business-case-netflix-sukanya

May 9, 2025

Netflix Business Case study:

Importing Python Libraries:

Python libraries allows us to accomplish tasks and run data analysis more efficiently by providing portions of crucial code already built for us.

[98]:
```
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/ori
ginal/netflix.csv
To: /content/netflix.csv
100% 3.40M/3.40M [00:00<00:00, 20.8MB/s]
```

[99]:
```
df=pd.read_csv('netflix.csv')
```

[154]:
```
df.describe()
```

[154]:

	release_year	duration_int	Movie_Minutes
count	8807.000000	8804.000000	6128.000000
mean	2014.180198	69.846888	99.577187
std	8.819312	50.814828	28.290593
min	1925.000000	1.000000	3.000000
25%	2013.000000	2.000000	87.000000
50%	2017.000000	88.000000	98.000000
75%	2019.000000	106.000000	114.000000
max	2021.000000	312.000000	312.000000

Data Cleaning:

[100]:
```
df.columns
```

[100]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

[101]:
```
duplicate=df.duplicated().value_counts()
print(duplicate)
```

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

[102]:
```
# NAN values replaced by 'Missing
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
```

[102]:
```
[['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'],
['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'],
…]
```

[103]:
```
# separating the director name based on title by setting title as index
b=pd.DataFrame(a,index=df['title'])
```

```
b
```

[103]:
```
                                0     1     2     3     4     5     6   \
title
Dick Johnson Is Dead     Kirsten Johnson  None  None  None  None  None  None
Blood & Water           unknown_director  None  None  None  None  None  None
Ganglands                Julien Leclercq  None  None  None  None  None  None
Jailbirds New Orleans   unknown_director  None  None  None  None  None  None
Kota Factory            unknown_director  None  None  None  None  None  None
…                                     …   …   …   …   …   …   …
Zodiac                     David Fincher  None  None  None  None  None  None
Zombie Dumb             unknown_director  None  None  None  None  None  None
Zombieland               Ruben Fleischer  None  None  None  None  None  None
Zoom                        Peter Hewitt  None  None  None  None  None  None
Zubaan                       Mozez Singh  None  None  None  None  None  None

                          7     8     9     10    11    12
title
Dick Johnson Is Dead    None  None  None  None  None  None
Blood & Water           None  None  None  None  None  None
Ganglands               None  None  None  None  None  None
Jailbirds New Orleans   None  None  None  None  None  None
Kota Factory            None  None  None  None  None  None
…                          …   …   …   …   …   …
Zodiac                  None  None  None  None  None  None
Zombie Dumb             None  None  None  None  None  None
Zombieland              None  None  None  None  None  None
Zoom                    None  None  None  None  None  None
Zubaan                  None  None  None  None  None  None

[8807 rows x 13 columns]
```

[104]:
```
#Using stack merging the columns to rows and shows 0,1 as per number of␣
 ↪directors
pd.DataFrame(a,index=df['title']).stack()
```

[104]:
```
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
```

[105]:
```
#On Stacking pandas create the index name for the unnamed one by using stack
pd.DataFrame(a,index=df['title']).stack().reset_index()
```

[105]:

	title	level_1	0
0	Dick Johnson Is Dead	0	Kirsten Johnson
1	Blood & Water	0	unknown_director
2	Ganglands	0	Julien Leclercq
3	Jailbirds New Orleans	0	unknown_director
4	Kota Factory	0	unknown_director
…	…	…	…
9607	Zodiac	0	David Fincher
9608	Zombie Dumb	0	unknown_director
9609	Zombieland	0	Ruben Fleischer
9610	Zoom	0	Peter Hewitt
9611	Zubaan	0	Mozez Singh

[9612 rows x 3 columns]

[106]:
```
director=pd.DataFrame(a,index=df['title']).stack().reset_index().drop(columns =␣
↪'level_1').rename(columns = {0:'director'})
director
```

[106]:

	title	director
0	Dick Johnson Is Dead	Kirsten Johnson
1	Blood & Water	unknown_director
2	Ganglands	Julien Leclercq
3	Jailbirds New Orleans	unknown_director
4	Kota Factory	unknown_director
…	…	…
9607	Zodiac	David Fincher
9608	Zombie Dumb	unknown_director
9609	Zombieland	Ruben Fleischer
9610	Zoom	Peter Hewitt
9611	Zubaan	Mozez Singh

[9612 rows x 2 columns]

[107]:
```
type_shows=df[['title','type']]
type_shows
```

[107]:

	title	type
0	Dick Johnson Is Dead	Movie
1	Blood & Water	TV Show
2	Ganglands	TV Show

```
3     Jailbirds New Orleans  TV Show
4                Kota Factory  TV Show
…                        …      …
8802                 Zodiac    Movie
8803            Zombie Dumb  TV Show
8804             Zombieland    Movie
8805                   Zoom    Movie
8806                 Zubaan    Movie

[8807 rows x 2 columns]
```

[108]:
```
date_columns=df[['title','date_added','release_year']]
date_columns
```

[108]:
```
                      title          date_added  release_year
0       Dick Johnson Is Dead  September 25, 2021          2020
1               Blood & Water  September 24, 2021          2021
2                  Ganglands  September 24, 2021          2021
3     Jailbirds New Orleans  September 24, 2021          2021
4                Kota Factory  September 24, 2021          2021
…                        …                 …             …
8802                 Zodiac   November 20, 2019          2007
8803            Zombie Dumb        July 1, 2019          2018
8804             Zombieland    November 1, 2019          2009
8805                   Zoom    January 11, 2020          2006
8806                 Zubaan       March 2, 2019          2015

[8807 rows x 3 columns]
```

[109]:
```
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(columns =␣
↪'level_1').rename(columns = {0:'cast'})
cast
```

[109]:
```
                      title                  cast
0       Dick Johnson Is Dead          unknown_actor
1               Blood & Water             Ama Qamata
2               Blood & Water           Khosi Ngema
3               Blood & Water         Gail Mabalane
4               Blood & Water        Thabang Molaba
…                        …                     …
64946                Zubaan       Manish Chaudhary
```

```
64947                Zubaan           Meghna Malik
64948                Zubaan          Malkeet Rauni
64949                Zubaan         Anita Shabdish
64950                Zubaan  Chittaranjan Tripathy

[64951 rows x 2 columns]
```

[110]:
```
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
```

[110]:
```
                      title              country
0       Dick Johnson Is Dead        United States
1              Blood & Water         South Africa
2                  Ganglands  Missing_countryname
3      Jailbirds New Orleans  Missing_countryname
4               Kota Factory                India
…                        …                    …
10840               Zodiac        United States
10841          Zombie Dumb  Missing_countryname
10842           Zombieland        United States
10843                 Zoom        United States
10844               Zubaan                India

[10845 rows x 2 columns]
```

[111]:
```
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().drop(columns␣
↪= 'level_1').rename(columns = {0:'listed_in'})
listed_in
```

[111]:
```
                      title                listed_in
0       Dick Johnson Is Dead            Documentaries
1              Blood & Water  International TV Shows
2              Blood & Water                TV Dramas
3              Blood & Water           TV Mysteries
```

```
4                 Ganglands            Crime TV Shows
…                       …                       …
19318                  Zoom  Children & Family Movies
19319                  Zoom                 Comedies
19320                Zubaan                   Dramas
19321                Zubaan      International Movies
19322                Zubaan          Music & Musicals

[19323 rows x 2 columns]
```

[112]:
```
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
```

[112]:
```
                     title  duration_int duration_type  Movie_Minutes
0      Dick Johnson Is Dead          90.0           min           90.0
1             Blood & Water           2.0       Seasons            NaN
2                 Ganglands           1.0        Season            NaN
3     Jailbirds New Orleans           1.0        Season            NaN
4              Kota Factory           2.0       Seasons            NaN
…                       …           …           …              …
8802                Zodiac         158.0           min         158.0
8803           Zombie Dumb           2.0       Seasons            NaN
8804            Zombieland          88.0           min          88.0
8805                  Zoom          88.0           min          88.0
8806                Zubaan         111.0           min         111.0

[8807 rows x 4 columns]
```

Data cleaning is crucial before analysis because it ensures the accuracy, consistency, and reliability of the data leading to more meaningful and trustworthy results. By removing errors, inconsistencies, and missing values data cleaning minimizes the risk of inaccurate analysis and helps in making

informed decisions.

Merging the columns:

[113]:
```
# 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
```

[113]:

	title	director	cast	country	listed_in	type	date_added	release_year	Movie_Minutes
0	Dick Johnson Is Dead	Kirsten Johnson	unknown_actor	United States	Documentaries	Movie	September 25, 2021	2020	90.0
1	Blood & Water	unknown_director	Ama Qamata	South Africa	International TV Shows	TV Show	September 24, 2021	2021	NaN
2	Blood & Water	unknown_director	Ama Qamata	South Africa	TV Dramas	TV Show	September 24, 2021	2021	NaN
3	Blood & Water	unknown_director	Ama Qamata	South Africa	TV Mysteries	TV Show	September 24, 2021	2021	NaN
4	Blood & Water	unknown_director	Khosi Ngema	South Africa	International TV Shows	TV Show	September 24, 2021	2021	NaN
…	…	…	…	…	…	…	…	…	…
201986	Zubaan	Mozez Singh	Anita Shabdish	India	International Movies	Movie	March 2, 2019		
201987	Zubaan	Mozez Singh	Anita Shabdish	India	Music & Musicals	Movie	March 2, 2019		
201988	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	Dramas	Movie	March 2, 2019		
201989	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	International Movies	Movie	March 2, 2019		
201990	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	Music & Musicals	Movie	March 2, 2019		

```
…              …             …
201986         2015          111.0
201987         2015          111.0
201988         2015          111.0
201989         2015          111.0
201990         2015          111.0

[201991 rows x 9 columns]
```

Merging is done to add variables to a dataset, append or add cases or observations to a dataset or remove duplicates and other incorrect information

Dropping Duplicates:

[114]:
```
cleaned_data.duplicated()
```

[114]:
```
0         False
1         False
2         False
3         False
4         False
          …
201986    False
201987    False
201988    False
201989    False
201990    False
Length: 201991, dtype: bool
```

[115]:
```
cleaned_data.loc[cleaned_data.duplicated()]
```

[115]:
```
                    title        director                 cast  \
39336           Rust Creek     Jen McGowan        Micah Hauptman
88516       Blood Will Tell    Miguel Cohan        Oscar Martínez
88517       Blood Will Tell    Miguel Cohan        Oscar Martínez
88518       Blood Will Tell    Miguel Cohan        Oscar Martínez
88519       Blood Will Tell    Miguel Cohan        Oscar Martínez
88520       Blood Will Tell    Miguel Cohan        Oscar Martínez
88521       Blood Will Tell    Miguel Cohan        Oscar Martínez
88522       Blood Will Tell    Miguel Cohan         Dolores Fonzi
88523       Blood Will Tell    Miguel Cohan         Dolores Fonzi
88524       Blood Will Tell    Miguel Cohan         Dolores Fonzi
88525       Blood Will Tell    Miguel Cohan         Dolores Fonzi
88526       Blood Will Tell    Miguel Cohan         Dolores Fonzi
88527       Blood Will Tell    Miguel Cohan         Dolores Fonzi
88528       Blood Will Tell    Miguel Cohan       Diego Velázquez
88529       Blood Will Tell    Miguel Cohan       Diego Velázquez
88530       Blood Will Tell    Miguel Cohan       Diego Velázquez
```

```
88531       Blood Will Tell    Miguel Cohan       Diego Velázquez
88532       Blood Will Tell    Miguel Cohan       Diego Velázquez
88533       Blood Will Tell    Miguel Cohan       Diego Velázquez
88534       Blood Will Tell    Miguel Cohan        Paulina Garcia
88535       Blood Will Tell    Miguel Cohan        Paulina Garcia
88536       Blood Will Tell    Miguel Cohan        Paulina Garcia
88537       Blood Will Tell    Miguel Cohan        Paulina Garcia
88538       Blood Will Tell    Miguel Cohan        Paulina Garcia
88539       Blood Will Tell    Miguel Cohan        Paulina Garcia
88540       Blood Will Tell    Miguel Cohan           Luis Gnecco
88541       Blood Will Tell    Miguel Cohan           Luis Gnecco
88542       Blood Will Tell    Miguel Cohan           Luis Gnecco
88543       Blood Will Tell    Miguel Cohan           Luis Gnecco
88544       Blood Will Tell    Miguel Cohan           Luis Gnecco
88545       Blood Will Tell    Miguel Cohan           Luis Gnecco
88546       Blood Will Tell    Miguel Cohan        Malena Sánchez
88547       Blood Will Tell    Miguel Cohan        Malena Sánchez
88548       Blood Will Tell    Miguel Cohan        Malena Sánchez
88549       Blood Will Tell    Miguel Cohan        Malena Sánchez
88550       Blood Will Tell    Miguel Cohan        Malena Sánchez
88551       Blood Will Tell    Miguel Cohan        Malena Sánchez
88552       Blood Will Tell    Miguel Cohan     Emilio Vodanovich
88553       Blood Will Tell    Miguel Cohan     Emilio Vodanovich
88554       Blood Will Tell    Miguel Cohan     Emilio Vodanovich
88555       Blood Will Tell    Miguel Cohan     Emilio Vodanovich
88556       Blood Will Tell    Miguel Cohan     Emilio Vodanovich
88557       Blood Will Tell    Miguel Cohan     Emilio Vodanovich
88558       Blood Will Tell    Miguel Cohan         Norman Briski
88559       Blood Will Tell    Miguel Cohan         Norman Briski
88560       Blood Will Tell    Miguel Cohan         Norman Briski
88561       Blood Will Tell    Miguel Cohan         Norman Briski
88562       Blood Will Tell    Miguel Cohan         Norman Briski
88563       Blood Will Tell    Miguel Cohan         Norman Briski
135609  300 Miles to Heaven  Maciej Dejczer  Adrianna Biedrzyńska
135610  300 Miles to Heaven  Maciej Dejczer  Adrianna Biedrzyńska
135611  300 Miles to Heaven  Maciej Dejczer  Adrianna Biedrzyńska
135612  300 Miles to Heaven  Maciej Dejczer  Adrianna Biedrzyńska
135613  300 Miles to Heaven  Maciej Dejczer  Adrianna Biedrzyńska
135614  300 Miles to Heaven  Maciej Dejczer  Adrianna Biedrzyńska

             country             listed_in   type         date_added  \
39336   United States             Thrillers  Movie  November 30, 2020
88516       Argentina                Dramas  Movie      June 21, 2019
88517       Argentina    Independent Movies  Movie      June 21, 2019
88518       Argentina  International Movies  Movie      June 21, 2019
88519   United States                Dramas  Movie      June 21, 2019
88520   United States    Independent Movies  Movie      June 21, 2019
```

```
88521   United States  International Movies  Movie      June 21, 2019
88522       Argentina                Dramas  Movie      June 21, 2019
88523       Argentina    Independent Movies  Movie      June 21, 2019
88524       Argentina  International Movies  Movie      June 21, 2019
88525   United States                Dramas  Movie      June 21, 2019
88526   United States    Independent Movies  Movie      June 21, 2019
88527   United States  International Movies  Movie      June 21, 2019
88528       Argentina                Dramas  Movie      June 21, 2019
88529       Argentina    Independent Movies  Movie      June 21, 2019
88530       Argentina  International Movies  Movie      June 21, 2019
88531   United States                Dramas  Movie      June 21, 2019
88532   United States    Independent Movies  Movie      June 21, 2019
88533   United States  International Movies  Movie      June 21, 2019
88534       Argentina                Dramas  Movie      June 21, 2019
88535       Argentina    Independent Movies  Movie      June 21, 2019
88536       Argentina  International Movies  Movie      June 21, 2019
88537   United States                Dramas  Movie      June 21, 2019
88538   United States    Independent Movies  Movie      June 21, 2019
88539   United States  International Movies  Movie      June 21, 2019
88540       Argentina                Dramas  Movie      June 21, 2019
88541       Argentina    Independent Movies  Movie      June 21, 2019
88542       Argentina  International Movies  Movie      June 21, 2019
88543   United States                Dramas  Movie      June 21, 2019
88544   United States    Independent Movies  Movie      June 21, 2019
88545   United States  International Movies  Movie      June 21, 2019
88546       Argentina                Dramas  Movie      June 21, 2019
88547       Argentina    Independent Movies  Movie      June 21, 2019
88548       Argentina  International Movies  Movie      June 21, 2019
88549   United States                Dramas  Movie      June 21, 2019
88550   United States    Independent Movies  Movie      June 21, 2019
88551   United States  International Movies  Movie      June 21, 2019
88552       Argentina                Dramas  Movie      June 21, 2019
88553       Argentina    Independent Movies  Movie      June 21, 2019
88554       Argentina  International Movies  Movie      June 21, 2019
88555   United States                Dramas  Movie      June 21, 2019
88556   United States    Independent Movies  Movie      June 21, 2019
88557   United States  International Movies  Movie      June 21, 2019
88558       Argentina                Dramas  Movie      June 21, 2019
88559       Argentina    Independent Movies  Movie      June 21, 2019
88560       Argentina  International Movies  Movie      June 21, 2019
88561   United States                Dramas  Movie      June 21, 2019
88562   United States    Independent Movies  Movie      June 21, 2019
88563   United States  International Movies  Movie      June 21, 2019
135609        Denmark                Dramas  Movie    October 1, 2019
135610        Denmark  International Movies  Movie    October 1, 2019
135611         France                Dramas  Movie    October 1, 2019
135612         France  International Movies  Movie    October 1, 2019
```

```
135613         Poland                Dramas  Movie    October 1, 2019
135614         Poland  International Movies  Movie    October 1, 2019

       release_year  Movie_Minutes
39336          2018          108.0
88516          2019          113.0
88517          2019          113.0
88518          2019          113.0
88519          2019          113.0
88520          2019          113.0
88521          2019          113.0
88522          2019          113.0
88523          2019          113.0
88524          2019          113.0
88525          2019          113.0
88526          2019          113.0
88527          2019          113.0
88528          2019          113.0
88529          2019          113.0
88530          2019          113.0
88531          2019          113.0
88532          2019          113.0
88533          2019          113.0
88534          2019          113.0
88535          2019          113.0
88536          2019          113.0
88537          2019          113.0
88538          2019          113.0
88539          2019          113.0
88540          2019          113.0
88541          2019          113.0
88542          2019          113.0
88543          2019          113.0
88544          2019          113.0
88545          2019          113.0
88546          2019          113.0
88547          2019          113.0
88548          2019          113.0
88549          2019          113.0
88550          2019          113.0
88551          2019          113.0
88552          2019          113.0
88553          2019          113.0
88554          2019          113.0
88555          2019          113.0
88556          2019          113.0
88557          2019          113.0
```

```
88558          2019          113.0
88559          2019          113.0
88560          2019          113.0
88561          2019          113.0
88562          2019          113.0
88563          2019          113.0
135609         1989           93.0
135610         1989           93.0
135611         1989           93.0
135612         1989           93.0
135613         1989           93.0
135614         1989           93.0
```

[116]:
```
cleaned_data.drop_duplicates(inplace=True)
```

[117]:
```
cleaned_data
```

[117]:
```
                     title          director                  cast  \
0       Dick Johnson Is Dead   Kirsten Johnson          unknown_actor
1              Blood & Water  unknown_director             Ama Qamata
2              Blood & Water  unknown_director             Ama Qamata
3              Blood & Water  unknown_director             Ama Qamata
4              Blood & Water  unknown_director            Khosi Ngema
…                        …               …                       …
201986               Zubaan       Mozez Singh         Anita Shabdish
201987               Zubaan       Mozez Singh         Anita Shabdish
201988               Zubaan       Mozez Singh  Chittaranjan Tripathy
201989               Zubaan       Mozez Singh  Chittaranjan Tripathy
201990               Zubaan       Mozez Singh  Chittaranjan Tripathy

              country                listed_in     type          date_added  \
0       United States           Documentaries    Movie  September 25, 2021
1        South Africa  International TV Shows  TV Show  September 24, 2021
2        South Africa              TV Dramas  TV Show  September 24, 2021
3        South Africa           TV Mysteries  TV Show  September 24, 2021
4        South Africa  International TV Shows  TV Show  September 24, 2021
…                 …                      …       …                     …
201986          India    International Movies    Movie       March 2, 2019
201987          India        Music & Musicals    Movie       March 2, 2019
201988          India                 Dramas    Movie       March 2, 2019
201989          India    International Movies    Movie       March 2, 2019
201990          India        Music & Musicals    Movie       March 2, 2019

        release_year  Movie_Minutes
0               2020           90.0
1               2021            NaN
2               2021            NaN
```

```
3               2021            NaN
4               2021            NaN
…               …               …
201986          2015          111.0
201987          2015          111.0
201988          2015          111.0
201989          2015          111.0
201990          2015          111.0

[201936 rows x 9 columns]
```

[118]:
```
cleaned_data['title'].nunique()
```

[118]:
```
8807
```

[119]:
```
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     201778 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 permanently in the cleaned_data.

Analysis and Recommendations:

Non-Graphical Analysis:

[120]:
```
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()
```

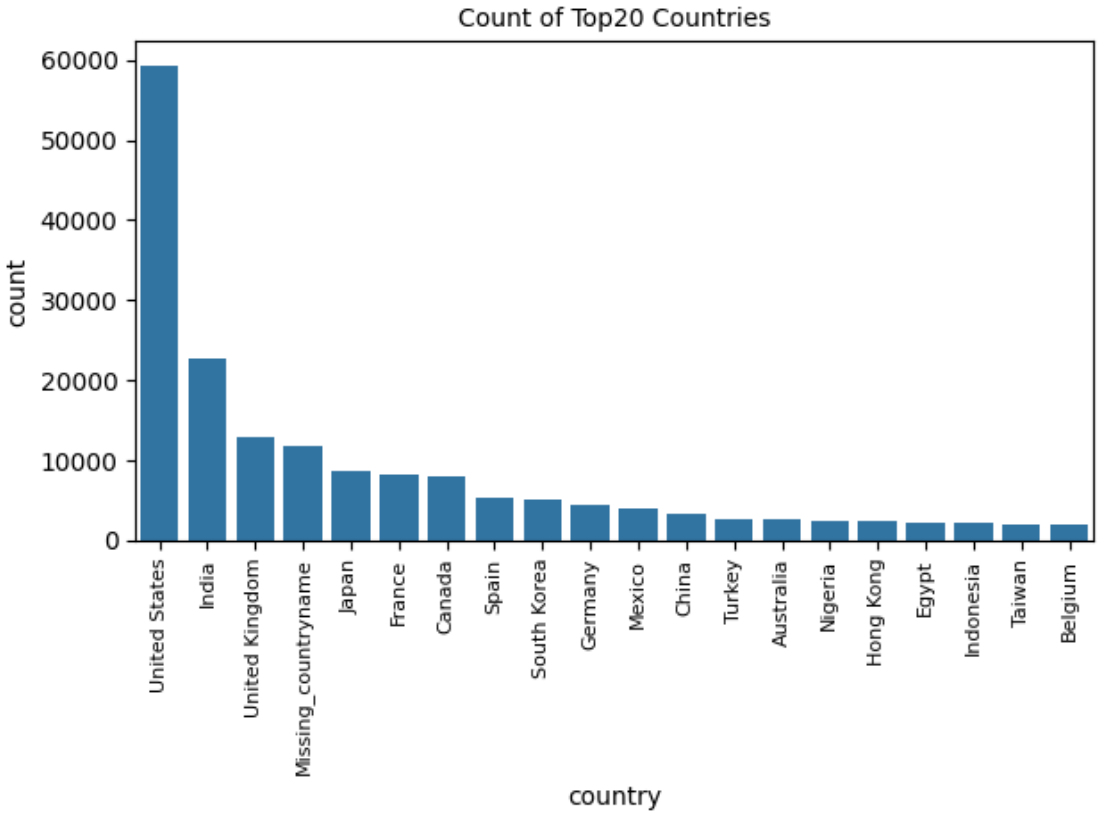
```
print(listed_in_counts)

type_counts = cleaned_data['type'].value_counts()
print(type_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
type
Movie      145788
TV Show     56148
Name: count, dtype: int64
```

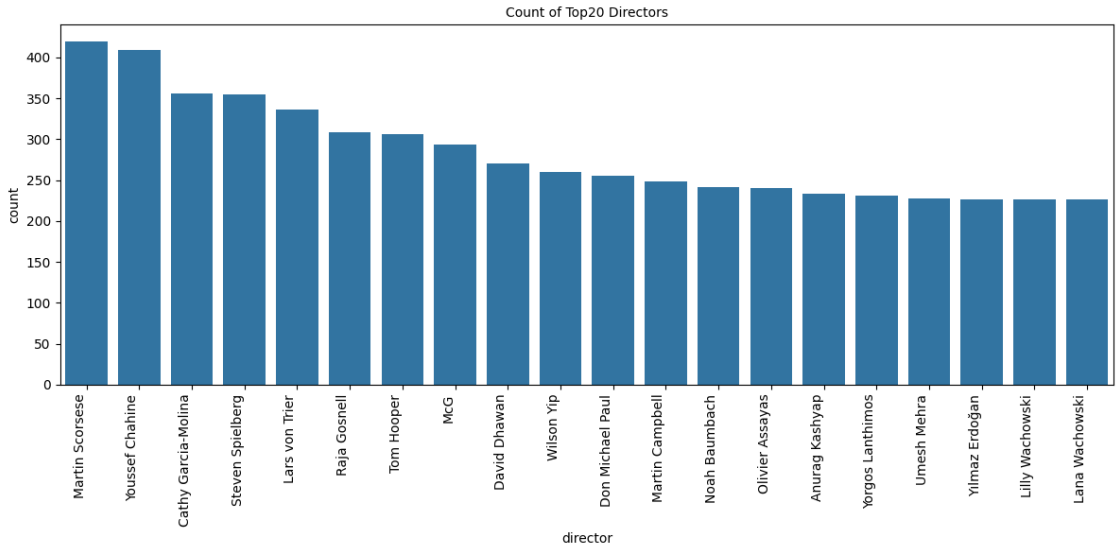
Graphical analysis (Univariate Analysis with bar plot):

[121]:
```
%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=8)
plt.title('Count of Top20 Countries',fontsize=10)
plt.tight_layout()
plt.show()
```

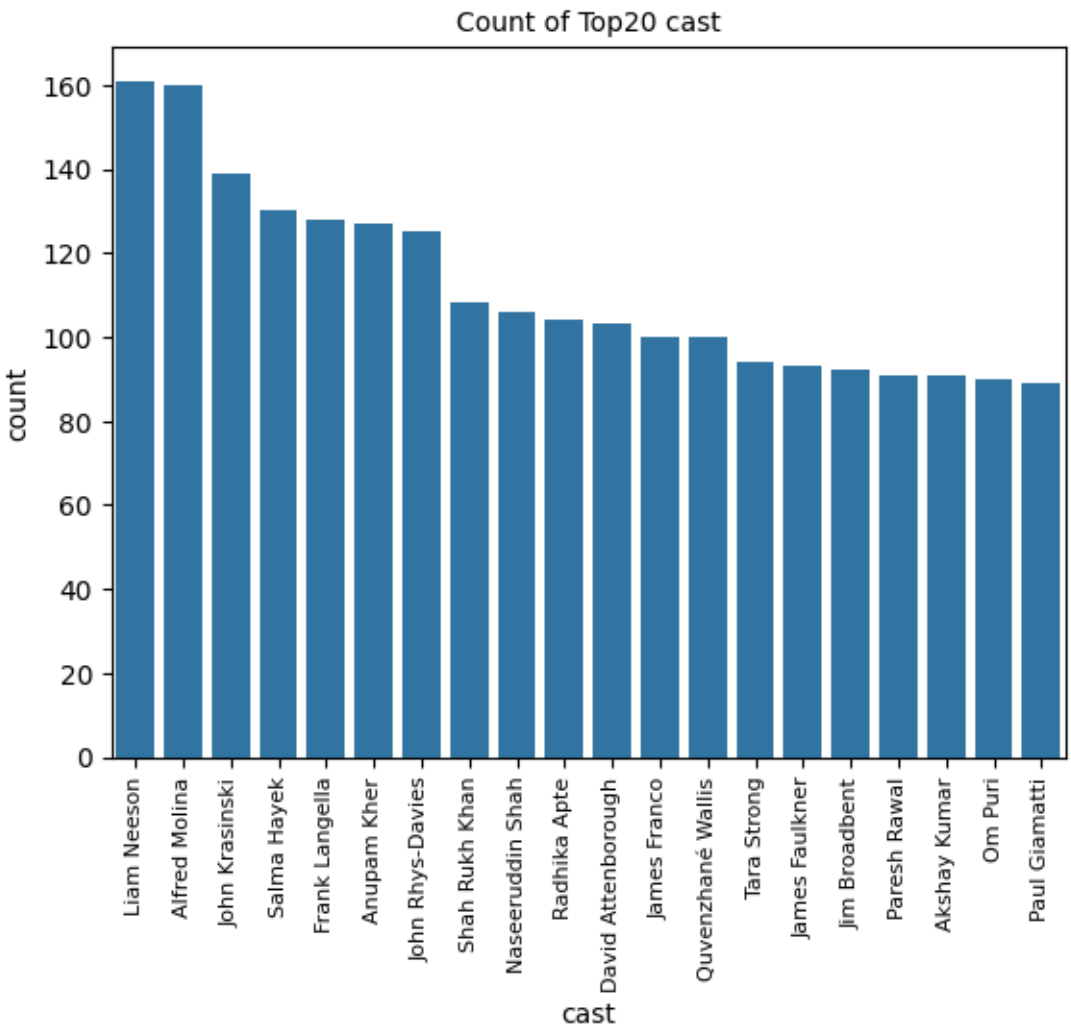


[122]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

# 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)
```

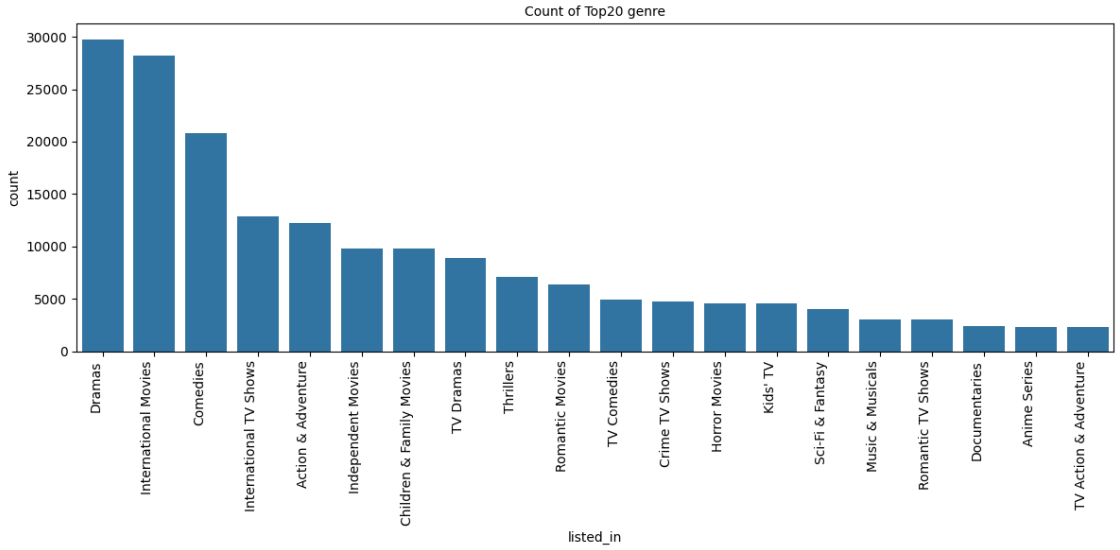
```
plt.xticks(rotation=90, ha='right')
plt.title('Count of Top20 Directors',fontsize=10)
plt.tight_layout()
plt.show()
```



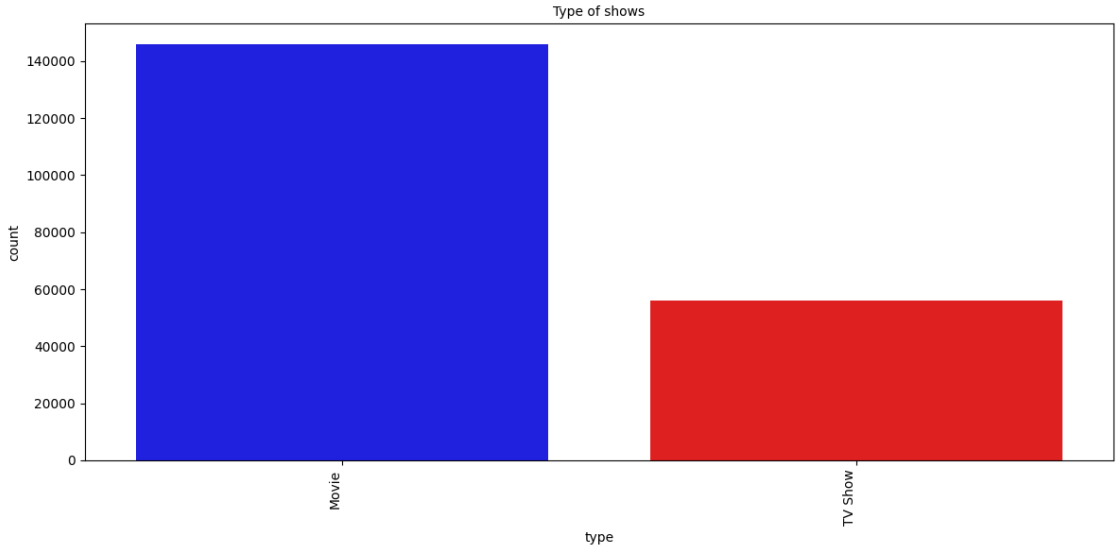
[123]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import re

# #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, width = 0.8)
plt.title('Count of Top20 cast',fontsize=10)
plt.xticks(rotation=90, fontsize=8)
plt.show()
```



[124]:
```
%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('Count of Top20 genre',fontsize=10)
plt.tight_layout()
plt.show()
```



[125]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

filtered_data = cleaned_data[cleaned_data['type'] != 'unknown_type']
top_type = filtered_data['type'].value_counts().nlargest(20).index

plt.figure(figsize=(12, 6))
sns.countplot(x='type', data=cleaned_data, order=top_type, hue='type',␣
 ↪palette={'Movie': 'blue', 'TV Show': 'red'})
plt.xticks(rotation=90, ha='right')
plt.title('Type of shows', fontsize=10)
plt.tight_layout()
plt.show()
```



Analysis from the counts of each categorical variable in both graphical and non-graphical format:

Below are the count of each category of the cleaned dataset,

United States have the more counts of TV shows and Movies

Martin Scorsese and **Youssef Chahine** are the top most directors with

more counts

Liam Neeson and **Affred Molina** are top actors with more count of movies and tv shows

Dramas and **International movies** are the most listed genres

Count of movies is greater than count of tv shows.

It is recommended that based on these counts Netflix can get an idea of how to improve the same logic in other countries too.

Comparison Analysis:

[126]:
```
groupby_movies=cleaned_data[cleaned_data['type']=='Movie'].
  ↪groupby('country')['title'].count()
groupby_movies.sort_values(ascending=False).head(10)
```

[126]:
```
country
United States          45791
India                  21411
United Kingdom          8560
France                  6605
Missing_countryname     6199
Canada                  5738
```

```
Japan                    3525
Spain                    3469
Germany                  3427
China                    2377
Name: title, dtype: int64
```

[127]:
```
groupby_tvShows=cleaned_data[cleaned_data['type']=='TV Show'].
  ↪groupby('country')['title'].count()
groupby_tvShows.sort_values(ascending=False).head(10)
```

[127]:
```
country
United States          13533
Missing_countryname     5698
Japan                   5154
United Kingdom          4385
South Korea             3754
Canada                  2177
Mexico                  2018
Spain                   1846
Taiwan                  1719
France                  1647
Name: title, dtype: int64
```

Analysis of comparison between tv shows and movies among each country:

US ranks1st in TV shows and Movies production in Netflix.

Hence Netflix can focus more on countries that have large production numbers like **US,India, United Kingdom, Japan, Canada, South Korea** to release tv shows or movies.

Analysis between Cast and shows:

[128]:
```
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
```

[129]:
```
filtered_cast_data = cleaned_data[cleaned_data['cast'] != 'unknown_actor']
groupby_cast_movie=filtered_cast_data[filtered_cast_data['type']=='Movie'].
  ↪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
```

Analysis on Cast with shows:

David Attenborough acted in most number of TVshows.

Liam Neeson acted in most number of Movies.

Netflix should focus more on releasing tvshows/Movies casted by the above actors to attract more subscribers

Analysis based on Directors:

[130]:
```
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)
```

[130]:
```
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
```

[131]:
```
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)
```

[131]:
```
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
```

Analysis on Directors with shows:

Noam Murro have directed more TV shows.

Martin Scorsese have directed more Movies

Netflix should focus more on releasing tvshows/Movies directed by the above actors to attract more subscribers to Neflix

Analysis on listed_in (genre):

[132]:
```
India_data = cleaned_data[cleaned_data['country'] == 'India']

# Group by listed_in and count
most_watched = India_data['listed_in'].value_counts().
 ↪sort_values(ascending=False)

# Printing the results
print(most_watched)
```

```
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
TV Dramas                        272
Children & Family Movies         225
TV Shows                         207
TV Comedies                      141
Sports Movies                    121
Sci-Fi & Fantasy                 111
Classic Movies                    98
Romantic TV Shows                 68
Crime TV Shows                    61
Kids' TV                          57
TV Action & Adventure             44
Cult Movies                       42
LGBTQ Movies                      33
Documentaries                     32
TV Horror                         28
TV Sci-Fi & Fantasy               27
Faith & Spirituality              20
British TV Shows                  19
Docuseries                        15
TV Mysteries                      11
Stand-Up Comedy & Talk Shows       8
Reality TV                         7
Stand-Up Comedy                    7
Teen TV Shows                      7
TV Thrillers                       3
Name: count, dtype: int64
```

Analysis on most watched genre in India:

International movies,Dramas,Comedies are the most watched genres in India. Hence Netflix can focus more on adding such genres in India.

Analysis on Duration based on minutes:

[133]:
```
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
```

Analysis of Avg duration of horror movies:

My friend wants to know the average duration of horror movies. So I made an analysis above and

the average duration of horror movies is **99 minutes**.

Analysis on Duration based on seasons:

[134]:
```
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']}")
print(f"Watch count: {top_show_by_seasons['watch_count']}")
```

```
The TV show with the most seasons is: Grey's Anatomy
Number of seasons: 17.0
Watch count: 30
```

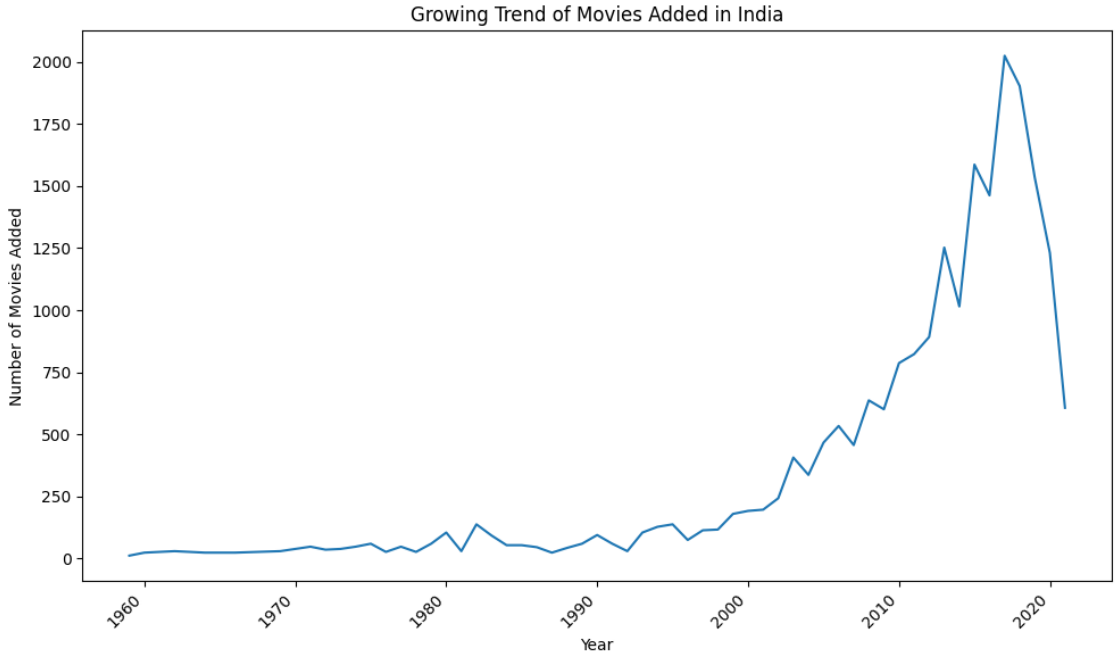
Analysis on TV shows with most seasons:

Grey's Anatomy is the tv show with most seasons with 17 seasons which can be preferred for binge watching

Graphical Analysis of growing trend:

[135]:
```
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()
```



Analysis on Growing Trend of Movies Added in India:

Since the year 2000, the growing trend of movies has been increased in India with upto 2000 movies

If Netflix focus on doing the same by adding more movies in other countries also same like India, subscribers will increase and Netflix can also see more profit on their side.

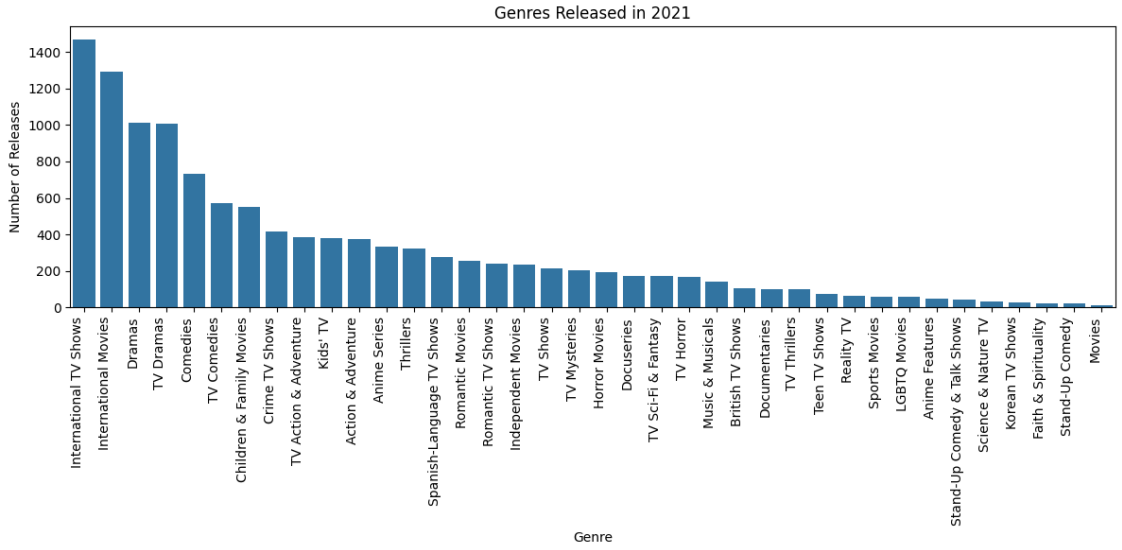
Graphical & Non-Graphical Analysis on recently added genre based on date_added:

[136]:
```
# 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 year added is: {most_recent_year}")
```

```
The most recent year added is: 2021.0
```

[137]:
```
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()
```



Analysis of Recently added Genre:

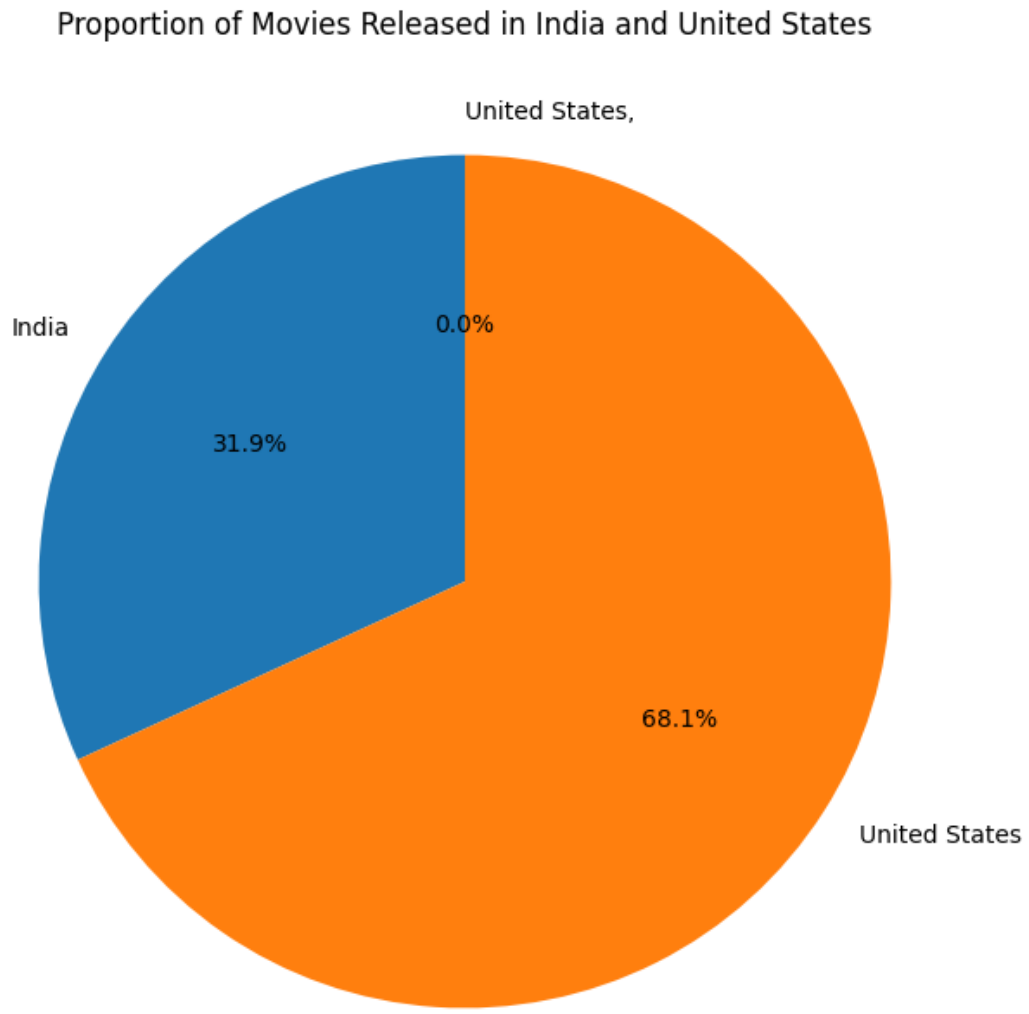
2021 is the recent year where genres are added.

In analysis, **International TV Shows** tops the list of recently added genre.

This analysis will help people who are looking out for recently added tv shows/movies in 2021

Graphical Analysis made on basis of Proportion:

[138]:
```
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=(8, 8))  # 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()
```



Analysis of proportion between US movie count and India movie count:

Even though India ranked 2nd in the count of movies released still it has much difference with the United States which ranked top.

Netflix can focus in increasing more movies in India.

Analysis based on Cast:

[139]:
```
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]
```

Analysis of actor with most released movies in India:

A.K. Hangal has the highest released movie count in India with 12 movie counts.

Indians are therefore having various recommendations for A.K.Hangal movies

Analysis based on year added of movies:

[140]:
```
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
```

Analysis on year with most movies added in Netflix:

Most movies were added in Netflix in the year **2019** with the count of **34,446 movies**.

Here subscribers have plenty of options to explore movies released in 2019.

Analysis based on cast and listed_in(genre):

[141]:
```
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]
```

Analysis on actors who acted in multiple genre:

Ron Perlman has acted in multiple genre with count of 17 followed by Kiernan Shipka, Gary Cole, Glenn close, Samuel L.Jackson

Analysis based on month:

[142]:
```
import pandas as pd
import calendar

cleaned_data['date_added'] = pd.to_datetime(cleaned_data['date_added'],␣
 ↪errors='coerce')
cleaned_data['release_month'] = cleaned_data['date_added'].dt.month

tv_shows = cleaned_data[cleaned_data['type'] == 'TV Show']
monthly_releases = tv_shows.groupby('release_month')['title'].count()

most_released_month_number = monthly_releases.idxmax()
```

```
most_released_month_name = calendar.month_name[int(most_released_month_number)]␣
 ↪ # Convert to int

print(f"The most released month for TV shows is: {most_released_month_name}")
```

```
The most released month for TV shows is: December
```

Analysis of most released month of a tv show:

This analysis shows that the maximum number of tv shows are released in the month of December.

Hence it recommends viewers to look into particular month if they need various options.

[143]:
```
import pandas as pd
import calendar

cleaned_data['date_added'] = pd.to_datetime(cleaned_data['date_added'],␣
 ↪errors='coerce')
cleaned_data['release_month'] = cleaned_data['date_added'].dt.month

movies = cleaned_data[cleaned_data['type'] == 'Movie']
monthly_releases = movies.groupby('release_month')['title'].count()

most_released_month_number = monthly_releases.idxmax()
most_released_month_name = calendar.month_name[int(most_released_month_number)]␣
 ↪ # Convert to int

print(f"The most released month for movies is: {most_released_month_name}")
```

```
The most released month for movies is: July
```

Analysis of most released month of a movie:

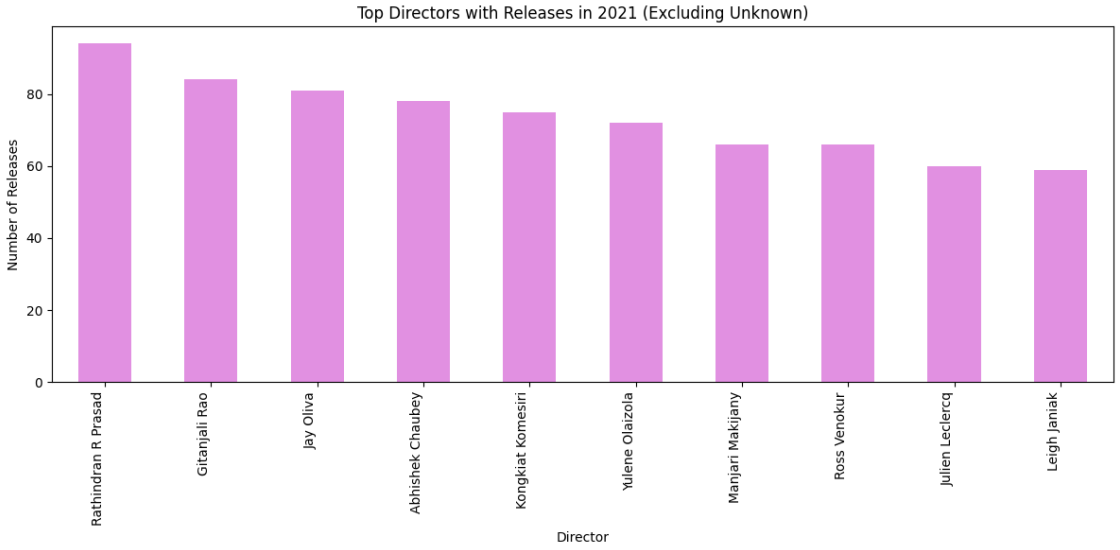
This analysis shows that the maximum number of movies are released in the month of July.

Hence it recommends viewers to look into particular month if they need various options in movies.

Graphical (Bivariate Cat-Num) Analysis based on Directors and release_year:

[144]:
```
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=(12,6))
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()
```



Analysis of Directors with recent released year:

It is found in analysis that Director Rathindran R Prasad has more movies released in 2021 which is the recent year.

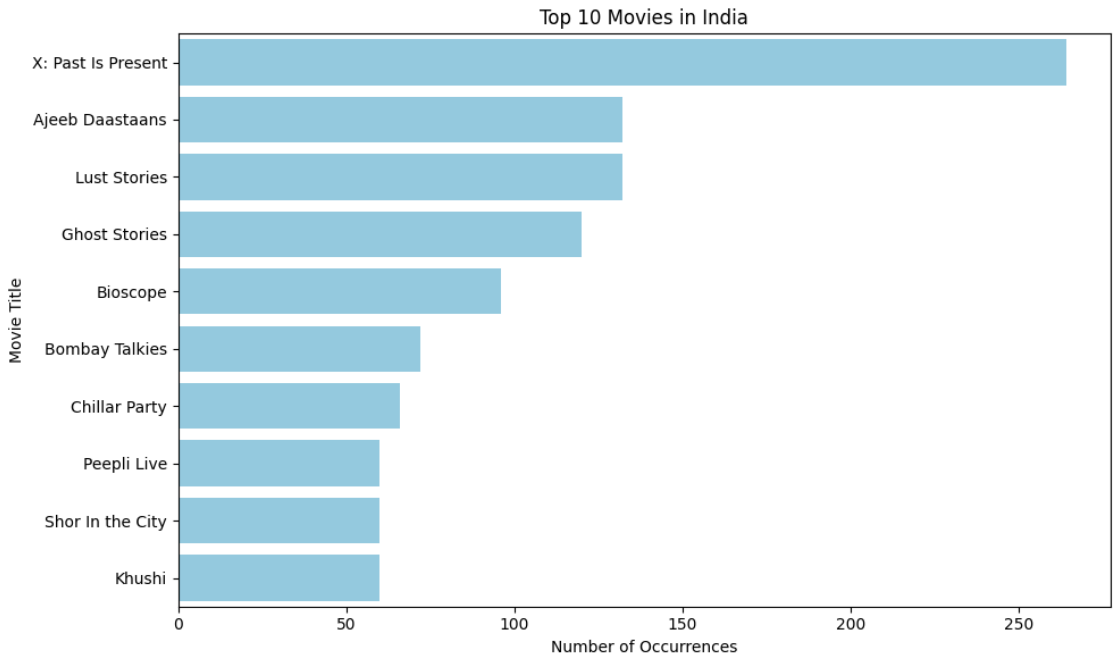
Other directors are also there with only minimum difference.

This analysis will help the audience and Netflix to focus more on those directors

Analysis based on Movies in India:

[145]:
```
import matplotlib.pyplot as plt
import seaborn as sns


india_movies = cleaned_data[(cleaned_data['country'] == 'India') &␣
 ↪(cleaned_data['type'] == 'Movie')]

top_10_movies = india_movies['title'].value_counts().head(10)
```

```
plt.figure(figsize=(10, 6))
sns.barplot(y=top_10_movies.index, x=top_10_movies.values, orient='h',␣
↪color='skyblue')
plt.title('Top 10 Movies in India')
plt.xlabel('Number of Occurrences')
plt.ylabel('Movie Title')
plt.tight_layout()
plt.show()
```



Analysis of top 10 movies in India:

Me and my friend wants to know which movie is in top with most number of occurences.

This analysis will give me the top 10 movies.

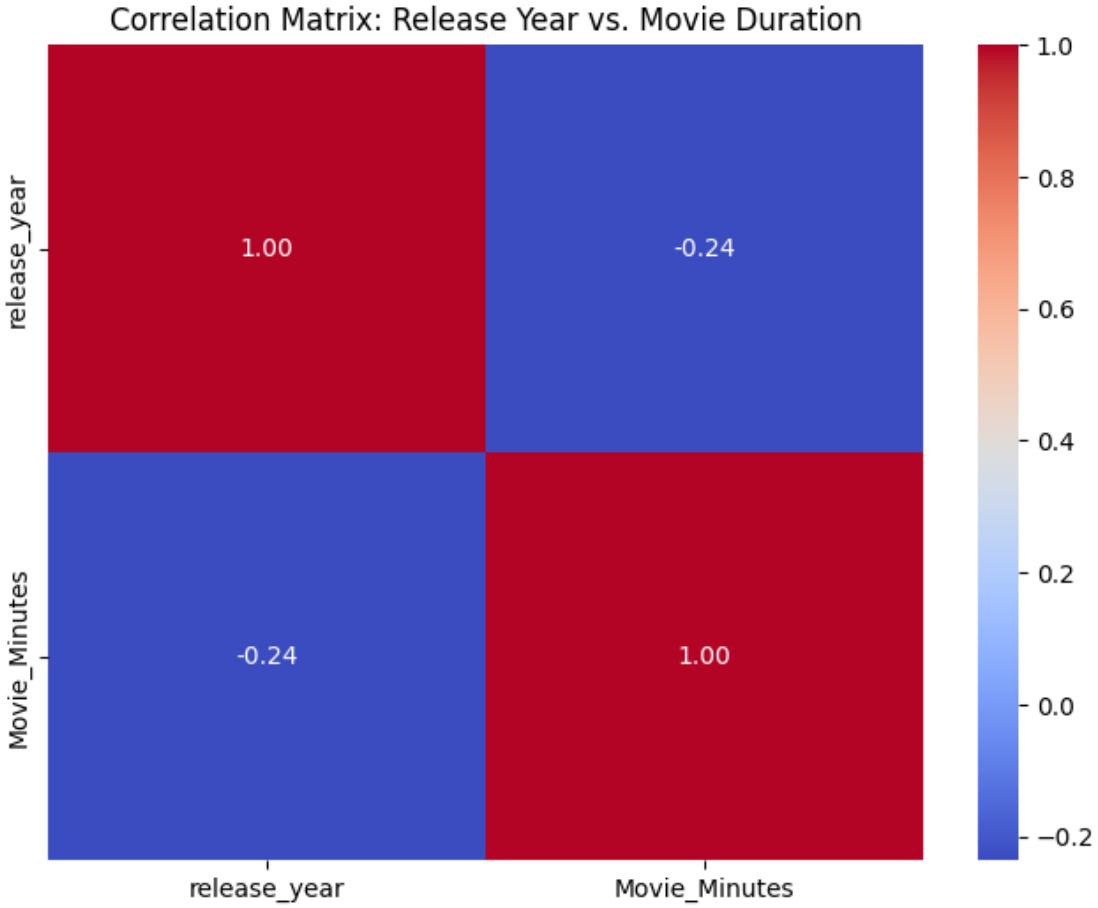
The movie **X: Past is Present** is the movie with most number of occurences followed by Ajeeb Daastaans, Lust stories, Ghost stories etc

This analysis will be helpful when people wants to watch movie with most number of occurences.

Corelation Analysis using heat map:

[149]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


movies_data = cleaned_data[cleaned_data['type'] == 'Movie']
```

```
correlation_data = movies_data[['release_year', 'Movie_Minutes']]

correlation_matrix = correlation_data.corr()

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix: Release Year vs. Movie Duration')
plt.show()
```



Analysis between movie release year and duration:

The heatmap generated by the code provides a visual representation of the correlation between movie release year and duration.

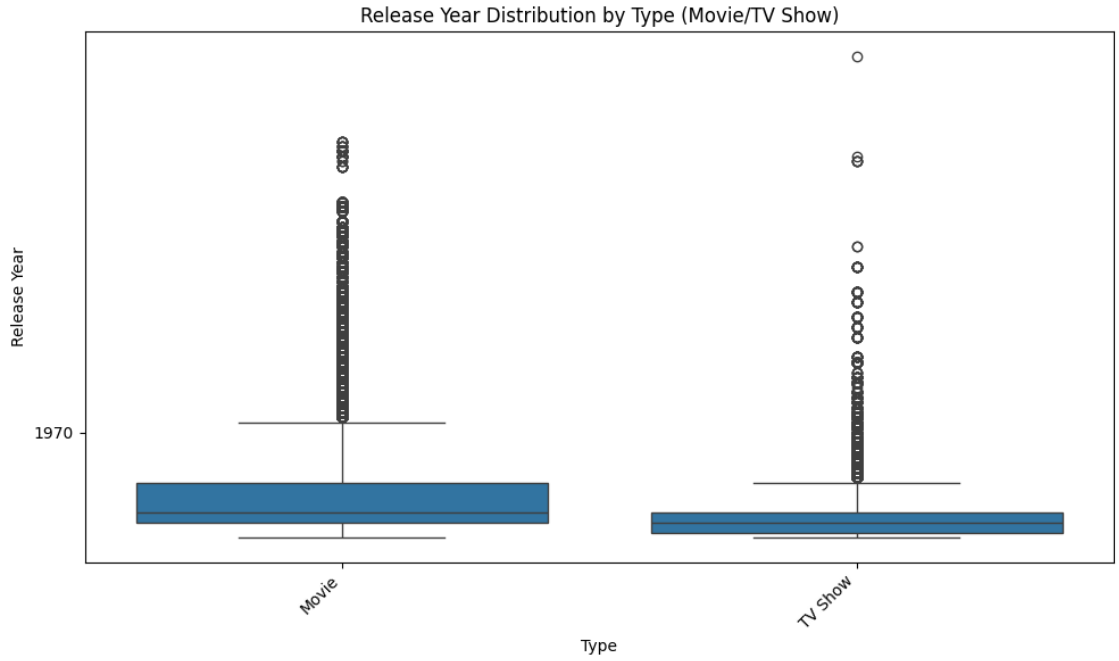
By analyzing the color intensity and the annotation value, we can gain insights into the strength and direction of this relationship, helping us understand potential trends in movie durations over time.

However, it's crucial to remember that correlation does not equal causation, and the interpretation

should be made cautiously considering the context of the data and research question.

Analysis of Categorical variables using Box plot:

[153]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.boxplot(x='type', y='release_year', data=cleaned_data)
plt.title('Release Year Distribution by Type (Movie/TV Show)')
plt.xlabel('Type')
plt.ylabel('Release Year')
plt.xticks(rotation=45, ha='right')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



Analysis of relationship between type (Movie or TV Show) and release_year using a box plot:

Movies on Netflix have a wider range of release years, including older titles.

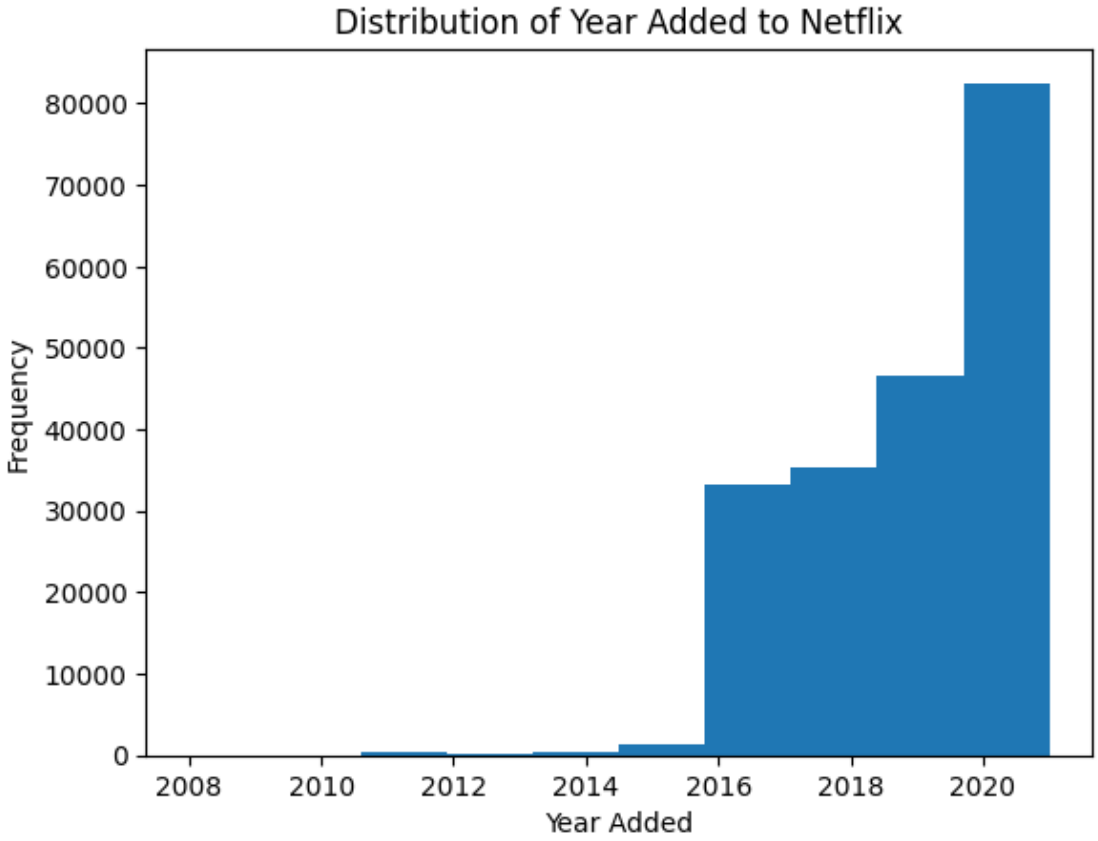
TV shows tend to have more recent releases compared to movies.

Outliers in release years could indicate unusual content or data anomalies.

Netflix could focus on acquiring more recent TV show releases, diversify content by including a mix of new and classic titles and investigate outliers for insights into content decisions or data quality

Graphical Analysis based on Histogram:

[157]:
```
import matplotlib.pyplot as plt

plt.hist(cleaned_data['year_added'], bins=10)
plt.xlabel('Year Added')
plt.ylabel('Frequency')
plt.title('Distribution of Year Added to Netflix')
plt.show()
```



Analysis of frequency and year added using Histogram:

Netflix has seen a significant increase in content additions over recent years, with a peak around 2019.

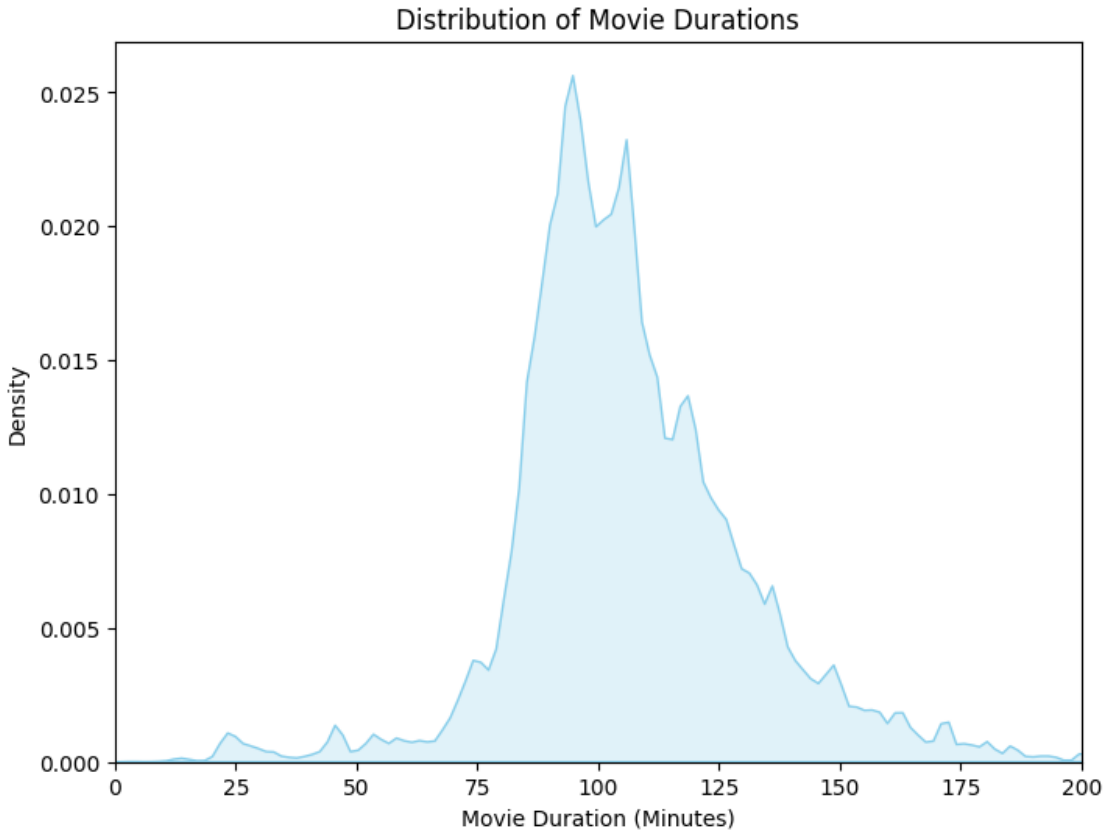
Recommendation: Leverage this trend by focusing on acquiring and promoting newer content.

Analyze audience preferences within specific years to further tailor content strategies and recommendations for user engagement.

Consider expanding content libraries with a balance of both recent and classic titles to cater to diverse viewer interests.

KDE analysis of Movie minutes:

[159]:
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
sns.kdeplot(data=cleaned_data, x='Movie_Minutes', fill=True, color='skyblue',␣
 ↪bw_adjust=0.5)
plt.title('Distribution of Movie Durations')
plt.xlabel('Movie Duration (Minutes)')
plt.ylabel('Density')
plt.xlim(0, 200)
plt.show()
```



Analysis of Movie minutes using KDE plot:

Most movies on Netflix have durations clustered around 90-100 minutes, with a gradual decrease in density for longer films.

Recommendation: Focus on acquiring movies within the popular duration range to cater to viewer preferences. Consider offering more diverse content with shorter or longer durations to expand audience reach. Analyze genre-specific duration preferences to further optimize content acquisition.

Promote movie duration as a search/filter option to enhance user experience.

Overall Insights:

Here's a detailed overview of the analysis I've performed on the Netflix dataset, broken down into steps:

1. Data Cleaning and Preparation: I started by importing the necessary libraries like pandas and downloaded the Netflix dataset using gdown and then performed data cleaning steps such as:

a) Handling missing values by replacing them with appropriate placeholders (e.g., 'unknown_director', 'unknown_actor').

b) Splitting comma-separated values in columns like 'director', 'cast', 'country', and 'listed_in' to create separate rows for each item.

c) Extracting numerical duration from the 'duration' column and creating separate columns for duration value and type (minutes or seasons).

d) Merging the cleaned dataframes into a single 'cleaned_data' dataframe.

e) Dropping duplicate rows.

2. Univariate Analysis (Counts and Distributions)

I've analyzed the frequency distributions of categorical variables such as 'director', 'cast', 'country', 'listed_in' and 'type' using both:

Non-graphical methods: Calculating and printing value counts.

Graphical methods: Creating bar plots to visualize the distributions, focusing on the top categories.

3. Bivariate Analysis (Relationships and Comparisons)

I've explored relationships between variables, including:

a) Comparing movie and TV show production by country.

b) Identifying top actors and directors for both movies and TV shows.

c) Analyzing the most watched genres in India.

d) Determining the average duration of horror movies.

e) Finding the TV show with the most seasons.

f) Examining the growing trend of movie releases in India over time using a line plot.

g) Analyzing the most recently added genres.

h) Comparing the proportion of movies released in India and the United States using a pie chart.

i) Identifying the year with the most movies added to Netflix.

j) Investigating actors who have acted in multiple genres.

k) Determining the months with the most releases for both TV shows and movies.

l) Identifying the correlation between movie release year and duration.

m) Finding relationship between type (Movie or TV Show) and release year.

n) Analyzing frequency and year added by Netflix.

o) Examining the length of movie minutes.

4. Key Insights and Recommendations

I've derived several insights from my analysis, including observations about top countries, directors, actors, genres and release trends.

I've provided recommendations for Netflix based on these insights, such as focusing on specific countries, genres or actors to attract more subscribers and increase viewership.

5. Visualization and Reporting

I effectively used visualizations (bar plots, line plots, pie charts) to present my findings in a clear and understandable manner.

I documented my analysis with Markdown cells, explaining the steps, insights and recommendations.

Overall, I've conducted a comprehensive data analysis of the Netflix dataset, starting from data cleaning to deriving valuable insights and providing actionable recommendations.

I've also effectively used a combination of techniques and visualizations to support my findings.

About Outlier treatment:

The Netflix analysis focused on initial exploration and visualization, where outlier treatment wasn't the main priority.

Extreme values might be valid in this context or implicitly handled by the chosen methods.

Outlier treatment is often more crucial for predictive modeling, which might be a later step.

The large dataset size could also reduce the impact of outliers on overall insights.