```
import pandas as pd
!gdown
https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940
/original/netflix.csv
```

```
Downloading...
From:
https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940
/original/netflix.csv
To: /content/netflix.csv
100% 3.40M/3.40M [00:01<00:00, 2.53MB/s]
```

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

Examination of the statistical summary

```
df.columns
```

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

```
df.info()
```

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

```
df.shape
```

```
(8807, 12)
```

```
df.nunique()
```

```
show_id         8807
type               2
title           8807
director        4528
cast            7692
country          748
date_added      1767
release_year      74
rating            17
duration         220
listed_in        514
description     8775
dtype: int64
```

```
duplicate=df.duplicated().value_counts()
print(duplicate)
```

```
False    8807
Name: count, dtype: int64
```

Unnesting a column in a Pandas DataFrame requires splitting list-like or iterable components within a column into independent rows. Filling the NAN values are replaced with meaningful values according to the column *name*

```
# NAN values replaced
df['director']=df['director'].fillna('unknown_director')
#Spliting the comma from the list of values
df['director'].apply(lambda x:x.split(', '))
#converting to list
a=df['director'].apply(lambda x:x.split(', ')).tolist()
a
```

```
[['Kirsten Johnson'],
 ['unknown_director'],
 ['Julien Leclercq'],
 ['unknown_director'],
 ['unknown_director'],
 ['Mike Flanagan'],
 ['Robert Cullen', 'José Luis Ucha'],
 ['Haile Gerima'],
 ['Andy Devonshire'],
 ['Theodore Melfi'],
 ['unknown_director'],
 ['Kongkiat Komesiri'],
 ['Christian Schwochow'],
 ['Bruno Garotti'],
 ['unknown_director'],
 ['unknown_director'],
 ['Pedro de Echave García', 'Pablo Azorín Williams'],
 ['unknown_director'],
```

```
['Adam Salky'],
['unknown_director'],
['Olivier Megaton'],
['unknown_director'],
['K.S. Ravikumar'],
['Alex Woo', 'Stanley Moore'],
['S. Shankar'],
['unknown_director'],
['Rajiv Menon'],
['Dennis Dugan'],
['Scott Stewart'],
['Robert Luketic'],
['Ashwiny Iyer Tiwari', 'Abhishek Chaubey', 'Saket Chaudhary'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Daniel Sandu'],
['Cédric Jimenez'],
['unknown_director'],
['George Nolfi'],
['unknown_director'],
['unknown_director'],
['Steven Spielberg'],
['Jeannot Szwarc'],
['Joe Alves'],
['Joseph Sargent'],
['Tyler Greco'],
['Daniel Espinosa'],
['Bunmi Ajakaiye'],
['Antoine Fuqua'],
['unknown_director'],
['unknown_director'],
['Toshiya Shinohara'],
['Toshiya Shinohara'],
['Toshiya Shinohara'],
['Toshiya Shinohara'],
['unknown_director'],
['Masahiko Murata'],
['Hajime Kamegaki'],
['Masahiko Murata'],
['Hajime Kamegaki'],
['Masahiko Murata'],
['Hirotsugu Kawasaki'],
['Toshiyuki Tsuru'],
['Tensai Okamura'],
['David Yarovesky'],
['unknown_director'],
['unknown_director'],
```

```
['unknown_director'],
['Hanns-Bruno Kammertöns', 'Vanessa Nöcker', 'Michael Wech'],
['unknown_director'],
['unknown_director'],
['David A. Vargas'],
['unknown_director'],
['Kemi Adetiba'],
['unknown_director'],
['Ben Simms'],
['unknown_director'],
['Prakash Satam'],
['Delhiprasad Deenadayalan'],
['Delhiprasad Deenadayalan'],
['Tomer Eshed'],
['Cedric Nicolas-Troyan'],
['unknown_director'],
['unknown_director'],
['JJC Skillz', 'Funke Akindele'],
['unknown_director'],
['Thomas Sieben'],
['unknown_director'],
['Marcus Clarke'],
['unknown_director'],
['Alice Waddington'],
['Mona Achache', 'Patricia Tourancheau'],
['unknown_director'],
['Alexis Almström'],
['Raja Gosnell'],
['unknown_director'],
['Stephen Kijak'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Chapman Way', 'Maclain Way'],
['Jason Hehir'],
['Yemi Amodu'],
['unknown_director'],
['Lijo Jose Pellissery'],
['unknown_director'],
['David de Vos'],
['unknown_director'],
['unknown_director'],
['Luis Alfaro', 'Javier Gómez Santander'],
['unknown_director'],
['Sara Colangelo'],
['Stephen Herek'],
['Rahul Rawail'],
['Jane Campion'],
```

```
['Nagesh Kukunoor'],
['Luke Holland'],
['Shanker Raman'],
['JP Habac'],
['unknown_director'],
['unknown_director'],
['Jane Campion'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Vidhu Vinod Chopra'],
['Mark Rosman'],
['Gilles Paquet-Brenner'],
['Lasse Hallström'],
['Scott Pleydell-Pearce'],
['Ridley Scott'],
['unknown_director'],
['Neill Blomkamp'],
['Phillip Noyce'],
['Renny Harlin'],
['Anthony Minghella'],
['Simon Wincer'],
['Lasse Hallström'],
['Spike Lee'],
['Sebastián Schindel'],
['Steven C. Miller'],
['Richard LaGravenese'],
['Martin Campbell'],
['Reginald Hudlin'],
['George Jackson', 'Doug McHenry'],
['Eric Meza'],
['unknown_director'],
['Gerhard Mostert'],
['Michael Martin'],
['Michael Rymer'],
['Andrew Lau Wai-keung', 'Alan Mak'],
['Brett Weiner'],
['unknown_director'],
['unknown_director'],
['Jim Henson'],
['Gary Winick'],
['Danishka Esterhazy'],
['Troy Byer'],
['Pang Ho-cheung'],
['unknown_director'],
['Tim Burton'],
['Reginald Hudlin'],
['David Zucker'],
['Kinka Usher'],
```

```
['unknown_director'],
['Sergio Leone'],
["Matthew O'Callaghan", 'Todd Wilderman'],
['Bobby Farrelly', 'Peter Farrelly'],
['Wolfgang Petersen'],
['Peter Spirer'],
['Michael Carney'],
['Richard Linklater'],
['Amy Rice'],
['Antoine Fuqua'],
['Randal Kleiser'],
['Michael Ritchie'],
['J. Lee Thompson'],
['Evan Goldberg', 'Seth Rogen'],
['Tom Shadyac'],
['Peter Segal'],
['unknown_director'],
['Malcolm D. Lee'],
['Wolfgang Petersen'],
['unknown_director'],
['Chapman Way', 'Maclain Way'],
['unknown_director'],
['unknown_director'],
['Ramzy Bedia', 'Éric Judor'],
['unknown_director'],
['Sharan Koppisetty'],
['Taylor Sheridan'],
['Sachin Yardi'],
['unknown_director'],
['unknown_director'],
['Saurabh Kabra'],
['Mark Waters'],
['unknown_director'],
['Kemi Adetiba'],
['Partho Mitra'],
['Santram Varma'],
['Anil V. Kumar', 'Anurag Basu'],
['Sangeeth Sivan'],
['Umesh Ghadge'],
['Sachin Yardi'],
['David Dhawan'],
['Dibakar Banerjee'],
['Apoorva Lakhia'],
['Milan Luthria'],
['Milan Luthria'],
['Pawan Kripalani'],
['Bhushan Patel'],
['unknown_director'],
['unknown_director'],
```

```
['Magnus Martens'],
['Apoorva Lakhia'],
['Raj Nidimoru', 'Krishna D.K.'],
['Milan Luthria'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Joshua Rofé'],
['Brad Anderson'],
['Mauricio Dias', 'Tatiana Villela'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Angel Kristi Williams'],
['Roger Donaldson'],
['Christopher Alender'],
['Rush Sturges'],
['David Oyelowo'],
['unknown_director'],
['Mark Lo'],
['unknown_director'],
['Crystal Moselle'],
['Rathindran R Prasad'],
['Rathindran R Prasad'],
['Rathindran R Prasad'],
['Rathindran R Prasad'],
['Han Kwang Il'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Karim El Shenawy'],
['unknown_director'],
['Yin Chen-hao'],
['Brian Andrew Mendoza'],
['unknown_director'],
['Dave Needham'],
['Veronica Velasco'],
['Drake Doremus'],
['Miguel Alexandre'],
['Mani Ratnam'],
['unknown_director'],
['Michael Harte'],
['Tosin Igho'],
['Mani Ratnam'],
['Alice Filippi'],
['Paakhi Tyrewala'],
['unknown_director'],
['Bruno Garotti'],
['Laura Brownson'],
```

```
 ['unknown_director'],
 ['Steve Brill'],
 ['unknown_director'],
 ['unknown_director'],
 ['Jane Campion'],
 ['Moses Inwang'],
 ['unknown_director'],
 ['Ferdinando Cito Filomarino'],
 ['unknown_director'],
 ['unknown_director'],
 ['Juan Carlos Medina'],
 ['unknown_director'],
 ['unknown_director'],
 ['Inma Torrente'],
 ['unknown_director'],
 ['Julián Gaviria'],
 ['Steven Yamamoto'],
 ['unknown_director'],
 ['Charles Uwagbai'],
 ['Julián Hernández'],
 ['Sam Hobkinson'],
 ['Vince Marcello'],
 ['Jonathan Teplitzky'],
 ['unknown_director'],
 ['Sakon Tiacharoen'],
 ['unknown_director'],
 ['unknown_director'],
 ['Floyd Russ'],
 ['unknown_director'],
 ['Dustin Hoffman'],
 ['Adze Ugah'],
 ['Hideaki Takizawa'],
 ['Quoc Bao Tran'],
 ['unknown_director'],
 ['Bejoy Nambiar',
  'Priyadarshan',
  'Karthik Narain',
  'Vasanth Sai',
  'Karthik Subbaraj',
  'Arvind Swamy',
  'Rathindran R Prasad',
  'Sarjun',
  'Gautham Vasudev Menon'],
 ['Kayode Kasum'],
 ['Just Philippot'],
 ['Kirk DeMicco', 'Brandon Jeffords'],
 ['Rohit Shetty'],
 ['Cavi Borges', 'Luciano Vidigal'],
 ['Alejandro Doria'],
```

```
['Laura Fairrie'],
['Marcelo Piñeyro'],
['Izu Ojukwu'],
['Peter Winther'],
['Susan Lacy'],
['unknown_director'],
['Billy Corben'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['James Mangold'],
['Chineze Anyaene'],
['Hsu Fu-chun'],
['Kristine Stolakis'],
['Eva Müller', 'Michael Schmitt'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Brian Levant'],
['Rod Daniel'],
['Robert Zemeckis'],
['Todor Chapkanov'],
['Steven Spielberg'],
['Phil Lord', 'Christopher Miller'],
['unknown_director'],
['Renny Harlin'],
['John Hughes'],
['Justin Baldoni'],
['Joe Roth'],
['unknown_director'],
['Mark Helfrich'],
['unknown_director'],
['Mag Hsu', 'Hsu Chih-yen'],
['Christopher Nolan'],
['Paul Thomas Anderson'],
['Nick Castle'],
['Howard Zieff'],
['Howard Zieff'],
['David Feiss'],
['David Gordon Green'],
['Jorge Blanco'],
['Zara Hayes'],
['Gary Ross'],
['Clint Eastwood'],
['Trey Parker'],
['Kelly Fremon Craig'],
```

```
['Tom Elkins'],
['Tony Scott'],
['Brad Furman'],
['Sylvain White'],
['Brad Anderson'],
['Irwin Winkler'],
['Spike Lee'],
['Garry Marshall'],
['unknown_director'],
['Kenneth Gyang'],
['Jaume Balagueró'],
['unknown_director'],
['Najwa Najjar'],
['unknown_director'],
['Selçuk Metin'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Najwa Najjar'],
['Keishi Otomo'],
['David Charhon'],
['Michelle Bello'],
['Steven Tsuchida'],
['unknown_director'],
['Daniel Markowicz'],
['Louie Schwartzberg'],
['unknown_director'],
['Glen Winter'],
['unknown_director'],
['unknown_director'],
['Ram Gopal Varma'],
['David Benullo'],
['unknown_director'],
['unknown_director'],
['Laxman Utekar'],
['Royale Watkins', 'Rich Schlansker'],
['Yuval Adler'],
['unknown_director'],
['unknown_director'],
['Quentin Tarantino'],
['Clay Glen'],
['Muzi Mthembu'],
['Marcos Bucay'],
['Peter Thorwarth'],
['unknown_director'],
['Kim Seong-hun'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
```

```
 ['Augustine Frizzell'],
 ['unknown_director'],
 ['unknown_director'],
 ['Sidheswar Shukla'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Binayak Das'],
 ['Arpan Sarkar', 'Shyamal Chaulia'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka', 'Owll Mina'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['Rajiv Chilaka'],
 ['unknown_director'],
 ['Ainsley Gardiner', 'Briar Grace-Smith'],
 ['Mahmoud Karim'],
 ['Kyohei Ishiguro'],
 ['Seyi Babatope'],
 ['unknown_director'],
 ['unknown_director'],
 ['Johane Matte', 'Andrew L. Schmidt', 'Francisco Ruiz Velasco'],
 ['Morgan Ingari'],
 ['unknown_director'],
 ['unknown_director'],
 ['Abdulaziz Alshlahei'],
 ['Edward Drake'],
 ['Kathryn Fasegha'],
 ['Sita Likitvanichkul',
  'Jetarin Ratanaserikiat',
  'Apirak Samudkitpaisan',
  'Thanabodee Uawithya',
  'Adirek Wattaleela'],
 ['unknown_director'],
 ['Leigh Janiak'],
 ['unknown_director'],
 ['Luis Estrada'],
 ['Garrett Bradley'],
 ['Sofia Coppola'],
 ['Colin Trevorrow'],
```

```
['Bill Condon'],
['Bill Condon'],
['David Slade'],
['Chris Weitz'],
['Catherine Hardwicke'],
['Hadrah Daeng Ratu'],
['unknown_director'],
['Tommy Chong'],
['Fred Ouro Preto'],
['unknown_director'],
['Mayye Zayed'],
['Alessandra de Rossi'],
['unknown_director'],
['Ash Brannon', 'Chris Buck'],
['Alaa Eddine Aljem'],
['Tom Donahue'],
['Roberto De Feo', 'Paolo Strippoli'],
['Navot Papushado'],
['unknown_director'],
['unknown_director'],
['Manuel Alcalá'],
['Ricardo Trogi'],
['Semi Chellas'],
['unknown_director'],
['Akay Mason', 'Abosi Ogba'],
['unknown_director'],
['Viridiana Lieberman'],
['Nima Nourizadeh'],
['Daniel Růžička'],
['Juraj Šajmovič'],
['unknown_director'],
['unknown_director'],
['Leigh Janiak'],
['Femi D. Ogunsanwo'],
['Douglas Attal'],
['unknown_director'],
['unknown_director'],
['Kim Joo-hyung'],
['Avi Federgreen'],
['Jericca Cleland', 'Kevin Munroe'],
['unknown_director'],
['unknown_director'],
['Ahmed Siddiqui'],
['unknown_director'],
['Hallie Meyers-Shyer'],
['Scott Speer'],
['unknown_director'],
['Barry Levinson'],
['Tolulope Itegboje'],
['Camille Delamarre'],
```

```
['unknown_director'],
['Pascal Atuma'],
['unknown_director'],
['Oleg Trofim'],
['Seyi Babatope'],
['Mahmood Ali-Balogun'],
['Jan Holoubek'],
['unknown_director'],
['Anurin Nwunembom', 'Musing Derrick'],
['Christine Luby'],
['Udoka Oyeka'],
['Yeo Siew Hua'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Hardik Mehta'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Rajveer Singh Maan', 'Harpeet Singh'],
['Youssef Chahine'],
['unknown_director'],
['Saw Teong Hin', 'Nik Amir Mustapha', 'M.S. Prem Nath'],
['unknown_director'],
['unknown_director'],
['Rano Karno'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Pramod Pawar'],
['Naresh Saigal'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
```

```
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['Leigh Janiak'],
 ['Vinil Mathew'],
 ['Neri Parenti'],
 ['unknown_director'],
 ['Ramsey Nouah'],
 ['Bong Joon Ho'],
 ['Kim Tae-hyung'],
 ['Michael Spierig', 'Peter Spierig'],
 ['Ernie Barbarash'],
 ['Wolfgang Petersen'],
 ['Matt Ogens'],
 ['Jay Roach'],
 ['Jay Roach'],
 ['Jay Roach'],
 ['Paul Thomas Anderson'],
 ['unknown_director'],
 ['McG'],
 ['Frank Marshall'],
 ['Nick Castle'],
 ['Victor Vu'],
 ['Chow Hin Yeung Roy'],
 ['unknown_director'],
 ['Joel Hopkins'],
 ['John Stevenson', 'Mark Osborne'],
 ['Jennifer Yuh Nelson'],
 ['Greg Berlanti'],
 ['Garth Davis'],
 ['Malik Nejer'],
 ['Rob Marshall'],
 ['Martin Brest'],
 ['Shuko Murase'],
 ['Paul W.S. Anderson'],
 ['Garry Marshall'],
 ['Ivan Reitman'],
 ['Joel Gallen'],
 ['Claire McCarthy'],
 ['Mary Lambert'],
 ['Sidharta Tata',
  'Aco Tenriyagelli',
  'Dian Sastrowardoyo',
  'Ifa Isfansyah',
  'Jason Iskandar'],
 ['unknown_director'],
 ['Trevor Nunn'],
 ['unknown_director'],
```

```
['Gabriele Muccino'],
['Jim Field Smith'],
['Chris Koch'],
['J.J. Abrams'],
['Rob Minkoff'],
['Lynn Shelton'],
['Adam McKay'],
['Anton Corbijn'],
['Robin Bissell'],
['David Fincher'],
['John G. Avildsen'],
['John G. Avildsen'],
['John G. Avildsen'],
['Alan Parker'],
['Walter Hill'],
['Stephen Frears'],
['Bryan Bertino'],
['Phil Alden Robinson'],
['Florian Henckel von Donnersmarck'],
['Len Wiseman'],
['Måns Mårlind', 'Björn Stein'],
['Patrick Tatopoulos'],
['Robert O. Peters'],
['Vincent Ward'],
['Gregory Nava'],
['Mary Harron'],
['unknown_director'],
['Matt Thompson'],
['Jameel Buari'],
['unknown_director'],
['Shen Leping'],
['Matt Aselton'],
['Jose Javier Reyes'],
['Jakub Piątek'],
['unknown_director'],
['John Dower'],
['unknown_director'],
['unknown_director'],
['B. T Thomas'],
['Andrew Dominik'],
['unknown_director'],
['Nibal Arakji'],
['Sofie Šustková'],
['Anissa Bonnefont'],
['Bahij Hojeij'],
['Jonathan Hensleigh'],
['Srijit Mukherji', 'Vasan Bala', 'Abhishek Chaubey'],
['unknown_director'],
['unknown_director'],
```

```
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['Justin P. Lange'],
 ['Kimmy Gatewood'],
 ['Tània Balló'],
 ['Manolo Caro'],
 ['unknown_director'],
 ['unknown_director'],
 ['Anurin Nwunembom'],
 ['Jayme Monjardim'],
 ['Kingsley Ogoro'],
 ['Kingsley Ogoro'],
 ['unknown_director'],
 ['Cristina Jacob'],
 ['Cristina Jacob'],
 ['Cristina Jacob'],
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['Fernando Moro'],
 ['unknown_director'],
 ['Ivan Andrew Payawal'],
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['Yoshiyuki Tomino', 'Yoshikazu Yasuhiko'],
 ['Yoshiyuki Tomino', 'Yoshikazu Yasuhiko'],
 ['Yoshiyuki Tomino'],
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['Rob Seidenglanz'],
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['unknown_director'],
 ['Simon Frederick'],
 ['unknown_director'],
 ['Michihito Fujii'],
 ['unknown_director'],
 ['Paul Weitz'],
 ['unknown_director'],
 ['Karthik Subbaraj'],
 ['Kayode Kasum'],
```

```
['Yoshiyuki Tomino', 'Yoshikazu Yasuhiko'],
['Keishi Otomo'],
['unknown_director'],
['Hsu Fu-chun'],
['Lucky Kuswandi'],
['Soudade Kaadan'],
['Soudade Kaadan'],
['unknown_director'],
['unknown_director'],
['Antoinette Jadaone'],
['unknown_director'],
['unknown_director'],
['Marie Clements'],
['David O. Russell'],
['unknown_director'],
['Achille Brice'],
['Max Jabs'],
['unknown_director'],
['Sarawut Wichiensarn'],
['Ricardo de Montreuil'],
['unknown_director'],
['unknown_director'],
['Peter Chelsom'],
['Michael Lockshin'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Francine Parker'],
['unknown_director'],
['Daniel Schechter'],
['unknown_director'],
['unknown_director'],
['Mike Gunther'],
['David Zeiger'],
['Michael Simon'],
['Jerry Rothwell'],
['unknown_director'],
['Joe Berlinger', 'Bruce Sinofsky'],
['Ian Cheney', 'Sharon Shattuck'],
['Bradley Parker'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Prabhakaran'],
['Manjari Makijany'],
['Jay Oliva'],
```

```
['unknown_director'],
['Chris Appelhans'],
['Michael Tiddes'],
['Lara Saba'],
['Bao Nhan', 'Namcito'],
['Mark Raso'],
['unknown_director'],
['Tariq Alkazim'],
['Mark Raso'],
['Kenneth Gyang'],
['unknown_director'],
['unknown_director'],
['Yulene Olaizola'],
['unknown_director'],
['Mark Waters'],
['unknown_director'],
['Matthew Heineman'],
['unknown_director'],
['Robert Peters'],
['Jonathan Clay'],
['Ally Pankiw'],
['George Ford'],
['George Ford'],
['unknown_director'],
['Lee Kae-byeok'],
['unknown_director'],
['Jayan Moodley'],
['Daniel Benmayor'],
['Alex Díaz'],
['unknown_director'],
['Helena Bergström'],
['Alejandro De Grazia', 'Juan Stadler'],
['Myriam Fares'],
['Chiaki Kon'],
['unknown_director'],
['Rae Red'],
['Lance Hool'],
['Nicole Conn'],
['Les Mayfield'],
['Ellen Seidler', 'Megan Siler'],
['unknown_director'],
['Roger Christian'],
['unknown_director'],
['Peter Galison'],
['Lance Young'],
['Leandro Neri'],
['Thom Fitzgerald'],
['unknown_director'],
['Don Michael Paul'],
```

```
['Andrzej Bartkowiak'],
['Harold Becker'],
['unknown_director'],
['Andrew Jenks'],
['Ric Roman Waugh'],
['Rob Reiner'],
['Andy Tennant'],
['Tade Ogidan'],
['unknown_director'],
['Scott Spiegel'],
['Jessie Nelson'],
['Theodore Witcher'],
['Clint Eastwood'],
['Aurora Guerrero'],
['Michael Jai White'],
['James McTeigue'],
['Takuya Igarashi'],
['Michelle MacLaren'],
['unknown_director'],
['Hidenori Inoue'],
['Hidenori Inoue'],
['Don Michael Paul'],
['Todd Phillips'],
['Walter Hill'],
['Steve Ball'],
['Dominic Sena'],
['unknown_director'],
['Alan Alda'],
['Mika Kaurismäki'],
['Sydney Pollack'],
['Lorene Scafaria'],
['Barbra Streisand'],
['Clint Eastwood'],
['unknown_director'],
['Michael Winterbottom'],
['Emma Tammi'],
['Peter Hutchings'],
['George Ratliff'],
['unknown_director'],
['Bo Burnham'],
['Ekene Som Mekwunye'],
['David Frankel'],
['Kevin Johnson'],
['unknown_director'],
['José Larraza', 'Marc Pons'],
['Gary Sing'],
['unknown_director'],
['Julio Quintana'],
['unknown_director'],
```

```
['Paween Purijitpanya'],
['unknown_director'],
['Pablo Faro'],
['Soudade Kaadan'],
['Letizia Lamartire'],
['Ray Jiang'],
['unknown_director'],
['unknown_director'],
['Daniel Vernon'],
['Steve Gukas'],
['Tim Johnson'],
['unknown_director'],
['unknown_director'],
['unknown_director'],
['Vishwesh Krishnamoorthy'],
['unknown_director'],
['unknown_director'],
['Zack Snyder'],
['unknown_director'],
['unknown_director'],
['Uduak-Obong Patrick'],
['unknown_director'],
['Mohamed Diab'],
['Amr Salama'],
['Christopher Amos'],
['Prakash Satam'],
['unknown_director'],
['Robert Rodriguez'],
['Milan Luthria'],
['David Ayer'],
['Eshom Nelms', 'Ian Nelms'],
['James Moll'],
['unknown_director'],
['unknown_director'],
['James Redford'],
['Kaashvie Nair'],
['J.D. Dillard'],
['Nikhil Pherwani'],
['unknown_director'],
['Mae Czarina Cruz'],
['unknown_director'],
['Praveen Kandregula'],
['Cecilia Verheyden'],
['Daniel Minahan'],
['unknown_director'],
['Donovan Marsh'],
['Brent Dawes'],
['unknown_director'],
['unknown_director'],
```

```
['Leli Maki'],
['Uzodinma Okpechi'],
['Daniel Prochaska'],
['unknown_director'],
['Joe Wright'],
['unknown_director'],
['Matthew Vaughn'],
['Aditya Kripalani'],
['Adriano Rudiman'],
['David Pablos'],
['Alexandre Aja'],
['unknown_director'],
['Cai Cong'],
['Samuel Olatunji'],
['Ramon Térmens'],
['unknown_director'],
['Svetlana Cvetko'],
['unknown_director'],
['Martin Prakkat'],
['Baran bo Odar'],
['Zhang Chong'],
['Yılmaz Erdoğan'],
['Shantrelle P. Lewis'],
['unknown_director'],
['Ivan Ayr'],
['Anthony Mandler'],
['Vijay Roche'],
['unknown_director'],
["Stanley Menino D'Costa"],
['Jennifer Brea'],
['Julia von Heinz'],
['Niels Arden Oplev'],
['Don Argott', 'Sheena M. Joyce'],
['unknown_director'],
['Joshua Zeman'],
['unknown_director'],
['unknown_director'],
['Duncan Skiles'],
['unknown_director'],
['Sean McNamara'],
['unknown_director'],
['Vondie Curtis-Hall'],
['unknown_director'],
['Robert Radler'],
['Roel Reiné'],
['Todd Phillips'],
['Dean Parisot'],
['Paul Greengrass'],
['Lasse Hallström'],
```

```
['Justin Kelly'],
['Eric Darnell', 'Tom McGrath', 'Conrad Vernon'],
['unknown_director'],
['Suhas Kadav'],
['Suhas Kadav'],
['Suhas Kadav'],
['Suhas Kadav'],
['Suhas Kadav'],
['Clint Eastwood'],
['Jeff Wadlow'],
['Charles Martin'],
['Stella Corradi'],
['Roland Emmerich'],
['Kevin Macdonald'],
['Ann Deborah Fishman'],
['Chris Gorak'],
['Peter Jackson'],
['Roger Kumble'],
['Jonathan Lynn'],
['Courtney Hunt'],
['Pierre Greco', 'Nancy Florence Savard'],
['Andrew Davis'],
['Kevin Smith'],
['unknown_director'],
['Tosin Igho'],
['Chaitanya Tamhane'],
['Oriol Paulo'],
['Mike Rianda', 'Jeff Rowe'],
['Johannes Roberts'],
['unknown_director'],
['Robert Pulcini', 'Shari Springer Berman'],
['unknown_director'],
['Pedro Antonio'],
['unknown_director'],
['unknown_director'],
['John Wells'],
['Jonathan Liebesman'],
['Maria Pulera'],
['unknown_director'],
['Santhosh Viswanath'],
['Seema Pahwa'],
['unknown_director'],
['Ozan Açıktan'],
['Meltem Bozoflu'],
['Hakan Algül'],
['Selçuk Aydemir', 'Birkan Pusa'],
['Selçuk Aydemir'],
['Ömer Faruk Sorak'],
['Şenol Sönmez'],
```

```
 ['Alexis Morante'],
 ['Burak Aksak'],
 ['Kıvanç Baruönü'],
 ['Kıvanç Baruönü'],
 ['Rindala Kodeih'],
 ['Kongkiat Khomsiri'],
 ['Bedran Güzel'],
 ['Hakan Algül'],
 ['Marwan Nabil'],
 ['MIKIKO', 'Daito Manabe'],
 ['unknown_director'],
 ['Kayode Kasum'],
 ['Yılmaz Erdoğan', 'Ömer Faruk Sorak'],
 ['Takashi Shimizu'],
 ['unknown_director'],
 ['unknown_director'],
 ['Joe Penna'],
 ...]
```

```
# separating the director name based on title by setting title as
index
b=pd.DataFrame(a,index=df['title'])
b
```


```
{"summary":"{\n  \"name\": \"b\",\n  \"rows\": 8807,\n  \"fields\": [\
n    {\n      \"column\": \"title\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 8807,\n
\"samples\": [\n          \"Game Over, Man!\",\n          \"Arsenio
Hall: Smart & Classy\",\n          \"Kazoops!\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 0,\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 4406,\n
\"samples\": [\n          \"Cate Shortland\",\n          \"Shawn
Rech\",\n          \"Cal Seville\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 1,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 541,\n
\"samples\": [\n          \"Matthew McNeil\",\n          \"Tyler
Measom\",\n          \"Viju Mane\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 2,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 69,\n
\"samples\": [\n          \"Alka Amarkant Dubey\",\n          \"Saket
Chaudhary\",\n          \"Hideki Futamura\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 3,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 34,\n
\"samples\": [\n          \"Erich Sturm\",\n          \"Lauren
MacMullan\",\n          \"Parkpoom Wongpoom\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": 4,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 20,\n
\"samples\": [\n          \"Karthik Subbaraj\",\n          \"Wong Fei-
Pang\",\n          \"Steve Brill\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 5,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 13,\n
\"samples\": [\n          \"James Duffy\",\n          \"Sarah Adina
Smith\",\n          \"Arvind Swamy\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 6,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 13,\n
\"samples\": [\n          \"Jonathan van Tulleken\",\n
\"Scott Stewart\",\n          \"Rathindran R Prasad\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 7,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 11,\n
\"samples\": [\n          \"Jon YonKondy\",\n          \"Sarjun\",\n
\"Elizabeth Banks\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 8,\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n          \"Joann
Sfar\",\n          \"Abdullah Al Noor\",\n          \"Drue Metz\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": 9,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 8,\n
\"samples\": [\n          \"Shinji Takagi\",\n          \"Koji
Sawai\",\n          \"Krishnendu Chattopadhyay\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 10,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 5,\n
\"samples\": [\n          \"Suparn Verma\",\n          \"Rusty
Cundieff\",\n          \"Roger Allers\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 11,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"Mike Gabriel\",\n          \"Hiroshi
Yamazaki\",\n          \"James Gunn\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 12,\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"Mark Henn\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"b"}
```


```
#Using stack merging the columns to rows and shows 0,1 as per number
of directors
pd.DataFrame(a,index=df['title']).stack()
```

```
title
Dick Johnson Is Dead   0     Kirsten Johnson
Blood & Water          0    unknown_director
Ganglands              0     Julien Leclercq
Jailbirds New Orleans  0    unknown_director
Kota Factory           0    unknown_director
                                   ...
Zodiac                 0       David Fincher
Zombie Dumb            0    unknown_director
Zombieland             0     Ruben Fleischer
Zoom                   0        Peter Hewitt
Zubaan                 0         Mozez Singh
Length: 9612, dtype: object
```

```
#On Stacking pandas create the index name for the unnamed one by using stack
pd.DataFrame(a,index=df['title']).stack().reset_index()
```


```
{"summary":"{\n  \"name\": \"pd\",\n  \"rows\": 9612,\n  \"fields\":
[\n    {\n      \"column\": \"title\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 8807,\n
\"samples\": [\n          \"Game Over, Man!\",\n          \"Arsenio
Hall: Smart & Classy\",\n          \"Kazoops!\"\n        ],\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\": 12,\n        \"num_unique_values\": 13,\
n        \"samples\": [\n          11,\n          9,\n          0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": 0,\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 4994,\n
\"samples\": [\n          \"Lasse Hallstr\\u00f6m\",\n
\"Terrie Samundra\",\n          \"Florian Schnell\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```
director=pd.DataFrame(a,index=df['title']).stack().reset_index().drop(
columns = 'level_1').rename(columns = {0:'director'})
director
```


```
{"summary":"{\n  \"name\": \"director\",\n  \"rows\": 9612,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 8807,\n        \"samples\": [\n          \"Game
Over, Man!\",\n          \"Arsenio Hall: Smart & Classy\",\n
\"Kazoops!\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"director\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 4994,\n
\"samples\": [\n          \"Lasse Hallstr\\u00f6m\",\n
```

\"Terrie Samundra\",\n   \"Florian Schnell\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"director"}

```
type_shows=df[['title','type']]
type_shows
```

{"summary":"{\n \"name\": \"type_shows\",\n \"rows\": 8807,\n
\"fields\": [\n   {\n   \"column\": \"title\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 8807,\n   \"samples\": [\n   \"Game
Over, Man!\",\n   \"Arsenio Hall: Smart & Classy\",\n
\"Kazoops!\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"type\",\n   \"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n   \"samples\": [\n   \"TV
Show\",\n   \"Movie\"\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"type_shows"}

```
# Convert 'date_added' to datetime objects, handling errors and
potential format issues
df['date_added'] = pd.to_datetime(df['date_added'], format='%B %d,
%Y', errors='coerce')

# If you have multiple date formats, you can try using
`format='mixed'`
# or a custom function to handle different formats individually.

# Format the datetime objects to a consistent string representation
df['date_added'] = df['date_added'].dt.strftime('%Y-%m-%d')

print(df)
```

```
     show_id     type                  title          director  \
0         s1    Movie   Dick Johnson Is Dead   Kirsten Johnson
1         s2  TV Show           Blood & Water  unknown_director
2         s3  TV Show               Ganglands   Julien Leclercq
3         s4  TV Show  Jailbirds New Orleans  unknown_director
4         s5  TV Show            Kota Factory  unknown_director
...      ...      ...                     ...              ...
8802   s8803    Movie                 Zodiac     David Fincher
8803   s8804  TV Show             Zombie Dumb  unknown_director
8804   s8805    Movie              Zombieland   Ruben Fleischer
8805   s8806    Movie                   Zoom      Peter Hewitt
8806   s8807    Movie                 Zubaan       Mozez Singh

                                                  cast        country
\
```

```
0                                                     NaN  United States

1     Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...   South Africa

2     Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...             NaN

3                                                     NaN             NaN

4     Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...           India

...                                                   ...             ...

8802  Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...  United States

8803                                                  NaN             NaN

8804  Jesse Eisenberg, Woody Harrelson, Emma Stone, ...  United States

8805  Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...  United States

8806  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...           India


      date_added  release_year rating   duration  \
0     2021-09-25          2020  PG-13     90 min
1     2021-09-24          2021  TV-MA  2 Seasons
2     2021-09-24          2021  TV-MA   1 Season
3     2021-09-24          2021  TV-MA   1 Season
4     2021-09-24          2021  TV-MA  2 Seasons
...          ...           ...    ...        ...
8802  2019-11-20          2007      R    158 min
8803  2019-07-01          2018  TV-Y7  2 Seasons
8804  2019-11-01          2009      R     88 min
8805  2020-01-11          2006     PG     88 min
8806  2019-03-02          2015  TV-14    111 min

                                                listed_in  \
0                                           Documentaries
1         International TV Shows, TV Dramas, TV Mysteries
2     Crime TV Shows, International TV Shows, TV Act...
3                               Docuseries, Reality TV
4     International TV Shows, Romantic TV Shows, TV ...
...                                                 ...
8802                Cult Movies, Dramas, Thrillers
8803            Kids' TV, Korean TV Shows, TV Comedies
8804                          Comedies, Horror Movies
8805              Children & Family Movies, Comedies
8806    Dramas, International Movies, Music & Musicals

                                              description
0     As her father nears the end of his life, filmm...
```

```
1     After crossing paths at a party, a Cape Town t...
2     To protect his family from a powerful drug lor...
3     Feuds, flirtations and toilet talk go down amo...
4     In a city of coaching centers known to train I...
...                                                 ...
8802  A political cartoonist, a crime reporter and a...
8803  While living alone in a spooky town, a young g...
8804  Looking to survive in a world taken over by zo...
8805  Dragged from civilian life, a former superhero...
8806  A scrappy but poor boy worms his way into a ty...

[8807 rows x 12 columns]
```

```
date_columns=df[['title','date_added','release_year']]
date_columns
```

```
{"summary":"{\n  \"name\": \"date_columns\",\n  \"rows\": 8807,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 8807,\n        \"samples\": [\n          \"Game
Over, Man!\",\n          \"Arsenio Hall: Smart & Classy\",\n
\"Kazoops!\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"date_added\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"num_unique_values\": 1699,\n
\"samples\": [\n          \"2019-10-29\",\n          \"2021-05-19\",\n
\"2021-04-24\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"release_year\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 8,\n        \"min\": 1925,\n
\"max\": 2021,\n        \"num_unique_values\": 74,\n
\"samples\": [\n          1996,\n          1969,\n          2003\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"date_columns"}
```

```
df['cast']=df['cast'].fillna('unknown_actor')
df['cast'].apply(lambda x:x.split(', '))
a1=df['cast'].apply(lambda x:x.split(', ')).tolist()
cast=pd.DataFrame(a1,index=df['title'])
pd.DataFrame(a1,index=df['title']).stack()
pd.DataFrame(a1,index=df['title']).stack().reset_index()
cast=pd.DataFrame(a1,index=df['title']).stack().reset_index().drop(col
umns = 'level_1').rename(columns = {0:'cast'})
cast
```

```
{"summary":"{\n  \"name\": \"cast\",\n  \"rows\": 64951,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 8807,\n        \"samples\": [\n          \"Game
Over, Man!\",\n          \"Arsenio Hall: Smart & Classy\",\n
```

\"Kazoops!\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"cast\",\n   \"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 36440,\n   \"samples\": [\n
\"Robert Catrini\",\n   \"Jonathan Sandor\",\n
\"Gautam Kurup\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"cast"}

```
df['country']=df['country'].fillna('Missing_countryname')
df['country'].apply(lambda x:x.split(', '))
a2=df['country'].apply(lambda x:x.split(', ')).tolist()
country=pd.DataFrame(a2,index=df['title'])
pd.DataFrame(a2,index=df['title']).stack()
pd.DataFrame(a2,index=df['title']).stack().reset_index()
country=pd.DataFrame(a2,index=df['title']).stack().reset_index().drop(
columns = 'level_1').rename(columns = {0:'country'})
country
```

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

```
df['listed_in']=df['listed_in'].fillna('unknown_genre')
df['listed_in'].apply(lambda x:x.split(', '))
a3=df['listed_in'].apply(lambda x:x.split(', ')).tolist()
listed_in=pd.DataFrame(a3,index=df['title'])
pd.DataFrame(a3,index=df['title']).stack()
pd.DataFrame(a3,index=df['title']).stack().reset_index()
listed_in=pd.DataFrame(a3,index=df['title']).stack().reset_index().dro
p(columns = 'level_1').rename(columns = {0:'listed_in'})
listed_in
```

{"summary":"{\n \"name\": \"listed_in\",\n \"rows\": 19323,\n
\"fields\": [\n   {\n   \"column\": \"title\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 8807,\n   \"samples\": [\n   \"Game
Over, Man!\",\n   \"Arsenio Hall: Smart & Classy\",\n
\"Kazoops!\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":

```
\"listed_in\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 42,\n
\"samples\": [\n          \"Action & Adventure\",\n
\"Independent Movies\",\n          \"Romantic TV Shows\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"listed_in"}
```


Data cleaning significantly impacts accuracy, efficiency, and decision-making. By addressing errors, inconsistencies, and missing data, data cleaning ensures the reliability of insights and supports better decision-making. This also leads to **improved data quality and enhanced analysis processes.**

```
def safe_int(x):
    try:
        return int(x.split(' ')[0])  # converting to an integer
    except (ValueError, AttributeError):
        return None  # Return None if conversion fails

df['duration'] = df['duration'].astype(str)
df['duration_int'] = df['duration'].apply(safe_int)  # Extract integer
part of duration
df['duration_type'] = df['duration'].str.extract(r'(\D+)')  # Extract
the Duration Type (min or Season)
df['Movie_Minutes'] = df[df.type=='Movie']['duration'].apply(safe_int)
# Select desired columns for the new DataFrame
new_df = df[['title', 'duration_int',
'duration_type','Movie_Minutes']]

# Display the new DataFrame
new_df
```


```
{"summary":"{\n  \"name\": \"new_df\",\n  \"rows\": 8807,\n
\"fields\": [\n    {\n      \"column\": \"title\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 8807,\n        \"samples\": [\n          \"Game
Over, Man!\",\n          \"Arsenio Hall: Smart & Classy\",\n
\"Kazoops!\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"duration_int\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 50.81482778918891,\n        \"min\":
1.0,\n        \"max\": 312.0,\n        \"num_unique_values\": 210,\n
\"samples\": [\n          99.0,\n          152.0,\n          148.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"duration_type\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          \"
Seasons\",\n          \"nan\",\n          \" min\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Movie_Minutes\",\n
```

\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
28.290593447417347,\n   \"min\": 3.0,\n   \"max\": 312.0,\n
\"num_unique_values\": 205,\n   \"samples\": [\n   110.0,\
n   166.0,\n   34.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"new_df"}

Merging cleaned datasets offers several key benefits, including enhanced analysis capabilities, improved data quality, and increased efficiency. By combining data from various sources after cleaning, you can uncover hidden relationships, gain a more comprehensive view of your data, and ensure accuracy for more reliable insights.

```
# Defining type_shows
type_shows = df[['title', 'type']]

# Merging dataframes
merge1 = director.merge(cast, on='title')
merge2 = merge1.merge(country, on='title')
merge3 = merge2.merge(listed_in, on='title')
merge4 = merge3.merge(type_shows, on='title') # type_shows used here
merge5 = merge4.merge(date_columns, on='title')
cleaned_data = merge5.merge(df[['title', 'Movie_Minutes']],
on='title')
cleaned_data
```

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

```
cleaned_data.duplicated()
```

```
0         False
1         False
2         False
3         False
4         False
          ...
201986    False
201987    False
201988    False
201989    False
201990    False
Length: 201991, dtype: bool
```

```
cleaned_data.loc[cleaned_data.duplicated()]
```

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

```
cleaned_data.drop_duplicates(inplace=True)

cleaned_data
```

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

```
cleaned_data['title'].nunique()
```

```
8807
```

```
cleaned_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 201936 entries, 0 to 201990
Data columns (total 9 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   title          201936 non-null  object 
 1   director       201936 non-null  object 
 2   cast           201936 non-null  object 
 3   country        201936 non-null  object 
 4   listed_in      201936 non-null  object 
 5   type           201936 non-null  object 
 6   date_added     200190 non-null  object 
 7   release_year   201936 non-null  int64  
 8   Movie_Minutes  145785 non-null  float64
dtypes: float64(1), int64(1), object(7)
memory usage: 15.4+ MB
```

Removed the Duplicates and made changes in cleaned data

```
director_counts = cleaned_data['director'].value_counts()
print(director_counts)
cast_counts = cleaned_data['cast'].value_counts()
print(cast_counts)

country_counts = cleaned_data['country'].value_counts()
print(country_counts)

listed_in_counts = cleaned_data['listed_in'].value_counts()
print(listed_in_counts)

type_counts = cleaned_data['type'].value_counts()
print(listed_in_counts)
```

```
director
unknown_director       50643
Martin Scorsese          419
Youssef Chahine          409
Cathy Garcia-Molina      356
Steven Spielberg         355
                        ...
Harvey Lilley              1
Jason Orley                1
Jeannie Gaffigan           1
Mario Rouleau              1
```

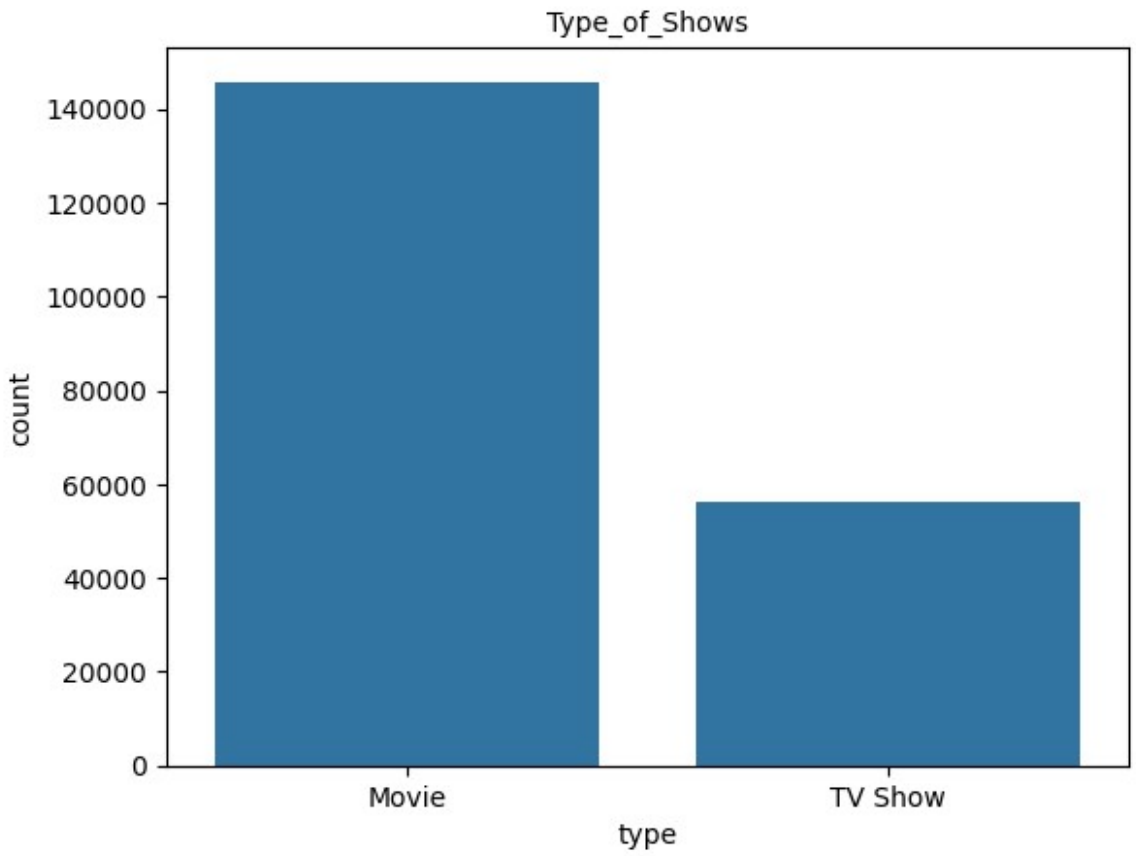
```
Richard Mears               1
Name: count, Length: 4994, dtype: int64
cast
unknown_actor      2146
Liam Neeson         161
Alfred Molina       160
John Krasinski      139
Salma Hayek         130
                   ...
Damien Echols         1
Anne Lamott           1
Duncan Trussell       1
Leather Storrs        1
Christian James       1
Name: count, Length: 36440, dtype: int64
country
United States          59324
India                  22814
United Kingdom         12945
Missing_countryname    11897
Japan                   8679
                       ...
Botswana                   2
United States,             1
Nicaragua                  1
Kazakhstan                 1
Uganda                     1
Name: count, Length: 128, dtype: int64
listed_in
Dramas                          29756
International Movies            28192
Comedies                        20829
International TV Shows          12845
Action & Adventure              12216
Independent Movies               9818
Children & Family Movies         9771
TV Dramas                        8942
Thrillers                        7106
Romantic Movies                  6412
TV Comedies                      4963
Crime TV Shows                   4733
Horror Movies                    4571
Kids' TV                         4568
Sci-Fi & Fantasy                 4037
Music & Musicals                 3077
Romantic TV Shows                3049
Documentaries                    2407
Anime Series                     2313
TV Action & Adventure            2288
```

```
Spanish-Language TV Shows        2126
British TV Shows                 1808
Sports Movies                    1531
Classic Movies                   1434
TV Mysteries                     1281
Korean TV Shows                  1122
Cult Movies                      1077
Anime Features                   1045
TV Sci-Fi & Fantasy              1045
TV Horror                         941
Docuseries                        845
LGBTQ Movies                      838
TV Thrillers                      768
Teen TV Shows                     742
Reality TV                        735
Faith & Spirituality              719
Stand-Up Comedy                   540
Movies                            412
TV Shows                          337
Classic & Cult TV                 272
Stand-Up Comedy & Talk Shows      268
Science & Nature TV               157
Name: count, dtype: int64
listed_in
Dramas                          29756
International Movies            28192
Comedies                        20829
International TV Shows          12845
Action & Adventure              12216
Independent Movies               9818
Children & Family Movies         9771
TV Dramas                        8942
Thrillers                        7106
Romantic Movies                  6412
TV Comedies                      4963
Crime TV Shows                   4733
Horror Movies                    4571
Kids' TV                         4568
Sci-Fi & Fantasy                 4037
Music & Musicals                 3077
Romantic TV Shows                3049
Documentaries                    2407
Anime Series                     2313
TV Action & Adventure            2288
Spanish-Language TV Shows        2126
British TV Shows                 1808
Sports Movies                    1531
Classic Movies                   1434
TV Mysteries                     1281
```

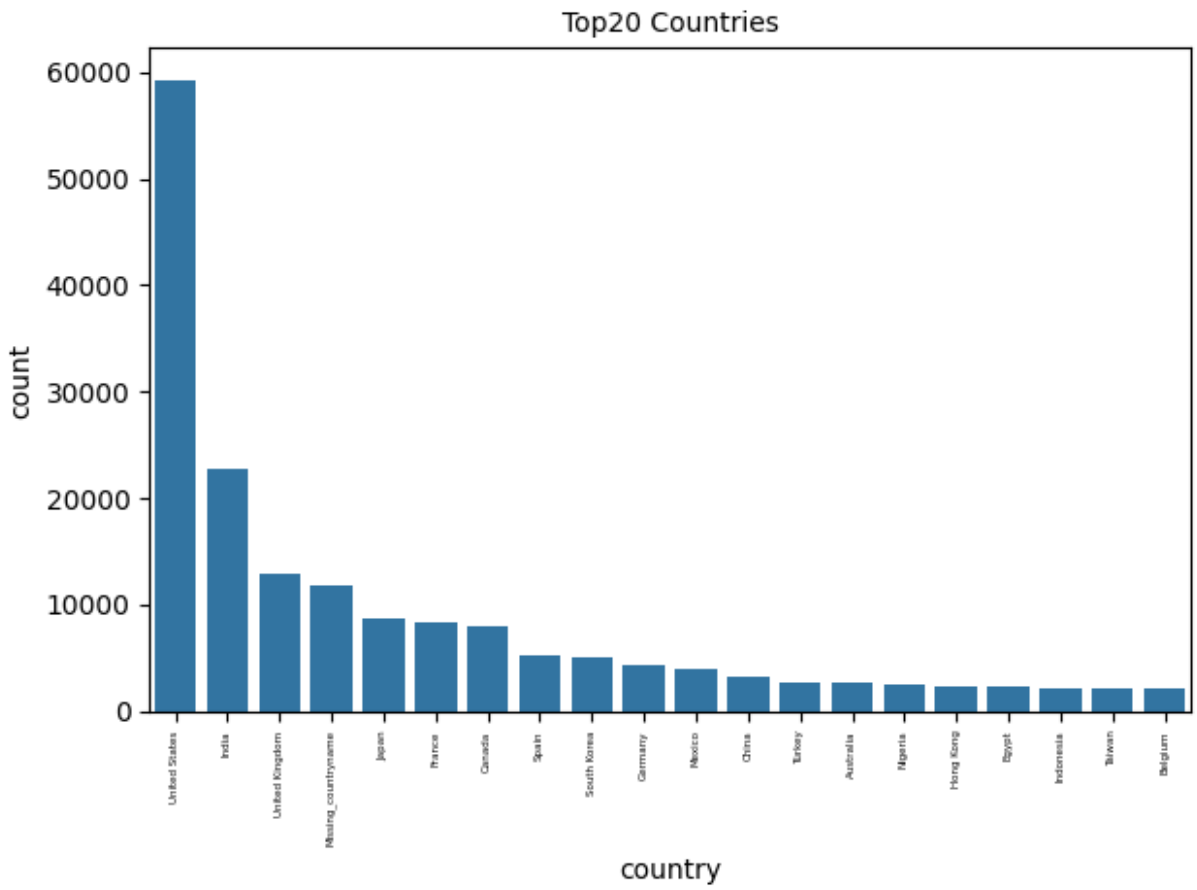
```
Korean TV Shows                    1122
Cult Movies                        1077
Anime Features                     1045
TV Sci-Fi & Fantasy                1045
TV Horror                           941
Docuseries                          845
LGBTQ Movies                        838
TV Thrillers                        768
Teen TV Shows                       742
Reality TV                          735
Faith & Spirituality                719
Stand-Up Comedy                     540
Movies                              412
TV Shows                            337
Classic & Cult TV                   272
Stand-Up Comedy & Talk Shows        268
Science & Nature TV                 157
Name: count, dtype: int64
```

PS1: Count of Categorical variable

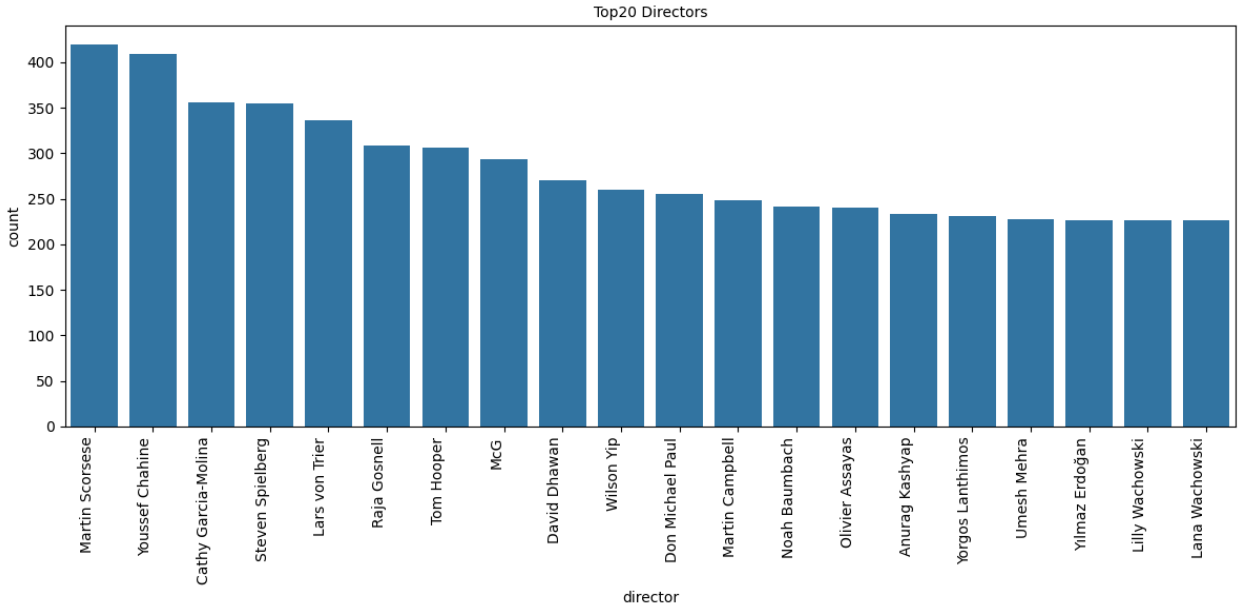
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns


sns.countplot(x='type',data=cleaned_data)
plt.title('Type_of_Shows',fontsize=10)
plt.show()
```




```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

top_countries =
cleaned_data['country'].value_counts().nlargest(20).index
sns.countplot(x='country',data=cleaned_data,order=top_countries)
plt.xticks(rotation=90, fontsize=4)
plt.title('Top20 Countries',fontsize=10)
plt.tight_layout()
plt.show()
```




```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

# Filter out 'unknown_director' rows before calculating top directors
filtered_data = cleaned_data[cleaned_data['director'] !=
'unknown_director']

top_directors =
filtered_data['director'].value_counts().nlargest(20).index
plt.figure(figsize=(12, 6))
sns.countplot(x='director', data=cleaned_data, order=top_directors)
plt.xticks(rotation=90, ha='right')
plt.title('Top20 Directors',fontsize=10)
plt.tight_layout()
plt.show()
```

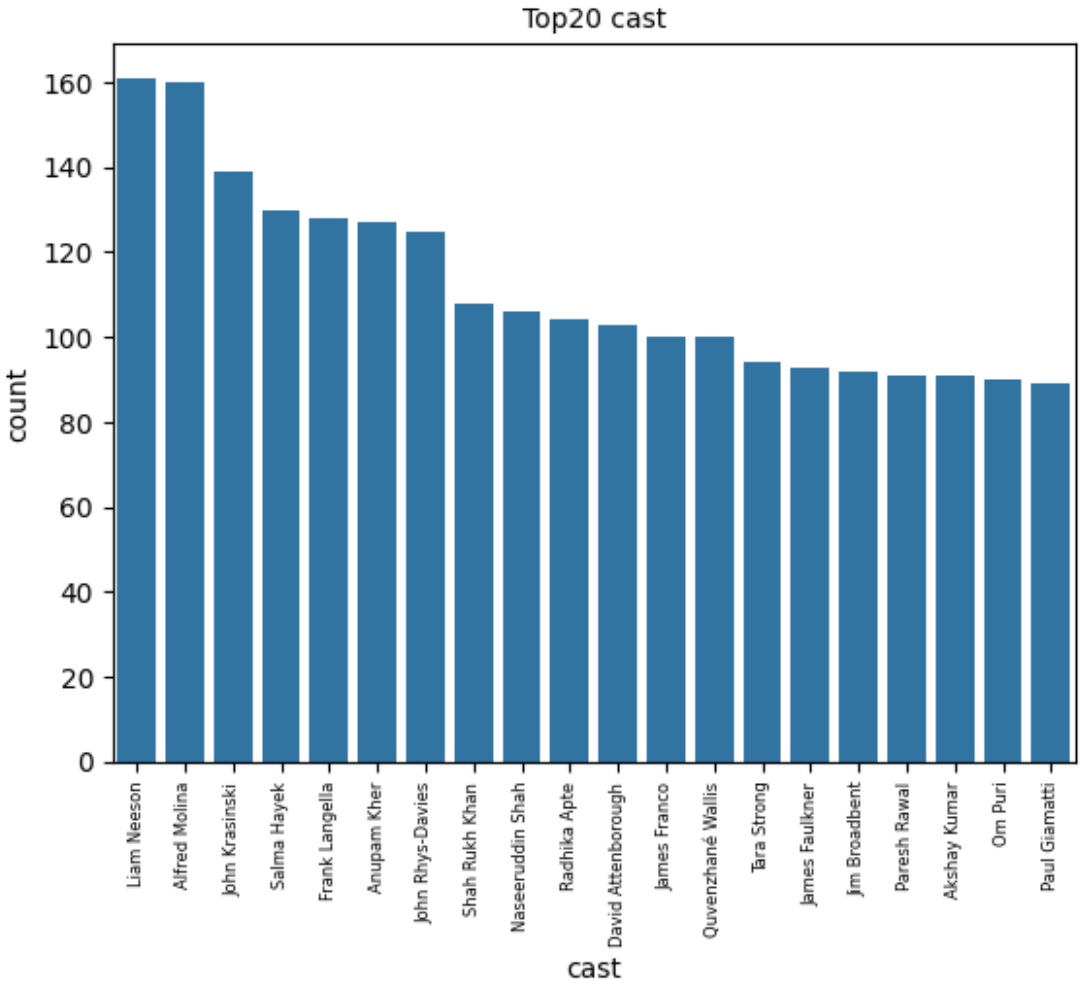


Insights from the counts of each categorical variable:

- **Type**: Movies are highly added in Netflix compared with TV_Shows
- **Country** United states have the more counts of TV shows and Movies
- **Directors** Martin Scorsese and Youssef Chahine are the top most directors with more number
- **Cast** Liam Neeson and Affred Molina are top actors with more count of movies and tv shows
- **Listed_in** Dramas and International movies are the most listed genres

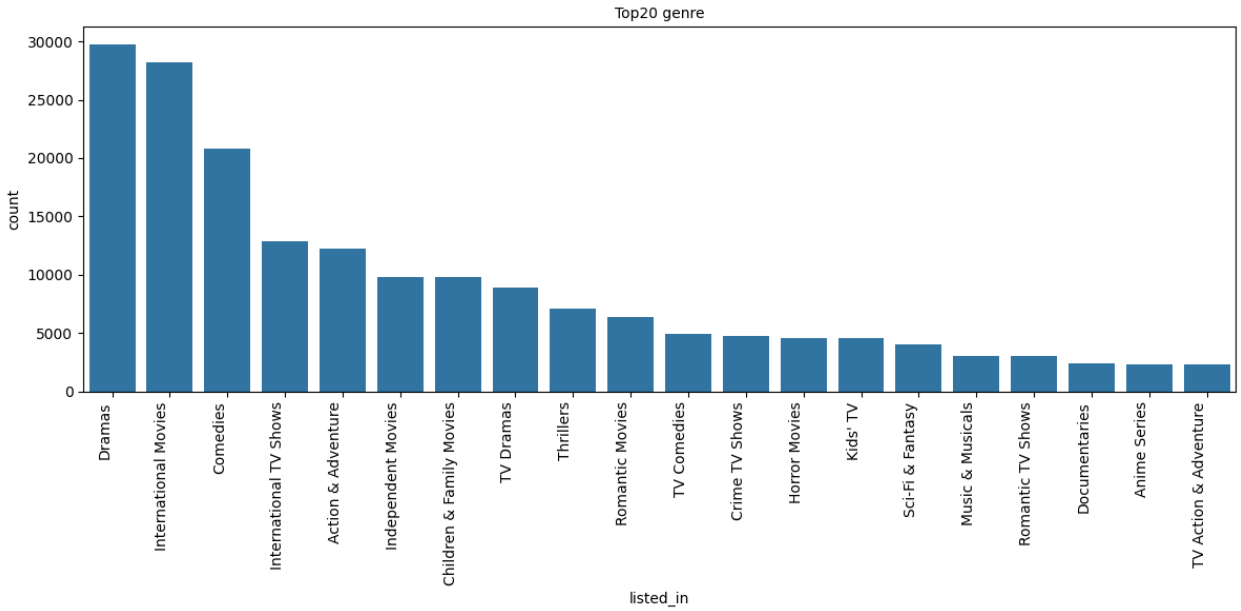
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import re

# #cast contains $ and we were getting value error while plotting
# cleaned_data['cast'] = cleaned_data['cast'].str.replace('$', '',
regex=True)

cleaned_data['cast'] = cleaned_data['cast'].str.replace('Joey Bada$
$','Joey Badass')
cleaned_data['cast'] = cleaned_data['cast'].str.replace('Too
$hort','Too Short')
filtered_cast = cleaned_data[cleaned_data['cast'] != 'unknown_actor']
top_casts = filtered_cast['cast'].value_counts().nlargest(20).index
sns.countplot(x='cast',data=cleaned_data,order=top_casts)
plt.title('Top20 cast',fontsize=10)
```

```
plt.xticks(rotation=90, fontsize=6)
plt.show()
```




```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

# Filter out 'unknown_director' rows before calculating top directors
filtered_data = cleaned_data[cleaned_data['listed_in'] !=
'unknown_genre']

top_genres =
filtered_data['listed_in'].value_counts().nlargest(20).index
plt.figure(figsize=(12, 6))
sns.countplot(x='listed_in', data=cleaned_data, order=top_genres)
plt.xticks(rotation=90, ha='right')
plt.title('Top20 genre',fontsize=10)
```

```
plt.tight_layout()
plt.show()
```



PS2: Count of Movies among each countries

```
filtered_data = cleaned_data[cleaned_data['country'] != 
'Missing_countryname']
groupby_movies=filtered_data[filtered_data['type']=='Movie'].groupby('
country')['title'].count()
groupby_movies.sort_values(ascending=False).head(10)
```

```
country
United States     45791
India             21411
United Kingdom     8560
France             6605
Canada             5738
Japan              3525
Spain              3469
Germany            3427
China              2377
Nigeria            2236
Name: title, dtype: int64
```

Insights: The United States has a significantly higher number of content available compared to other countries. Netflix should prioritize regional content acquisition and production, particularly in high-potential markets

PS3:Count of TV shows among each countries

```
filtered_data = cleaned_data[cleaned_data['country'] !=
'Missing_countryname']
groupby_tvShows=filtered_data[filtered_data['type']=='TV
Show'].groupby('country')['title'].count()
groupby_tvShows.sort_values(ascending=False).head(10)
```

```
country
United States     13533
Japan              5154
United Kingdom     4385
South Korea        3754
Canada             2177
Mexico             2018
Spain              1846
Taiwan             1719
France             1647
India              1403
Name: title, dtype: int64
```

Insights: The United States has the highest number of content on Netflix (13,533), followed by Japan (5,154) and the United Kingdom (4,385). Several other countries, like India and France, are also contributing significantly to content volume.

Netflix should consider adding more tv shows and movies from other countries as well to increase diversity.

PS4:Cast contribution on TV shows

```
filtered_cast_data = cleaned_data[cleaned_data['cast'] !=
'unknown_actor']
groupby_cast_tvShow=filtered_cast_data[filtered_cast_data['type']=='TV
Show'].groupby('cast')['title'].count()
groupby_cast_tvShow =
groupby_cast_tvShow.sort_values(ascending=False).head(10)
print(groupby_cast_tvShow)
```

```
cast
David Attenborough    82
Takahiro Sakurai      56
Yuki Kaji             45
Ai Kayano             41
Junichi Suwabe        39
Daisuke Ono           38
Yuichi Nakamura       38
Jun Fukuyama          38
Kate Harbour          37
Amandla Stenberg      35
Name: title, dtype: int64
```

Insights: These actors are in high demand in TV shows, and Netflix could prioritize collaborations with them to attract viewers. Their presence in multiple productions indicates their ability to draw an audience.

PS5:Cast Contribution in Movies

```
filtered_cast_data = cleaned_data[cleaned_data['cast'] !=
'unknown_actor']
groupby_cast_movie=filtered_cast_data[filtered_cast_data['type']=='Mov
ie'].groupby('cast')['title'].count()
groupby_cast_movie =
groupby_cast_movie.sort_values(ascending=False).head(10)
print(groupby_cast_movie)
```

```
cast
Liam Neeson           161
Alfred Molina         157
John Krasinski        138
Salma Hayek           130
Frank Langella        128
Anupam Kher           118
John Rhys-Davies      116
Shah Rukh Khan        108
Naseeruddin Shah      106
Quvenzhané Wallis     100
Name: title, dtype: int64
```

Insights: These actors are in high demand in Movies, and Netflix could prioritize collaborations with them to attract viewers. Their presence in multiple productions indicates their ability to draw an audience.

PS6:Top Directors of TVshows

```
filtered_director_data = cleaned_data[cleaned_data['director'] !=
'unknown_director']
groupby_director_tvshow=filtered_director_data[filtered_director_data[
'type']=='TV Show'].groupby('director')['title'].count()
groupby_director_tvshow.sort_values(ascending=False).head(10)
```

```
director
Noam Murro           189
Thomas Astruc        160
Damien Chazelle      104
Alan Poul            104
Houda Benyamina      104
Laïla Marrakchi      104
Rob Seidenglanz      103
Alejandro Lozano      90
Jay Oliva             81
```

```
Manolo Caro           78
Name: title, dtype: int64
```

The high TV Show counts of some directors highlight their significant contribution to Netflix's content volume

PS6:Top Directors in Movies

```
filtered_director_data = cleaned_data[cleaned_data['director'] !=
'unknown_director']
groupby_director_movie=filtered_director_data[filtered_director_data['
type']=='Movie'].groupby('director')['title'].count()
groupby_director_movie.sort_values(ascending=False).head(10)
```

```
director
Martin Scorsese         419
Youssef Chahine         409
Cathy Garcia-Molina     356
Steven Spielberg        355
Lars von Trier          336
Raja Gosnell            308
Tom Hooper              306
McG                     293
David Dhawan            270
Wilson Yip              260
Name: title, dtype: int64
```

Insights:Analysis on Directors with shows

- **Noam Murro** have directed more TV shows
- **Martin Scorsese** have directed more Movies Netflix should consider prioritizing collaborations with directors who have demonstrated high levels of productivity and involvement on the platform. This can ensure a steady flow of new content and potentially attract a loyal audience base.

PS7:Best Month to add Movies and TV Shows

```
cleaned_data['date_added'] =
pd.to_datetime(cleaned_data['date_added'], errors='coerce')
# cleaned_data['release_year'] =
pd.to_datetime(cleaned_data['release_year'], errors='coerce')

cleaned_data['date_added_month'] =
cleaned_data['date_added'].dt.month_name()
cleaned_data
```

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

```
month_groupby_tvShow=cleaned_data[cleaned_data['type']=='TV
Show'].groupby('date_added_month')['title'].count()
month_groupby_tvShow.sort_values(ascending=False).head(10)
```

```
date_added_month
December     5341
July         5129
August       5029
June         4959
September    4818
April        4460
November     4428
March        4201
October      4199
May          4111
Name: title, dtype: int64
```

```
month_groupby_movie=cleaned_data[cleaned_data['type']=='Movie'].groupb
y('date_added_month')['title'].count()
month_groupby_movie.sort_values(ascending=False).head(10)
```

```
date_added_month
July         15049
January      13947
October      13508
September    13219
December     12768
April        12538
August       11924
June         11568
March        11489
November     11062
Name: title, dtype: int64
```

Insights:Analysis for best Month to add TV Shows and Movies

- Netflix should continue to prioritize content additions during December and July to cater to audience demand.
- Marketing and promotional activities can be aligned with content additions during peak periods to further boost engagement.

PS8:Top Genres in the dataset

```
India_groupby_genre=cleaned_data[cleaned_data['country']=='India'].gro
upby('listed_in')['listed_in'].count()
India_groupby_genre.sort_values(ascending=False).head(10)
```

```
listed_in
International Movies      7059
Dramas                    5569
```

```
Comedies                   2685
Independent Movies         1394
Action & Adventure         1187
Romantic Movies             931
Music & Musicals            847
Thrillers                   743
International TV Shows      428
Horror Movies               307
Name: listed_in, dtype: int64
```

Netflix should Focus on acquiring and producing more content within the dominant genres ("International Movies," "Dramas," and "Comedies") to cater to the largest user base.

PS9:Country with highest Horror Movie count

```
filtered_data = cleaned_data[cleaned_data['country'] !=
'Missing_countryname']
Horror_groupby_genre=filtered_data[filtered_data['listed_in']=='Horror
Movies'].groupby('country')['country'].count()
Horror_groupby_genre.sort_values(ascending=False).head(10)
```

```
country
United States     2078
Canada             363
India              307
United Kingdom     217
Thailand           187
Ireland            145
France              93
Indonesia           91
Mexico              75
Spain               74
Name: country, dtype: int64
```

Insights: Netflix should prioritize acquiring and producing more horror movies from regions with a growing interest in the genre, like India, Canada, and the UK. This will cater to diverse audience preferences and potentially uncover new talent.

PS10:Average duration of Horror Movie

```
type_shows = df[['title', 'type', 'Movie_Minutes']]

horror_movies =
cleaned_data[cleaned_data['listed_in'].str.contains('Horror')]

# Calculating the average duration
average_duration = horror_movies['Movie_Minutes'].mean()

# Printing the result
```

```
print(f"The average duration of horror movies is:{average_duration}
minutes")
```

```
The average duration of horror movies is:99.01903303434698 minutes
```

Insights: The average duration of a horror movie is found to be 99 minutes. This information will be useful for my friend who is planning to direct a horror movie.

PS11:TV shows with more number of seasons

```
tv_shows = cleaned_data[cleaned_data['type'] == 'TV Show']
tv_show_counts = tv_shows.groupby('title')
['title'].count().reset_index(name='watch_count')

# Merging with original dataframe to get duration (number of seasons)
tv_show_counts = pd.merge(tv_show_counts, df[['title', 'duration']],
on='title', how='left')

# Converting duration to numeric (number of seasons) and handling non-
numeric values
tv_show_counts['duration'] =
tv_show_counts['duration'].str.extract('(\d+)').astype(float)
# Sorting by duration (number of seasons) and then watch count
tv_show_counts = tv_show_counts.sort_values(['duration',
'watch_count'], ascending=[False, False])


# To get the top show based on the highest number of seasons
top_show_by_seasons = tv_show_counts.iloc[0]


# Printing the result

print(f"The TV show with the most seasons is:
{top_show_by_seasons['title']}")
print(f"Number of seasons: {top_show_by_seasons['duration']}")
```

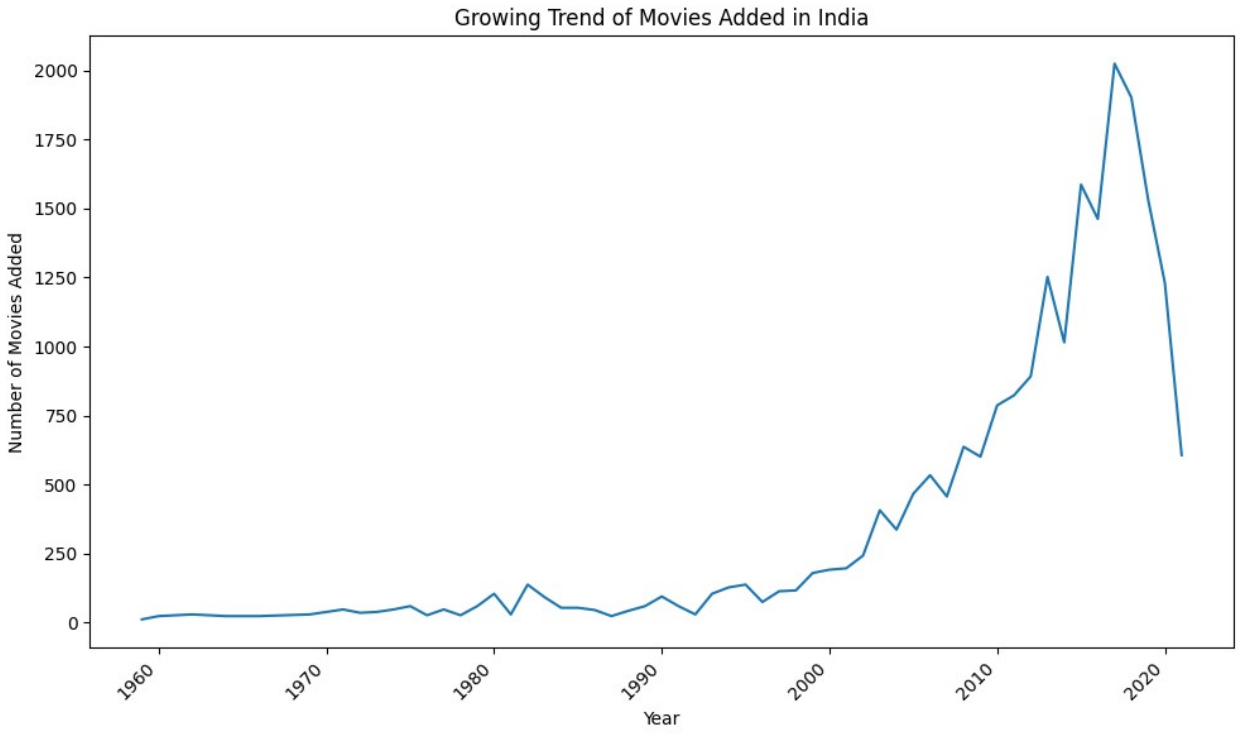
```
The TV show with the most seasons is: Grey's Anatomy
Number of seasons: 17.0
```

Insights: It is found that **Grey's Anatomy** is a TV show with highest number of seasons By leveraging the success of "Grey's Anatomy," Netflix can further enhance its content offering and engage viewers who enjoy long-running, medical dramas with compelling storylines and characters.

PS12:Growing Trend of Movies added in India

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Filtering data for movies in India
india_movies = cleaned_data[(cleaned_data['country'] == 'India') &
(cleaned_data['type'] == 'Movie')]

# Group by year added and count occurrences
movie_counts_by_year = india_movies.groupby('release_year')
['title'].count().reset_index(name='count')

# Creating a line plot
plt.figure(figsize=(10, 6))
sns.lineplot(x='release_year', y='count', data=movie_counts_by_year)
plt.title('Growing Trend of Movies Added in India')
plt.xlabel('Year')
plt.ylabel('Number of Movies Added')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Insights: Netflix should continue to capitalize on the growing interest in Indian cinema by acquiring and producing more high-quality Indian movies across various genres Netflix should investigate the reasons behind the recent decline in Indian movie additions and address any underlying issues to maintain a steady flow of new content.

PS13: Find the recently added year in Netflix dataset

```
# Converting 'date_added' to datetime objects
cleaned_data['date_added'] =
```

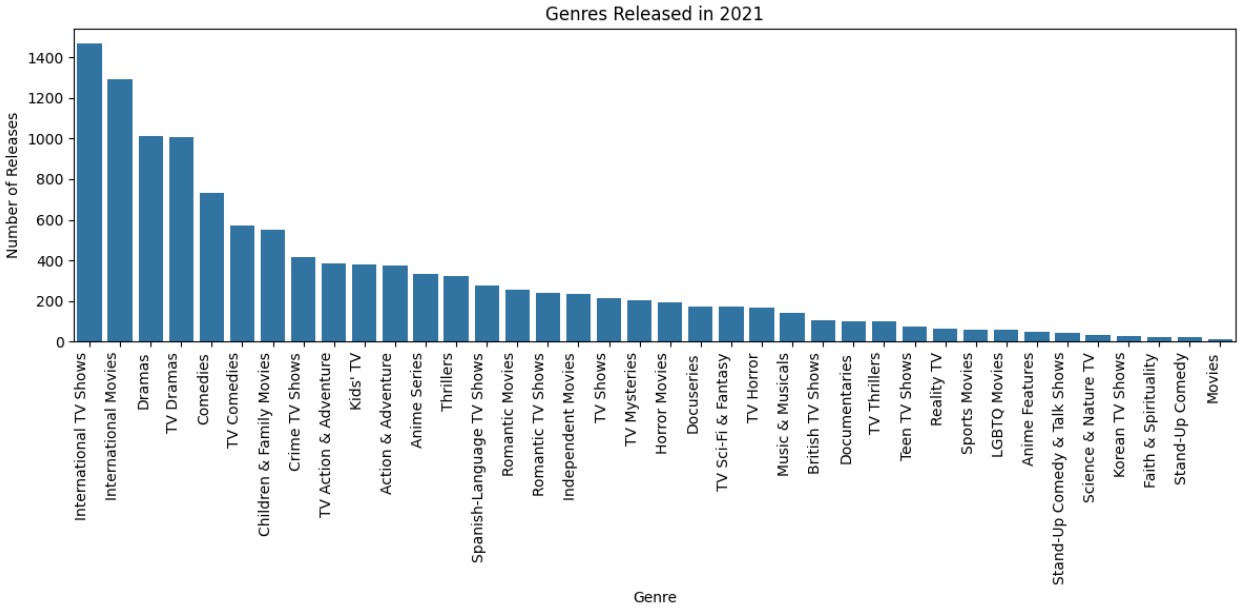
```
pd.to_datetime(cleaned_data['date_added'], errors='coerce')

# Extracting the year
cleaned_data['year_added'] = cleaned_data['date_added'].dt.year

# To Find the most recent year
most_recent_year = cleaned_data['year_added'].max()

# Printing the result
print(f"The most recent added year as per netflix dataset is:
{most_recent_year}")
```

```
The most recent added year as per netflix dataset is: 2021.0
```

2021 is the most recent year added as per netfix dataset

PS14:Geners released in 2021

```
import matplotlib.pyplot as plt
import seaborn as sns

# Filtering data for content released in 2021
released_2021 = cleaned_data[cleaned_data['release_year'] == 2021]

# Group by genre and count occurrences
genre_counts = released_2021.groupby('listed_in')
['title'].count().reset_index(name='count')

# Sort by count in descending order
genre_counts = genre_counts.sort_values('count', ascending=False)

# Create a bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x='listed_in', y='count', data=genre_counts)
plt.title('Genres Released in 2021')
plt.xlabel('Genre')
plt.ylabel('Number of Releases')
plt.xticks(rotation=90, ha='right')
plt.tight_layout()
plt.show()
```

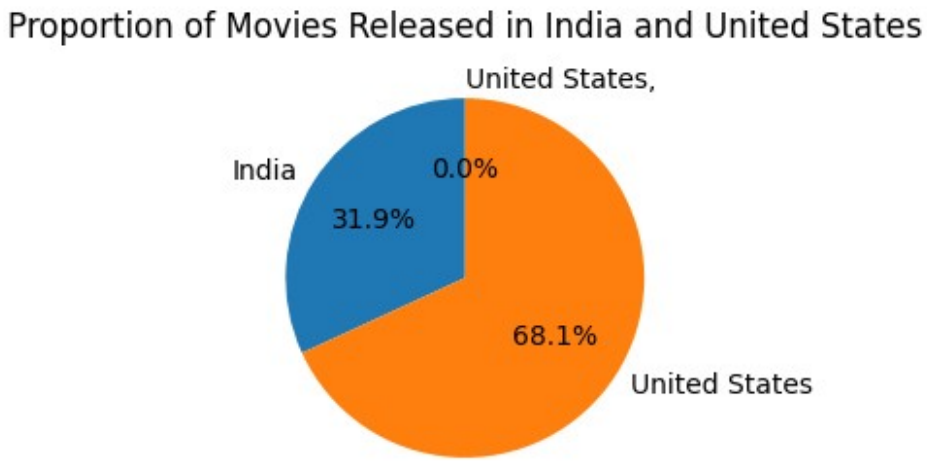


Insights: International TV Shows,Movies and Dramas genres are added in the year 2021.

While dramas and comedies remain popular, Netflix could explore other genres to cater to niche audiences. This can involve acquiring or producing content in genres like thrillers, documentaries, and sci-fi, ensuring a diverse and engaging content library.

PS15:Proportion of movie released in India and US

```
import matplotlib.pyplot as plt

# Filtering data for movies
movies_data = cleaned_data[cleaned_data['type'] == 'Movie']

# Filtering for India and United States
india_us_movies = movies_data[movies_data['country'].apply(lambda x:
'India' in x or 'United States' in x)]

# Group by country and count occurrences
country_counts = india_us_movies.groupby('country')['title'].count()

# Creating a pie chart
plt.figure(figsize=(3, 3))  # Adjust figure size as needed
plt.pie(country_counts, labels=country_counts.index, autopct='%1.1f%
%', startangle=90)
plt.title('Proportion of Movies Released in India and United States')
plt.show()
```



Insights: While the United States has a dominant presence, Netflix should prioritize acquiring and producing more content from other regions, particularly those with high growth potential like India, Japan, and South Korea. This can attract a wider audience and cater to diverse preferences.

PS16:Indian Movies cast count

```
indian_movies = cleaned_data[(cleaned_data['country'] == 'India') &
(cleaned_data['type'] == 'Movie')]
filtered_cast_data = indian_movies[indian_movies['cast'] !=
'unknown_actor']
actor_counts = filtered_cast_data.groupby('cast')
['title'].count().reset_index(name='movie_count')
print (actor_counts)
```

```
                      cast  movie_count
0               A.K. Hangal           12
1               A.R. Rahman            3
2           A.S. Sasi Kumar            3
3              Aabhas Yadav            3
4             Aachal Munjal            2
...                     ...          ...
3677           Zohra Sehgal            3
3678           Zoya Hussain            3
3679            Zul Vellani            3
3680  Ólafur Darri Ólafsson            2
3681            Şafak Sezer            3

[3682 rows x 2 columns]
```

A.K Hangal is the actor casted in majority of movies added in India

PS17:Year when Maximum number of movies added

```
movies_data = cleaned_data[cleaned_data['type'] == 'Movie']
import pandas as pd

cleaned_data['date_added'] =
pd.to_datetime(cleaned_data['date_added'], errors='coerce')
cleaned_data['year_added'] = cleaned_data['date_added'].dt.year
max_year = cleaned_data[cleaned_data['type'] ==
'Movie'].groupby('year_added')
['title'].count().reset_index(name='movie_count').loc[lambda df:
df['movie_count'].idxmax()]

print(f"The year with the most movies added to Netflix is:
{max_year['year_added']}")
print(f"Number of movies added: {max_year['movie_count']}")
```

```
The year with the most movies added to Netflix is: 2019.0
Number of movies added: 34392.0
```

2019 is the maximum number of Movies added in Netflix

PS18:Actors acted in multiple genres

```
exploded_genres = cleaned_data.explode('listed_in')

filtered_cast_data = exploded_genres[exploded_genres['cast'] !=
'unknown_actor']

actor_genres = filtered_cast_data.groupby('cast')
['listed_in'].nunique().reset_index(name='genre_count')

multi_genre_actors = actor_genres[actor_genres['genre_count'] > 1]

multi_genre_actors = multi_genre_actors.sort_values('genre_count',
ascending=False)
print("Actors who have acted in multiple genres:")
print(multi_genre_actors)
```

```
Actors who have acted in multiple genres:
                     cast  genre_count
28716          Ron Perlman           17
18153       Kiernan Shipka           16
11174            Gary Cole           16
11641          Glenn Close           15
29600    Samuel L. Jackson           14
...                    ...          ...
25892         Pascal Atuma            2
25882       Parvati Sehgal            2
25876     Parthveer Shukla            2
25906        Pasi Ruohonen            2
25905  Pasha D. Lychnikoff            2
```

```
[32665 rows x 2 columns]
```

Insights:

Ron Perlman acted in different types of genres, showcasing various emotions. Netflix should prioritize collaborations with actors identified as multi-genre performers. These actors demonstrate flexibility and appeal to a broader audience, potentially increasing viewership and engagement. Netflix could offer them diverse roles across different genres to leverage their talent and attract a wider fanbase.

PS19:Analysis of Directors with recent released year

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

most_recent_year = cleaned_data['release_year'].max()
recent_releases = cleaned_data[(cleaned_data['release_year'] ==
most_recent_year) & (cleaned_data['director'] != 'unknown_director')]

director_counts = recent_releases.groupby('director')
['title'].count().reset_index(name='count')
top_directors = director_counts.sort_values('count',
ascending=False).head(10)

plt.figure(figsize=(9,4))
sns.barplot(x='director', y='count', data=top_directors,
color='violet', width = 0.5)
plt.xticks(rotation=90, ha='right')
plt.title(f'Top Directors with Releases in {most_recent_year}
(Excluding Unknown)')
plt.xlabel('Director')
plt.ylabel('Number of Releases')
plt.tight_layout()
plt.show()
```

Insights: Netflix should strengthen relationships with the top directors identified in the analysis. This could involve offering exclusive deals, co-production opportunities, or creative workshops to foster long-term collaborations and ensure a steady flow of high-quality content.