MA\"\n        ],\n        \"num_unique_values\": 17,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"duration\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"samples\": [\n
\"37 min\",\n          \"177 min\"\n        ],\n
\"num_unique_values\": 220,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"listed_in\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"Crime TV Shows,
```

```
International TV Shows, TV Mysteries\",\n          \"Children & Family
Movies, Classic Movies, Dramas\"\n        ],\n
\"num_unique_values\": 514,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"description\",\n      \"properties\": {\n        \"dtype\":
\"string\",\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
\"num_unique_values\": 8775,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

```
df.shape
```

```
(8807, 12)
```

```
df.columns
```

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

```
#stats summary
df.dtypes
```

```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration        object
listed_in       object
description     object
dtype: object
```

```
df['date_added'] = pd.to_datetime(df['date_added'])  #dtype for
date_added column should be datetime*
df.dtypes
```

```
show_id                 object
type                    object
title                   object
```

```
director                  object
cast                      object
country                   object
date_added        datetime64[ns]
release_year               int64
rating                    object
duration                  object
listed_in                 object
description               object
dtype: object
```

```
columns = ['director', 'cast', 'country', 'listed_in', 'rating']
#filling blanks with unknown for mentioned columns
for i in columns:
  df[i] = df[i].fillna("Unknown")
df
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8807,\n  \"fields\":
[\n    {\n      \"column\": \"show_id\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"samples\": [\n          \"s4971\",\n
\"s3363\",\n          \"s5495\"\n        ],\n
\"num_unique_values\": 8807,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"type\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"samples\": [\n          \"TV Show\",\n          \"Movie\"\
n        ],\n        \"num_unique_values\": 2,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"title\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"samples\": [\n
\"Game Over, Man!\",\n          \"Arsenio Hall: Smart & Classy\"\n
],\n        \"num_unique_values\": 8807,\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"director\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"samples\": [\n          \"Babak Anvari\",\n
\"Maria Ripoll\"\n        ],\n        \"num_unique_values\": 4529,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"cast\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"samples\": [\n          \"Vittoria
Puccini, Francesco Scianna, Camilla Filippi, Simone Colombari,
Maurizio Lastrico, Alessandro Averone, Euridice Axen, Marco Baliani,
Pia Lanciotti, Giordano De Plano, Roberto Herlitzka, Tommaso Ragno,
Margherita Caviezel, Michele Morrone\",\n          \"Banky Wellington,
Rahama Sadau, Kanayo O. Kanayo, Ibrahim Suleiman, Michelle Dede,
Adesua Etomi, Hilda Dokubo, Akin Lewis\"\n        ],\n
\"num_unique_values\": 7693,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"country\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"United States,
United Kingdom, Denmark, Sweden\",\n          \"Spain, France\"\n
],\n        \"num_unique_values\": 749,\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   \"samples\": [\n
\"2016-04-20 00:00:00\",\n   \"2017-01-10 00:00:00\"\
n   ],\n   \"num_unique_values\": 1714,\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
\"samples\": [\n   1996,\n   1969\n   ],\n
\"num_unique_values\": 74,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"rating\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"samples\": [\n   \"PG-13\",\n
\"TV-MA\"\n   ],\n   \"num_unique_values\": 18,\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"duration\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"samples\": [\n
\"37 min\",\n   \"177 min\"\n   ],\n
\"num_unique_values\": 220,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"listed_in\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"samples\": [\n   \"Crime TV Shows,
International TV Shows, TV Mysteries\",\n   \"Children & Family
Movies, Classic Movies, Dramas\"\n   ],\n
\"num_unique_values\": 514,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"description\",\n   \"properties\": {\n   \"dtype\":
\"string\",\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
\"num_unique_values\": 8775,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

```
df.loc[df['duration'].isnull()]  #duration values have been place in
rating column inadvertently
```

{"repr_error":"0","type":"dataframe"}

```
df.loc[df["duration"].isnull(), "duration"] =
df.loc[df["duration"].isnull(), "rating"]
```

```
df.loc[[5541,5794,5813], 'rating'] = 'Unknown'
```

```
df.loc[[5541,5794,5813]]
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 3,\n  \"fields\": [\n
{\n      \"column\": \"show_id\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"samples\": [\n          \"s5542\",\n
\"s5795\",\n          \"s5814\"\n        ],\n
\"num_unique_values\": 3,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"type\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"samples\": [\n          \"Movie\"\n        ],\n
\"num_unique_values\": 1,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"title\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"samples\": [\n          \"Louis C.K. 2017\"\n        ],\n
\"num_unique_values\": 3,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"director\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"Louis C.K.\"\n
],\n        \"num_unique_values\": 1,\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"cast\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"Louis C.K.\"\n
],\n        \"num_unique_values\": 1,\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"country\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"United States\"\n
],\n        \"num_unique_values\": 1,\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"date_added\",\n      \"properties\": {\n
\"dtype\": \"date\",\n        \"min\": \"2016-08-15 00:00:00\",\n
\"max\": \"2017-04-04 00:00:00\",\n        \"samples\": [\n
\"2017-04-04 00:00:00\"\n        ],\n        \"num_unique_values\":
3,\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"release_year\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 2010,\n        \"max\": 2017,\n
\"samples\": [\n          2017\n        ],\n
\"num_unique_values\": 3,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"Unknown\"\
n        ],\n        \"num_unique_values\": 1,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"duration\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"samples\": [\n
\"74 min\"\n        ],\n        \"num_unique_values\": 3,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"listed_in\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"samples\": [\n          \"Movies\"\n        ],\n
\"num_unique_values\": 1,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

\"description\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"samples\": [\n   \"Louis C.K. muses on
religion, eternal love, giving dogs drugs, email fights, teachers and
more in a live performance from Washington, D.C.\"\n   ],\n
\"num_unique_values\": 3,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
#Missing value detection
df.isnull().sum().sort_values(ascending = False)  #only date column is
left, missing blanks should replace wd mode of date column else string
value cn chgne dtype
```

```
date_added      10
show_id          0
type             0
title            0
director         0
cast             0
country          0
release_year     0
rating           0
duration         0
listed_in        0
description      0
dtype: int64
```

```
df["date_added"].fillna(df["date_added"].mode()[0],inplace = True)
df
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 8807,\n \"fields\":
[\n   {\n   \"column\": \"show_id\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"samples\": [\n   \"s4971\",\n
\"s3363\",\n   \"s5495\"\n   ],\n
\"num_unique_values\": 8807,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"type\",\n   \"properties\": {\n   \"dtype\": \"category\",\n
\"samples\": [\n   \"TV Show\",\n   \"Movie\"\
n   ],\n   \"num_unique_values\": 2,\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"title\",\n   \"properties\": {\
n   \"dtype\": \"string\",\n   \"samples\": [\n
\"Game Over, Man!\",\n   \"Arsenio Hall: Smart & Classy\"\n
],\n   \"num_unique_values\": 8807,\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"director\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"samples\": [\n   \"Babak Anvari\",\n
\"Maria Ripoll\"\n   ],\n   \"num_unique_values\": 4529,\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"cast\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"samples\": [\n   \"Vittoria

```
Puccini, Francesco Scianna, Camilla Filippi, Simone Colombari,
Maurizio Lastrico, Alessandro Averone, Euridice Axen, Marco Baliani,
Pia Lanciotti, Giordano De Plano, Roberto Herlitzka, Tommaso Ragno,
Margherita Caviezel, Michele Morrone\",\n          \"Banky Wellington,
Rahama Sadau, Kanayo O. Kanayo, Ibrahim Suleiman, Michelle Dede,
Adesua Etomi, Hilda Dokubo, Akin Lewis\"\n        ],\n
\"num_unique_values\": 7693,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"country\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"United States,
United Kingdom, Denmark, Sweden\",\n          \"Spain, France\"\n
],\n        \"num_unique_values\": 749,\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        \"samples\": [\n
\"2016-04-20 00:00:00\",\n          \"2017-01-10 00:00:00\"\
n        ],\n        \"num_unique_values\": 1714,\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
\"samples\": [\n          1996,\n          1969\n        ],\n
\"num_unique_values\": 74,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"G\",\n
\"Unknown\"\n        ],\n        \"num_unique_values\": 15,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"duration\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"samples\": [\n
\"37 min\",\n          \"177 min\"\n        ],\n
\"num_unique_values\": 220,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"listed_in\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"Crime TV Shows,
International TV Shows, TV Mysteries\",\n          \"Children & Family
Movies, Classic Movies, Dramas\"\n        ],\n
\"num_unique_values\": 514,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"description\",\n      \"properties\": {\n        \"dtype\":
\"string\",\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
\"num_unique_values\": 8775,\n        \"semantic_type\": \"\",\n
```

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

```
#Missing value detection
df.isnull().sum().sort_values(ascending = False)  #only date column is left, missing blanks should replace wd mode of date column else string value cn chgne dtype
```

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

```
df
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 8807,\n \"fields\":
[\n   {\n   \"column\": \"show_id\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"samples\": [\n   \"s4971\",\n
\"s3363\",\n   \"s5495\"\n   ],\n
\"num_unique_values\": 8807,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"type\",\n   \"properties\": {\n   \"dtype\": \"category\",\n
\"samples\": [\n   \"TV Show\",\n   \"Movie\"\
n   ],\n   \"num_unique_values\": 2,\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"title\",\n   \"properties\": {\
n   \"dtype\": \"string\",\n   \"samples\": [\n
\"Game Over, Man!\",\n   \"Arsenio Hall: Smart & Classy\"\n
],\n   \"num_unique_values\": 8807,\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"director\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"samples\": [\n   \"Babak Anvari\",\n
\"Maria Ripoll\"\n   ],\n   \"num_unique_values\": 4529,\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"cast\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"samples\": [\n   \"Vittoria
Puccini, Francesco Scianna, Camilla Filippi, Simone Colombari,
Maurizio Lastrico, Alessandro Averone, Euridice Axen, Marco Baliani,
Pia Lanciotti, Giordano De Plano, Roberto Herlitzka, Tommaso Ragno,
Margherita Caviezel, Michele Morrone\",\n   \"Banky Wellington,
Rahama Sadau, Kanayo O. Kanayo, Ibrahim Suleiman, Michelle Dede,

```
Adesua Etomi, Hilda Dokubo, Akin Lewis\"\n        ],\n
\"num_unique_values\": 7693,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"country\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"United States,
United Kingdom, Denmark, Sweden\",\n          \"Spain, France\"\n
],\n        \"num_unique_values\": 749,\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        \"samples\": [\n
\"2016-04-20 00:00:00\",\n          \"2017-01-10 00:00:00\"\
n        ],\n        \"num_unique_values\": 1714,\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
\"samples\": [\n          1996,\n          1969\n        ],\n
\"num_unique_values\": 74,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"G\",\n
\"Unknown\"\n        ],\n        \"num_unique_values\": 15,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"duration\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"samples\": [\n
\"37 min\",\n          \"177 min\"\n        ],\n
\"num_unique_values\": 220,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"listed_in\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"Crime TV Shows,
International TV Shows, TV Mysteries\",\n          \"Children & Family
Movies, Classic Movies, Dramas\"\n        ],\n
\"num_unique_values\": 514,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"description\",\n      \"properties\": {\n        \"dtype\":
\"string\",\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
\"num_unique_values\": 8775,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
con1 = df['director'].apply(lambda x: str(x).split(', ')).tolist()

df_new1=pd.DataFrame(con1,index=df['title'])
df_new1
```

```
{"summary":"{\n  \"name\": \"df_new1\",\n  \"rows\": 8807,\n
\"fields\": [\n    {\n      \"column\": 0,\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"samples\": [\n          \"Cate
Shortland\",\n          \"Shawn Rech\",\n          \"Cal Seville\"\n
],\n        \"num_unique_values\": 4406,\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": 1,\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"Matthew McNeil\",\n
\"Tyler Measom\",\n          \"Viju Mane\"\n        ],\n
\"num_unique_values\": 541,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 2,\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"samples\": [\n          \"Alka Amarkant Dubey\",\n          \"Saket
Chaudhary\",\n          \"Hideki Futamura\"\n        ],\n
\"num_unique_values\": 69,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 3,\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"samples\": [\n          \"Erich Sturm\",\n          \"Lauren
MacMullan\",\n          \"Parkpoom Wongpoom\"\n        ],\n
\"num_unique_values\": 34,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 4,\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"samples\": [\n          \"Karthik Subbaraj\",\n          \"Wong Fei-
Pang\",\n          \"Steve Brill\"\n        ],\n
\"num_unique_values\": 20,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 5,\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"samples\": [\n          \"James Duffy\",\n          \"Sarah Adina
Smith\",\n          \"Arvind Swamy\"\n        ],\n
\"num_unique_values\": 13,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 6,\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"samples\": [\n          \"Jonathan van Tulleken\",\n
\"Scott Stewart\",\n          \"Rathindran R Prasad\"\n        ],\n
\"num_unique_values\": 13,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 7,\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"samples\": [\n          \"Jon YonKondy\",\n          \"Sarjun\",\n
\"Elizabeth Banks\"\n        ],\n        \"num_unique_values\": 11,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 8,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"samples\": [\n          \"Joann
Sfar\",\n          \"Abdullah Al Noor\",\n          \"Drue Metz\"\n
],\n        \"num_unique_values\": 10,\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": 9,\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"samples\": [\n          \"Shinji Takagi\",\n
\"Koji Sawai\",\n          \"Krishnendu Chattopadhyay\"\n        ],\n
\"num_unique_values\": 8,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 10,\n
```

\"properties\": {\n   \"dtype\": \"category\",\n
\"samples\": [\n   \"Suparn Verma\",\n   \"Rusty
Cundieff\",\n   \"Roger Allers\"\n   ],\n
\"num_unique_values\": 5,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": 11,\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"samples\": [\n   \"Mike Gabriel\",\n   \"Hiroshi
Yamazaki\",\n   \"James Gunn\"\n   ],\n
\"num_unique_values\": 3,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": 12,\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"samples\": [\n   \"Mark Henn\"\n   ],\n
\"num_unique_values\": 1,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df_new1"}

```
constraint1 = df['director'].apply(lambda x: str(x).split(',
')).tolist()
df_new1=pd.DataFrame(constraint1,index=df['title'])
df_new1=df_new1.stack()
df_new1=pd.DataFrame(df_new1.reset_index())
df_new1.sample(20)
```

{"summary":"{\n \"name\": \"df_new1\",\n \"rows\": 20,\n
\"fields\": [\n   {\n   \"column\": \"title\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"samples\": [\n   \"Harvey Street Kids\",\n   \"2015
Dream Concert\",\n   \"You Are My Spring\"\n   ],\n
\"num_unique_values\": 20,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"level_1\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 0,\n
\"samples\": [\n   0\n   ],\n
\"num_unique_values\": 1,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": 0,\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"samples\": [\n   \"Bilal Lashari\"\n   ],\n
\"num_unique_values\": 8,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
#unnesting the directors column, i.e- creating separate lines for each
director in a movie
constraint1 = df['director'].apply(lambda x: str(x).split(',
')).tolist()
df_new1=pd.DataFrame(constraint1,index=df['title'])
df_new1=df_new1.stack()
df_new1=pd.DataFrame(df_new1.reset_index())
df_new1.rename(columns={0:'Directors'},inplace=True)
df_new1.drop(['level_1'],axis=1,inplace=True)
df_new1.head()
```

{"summary":"{\n \"name\": \"df_new1\",\n \"rows\": 9612,\n
\"fields\": [\n   {\n   \"column\": \"title\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"samples\": [\n   \"Game Over, Man!\",\n   \"Arsenio
Hall: Smart & Classy\",\n   \"Kazoops!\"\n   ],\n
\"num_unique_values\": 8807,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Directors\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"samples\": [\n   \"Lasse Hallstr\\
u00f6m\",\n   \"Terrie Samundra\",\n   \"Florian
Schnell\"\n   ],\n   \"num_unique_values\": 4994,\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"df_new1"}

```
#unnesting the cast column, i.e- creating separate lines for each cast member in a movie
constraint2=df['cast'].apply(lambda x: str(x).split(', ')).tolist()
df_new2=pd.DataFrame(constraint2,index=df['title'])
df_new2=df_new2.stack()
df_new2=pd.DataFrame(df_new2.reset_index())
df_new2.rename(columns={0:'Actors'},inplace=True)
df_new2.drop(['level_1'],axis=1,inplace=True)
df_new2.head()
```

{"summary":"{\n \"name\": \"df_new2\",\n \"rows\": 64951,\n
\"fields\": [\n   {\n   \"column\": \"title\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"samples\": [\n   \"Game Over, Man!\",\n   \"Arsenio
Hall: Smart & Classy\",\n   \"Kazoops!\"\n   ],\n
\"num_unique_values\": 8807,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Actors\",\n   \"properties\": {\n   \"dtype\": \"string\",\n
\"samples\": [\n   \"Robert Catrini\",\n   \"Jonathan
Sandor\",\n   \"Gautam Kurup\"\n   ],\n
\"num_unique_values\": 36440,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df_new2"}

```
#unnesting the listed_in column, i.e- creating separate lines for each genre in a movie
constraint3=df['listed_in'].apply(lambda x: str(x).split(',
')).tolist()
df_new3=pd.DataFrame(constraint3,index=df['title'])
df_new3=df_new3.stack()
df_new3=pd.DataFrame(df_new3.reset_index())
df_new3.rename(columns={0:'Genre'},inplace=True)
df_new3.drop(['level_1'],axis=1,inplace=True)
df_new3.head()
```

{"summary":"{\n \"name\": \"df_new3\",\n \"rows\": 19323,\n
\"fields\": [\n   {\n   \"column\": \"title\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"samples\": [\n   \"Game Over, Man!\",\n   \"Arsenio
Hall: Smart & Classy\",\n   \"Kazoops!\"\n   ],\n
\"num_unique_values\": 8807,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Genre\",\n   \"properties\": {\n   \"dtype\": \"category\",\
n   \"samples\": [\n   \"Action & Adventure\",\n
\"Independent Movies\",\n   \"Romantic TV Shows\"\n   ],\n
\"num_unique_values\": 42,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df_new3"}

```
#unnesting the country column, i.e- creating separate lines for each
country in a movie
constraint4=df['country'].apply(lambda x: str(x).split(', ')).tolist()
df_new4=pd.DataFrame(constraint4,index=df['title'])
df_new4=df_new4.stack()
df_new4=pd.DataFrame(df_new4.reset_index())
df_new4.rename(columns={0:'country'},inplace=True)
df_new4.drop(['level_1'],axis=1,inplace=True)
df_new4.head()
```

{"summary":"{\n \"name\": \"df_new4\",\n \"rows\": 10845,\n
\"fields\": [\n   {\n   \"column\": \"title\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"samples\": [\n   \"Game Over, Man!\",\n   \"Arsenio
Hall: Smart & Classy\",\n   \"Kazoops!\"\n   ],\n
\"num_unique_values\": 8807,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"country\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"samples\": [\n   \"Syria\",\n
\"Bulgaria\",\n   \"Spain\"\n   ],\n
\"num_unique_values\": 128,\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df_new4"}

```
#merging the unnested director data with unnested actors data
df_new5=df_new2.merge(df_new1,on=['title'],how='inner')
#merging the above merged data with unnested genre data
df_new6=df_new5.merge(df_new3,on=['title'],how='inner')
#merging the above merged data with unnested country data
df_new=df_new6.merge(df_new4,on=['title'],how='inner')
#replacing nan values of director and actor by Unknown Actor and
Director
df_new['Actors'].replace(['nan'],['Unknown Actor'],inplace=True)
df_new['Directors'].replace(['nan'],['Unknown Director'],inplace=True)
df_new['country'].replace(['nan'],[np.nan],inplace=True)
df_new.head()
```

```
{"type":"dataframe","variable_name":"df_new"}

#merging our nested data with original data
df_final=df_new.merge(df[['show_id', 'type', 'title', 'date_added',
       'release_year', 'rating', 'duration']],on=['title'],how='left')
df_final.head()

{"type":"dataframe","variable_name":"df_final"}

df_final.shape

(201991, 11)
```

Univariate Analysis for each column- type, genre, actors, country, directors

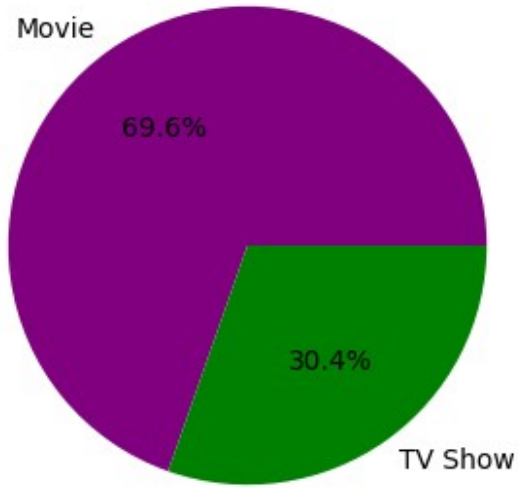
```
#number of distinct titles on the basis of type
df_final.groupby('type').agg({"title":"nunique"})

{"summary":"{\n  \"name\": \"df_final\",\n  \"rows\": 2,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2443,\n        \"min\": 2676,\n        \"max\": 6131,\n
\"samples\": [\n          2676,\n          6131\n        ],\n
\"num_unique_values\": 2,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}

df_type=df_final.groupby('type').agg({"title":"nunique"}).reset_index(
)
df_type

{"summary":"{\n  \"name\": \"df_type\",\n  \"rows\": 2,\n  \"fields\":
[\n    {\n      \"column\": \"type\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"samples\": [\n          \"TV
Show\",\n          \"Movie\"\n        ],\n
\"num_unique_values\": 2,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"title\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 2443,\n        \"min\": 2676,\n        \"max\": 6131,\n
\"samples\": [\n          2676,\n          6131\n        ],\n
\"num_unique_values\": 2,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df_type"}

plt.figure(figsize = (4,4))
plt.pie(df_type['title'], explode = None, labels=
df_type['type'],colors=['purple','green'],autopct='%.1f%%')
plt.show()
```



There is 70:30 ratio of Movies and TV Shows in data

```
#number of distinct titles on the basis of genre
df_final.groupby('Genre').agg({'title':'nunique'})
```

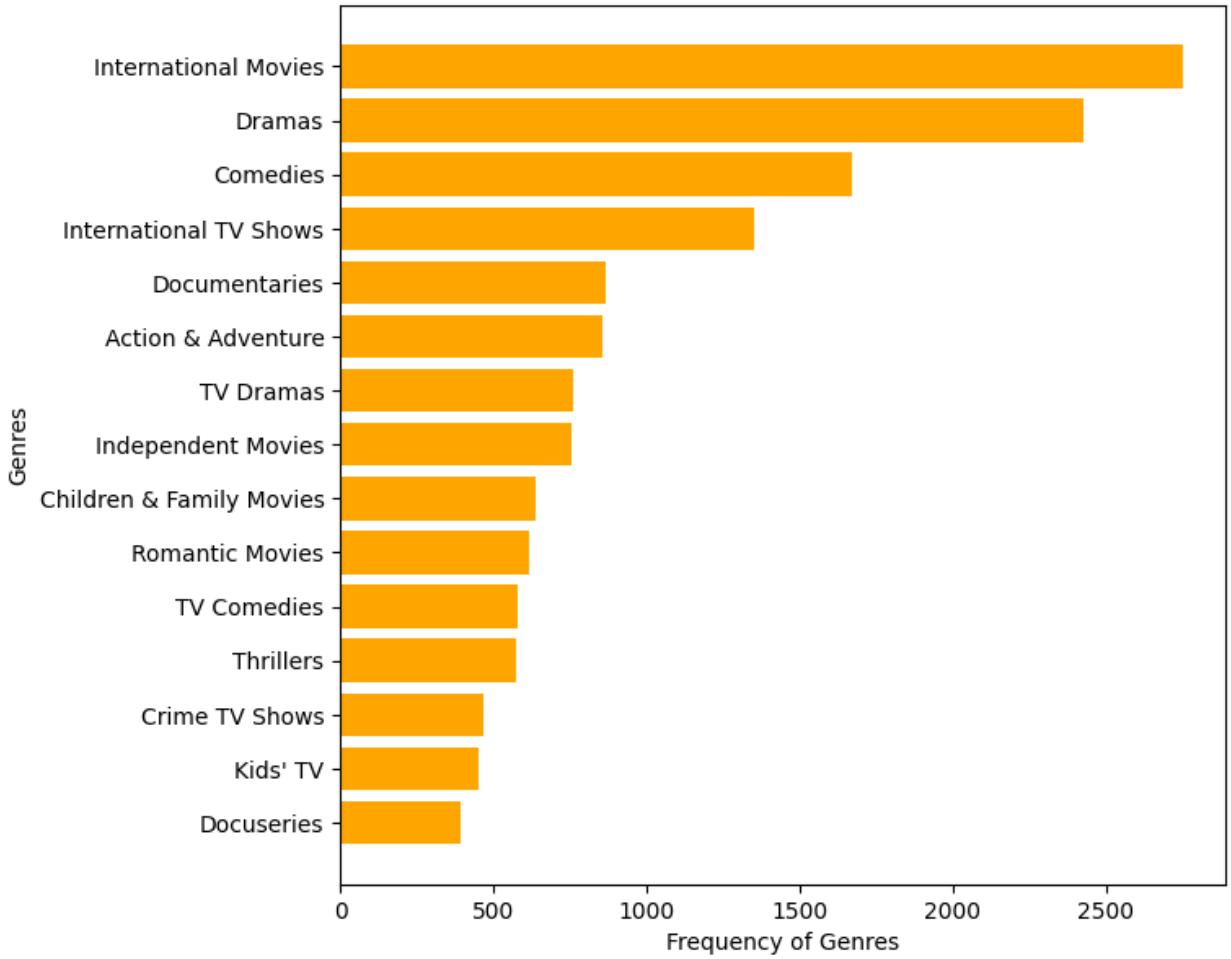

```
{"summary":"{\n  \"name\": \"df_final\",\n  \"rows\": 42,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
601,\n        \"min\": 16,\n        \"max\": 2752,\n
\"samples\": [\n          102,\n          1351,\n          2752\n
],\n        \"num_unique_values\": 40,\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```


```
df_genre =
df_final.groupby('Genre').agg({'title':'nunique'}).reset_index().sort_
values(by=['title'],ascending=False)[:15]
df_genre
```


```
{"summary":"{\n  \"name\": \"df_genre\",\n  \"rows\": 15,\n
\"fields\": [\n    {\n      \"column\": \"Genre\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"samples\": [\n          \"Romantic Movies\",\n
\"Thrillers\",\n          \"International Movies\"\n        ],\n
\"num_unique_values\": 15,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"title\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 727,\n        \"min\": 395,\n        \"max\": 2752,\n
\"samples\": [\n          616,\n          577,\n          2752\n
],\n        \"num_unique_values\": 15,\n        \"semantic_type\":
```

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

```
plt.figure(figsize = (7,7))
plt.barh(df_genre[::-1]['Genre'], df_genre[::-1]
['title'],color=['orange'])
plt.xlabel('Frequency of Genres')
plt.ylabel('Genres')
plt.show()
```



International Movies, Dramas, Comedies are the genres with high frequency

```
#number of distinct titles on the basis of country
df_final.groupby('country').agg({'title':'nunique'})
```

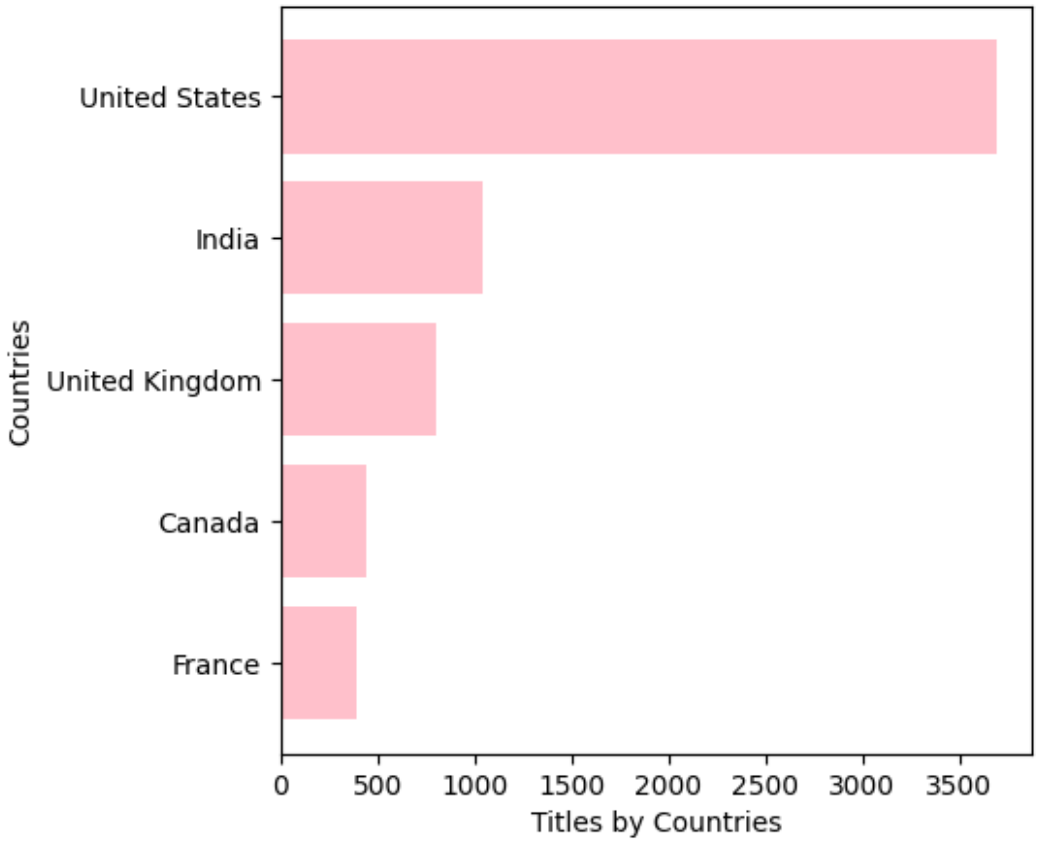
{"summary":"{\n \"name\": \"df_final\",\n \"rows\": 128,\n
\"fields\": [\n   {\n   \"column\": \"title\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
354,\n   \"min\": 1,\n   \"max\": 3689,\n
\"samples\": [\n   2,\n   12,\n   50\n   ],\

```
n        \"num_unique_values\": 56,\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```

```
df_country = df_final[df_final['country']!=
'Unknown'].groupby('country').agg({'title':'nunique'}).reset_index().s
ort_values(by = ['title'], ascending = False)[:5]
df_country
```

```
{"summary":"{\n  \"name\": \"df_country\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"country\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"samples\": [\n          \"India\",\n          \"France\",\n
\"United Kingdom\"\n        ],\n        \"num_unique_values\": 5,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"title\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 1375,\n
\"min\": 393,\n        \"max\": 3689,\n        \"samples\": [\n
1046,\n          393,\n          804\n        ],\n
\"num_unique_values\": 5,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df_country"}
```

```
plt.figure(figsize = (5,5))
plt.barh(df_country[::-1]['country'], df_country[::-1]['title'], color
= 'pink')
plt.xlabel('Titles by Countries')
plt.ylabel('Countries')
plt.show()
```



US,India,UK,Canada and France are leading countries in Content Creation

```
##number of distinct titles on the basis of directors
df_final.groupby('Directors').agg({'title':'nunique'})
```

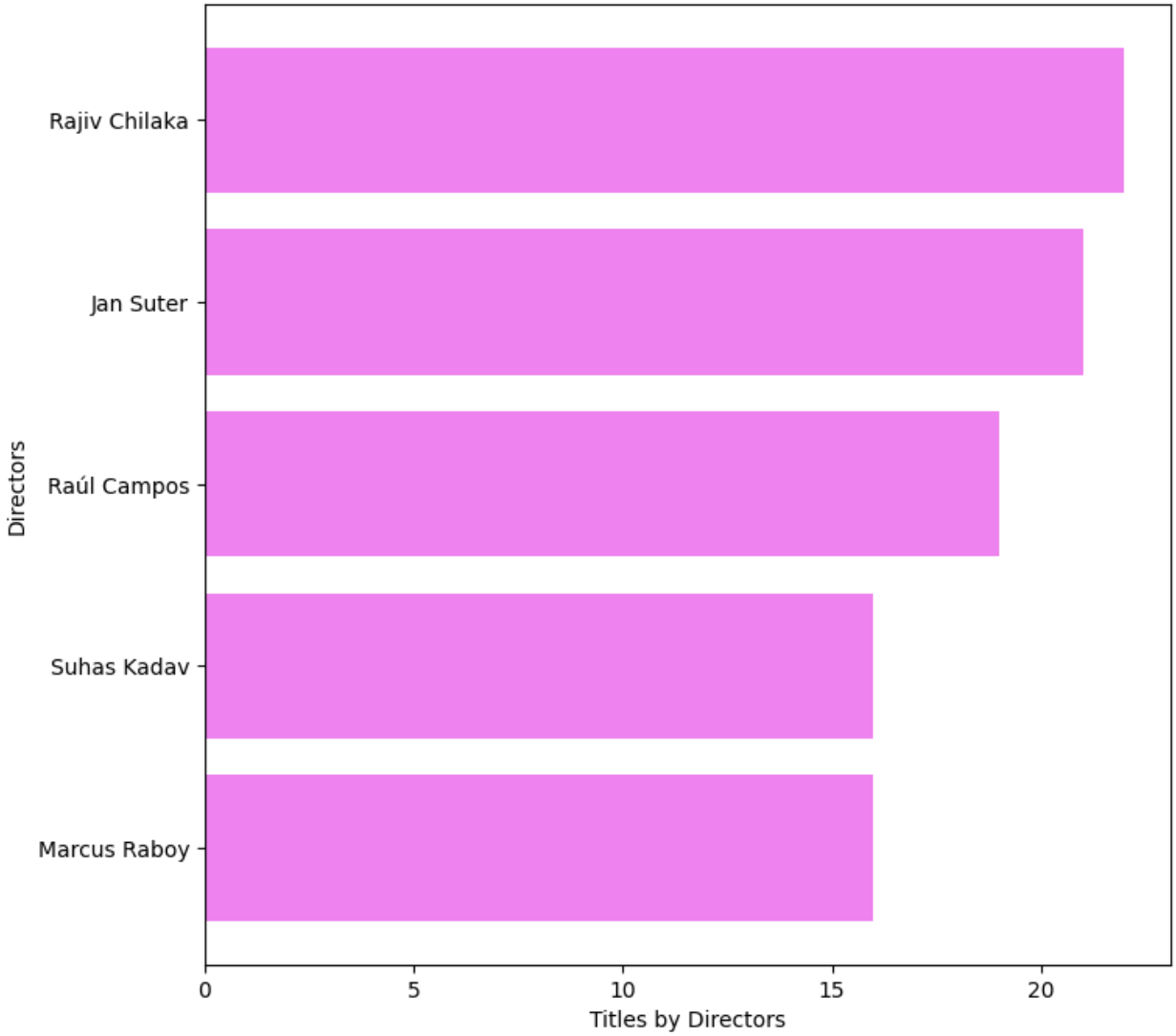

```
{"summary":"{\n  \"name\": \"df_final\",\n  \"rows\": 4994,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
37,\n        \"min\": 1,\n        \"max\": 2634,\n        \"samples\":
[\n          2,\n          6,\n          21\n        ],\n
\"num_unique_values\": 19,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
df_director = df_final[df_final['Directors'] !=
'Unknown'].groupby('Directors').agg({'title':'nunique'}).reset_index()
.sort_values(by = ['title'], ascending = False)[:5]
df_director
```


```
{"summary":"{\n  \"name\": \"df_director\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"Directors\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"samples\": [\n          \"Jan Suter\",\n          \"Marcus Raboy\",\
n          \"Ra\\u00fal Campos\"\n        ],\n
\"num_unique_values\": 5,\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"title\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 2,\n        \"min\": 16,\n        \"max\": 22,\n
\"samples\": [\n          21,\n          16,\n          22\
n        ],\n        \"num_unique_values\": 4,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df_director"}
```

```
plt.figure(figsize = (8,8))
plt.barh(df_director[::-1]['Directors'], df_director[::-1]['title'],
color = 'violet')
plt.xlabel('Titles by Directors')
plt.ylabel('Directors')
plt.show()
```



Rajiv Chilaka, Jan Suter and Raul Campos are the most popular directors across Netflix

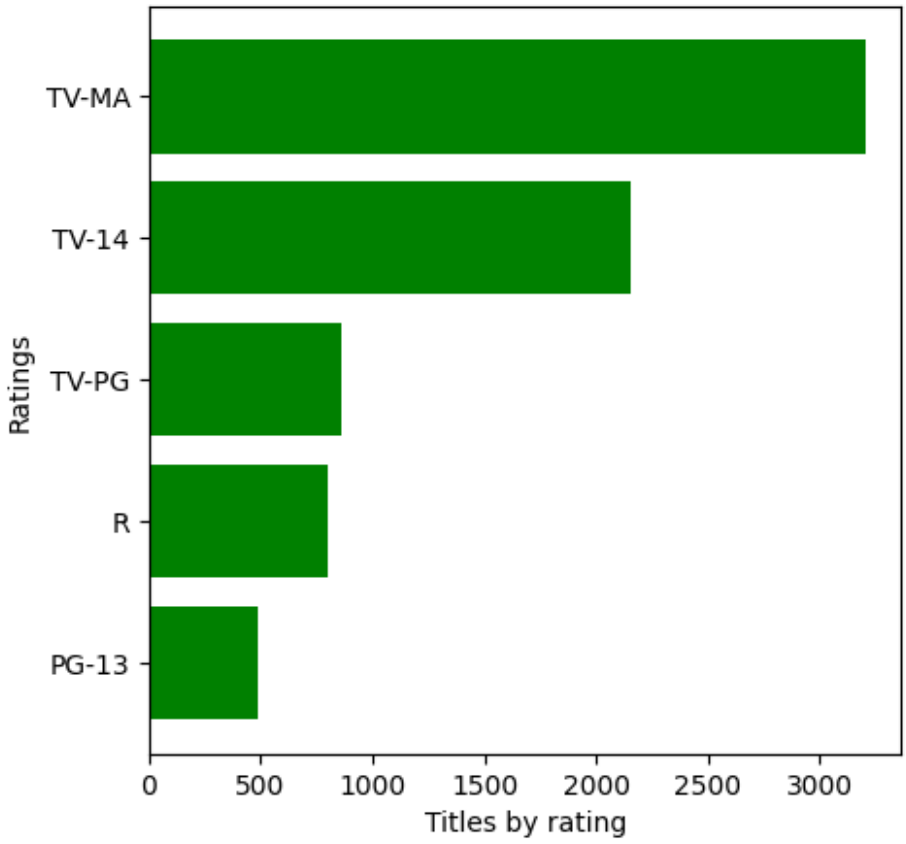
```
##number of distinct titles on the basis of rating
df_final.groupby('rating').agg({'title':'nunique'})
```


```
{"summary":"{\n  \"name\": \"df_final\",\n  \"rows\": 15,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
916,\n        \"min\": 3,\n        \"max\": 3207,\n
\"samples\": [\n          863,\n          334,\n          41\
n        ],\n        \"num_unique_values\": 14,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```
df_rating =
df_final.groupby('rating').agg({'title':'nunique'}).sort_values(by =
['title'], ascending = False).reset_index()[:5]
df_rating
```


```
{"summary":"{\n  \"name\": \"df_rating\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"rating\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"samples\": [\n          \"TV-14\",\n          \"PG-13\",\n
\"TV-PG\"\n        ],\n        \"num_unique_values\": 5,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"title\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 1147,\n
\"min\": 490,\n        \"max\": 3207,\n        \"samples\": [\n
2160,\n          490,\n          863\n        ],\n
\"num_unique_values\": 5,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df_rating"}
```


```
plt.figure(figsize = (5,5))
plt.barh(df_rating[::-1]['rating'], df_rating[::-1]['title'], color =
'green')
plt.xlabel('Titles by rating')
plt.ylabel('Ratings')
plt.show()
```

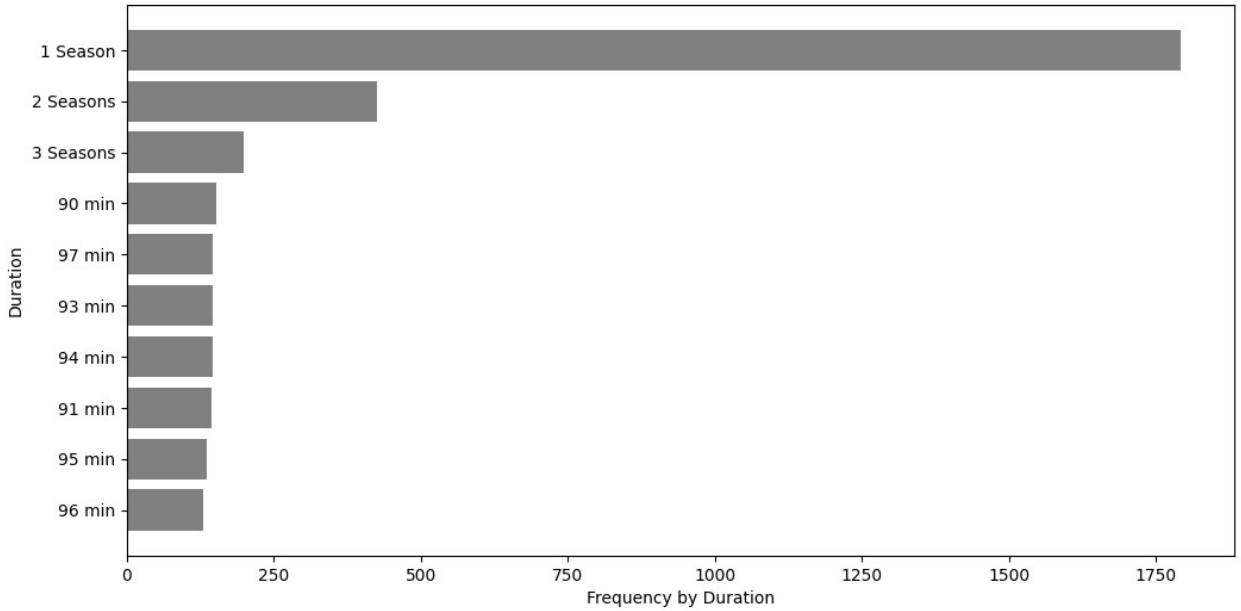


TV-MA, TV-14, TV-PG, R and PG-13 are the top ratings among the Netflix content

```
#number of distinct titles on the basis of duration
df_final.groupby(['duration']).agg({"title":"nunique"})
```


```
{"summary":"{\n  \"name\": \"df_final\",\n  \"rows\": 220,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
127,\n        \"min\": 1,\n        \"max\": 1793,\n
\"samples\": [\n          32,\n          1793,\n          3\
n        ],\n        \"num_unique_values\": 80,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```
df_duration=df_final.groupby(['duration']).agg({"title":"nunique"}).re
set_index().sort_values(by=['title'],ascending=False)[:10]
plt.figure(figsize=(12,6))
plt.barh(df_duration[::-1]['duration'], df_duration[::-1]
['title'],color='grey')
plt.xlabel('Frequency by Duration')
plt.ylabel('Duration')
plt.show()
```

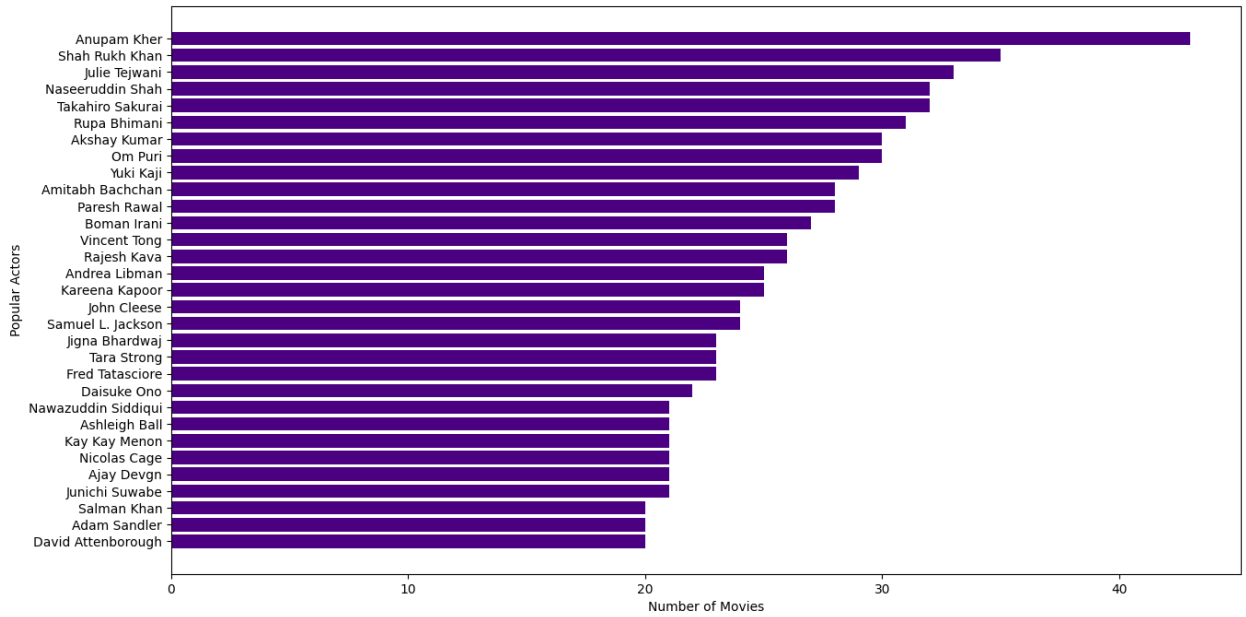


Season wise- content with one season is most watched and 90-100 mins is most watched Movies/TV-show on minutes basis.

```
#number of distinct titles on the basis of Actors
df_final.groupby(['Actors']).agg({"title":"nunique"})
```


```
{"summary":"{\n  \"name\": \"df_final\",\n  \"rows\": 36440,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
4,\n        \"min\": 1,\n        \"max\": 825,\n        \"samples\":
[\n          29,\n          12,\n          22\n        ],\n
\"num_unique_values\": 36,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
df_actors=df_final[df_final['Actors']!
='Unknown'].groupby(['Actors']).agg({"title":"nunique"}).reset_index()
.sort_values(by=['title'],ascending=False)[:31]
plt.figure(figsize=(15,8))
plt.barh(df_actors[::-1]['Actors'], df_actors[::-1]
['title'],color=['indigo'])
plt.xlabel('Number of Movies')
plt.ylabel('Popular Actors')
plt.show()
```



Anupam Kher, SRK and Julie tejwani are top 3 actors whose content has been watched the most.

```
df_final
```


```
{"type":"dataframe","variable_name":"df_final"}
```


```
# Convert the date_added column to datetime format
df_final['date_added'] = pd.to_datetime(df_final['date_added'],
format="%Y=%m-%d")

# Change the format to YYYY-MM-DD
df_final['date_added'] = df_final['date_added'].dt.strftime('%Y-%m-
%d')

#number of distinct titles on the basis of year of date_added
df_final['year_only'] = df_final['date_added'].apply(lambda x:
x.split('-')[0])
df_final['month_only'] = df_final['date_added'].apply(lambda x:
x.split('-')[1])
df_final.head()
```

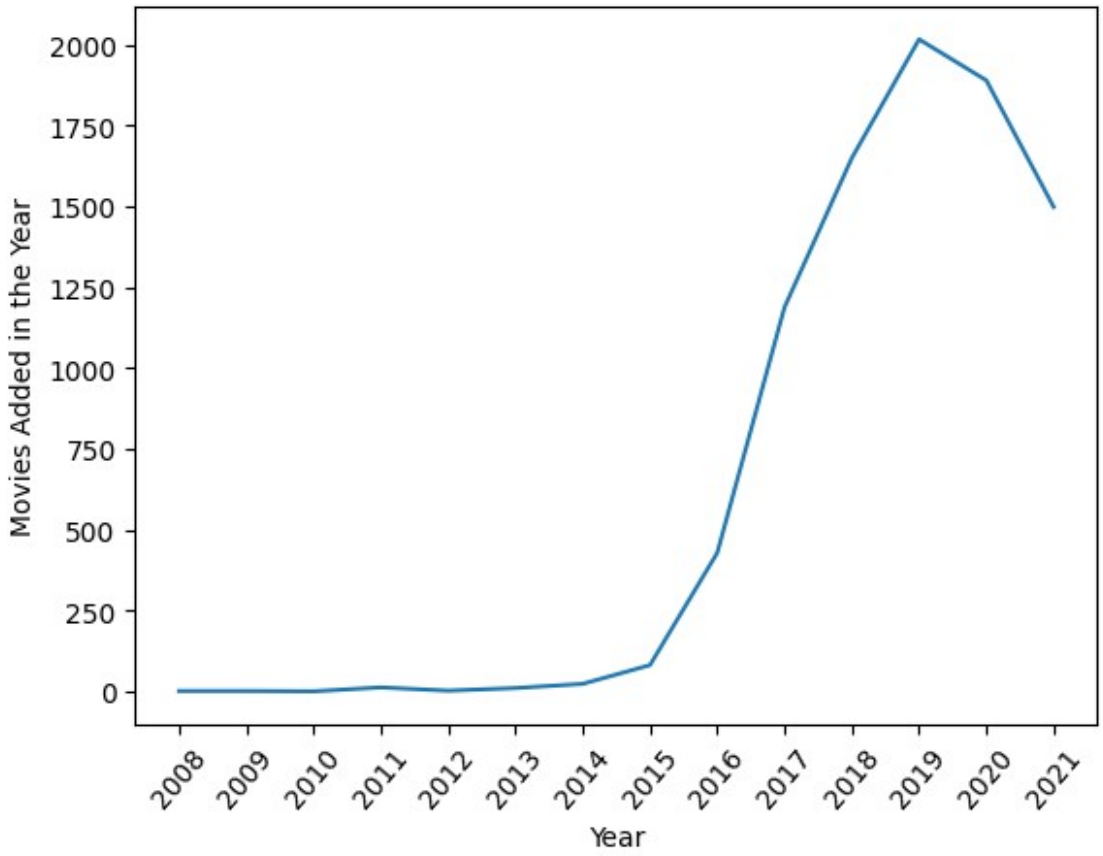

```
{"type":"dataframe","variable_name":"df_final"}
```


```
#number of distinct titles on the basis of year
df_final.groupby(['year_only']).agg({"title":"nunique"})
```


```
{"summary":"{\n  \"name\": \"df_final\",\n  \"rows\": 14,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
816,\n        \"min\": 1,\n        \"max\": 2016,\n
\"samples\": [\n          1889,\n          1649,\n          2\n
],\n        \"num_unique_values\": 13,\n        \"semantic_type\":
```

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

```
df_year=df_final.groupby(['year_only']).agg({"title":"nunique"}).reset
_index()
sns.lineplot(data=df_year, x='year_only', y='title')
plt.ylabel("Movies Added in the Year")
plt.xlabel("Year")
plt.xticks(rotation = 50)
plt.show()
```



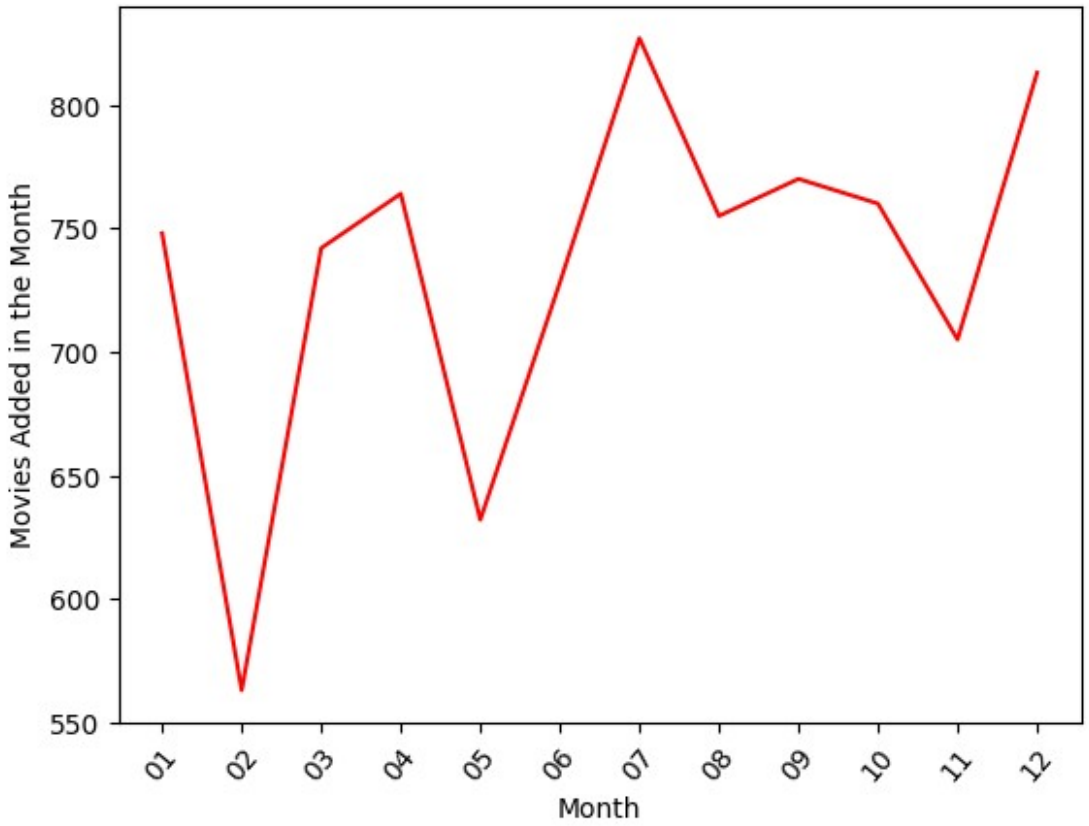
Highest number of movies/tv_shows were added from year 2015-2019 but decreased after probabily due to Covid.

```
#number of distinct titles on the basis of month
df_final.groupby(['month_only']).agg({"title":"nunique"})
```

{"summary":"{\n \"name\": \"df_final\",\n \"rows\": 12,\n
\"fields\": [\n   {\n   \"column\": \"title\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
73,\n   \"min\": 563,\n   \"max\": 827,\n
\"samples\": [\n   705,\n   760,\n   748\n
],\n   \"num_unique_values\": 12,\n   \"semantic_type\":

```
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```

```
df_month=df_final.groupby(['month_only']).agg({"title":"nunique"}).res
et_index()
sns.lineplot(data=df_month, x='month_only', y='title', color = 'red')
plt.ylabel("Movies Added in the Month")
plt.xlabel("Month")
plt.xticks(rotation = 50)
plt.show()
```



the amount of content has been increased from Feb-April, May to July month and Nov-Dec.

#Comparison of TV Shows vs Movies

```
df_shows = df_final[df_final['type']== 'TV Show']
df_movies= df_final[df_final['type']=='Movie']
df_movies
```

```
{"type":"dataframe","variable_name":"df_movies"}
```

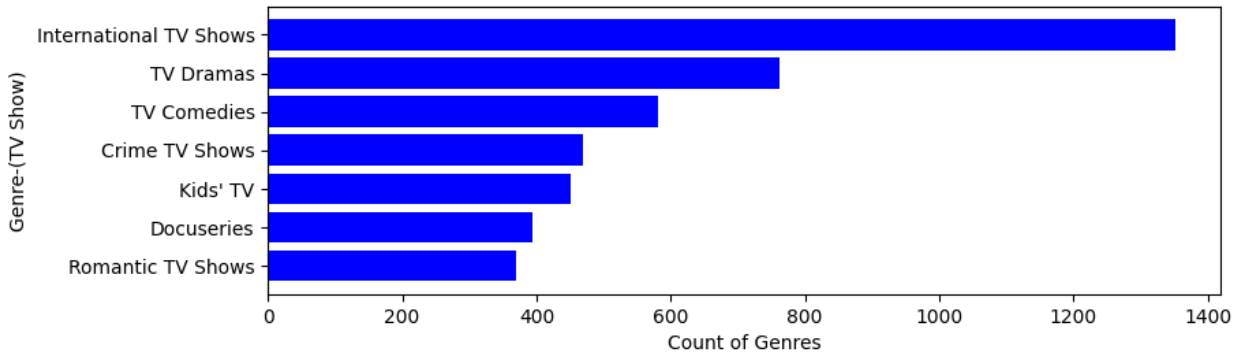
```
df_genre_shows = df_shows.groupby('Genre').agg({'title': 'nunique'})
df_genre_movies = df_movies.groupby('Genre').agg({'title': 'nunique'})
df_country_shows = df_shows.groupby('country').agg({'title':
```

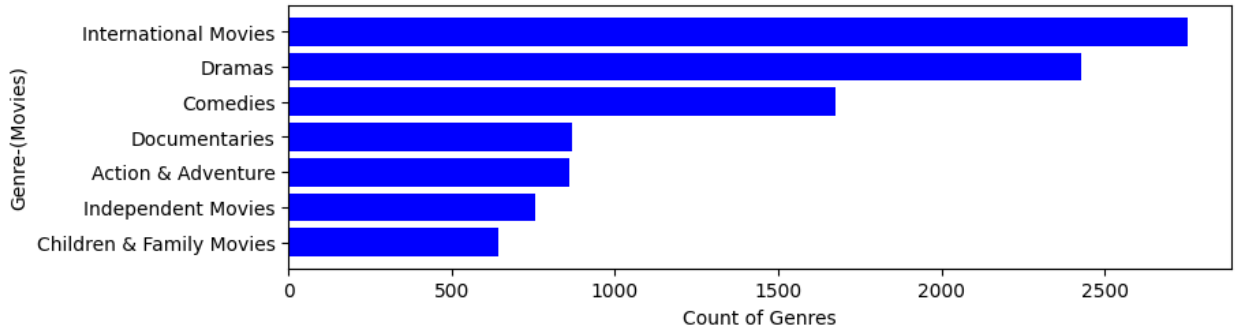
```
'nunique'})
df_country_movies = df_movies.groupby('country').agg({'title':
'nunique'})
df_rating_shows = df_shows.groupby('rating').agg({'title': 'nunique'})
df_rating_movies = df_movies.groupby('rating').agg({'title':
'nunique'})
df_directors_shows = df_shows.groupby('Directors').agg({'title':
'nunique'})
df_directors_movies = df_movies.groupby('Directors').agg({'title':
'nunique'})
df_rating_movies
```


```
{"summary":"{\n  \"name\": \"df_rating_movies\",\n  \"rows\": 15,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
603,\n        \"min\": 3,\n        \"max\": 2062,\n
\"samples\": [\n          139,\n          540,\n          41\
n        ],\n        \"num_unique_values\": 13,\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df_rating_movies"}
```


```
df_genre_shows = df_shows.groupby('Genre').agg({'title':
'nunique'}).reset_index().sort_values(by=['title'],ascending=False)
[:7]
df_genre_movies = df_movies.groupby('Genre').agg({'title':
'nunique'}).reset_index().sort_values(by=['title'],ascending=False)
[:7]
plt.figure(figsize= (9,6))
plt.subplot(2,1,1)
plt.barh(df_genre_shows[::-1]['Genre'], df_genre_shows[::-1]['title'],
color = 'blue')
plt.ylabel('Genre-(TV Show)')
plt.xlabel("Count of Genres")
plt.show()

plt.figure(figsize= (8.92,5.5))
plt.subplot(2,1,2)
plt.barh(df_genre_movies[::-1]['Genre'], df_genre_movies[::-1]
['title'], color = 'blue')
plt.ylabel('Genre-(Movies)')
plt.xlabel("Count of Genres")
plt.show()
```





For both TV shows and Movies- International Movies, Dramas and Comedies are most watched content

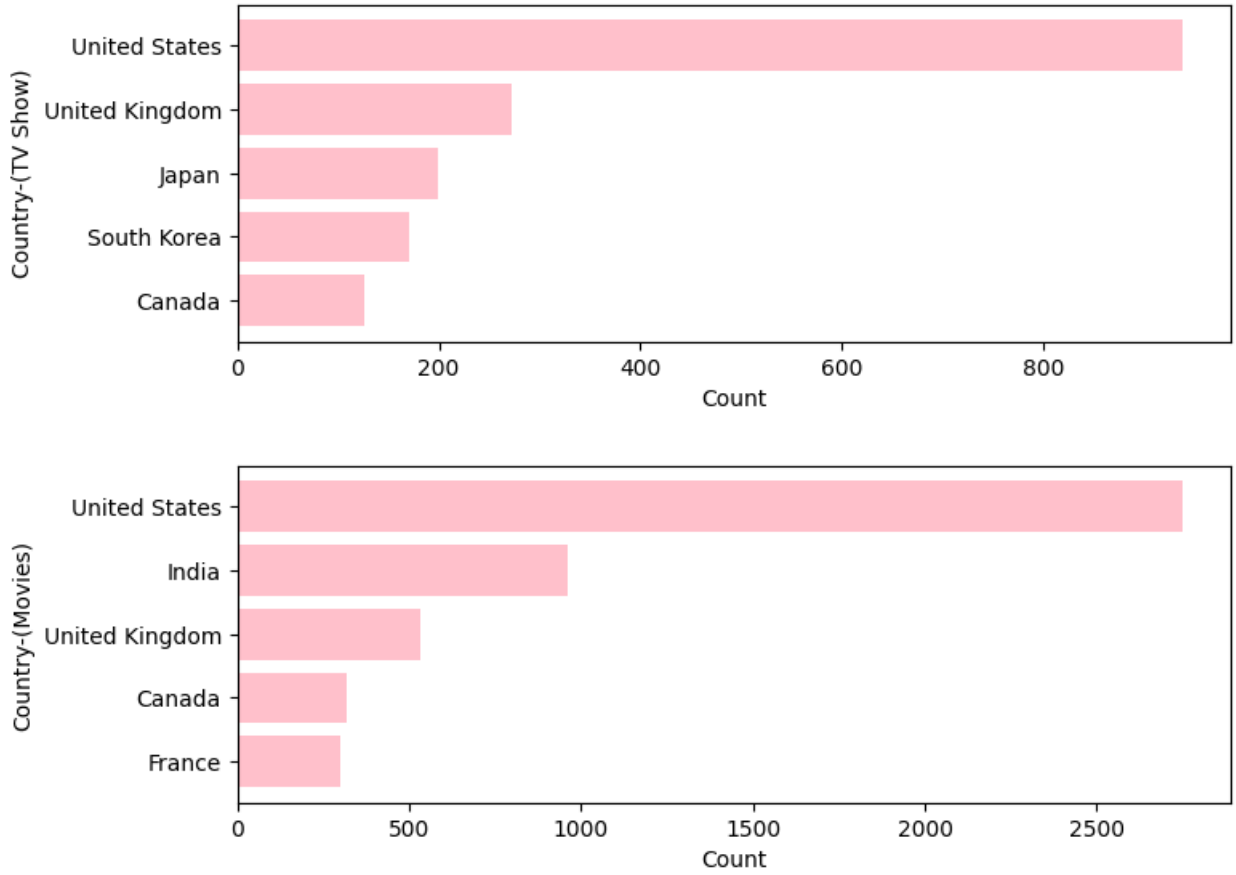
```
df_country_shows = df_shows[df_shows !=
'Unknown'].groupby('country').agg({'title':
'nunique'}).reset_index().sort_values(by=['title'],ascending=False)
[:5]
df_country_movies = df_movies[df_movies !=
'Unknown'].groupby('country').agg({'title':
'nunique'}).reset_index().sort_values(by=['title'],ascending=False)
[:5]

plt.figure(figsize = (8,6))

plt.subplot(2,1,1)
plt.barh(df_country_shows[::-1]['country'], df_country_shows[::-1]
['title'], color = 'pink')
plt.ylabel('Country-(TV Show)')
plt.xlabel("Count")
plt.show()

plt.figure(figsize = (8,6))
plt.subplot(2,1,2)
plt.barh(df_country_movies[::-1]['country'], df_country_movies[::-1]
['title'], color = 'pink')
plt.ylabel('Country-(Movies)')
```

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



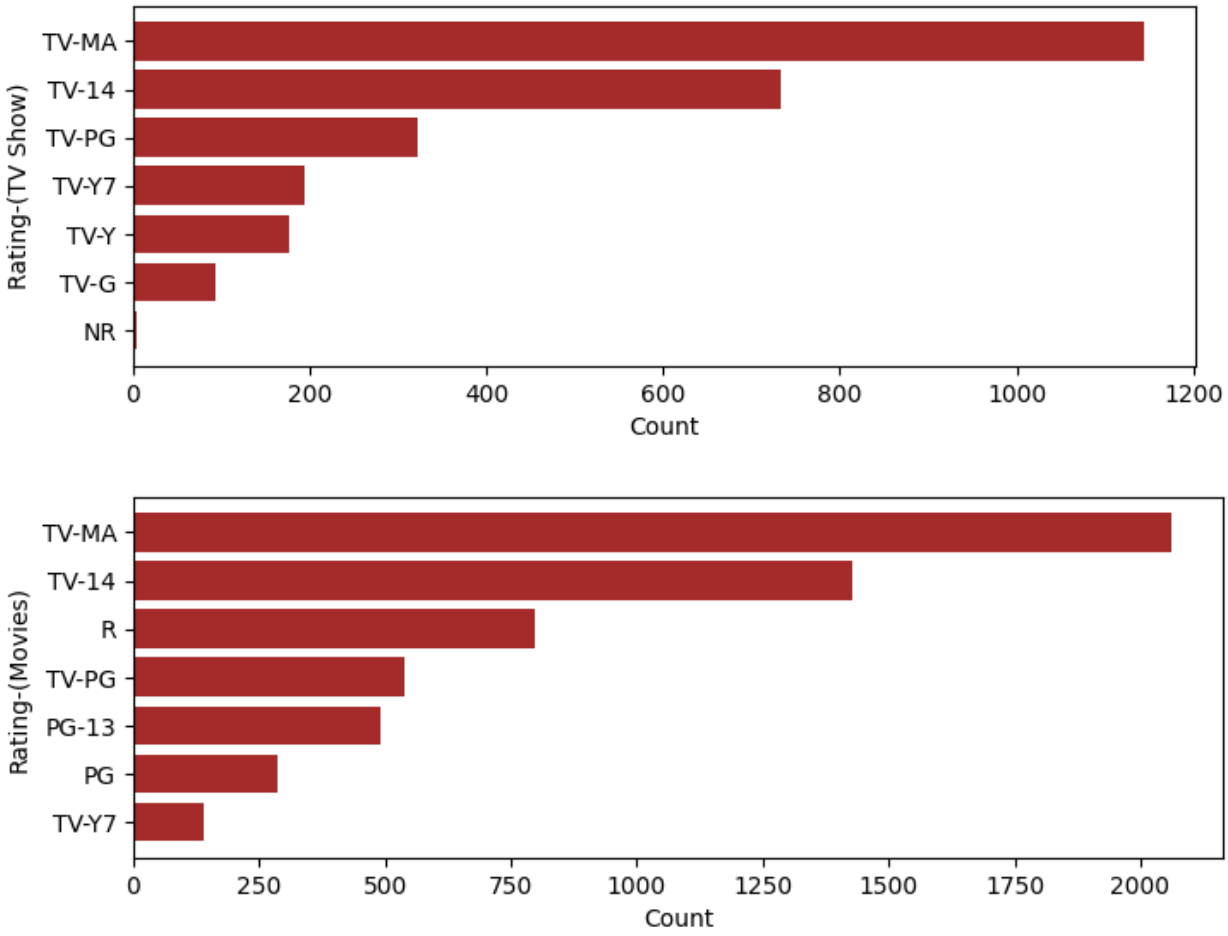
Most TV Shows are produced in US, UK, Japan, South Korea and Japan Most Movies are produced in US, UK, India, Canada and France

```
df_rating_shows = df_shows.groupby('rating').agg({'title':
'nunique'}).reset_index().sort_values(by=['title'],ascending=False)
[:7]
df_rating_movies = df_movies.groupby('rating').agg({'title':
'nunique'}).reset_index().sort_values(by=['title'],ascending=False)
[:7]

plt.figure(figsize = (8,6))
plt.subplot(2,1,1)
plt.barh(df_rating_shows[::-1]['rating'], df_rating_shows[::-1]
['title'], color = 'brown')
plt.ylabel('Rating-(TV Show)')
plt.xlabel("Count")
plt.show()

plt.figure(figsize = (8.2,6))
```

```
plt.subplot(2,1,2)
plt.barh(df_rating_movies[::-1]['rating'], df_rating_movies[::-1]
['title'], color = 'brown')
plt.ylabel('Rating-(Movies)')
plt.xlabel("Count")
plt.show()
```



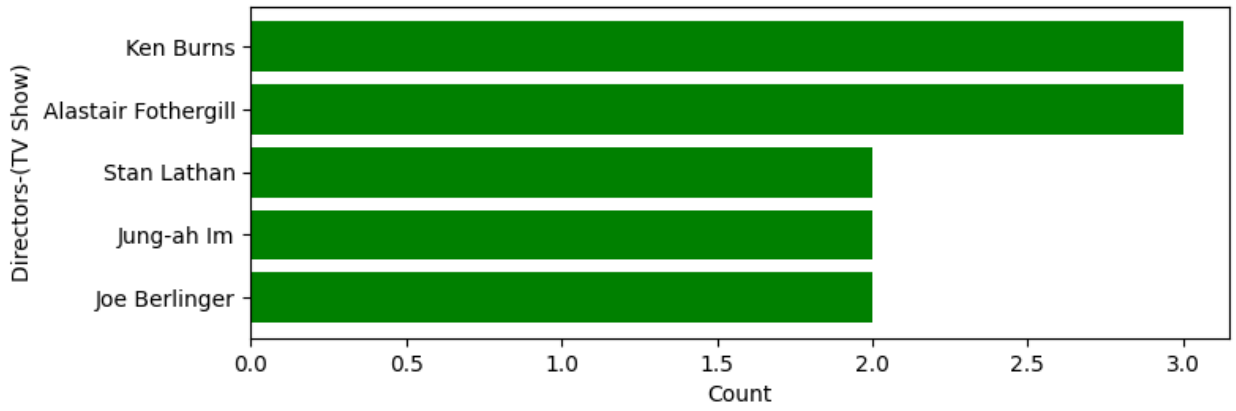
TV_mature adults, TV_above 14 are top rating for both Movies and TV Shows

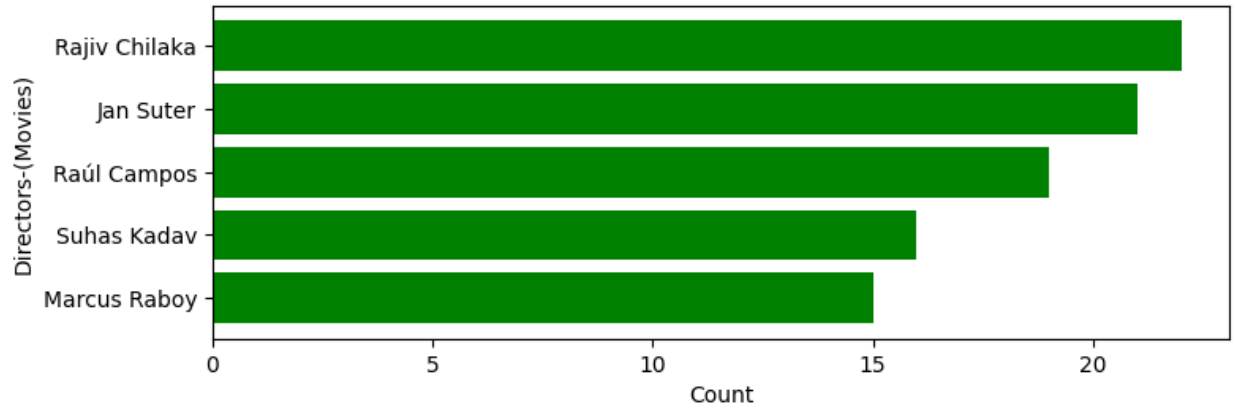
```
df_director_shows = df_shows[df_shows !=
'Unknown'].groupby('Directors').agg({'title':
'nunique'}).reset_index().sort_values(by=['title'],ascending=False)
[:5]
df_director_movies = df_movies[df_movies !=
'Unknown'].groupby('Directors').agg({'title':
'nunique'}).reset_index().sort_values(by=['title'],ascending=False)
[:5]

plt.figure(figsize = (8,6))
plt.subplot(2,1,1)
```

```
plt.barh(df_director_shows[::-1]['Directors'], df_director_shows[::-1]
['title'], color = 'green')
plt.ylabel('Directors-(TV Show)')
plt.xlabel("Count")
plt.show()

plt.figure(figsize = (8.3,6))
plt.subplot(2,1,2)
plt.barh(df_director_movies[::-1]['Directors'], df_director_movies[::-
1]['title'], color = 'green')
plt.ylabel('Directors-(Movies)')
plt.xlabel("Count")
plt.show()
```





Ken Burns, Alastair Fothergill, Stan Lathan, Joe Barlinger are popular directors across TV Shows on Netflix Rajiv Chilka, Jan Suter, Raul Campos, Suhas Kadav are popular directors across movies

```
df_shows.groupby('year_only').agg({"title":"nunique"})
```


```
{"summary":"{\n  \"name\": \"df_shows\",\n  \"rows\": 10,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
253,\n        \"min\": 1,\n        \"max\": 605,\n        \"samples\":
```

```
[\n          605,\n          5,\n          412\n        ],\n
\"num_unique_values\": 9,\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

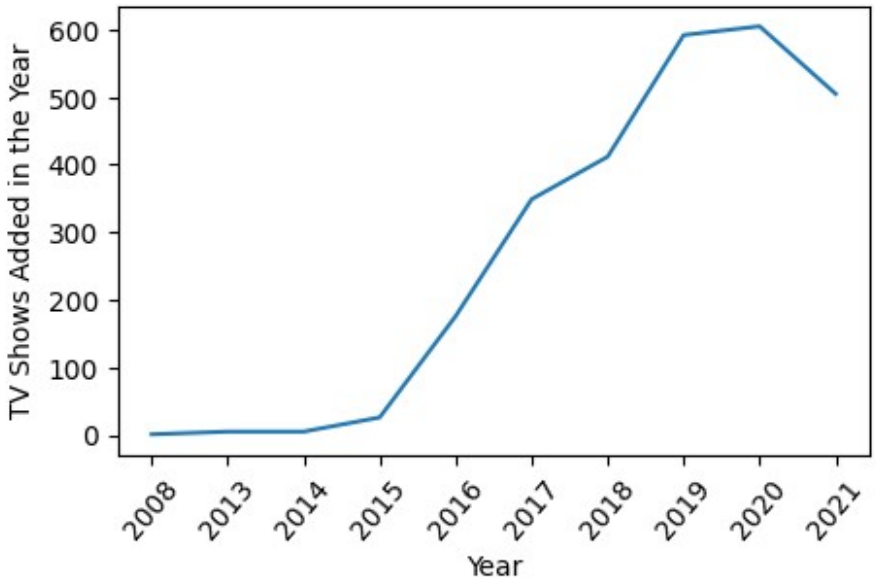
```
df_movies.groupby('year_only').agg({"title":"nunique"})
```

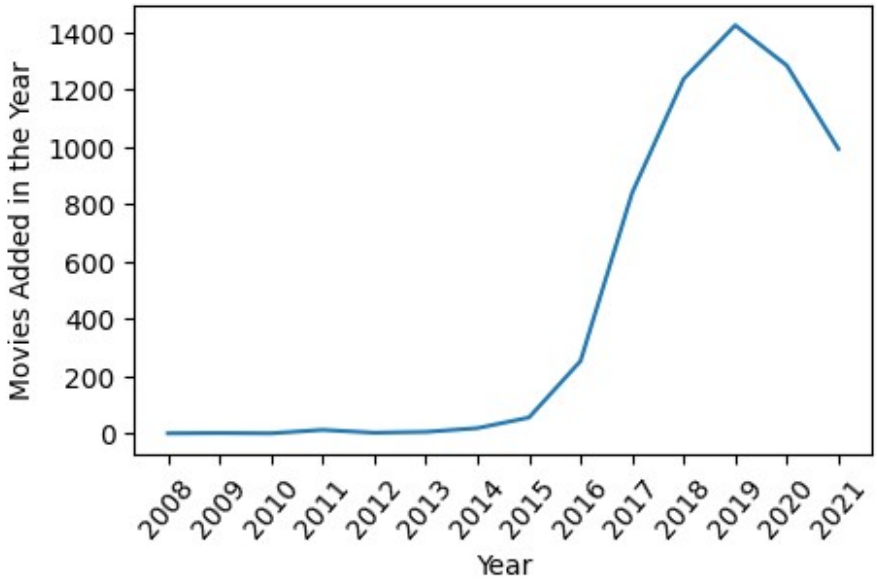
```
{"summary":"{\n  \"name\": \"df_movies\",\n  \"rows\": 14,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
573,\n        \"min\": 1,\n        \"max\": 1424,\n
\"samples\": [\n          1284,\n          1237,\n          1\n
],\n        \"num_unique_values\": 13,\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```

```
df_year_shows =
df_shows.groupby('year_only').agg({"title":"nunique"}).reset_index()
df_year_movies =
df_movies.groupby('year_only').agg({"title":"nunique"}).reset_index()

plt.figure(figsize = (5,3))
sns.lineplot(data=df_year_shows, x='year_only', y='title')
plt.ylabel("TV Shows Added in the Year")
plt.xlabel("Year")
plt.xticks(rotation = 50)
plt.show()

plt.figure(figsize = (4.9,3))
sns.lineplot(data=df_year_movies, x='year_only', y='title')
plt.ylabel("Movies Added in the Year")
plt.xlabel("Year")
plt.xticks(rotation = 50)
plt.show()
```



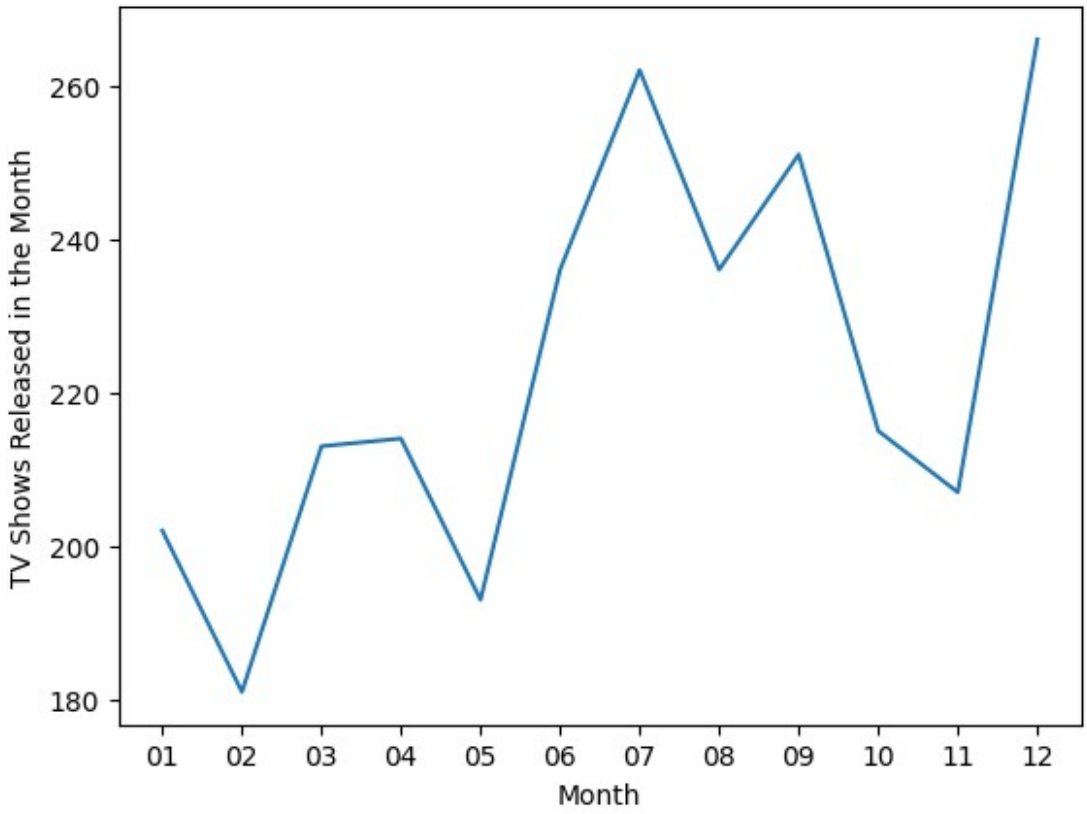


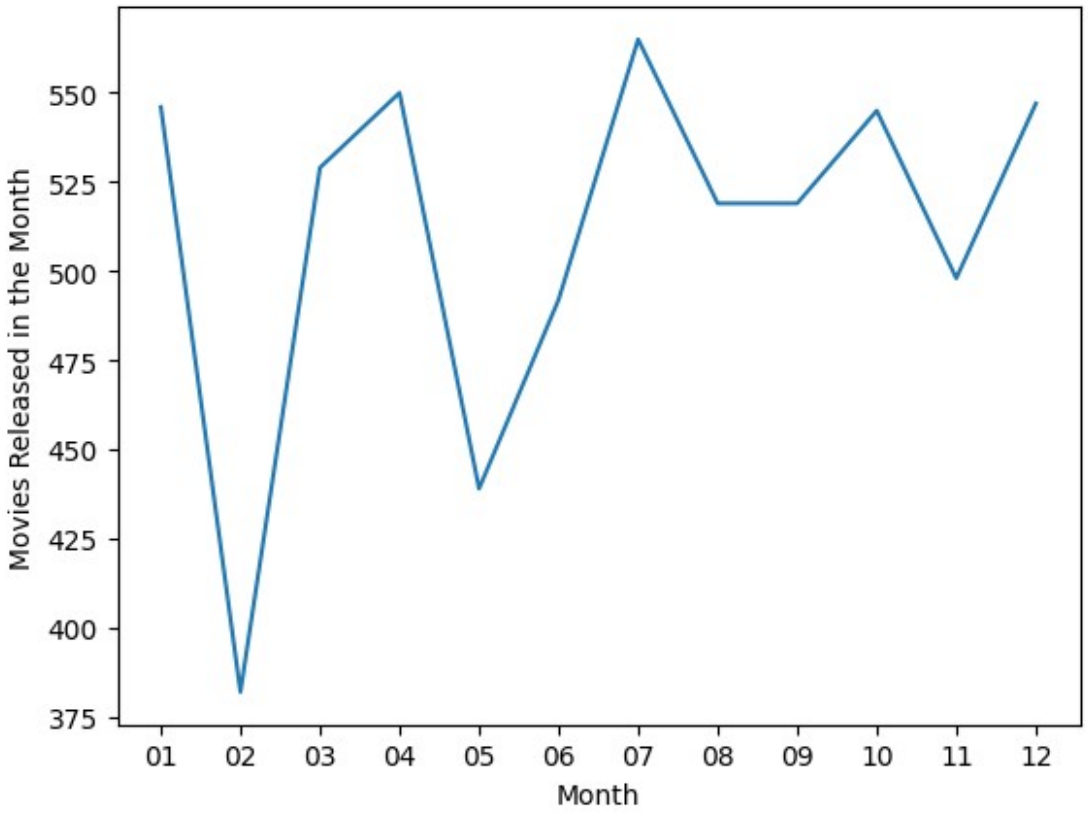
Till 2019, overall content across Netflix was increasing but due to Covid in 2020, though TV Shows didn't take a dip then Movies did take a dip. Well later in 2021, content across both was reduced significantly

```
df_month=df_shows.groupby(['month_only']).agg({"title":"nunique"}).res
et_index()
sns.lineplot(data=df_month, x='month_only', y='title')
plt.ylabel("TV Shows Released in the Month")
plt.xlabel("Month")
plt.show()

df_month=df_movies.groupby(['month_only']).agg({"title":"nunique"}).re
```

```
set_index()
sns.lineplot(data=df_month, x='month_only', y='title')
plt.ylabel("Movies Released in the Month")
plt.xlabel("Month")
plt.show()
```





TV Shows are added in Netflix hugely during mid of months of the year, i.e- July

Movies are added in Netflix hugely in first week/last month of current year and first month of next year

```
plt.figure(figsize=(15, 10))
plt.subplot(1, 2, 1)
movie_genre = df_final[df_final['type'] == 'Movie'].groupby("Genre")
["title"].nunique().sort_values(ascending=False)
text_m = ' '.join(movie_genre.index)

wordcloud_m = WordCloud(width=800, height=400,
background_color='black',
                        prefer_horizontal=1.0,
min_font_size=8).generate_from_frequencies(movie_genre.to_dict())

plt.imshow(wordcloud_m, interpolation='bilinear')
plt.axis('off')
plt.title('Movie Genre Word Cloud')


plt.subplot(1, 2, 2)
tv_genre = df_final[df_final['type'] == 'TV Show'].groupby("Genre")
["show_id"].nunique().sort_values(ascending=False)
text_tv = ' '.join(tv_genre.index)
```

```
wordcloud_tv = WordCloud(width=1000, height=600,
background_color='black',
                         prefer_horizontal=1.0,
min_font_size=8).generate_from_frequencies(tv_genre.to_dict())

plt.imshow(wordcloud_tv, interpolation='bilinear')
plt.axis('off')
plt.title('TV Show Genre Word Cloud')

plt.show()
```

Movie Genre Word Cloud

TV Show Genre Word Cloud

Movies:

The dataset reveals a diverse array of movie genres, indicating a broad audience interest. Popular genres include dramas and comedies, reflecting a preference for emotionally engaging narratives and lighthearted content. There's a significant presence of international and independent films, suggesting an openness to diverse cultural experiences and unique storytelling.

TV Shows:

The data displays a diverse range of TV show genres, reflecting varied viewer preferences. International TV Shows are prominent, indicating a global audience's interest. TV Dramas and TV Comedies are popular genres, showcasing a preference for emotionally engaging narratives and lighthearted content. The dataset includes various genres, such as crime shows, docuseries, and reality TV, catering to diverse interests.

Recommendations

1) The most popular Genres across the countries and in both TV Shows and Movies are Drama, Comedy and International TV Shows/Movies, so content aligning to that is recommended.

2)Add TV Shows in July/August and Movies in last week of the year/first month of the next year.

3)For most audience 80-120 mins is the recommended length for movies and Kids TV Shows are also popular along with the genres in first point, hence recommended.

4)The target audience in USA and India is recommended to be 14+ and above ratings while for UK, its recommended to be completely Mature/R content .