BIG DATA MANAGEMENT

PROJECT 2

Vasilis Konstantakos

AID: 2022202004009

Grigoris Bouziotopoulos

AID: 2022202004016

Professor: C. Trifonopoulos

System Requirements

The project was developed on Linux and especially on Pop!_os 21.04 distribution (based on ubuntu) using x86_64 architecture [1], which is supported by MongoDB [2]. The following tools were used:

1. **Pycharm Professional** for Development (Free Student version) [3]
2. **Datagrip** for Database interaction (Free Student Version) [4]
3. **mongoimport** for importing the data into the database [5]
4. **Python** *version 3.9.5* for converting csv to json [6]
5. Google's **Collaboratory** for creating visualizations [7]

Deliverable Structure

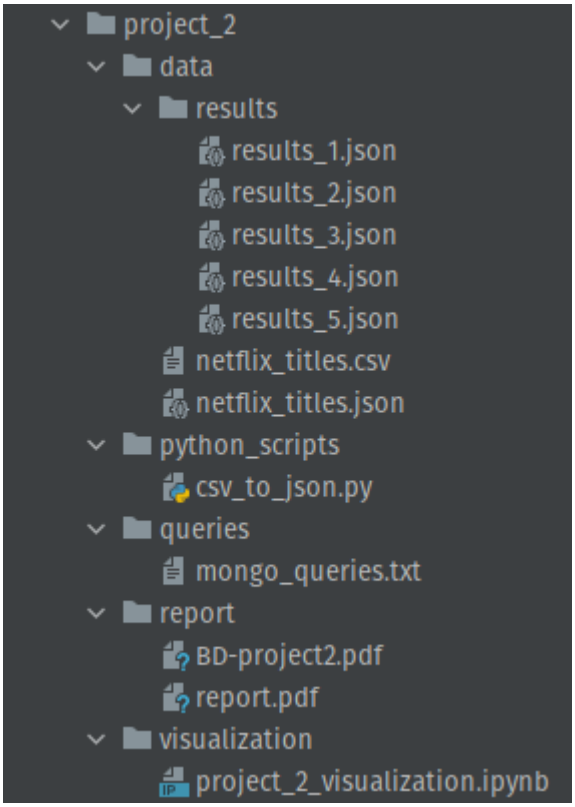

Figure 1: Project structure

The structure is outlined below:

1. **data directory –** contains all the data files, including the given *netflix_titles.csv,* its converted version to json, *netflix_titles.json* and the subdirectory results.
2. **results –** contains the result files from the 5 mongo queries
3. **python scripts –** contains the script that converts the *netflix_titles.csv* to json
4. **queries –** contains a text file containing the mongo db queries
5. **report –** contains the report files
6. **visualization –** contains the jupyter notebook used for data visualization

Data Storage in MongoDB

In order to install and start the MongoDB 4.4 the official documentation tutorial was used [8]. For converting the csv data to json, a simple python script was used. It parses the file *netflix_titles.csv* inside **project_2/python_scripts** and generates *netflix_titles.json* file.

It can be executed with the following command from inside **project_2/python_scripts** directory (important to be executed from inside this directory as it has hardcoded file paths and will not work otherwise):

python3 csv_to_json.py

In order to import the data, **mongoimport** was used via the following command, executed from the project directory:

mongoimport --db netflix_titles --collection shows --drop --file data/netflix_titles.json --jsonArray

for more information about the arguments used, please refer to the documentation of mongoimport [5]. The above command is demonstrated in Figure 2.

```
grigorisbouziotopoulos@pop-os ~/P/b/project_2 (main)>
mongoimport --db netflix_titles --collection shows --drop --file data/net
flix_titles.json --jsonArray
2021-07-11T17:56:47.774+0300    connected to: mongodb://localhost/
2021-07-11T17:56:47.774+0300    dropping: netflix_titles.shows
2021-07-11T17:56:47.976+0300    6234 document(s) imported successfully. 0
 document(s) failed to import.
grigorisbouziotopoulos@pop-os ~/P/b/project_2 (main)>
```

Figure 2: mongoimport Execution

An image of the data is presented in Figure 3.

	{} _id	{} cast	{} country	{} date_added	{} description
1	60eb06afde15cdc90026aa66	["Alan Marriott", " Andrew Toth", " Brian Dobson", " Cole Howard", " Jennifer Came	["United States", " India", " South Korea", " China"]	2019-09-09	Before planning an awesome
2	60eb06afde15cdc90026aa67	["Jandino Asporaat"]	["United Kingdom"]	2016-09-09	Jandino Asporaat riffs on
3	60eb06afde15cdc90026aa68	["Peter Cullen", " Sumalee Montano", " Frank Welker", " Jeffrey Combs", " Kevin Mi	["United States"]	2018-09-08	With the help of three huma
4	60eb06afde15cdc90026aa69	["Will Friedle", " Darren Criss", " Constance Zimmer", " Khary Payton", " Mitchell	["United States"]	2018-09-08	When a prison ship crash u
5	60eb06afde15cdc90026aa6a	["Nesta Cooper", " Kate Walsh", " John Michael Higgins", " Keith Powers", " Alicia	["United States"]	2017-09-08	When nerdy high schooler D
6	60eb06afde15cdc90026aa6b	["Alberto Ammann", " Eloy Azorín", " Verónica Echegui", " Lucía Jiménez", " Claudi	["Spain"]	2017-09-08	A young journalist is forc
7	60eb06afde15cdc90026aa6c	["Fabrizio Copano"]	["Chile"]	2017-09-08	Fabrizio Copano takes audi
8	60eb06afde15cdc90026aa6d	<unset>	["United States"]	2017-09-08	As California's 2016 fire
9	60eb06afde15cdc90026aa6e	["Joaquín Reyes"]	<unset>	2017-09-08	Comedian and celebrity imp
10	60eb06afde15cdc90026aa6f	["James Franco", " Kate Hudson", " Tom Wilkinson", " Omar Sy", " Sam Spruell", " A	["United States", " United Kingdom", " Denmark", " Sweden"]	2017-09-08	A struggling couple can't
11	60eb06afde15cdc90026aa70	["Jim Sturgess", " Sam Worthington", " Ryan Kwanten", " Anthony Hopkins", " Mark v	["Netherlands", " Belgium", " United Kingdom", " United States"]	2017-09-08	When beer magnate Alfred "
12	60eb06afde15cdc90026aa71	["Damandeep Singh Baggan", " Smita Malhotra", " Baba Sehgal"]	<unset>	2017-09-08	A team of minstrels, inclu
13	60eb06afde15cdc90026aa72	["Antonio Banderas", " Dylan McDermott", " Melanie Griffith", " Birgitte Hjort Sør	["Bulgaria", " United States", " Spain", " Canada"]	2017-09-08	In a dystopian future, an
14	60eb06afde15cdc90026aa73	["Damandeep Singh Baggan", " Smita Malhotra", " Baba Sehgal", " Deepak Chachra"]	<unset>	2017-09-08	An artisan is cheated of h
15	60eb06afde15cdc90026aa74	["Damandeep Singh Baggan", " Smita Malhotra", " Deepak Chachra"]	<unset>	2017-09-08	A cat, monkey and donkey t
16	60eb06afde15cdc90026aa75	["Damandeep Singh Baggan", " Smita Malhotra", " Baba Sehgal"]	<unset>	2017-09-08	Animal minstrels narrate s
17	60eb06afde15cdc90026aa76	["Rishi Gambhir", " Smita Malhotra", " Deepak Chachra"]	<unset>	2017-09-08	A cat, monkey and donkey l
18	60eb06afde15cdc90026aa77	["Damandeep Singh Baggan", " Smita Malhotra", " Baba Sehgal"]	<unset>	2017-09-08	In three comic-strip-style
19	60eb06afde15cdc90026aa78	["Damandeep Singh Baggan", " Smita Malhotra", " Baba Sehgal"]	<unset>	2017-09-08	The consequences of tricke
20	60eb06afde15cdc90026aa79	<unset>	["United States", " Uruguay"]	2017-09-08	As the newspaper industry

Figure 3: inserted data

MongoDB Queries

Since Datagrip [4] was used in order to query the database, mongo shell queries were used.

Content of 2019 query

```
db.shows.find({date_added: {$regex: '^2019'}},
{show_id: 1, type: 1, title: 1, _id: 0});
```

Figure 4: first query

Content of 2019 query first 20 results

	{} show_id	{} title	{} type
1	81145628	Norm of the North: King Sized Adventure	Movie
2	80221550	Archibald's Next Big Thing	TV Show
3	80178151	The Spy	TV Show
4	81154455	Article 15	Movie
5	81176188	American Factory: A Conversation with the Obamas	Movie
6	81113928	Care of Kancharapalem	Movie
7	81132437	Kill Me If You Dare	Movie
8	81052275	Ee Nagaraniki Emaindi	Movie
9	81160036	Saawan	Movie
10	81155784	Watchman	Movie
11	81078908	The World We Make	Movie
12	81054495	Mo Gilligan: Momentum	Movie
13	81173255	The Heretics	Movie
14	81053893	Cultivating the Seas: History and Future of the Full-Cycle Cultured Kindai Tuna	Movie
15	80200087	Domino	Movie
16	81053892	TUNA GIRL	Movie
17	81132443	Deliha 2	Movie
18	80225885	Bard of Blood	TV Show
19	81171121	Sturgill Simpson Presents Sound & Fury	Movie
20	80231903	In the Shadow of the Moon	Movie

Figure 5: first query – 20 results

Some notes on the query:

- it uses a regular expression to match all dates that start with the year 2019

The full results of the query are stored in *project_2/data/results/results_1.json* file.

TV Shows per Country query

```
db.shows.aggregate([
  {$match: { type: 'TV Show' }},
  {$project: { _id: 0, country: 1} },
  {$unwind: "$country" },
  {$group: { _id: { $trim: { input: "$country" } }, country_count: { $sum: 1 } }},
  {$project: { _id: 0, country: "$_id", country_count: 1 } },
  {$sort: { country_count: -1 } }
  ]);
```

Figure 6: second query

TV Shows per Country first 20 results

	{} country	{} country_count
1	United States	686
2	United Kingdom	223
3	Japan	156
4	South Korea	116
5	Canada	107
6	France	70
7	Taiwan	65
8	India	55
9	Australia	50
10	Mexico	45
11	Spain	45
12	China	36
13	Germany	25
14	Turkey	25
15	Colombia	23
16	Thailand	18
17	Brazil	18
18	Italy	15
19	Russia	14
20	Argentina	14

Figure 7: second query – 20 results

Some notes on the query:

- it filters based on the content
- selects only the countries
- converts each country in the json array into a separate record
- groups based on the trimmed country (devoid of trailing and leading whitespaces) and counts the countries for each show
- keeps only the columns of interest
- performs descending sorting based on the count

The full results of the query are stored in *project_2/data/results/results_2.json* file.

Genre query

```
db.shows.aggregate([
  {$project: { _id: 0, listed_in: 1} },
  {$unwind: "$listed_in" },
  {$group: {
        _id: { $trim: { input: "$listed_in" } },
        count: { "$sum": 1 }
    }},
  {$project: { _id: 0, genre: '$_id', count: 1} },
  {$sort: { count: -1 } }
]);
```

Figure 8: third query

Genre first 20 results

	{} count	{} genre
1	1927	International Movies
2	1623	Dramas
3	1113	Comedies
4	1001	International TV Shows
5	668	Documentaries
6	599	TV Dramas
7	597	Action & Adventure
8	552	Independent Movies
9	436	TV Comedies
10	392	Thrillers
11	378	Children & Family Movies
12	376	Romantic Movies
13	363	Crime TV Shows
14	328	Kids' TV
15	281	Stand-Up Comedy
16	279	Docuseries
17	278	Romantic TV Shows
18	262	Horror Movies
19	243	Music & Musicals
20	210	British TV Shows

Figure 9: third query – 20 results

Some notes on the query:

- selects only the listed_in column
- converts each genre in the json array into a separate record
- groups based on the trimmed genre (devoid of trailing and leading whitespaces) and counts the genre for each show
- keeps only the columns of interest
- performs descending sorting based on the count

The full results of the query are stored in *project_2/data/results/results_3.json* file.

Most frequent Actors query

```
db.shows.aggregate([
  {$project: { _id: 0, cast: 1} },
  {$unwind: "$cast" },
  {$group: {
        _id: { $trim: { input: "$cast" } },
        count: { "$sum": 1 }
    }},
  {$project: { _id: 0, actor: "$_id", movie_count: "$count"} },
  {$sort: { movie_count: -1 } },
  {$limit : 20 }
 ]);
```

Figure 10: fourth query

Most frequent Actors results

	{} actor	{} movie_count
1	Anupam Kher	33
2	Shah Rukh Khan	30
3	Naseeruddin Shah	27
4	Om Puri	27
5	Akshay Kumar	26
6	Yuki Kaji	26
7	Takahiro Sakurai	25
8	Paresh Rawal	25
9	Amitabh Bachchan	24
10	Boman Irani	23
11	Andrea Libman	22
12	John Cleese	22
13	Ashleigh Ball	22
14	Kareena Kapoor	19
15	Gulshan Grover	18
16	Fred Tatasciore	18
17	Daisuke Ono	18
18	Erin Fitzgerald	18
19	Vincent Tong	18
20	David Attenborough	18

Figure 11: fourth query – 20 results

Some notes on the query:

- selects only the cast column
- converts each actor in the json array into a separate record
- groups based on the trimmed actor (devoid of trailing and leading whitespaces) and counts the actor for each show
- keeps only the columns of interest
- performs descending sorting based on the count
- limits the results to 20

The full results of the query are stored in *project_2/data/results/results_4.json* file.

Most frequent Genre per Actor query

```
{$project: { _id: 0, cast: 1, listed_in: 1} },
{$unwind: "$cast" },
{$unwind: "$listed_in" },
{$group: {
      _id: {actor: { $trim: { input: "$cast" } },
              genre: { $trim: { input: "$listed_in" } }},
      count: { "$sum": 1 }
  }},
 {$sort: {
    count: -1
  }},
{$group: {
    _id: {
      actor: "$_id.actor"
    },
    genre: {
      "$first": "$_id.genre"
    },
    count: {
      "$first": "$count"
    }
  }
},
{$project: { _id: 0, actor: "$_id.actor", genre: "$genre", count: "$count"} },
{$sort: { actor: 1 } }
]);
```

Figure 11: fifth query

Most frequent Genre per Actor first 20 results

	{} actor	{} count	{} genre
1	2 Chainz	1	Docuseries
2	4Minute	1	Music & Musicals
3	50 Cent	2	Action & Adventure
4	A Boogie Wit tha Hoodie	1	Docuseries
5	A-ra Go	1	Korean TV Shows
6	A. Murat Özgen	1	International Movies
7	A.C. Peterson	1	Dramas
8	A.D. Miles	2	TV Comedies
9	A.J. Cook	1	Crime TV Shows
10	A.J. LoCascio	2	Kids' TV
11	A.K. Hangal	3	International Movies
12	A.R. Rahman	1	International Movies
13	A.S. Sasi Kumar	1	International Movies
14	AFRA	1	International TV Shows
15	AJ Bowen	1	Independent Movies
16	AJ Michalka	1	TV Sci-Fi & Fantasy
17	AJ Rivera	1	TV Action & Adventure
18	Aabhas Yadav	1	Music & Musicals
19	Aachal Munjal	1	Dramas
20	Aadarsh Balakrishna	2	International Movies

Figure 12: fifth query

Some notes on the query:

- selects only the cast and liste_in columns
- converts each actor and genre in the json arrays into a separate record
- groups based on the trimmed actor (devoid of trailing and leading whitespaces) and the trimmed genre and counts the total amount of genres per actor
- sorts based on the count in descending order
- groups the actors from the previous group and keeps the genre with the most count
- keeps the actor, genre and count columns
- performs ascending sorting based on the actor's name

The full results of the query are stored in *project_2/data/results/results_5.json* file.

Visualization

For visualization purposes a jupyter notebook was used, created by utilizing the google colab echosystem [7]. It is included in the project files in ***project_2/visualization/project_2_visualization.ipynb*** but can be also viewed from the following link

i. Netflix 2019 Content

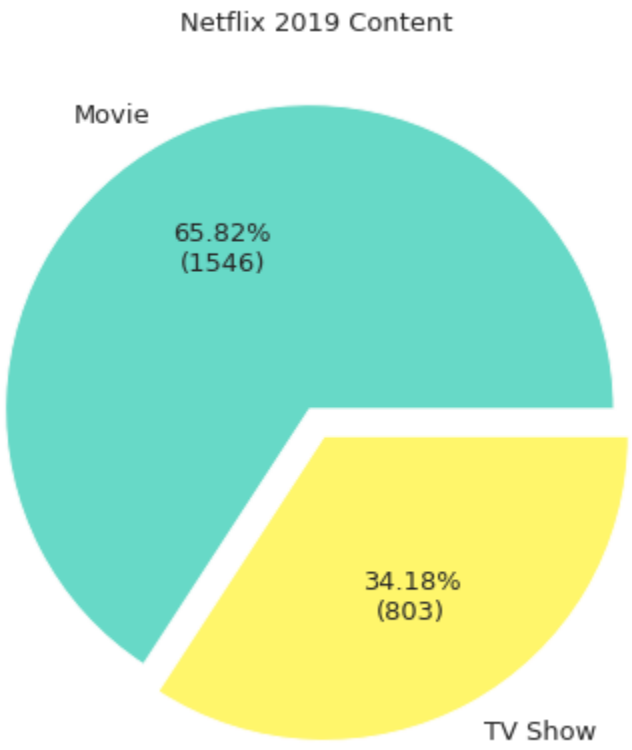

Figure 13: Netflix 2019 Content Pie

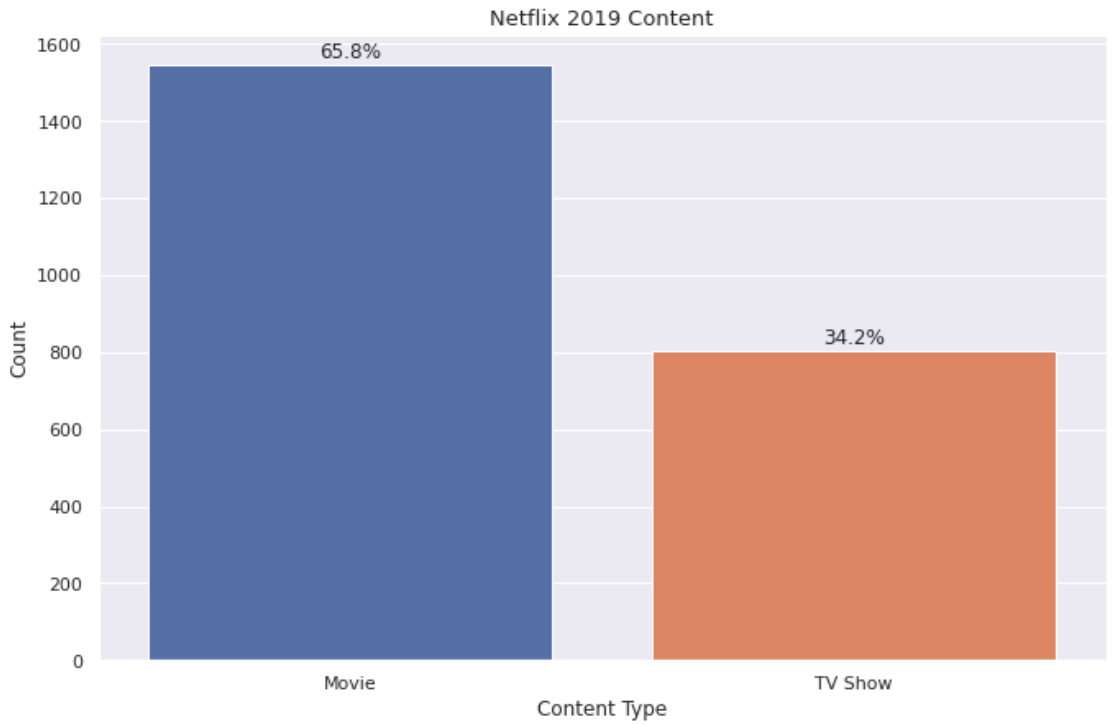

Figure 13: Netflix 2019 Content Barplot

As we can see the majority of netflix shows for 2019 are consisted of movies. This is expected since, usually movies require a lot less dedication to create than entire shows consisted of many possible seasons.

ii. **Content per Country**

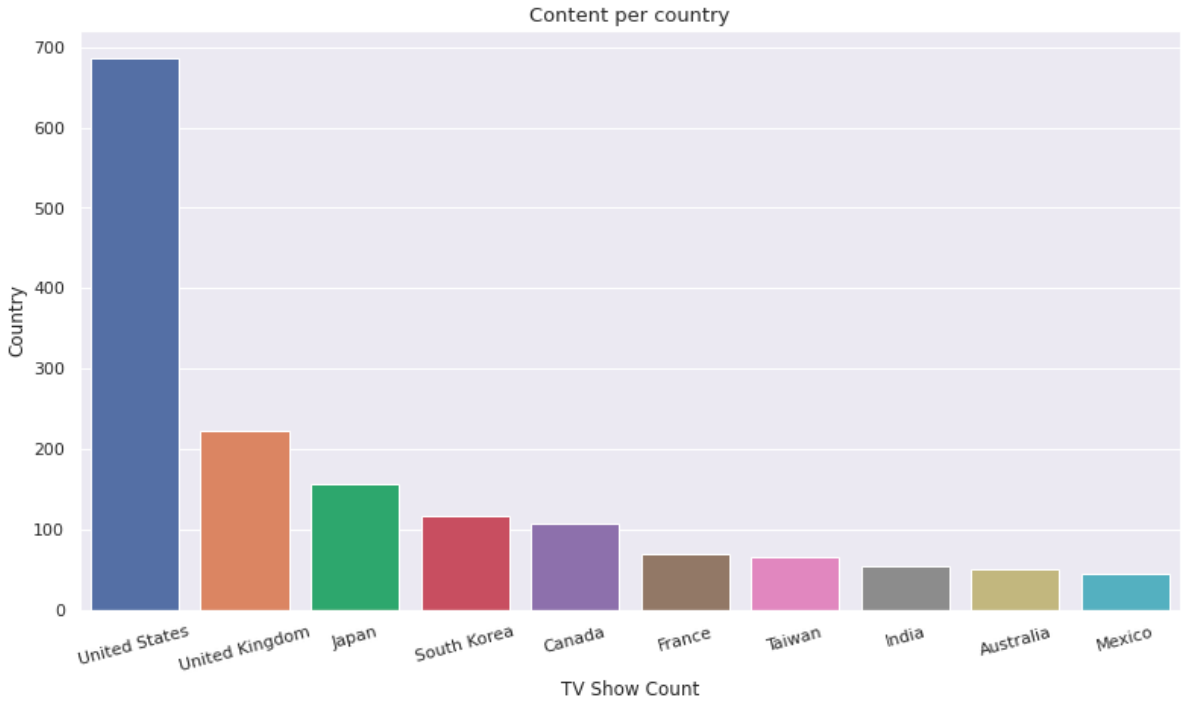

Figure 14: Content per Country Barplot

USA is dominating Netflix's productions overall. That comes as no surprise since Netflix originally began in USA, not to mention that USA invests humongous budgets on movies and shows. The most famous / known TV shows are usually produced in America. United Kingdom comes next and Japan afterwards. Surprising is also the fact that India, known for a handful of Bollywood blockbusters, comes in the 8th place in Netflix concerning the TV shows, after Taiwan which is in the 7th place. As a result, we can safely assume that TV show productions are on an entirely different level than movie productions.

iii. **Most common Genre**

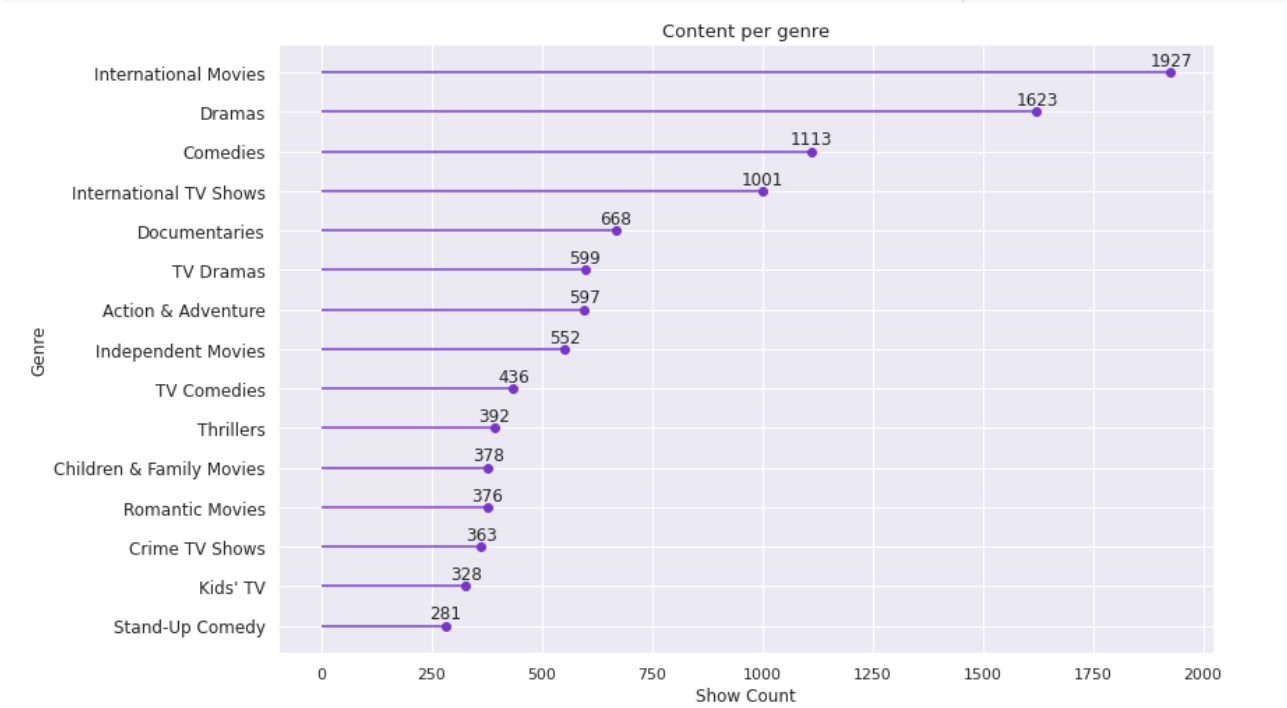

Figure 15: Most common Genre plot

Regarding the genres, the most prevalent ones are international movies, dramas and comedies. That makes a lot of sense since a show can belong in many genres at a time. One would expect thrillers to be a bit higher in the ranking though.

iv. Most frequent actors

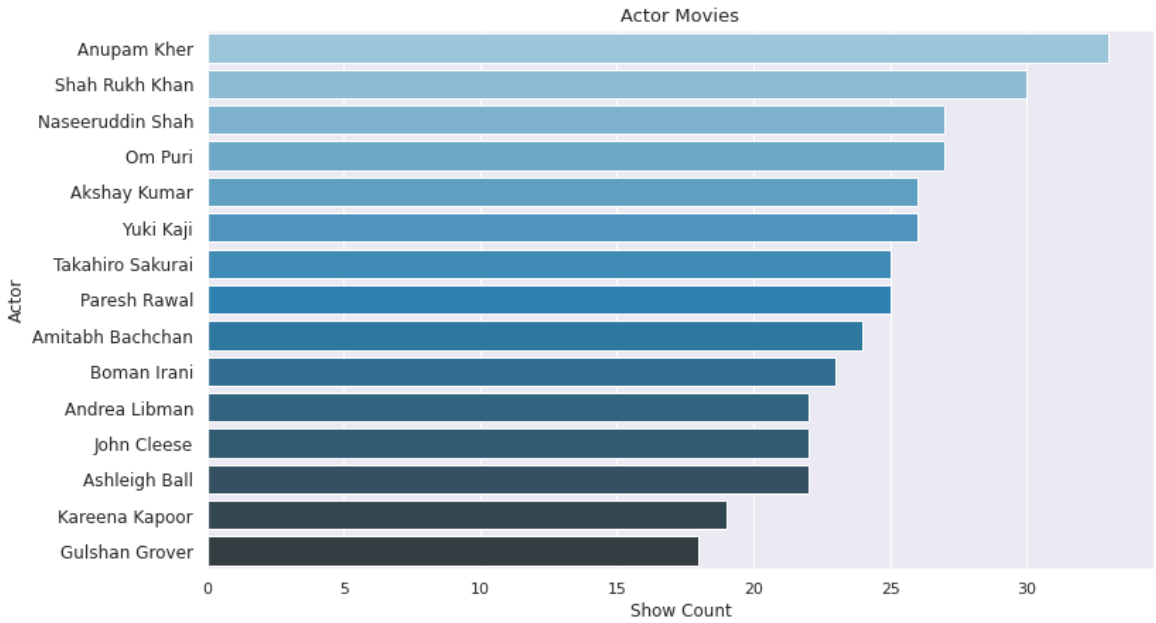

Figure 16: Most frequent actors barplot

The major surprise here is that there is not a single well-known name from blockbuster movies/shows in the top results, as one would expect. Rather most of them come from India. That may be, because Netflix has selected fewer blockbusters due to expensive copyrights. However, movies from India may be cheaper to buy.

v. Actors with the most movies/shows per genre

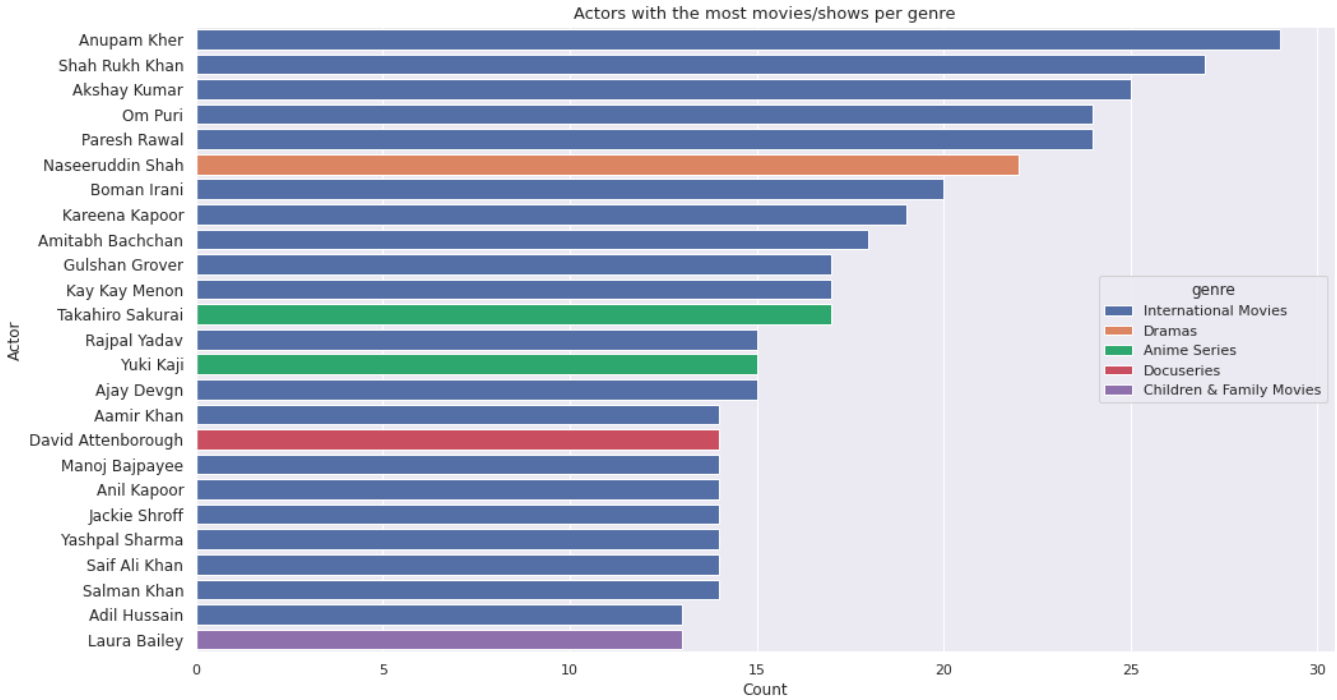

Figure 17: Actors with the most movies/shows per genre

The above plot complements Figure 16. We can see that the most frequent actors have played in international movies/TV Shows.

References

1. https://pop.system76.com/
2. https://www.mongodb.com/
3. https://www.jetbrains.com/pycharm/
4. https://www.jetbrains.com/datagrip/
5. https://docs.mongodb.com/database-tools/mongoimport/
6. https://www.python.org/downloads/release/python-395/
7. https://research.google.com/colaboratory/
8. https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/