ΔΗΜΟΚΡΙΤΟΣ
DEMOKRITOS

BIG DATA MANAGEMENT

2ND PROJECT 2020-2021

TEAM MEMBERS
MALFA ILIA AIKATERINI
LERTAS GEORGIOS

Table of Contents

1. Data Preprocessing

In this assignment, we had to process the Netflix dataset for the years from 2013 to 2019 and run some queries in mongodb. Below, we can see the features that make up the dataset:

- show_id - Unique ID for every entry
- type - An Identifier; it can be either Movie or TV Show
- title - Title of the Movie / TV Show
- director - Director of the Movie
- cast - Actors involved in the Movie / TV show
- country - Country where the Movie / TV show was produced
- date_added - Date the Movie / TV show was added on Netflix
- release_year - Actual Release year of the Movie / TV show
- rating - TV Rating of the Movie / TV show
- duration - Total Duration in minutes for movies or number of seasons for TV shows
- listed_in - Genre (Documentaries, Stand-Up Comedy, Dramas, Comedies, Kids' TV, International Movies, Independent Movies, etc.)
- description - The summary description

For the pre-processing of our data, we loaded the csv file in Python with a Python script, with which, after we processed the data, we export a json file to import it in mongodb.
The script file is available along with the report and is called preprocessing.py
Below we see the content from the script and justify our actions to pre-process the data.

In the beginning, we converted the values of the 'date_added' attribute from string type to date type such as json code. We made this change to be the feature in the appropriate type. Also, initially, we convert from string type to Timestamp type, because python cannot convert it from string to date type such as json code. Below, we have the code:

```
df.date_added=pd.to_datetime(df.date_added, unit='ns')
for i, j in df.date_added.items():
   df.date_added[i] = df.date_added[i].isoformat()
```

Then, we converted values of column 'cast' from string to list, because a value from this attribute can have multiple actor names. Below, we have the code:

```
for label, content in df['cast'].items():
   if type(content)==str:
      df['cast'][label]=content.split(", ")
```

And we do the same as before, with the attribute 'listed_in', for the same reason. Below, we have the code:

```
for label, content in df['listed_in'].items():
   if type(content)==str:
      df['listed_in'][label]=content.split(", ")
```

Also, for the country attribute, same reason and same procedure, because some rows have more than one country. Below, we have the code:

```
for label, content in df['country'].items():
   if type(content)==str:
        df['country'][label]=content.split(", ")
```
And the same for the director attribute. Below, we have the code:

```
for label, content in df['director'].items():
    if type(content)==str:
        df['director'][label]=content.split(", ")
```

In the above examples, we chose to use robbers in attributes because we considered it the most ideal solution. However, we thought of using another way of representation such as dictionaries (how call in Python), but we rejected this way because we had individual values and could not represent them in the key-value format, we had to completely change the structure of each attribute.

Finally, we converted the dataframe to a json string with the following code, because mongodb accepts json files:

```
result = df.to_json(orient="records")
jsonArray = json.loads(result)
with open('netflix.json', 'w', encoding='utf-8') as json_file:
    jsonString = json.dumps(jsonArray, indent=4)
    json_file.write(jsonString)
```

2. Data uploading

To upload the data we ran the following command:

```
mongoimport -d project -c netflix --drop --jsonArray --file
"/Users/giorgoslertas/Desktop/Lertas files/data science/Semester 2/Big data
management/Projects/BDM_Project2/netflix.json" --port 27017 --host 127.0.0.1
```

More specifically,

-d: Specifies the name of the database, we give name 'project'
-c: Specifies the name of the collection, we give name 'netflix'
--drop: Modifies the import process so that the target instance drops the collection before importing the data from the input.
--type: Specifies the file type to import.
--jsonArray: Accepts the import of data expressed with multiple MongoDB documents within a single json array. Limited to imports of 16 MB or smaller.
--file: Specifies the location and name of a file containing the data to import.
--port: Specifies the TCP port on which the MongoDB instance listens for client connections.
--host: Specifies the resolvable hostname of the MongoDB deployment.
-- And in the quotation marks (" ") we have the path of the file

3. Data Analysis

3.1 Content available in 2019

For the first query, we used find operation from mongodb, to return us the rows in year 2019. Also, we got only show_id, type and title of each row in json file.

db.netflix.find({date_added:/2019/}, {show_id:1, type:1, title:1, _id:0})

3.2 Series production countries

Then, for the next query we used aggregate operation to process the data that returns the computed results. More specifically, we used match to return us the rows with type equals to "TV Show". Also, we used unwind to open the list's values for country attribute, that return us a document for each element from an array field from the input documents, and sortByCount, which is equivalent to $group + $sort sequence. We used sortByCount to group, count and sort the country attribute by descending.

**db.netflix.aggregate([{$match:{"type":"TV Show"}},
{$unwind:"$country"},{$sortByCount:"$country"}])**

3.3 Types of content available

In the next query, we used aggregate operation and inside it, we used unwind for the attribute with the lists of genres for the movies. Afterward, we used sortByCount to group, count and sort the listed_in attribute by descending.

db.netflix.aggregate([{$unwind:"$listed_in"},{$sortByCount:"$listed_in"}])

3.4 Appearing actors

Continuing to the next query, we used aggregate operation. In this operation, we used, again unwind and sortByCount for the cast attribute, for same reasons as we explain in the above queries. Also, we used limit equal to 20, to return us only the first 20 rows.

db.netflix.aggregate([{$unwind:"$cast"},{$sortByCount:"$cast"},{$limit:20}])

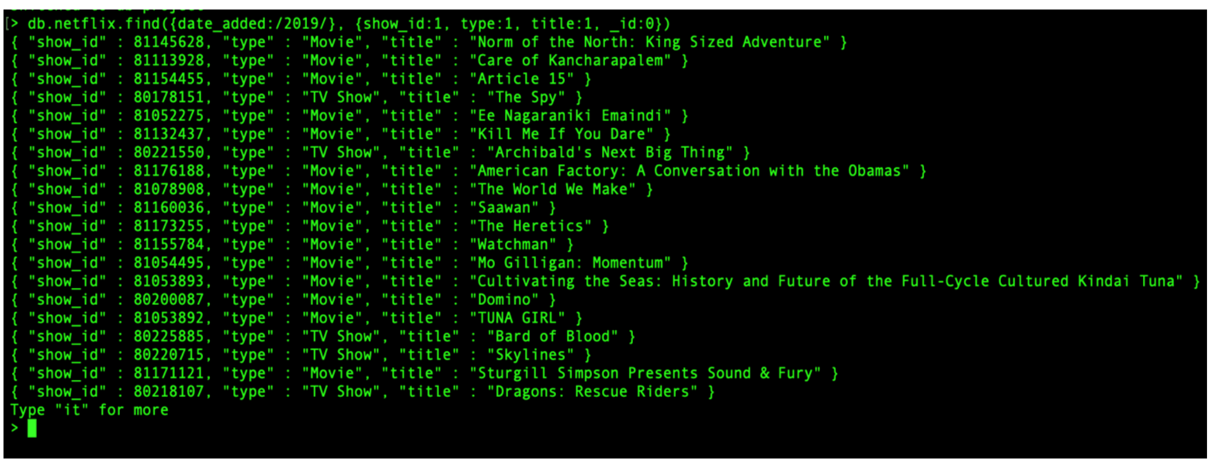
3.5 Top actors' preferences

At last query, we used again aggregate operation. Subsequently, we used unwind for two columns to get all the values flatten. Then, we create a group with these two values, cast and listed_in and we count these pairs. Afterwards, we sort them first by actor's name by ascending and then with the counter by descending, to get in the next group the most common genre from each actor, and in the last sorting issue we used to sort listed_in values by ascending, for the reason that we got equal count numbers. Closing, we created a group with these 3 columns, but for each actor we keep only first genre and first count and then we sort them based on the names of actors by ascending.

**db.netflix.aggregate([{$unwind:"$cast"},{$unwind:"$listed_in"},
{$group:{_id:{cast:"$cast", listed_in:"$listed_in"},counter:{$sum:1}}},
{$sort:{"_id.cast":1, counter:-1, "_id.listed_in":1}}, {$group:{_id:{actor:"$_id.cast"},
genre:{$first:"$_id.listed_in"},counter:{$first:"$counter"}}},{$sort:{"_id.actor":1}}])**

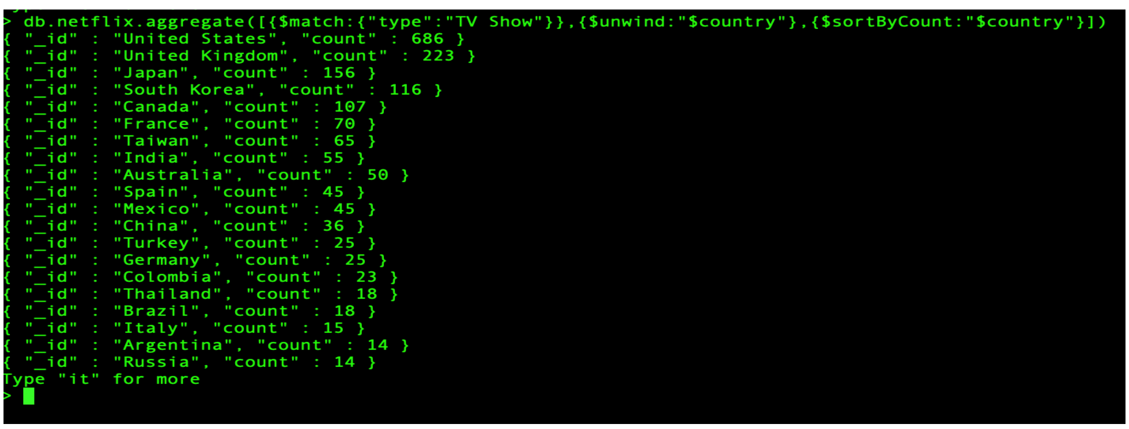
4. Screenshots with the results

4.1 Screenshot task1



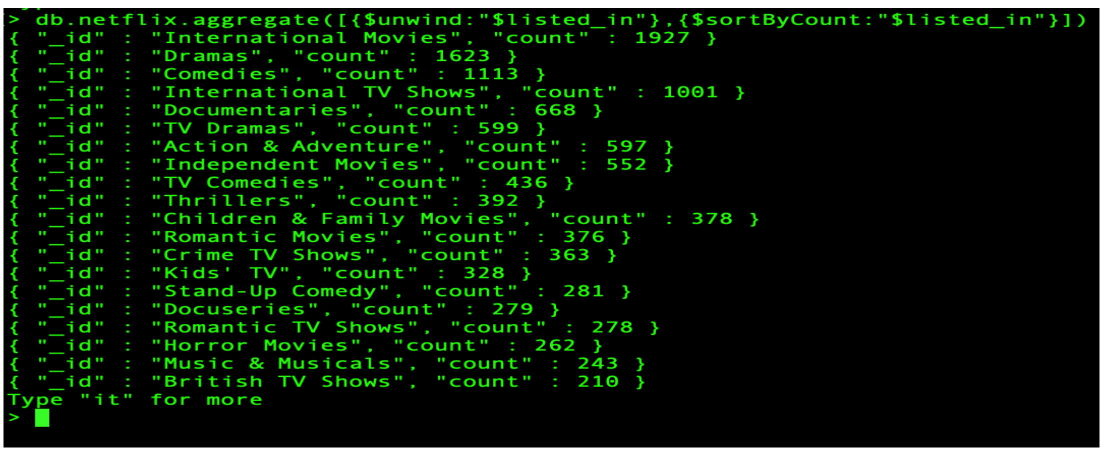
```
[> db.netflix.find({date_added:/2019/}, {show_id:1, type:1, title:1, _id:0})
{ "show_id" : 81145628, "type" : "Movie", "title" : "Norm of the North: King Sized Adventure" }
{ "show_id" : 81113928, "type" : "Movie", "title" : "Care of Kancharapalem" }
{ "show_id" : 81154455, "type" : "Movie", "title" : "Article 15" }
{ "show_id" : 80178151, "type" : "TV Show", "title" : "The Spy" }
{ "show_id" : 81052275, "type" : "Movie", "title" : "Ee Nagaraniki Emaindi" }
{ "show_id" : 81132437, "type" : "Movie", "title" : "Kill Me If You Dare" }
{ "show_id" : 80221550, "type" : "TV Show", "title" : "Archibald's Next Big Thing" }
{ "show_id" : 81176188, "type" : "Movie", "title" : "American Factory: A Conversation with the Obamas" }
{ "show_id" : 81078908, "type" : "Movie", "title" : "The World We Make" }
{ "show_id" : 81160036, "type" : "Movie", "title" : "Saawan" }
{ "show_id" : 81173255, "type" : "Movie", "title" : "The Heretics" }
{ "show_id" : 81155784, "type" : "Movie", "title" : "Watchman" }
{ "show_id" : 81054495, "type" : "Movie", "title" : "Mo Gilligan: Momentum" }
{ "show_id" : 81053893, "type" : "Movie", "title" : "Cultivating the Seas: History and Future of the Full-Cycle Cultured Kindai Tuna" }
{ "show_id" : 80200087, "type" : "Movie", "title" : "Domino" }
{ "show_id" : 81053892, "type" : "Movie", "title" : "TUNA GIRL" }
{ "show_id" : 80225885, "type" : "TV Show", "title" : "Bard of Blood" }
{ "show_id" : 80220715, "type" : "TV Show", "title" : "Skylines" }
{ "show_id" : 81171121, "type" : "Movie", "title" : "Sturgill Simpson Presents Sound & Fury" }
{ "show_id" : 80218107, "type" : "TV Show", "title" : "Dragons: Rescue Riders" }
Type "it" for more
>
```

4.2 Screenshot task2



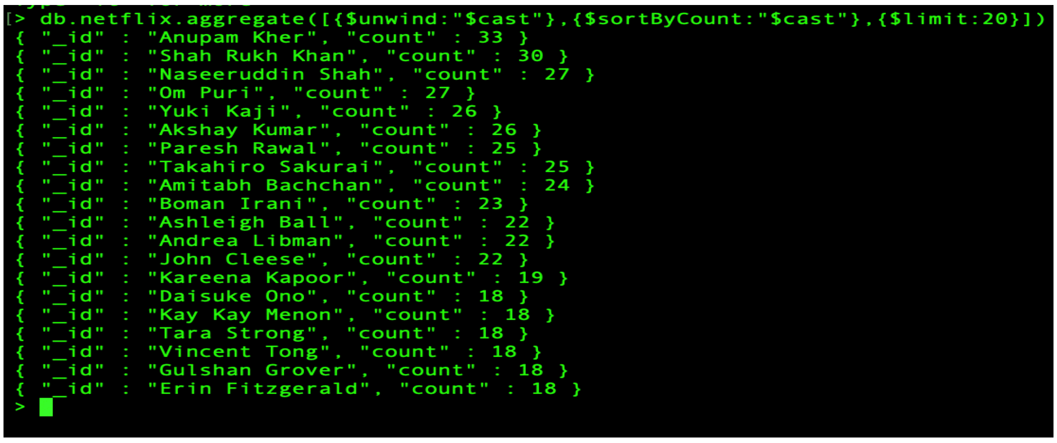
```
> db.netflix.aggregate([{$match:{"type":"TV Show"}},{$unwind:"$country"},{$sortByCount:"$country"}])
{ "_id" : "United States", "count" : 686 }
{ "_id" : "United Kingdom", "count" : 223 }
{ "_id" : "Japan", "count" : 156 }
{ "_id" : "South Korea", "count" : 116 }
{ "_id" : "Canada", "count" : 107 }
{ "_id" : "France", "count" : 70 }
{ "_id" : "Taiwan", "count" : 65 }
{ "_id" : "India", "count" : 55 }
{ "_id" : "Australia", "count" : 50 }
{ "_id" : "Spain", "count" : 45 }
{ "_id" : "Mexico", "count" : 45 }
{ "_id" : "China", "count" : 36 }
{ "_id" : "Turkey", "count" : 25 }
{ "_id" : "Germany", "count" : 25 }
{ "_id" : "Colombia", "count" : 23 }
{ "_id" : "Thailand", "count" : 18 }
{ "_id" : "Brazil", "count" : 18 }
{ "_id" : "Italy", "count" : 15 }
{ "_id" : "Argentina", "count" : 14 }
{ "_id" : "Russia", "count" : 14 }
Type "it" for more
>
```

4.3 Screenshot task3



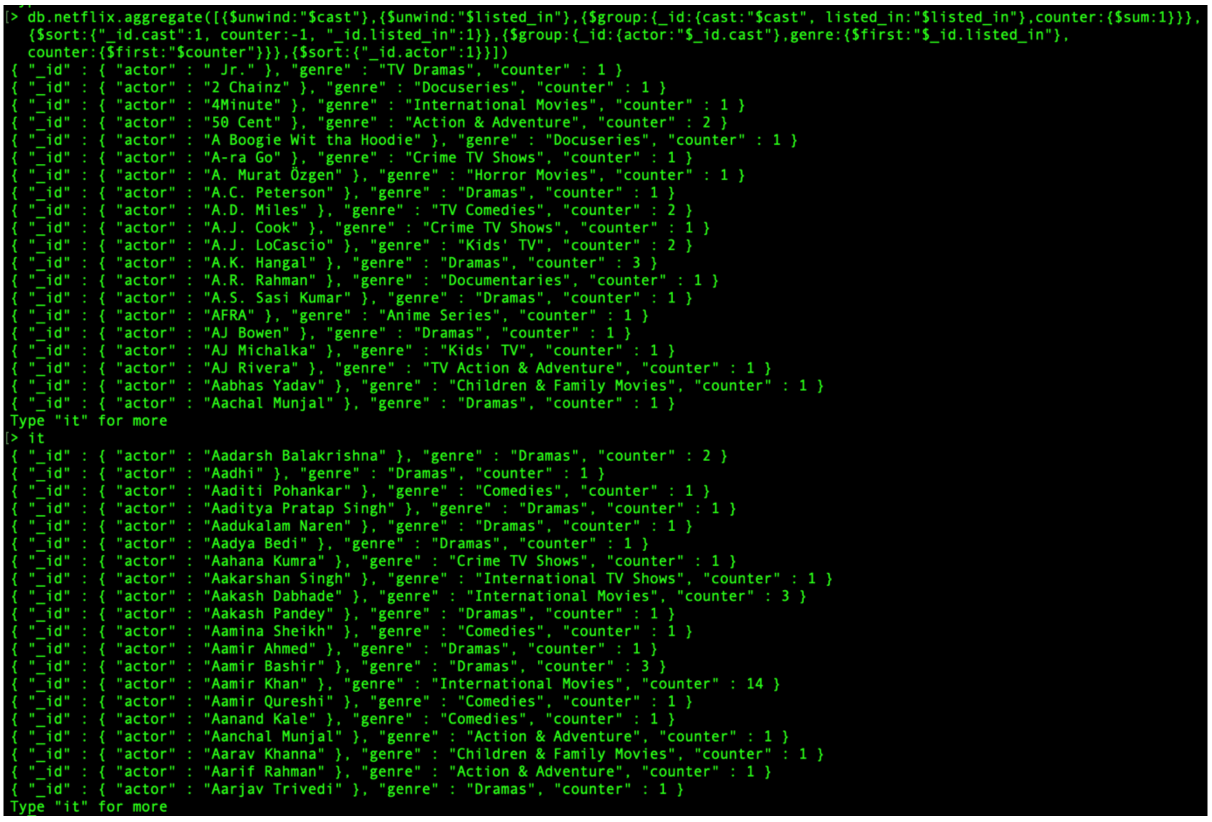
```
> db.netflix.aggregate([{$unwind:"$listed_in"},{$sortByCount:"$listed_in"}])
{ "_id" : "International Movies", "count" : 1927 }
{ "_id" : "Dramas", "count" : 1623 }
{ "_id" : "Comedies", "count" : 1113 }
{ "_id" : "International TV Shows", "count" : 1001 }
{ "_id" : "Documentaries", "count" : 668 }
{ "_id" : "TV Dramas", "count" : 599 }
{ "_id" : "Action & Adventure", "count" : 597 }
{ "_id" : "Independent Movies", "count" : 552 }
{ "_id" : "TV Comedies", "count" : 436 }
{ "_id" : "Thrillers", "count" : 392 }
{ "_id" : "Children & Family Movies", "count" : 378 }
{ "_id" : "Romantic Movies", "count" : 376 }
{ "_id" : "Crime TV Shows", "count" : 363 }
{ "_id" : "Kids' TV", "count" : 328 }
{ "_id" : "Stand-Up Comedy", "count" : 281 }
{ "_id" : "Docuseries", "count" : 279 }
{ "_id" : "Romantic TV Shows", "count" : 278 }
{ "_id" : "Horror Movies", "count" : 262 }
{ "_id" : "Music & Musicals", "count" : 243 }
{ "_id" : "British TV Shows", "count" : 210 }
Type "it" for more
>
```

4.4 Screenshot task4



```
[> db.netflix.aggregate([{$unwind:"$cast"},{$sortByCount:"$cast"},{$limit:20}])
{ "_id" : "Anupam Kher", "count" : 33 }
{ "_id" : "Shah Rukh Khan", "count" : 30 }
{ "_id" : "Naseeruddin Shah", "count" : 27 }
{ "_id" : "Om Puri", "count" : 27 }
{ "_id" : "Yuki Kaji", "count" : 26 }
{ "_id" : "Akshay Kumar", "count" : 26 }
{ "_id" : "Paresh Rawal", "count" : 25 }
{ "_id" : "Takahiro Sakurai", "count" : 25 }
{ "_id" : "Amitabh Bachchan", "count" : 24 }
{ "_id" : "Boman Irani", "count" : 23 }
{ "_id" : "Ashleigh Ball", "count" : 22 }
{ "_id" : "Andrea Libman", "count" : 22 }
{ "_id" : "John Cleese", "count" : 22 }
{ "_id" : "Kareena Kapoor", "count" : 19 }
{ "_id" : "Daisuke Ono", "count" : 18 }
{ "_id" : "Kay Kay Menon", "count" : 18 }
{ "_id" : "Tara Strong", "count" : 18 }
{ "_id" : "Vincent Tong", "count" : 18 }
{ "_id" : "Gulshan Grover", "count" : 18 }
{ "_id" : "Erin Fitzgerald", "count" : 18 }
>
```

4.5 Screenshot task5



```
[> db.netflix.aggregate([{$unwind:"$cast"},{$unwind:"$listed_in"},{$group:{_id:{cast:"$cast", listed_in:"$listed_in"},counter:{$sum:1}}},
  {$sort:{"_id.cast":1, counter:-1, "_id.listed_in":1}},{$group:{_id:{actor:"$_id.cast"},genre:{$first:"$_id.listed_in"},
  counter:{$first:"$counter"}}},{$sort:{"_id.actor":1}}])
{ "_id" : { "actor" : " Jr." }, "genre" : "TV Dramas", "counter" : 1 }
{ "_id" : { "actor" : "2 Chainz" }, "genre" : "Docuseries", "counter" : 1 }
{ "_id" : { "actor" : "4Minute" }, "genre" : "International Movies", "counter" : 1 }
{ "_id" : { "actor" : "50 Cent" }, "genre" : "Action & Adventure", "counter" : 2 }
{ "_id" : { "actor" : "A Boogie Wit tha Hoodie" }, "genre" : "Docuseries", "counter" : 1 }
{ "_id" : { "actor" : "A-ra Go" }, "genre" : "Crime TV Shows", "counter" : 1 }
{ "_id" : { "actor" : "A. Murat Özgen" }, "genre" : "Horror Movies", "counter" : 1 }
{ "_id" : { "actor" : "A.C. Peterson" }, "genre" : "Dramas", "counter" : 1 }
{ "_id" : { "actor" : "A.D. Miles" }, "genre" : "TV Comedies", "counter" : 2 }
{ "_id" : { "actor" : "A.J. Cook" }, "genre" : "Crime TV Shows", "counter" : 1 }
{ "_id" : { "actor" : "A.J. LoCascio" }, "genre" : "Kids' TV", "counter" : 2 }
{ "_id" : { "actor" : "A.K. Hangal" }, "genre" : "Dramas", "counter" : 3 }
{ "_id" : { "actor" : "A.R. Rahman" }, "genre" : "Documentaries", "counter" : 1 }
{ "_id" : { "actor" : "A.S. Sasi Kumar" }, "genre" : "Dramas", "counter" : 1 }
{ "_id" : { "actor" : "AFRA" }, "genre" : "Anime Series", "counter" : 1 }
{ "_id" : { "actor" : "AJ Bowen" }, "genre" : "Dramas", "counter" : 1 }
{ "_id" : { "actor" : "AJ Michalka" }, "genre" : "Kids' TV", "counter" : 1 }
{ "_id" : { "actor" : "AJ Rivera" }, "genre" : "TV Action & Adventure", "counter" : 1 }
{ "_id" : { "actor" : "Aabhas Yadav" }, "genre" : "Children & Family Movies", "counter" : 1 }
{ "_id" : { "actor" : "Aachal Munjal" }, "genre" : "Dramas", "counter" : 1 }
Type "it" for more
[> it
{ "_id" : { "actor" : "Aadarsh Balakrishna" }, "genre" : "Dramas", "counter" : 2 }
{ "_id" : { "actor" : "Aadhi" }, "genre" : "Dramas", "counter" : 1 }
{ "_id" : { "actor" : "Aaditi Pohankar" }, "genre" : "Comedies", "counter" : 1 }
{ "_id" : { "actor" : "Aaditya Pratap Singh" }, "genre" : "Dramas", "counter" : 1 }
{ "_id" : { "actor" : "Aadukalam Naren" }, "genre" : "Dramas", "counter" : 1 }
{ "_id" : { "actor" : "Aadya Bedi" }, "genre" : "Dramas", "counter" : 1 }
{ "_id" : { "actor" : "Aahana Kumra" }, "genre" : "Crime TV Shows", "counter" : 1 }
{ "_id" : { "actor" : "Aakarshan Singh" }, "genre" : "International TV Shows", "counter" : 1 }
{ "_id" : { "actor" : "Aakash Dabhade" }, "genre" : "International Movies", "counter" : 3 }
{ "_id" : { "actor" : "Aakash Pandey" }, "genre" : "Dramas", "counter" : 1 }
{ "_id" : { "actor" : "Aamina Sheikh" }, "genre" : "Comedies", "counter" : 1 }
{ "_id" : { "actor" : "Aamir Ahmed" }, "genre" : "Dramas", "counter" : 1 }
{ "_id" : { "actor" : "Aamir Bashir" }, "genre" : "Dramas", "counter" : 3 }
{ "_id" : { "actor" : "Aamir Khan" }, "genre" : "International Movies", "counter" : 14 }
{ "_id" : { "actor" : "Aamir Qureshi" }, "genre" : "Comedies", "counter" : 1 }
{ "_id" : { "actor" : "Aanand Kale" }, "genre" : "Comedies", "counter" : 1 }
{ "_id" : { "actor" : "Aanchal Munjal" }, "genre" : "Action & Adventure", "counter" : 1 }
{ "_id" : { "actor" : "Aarav Khanna" }, "genre" : "Children & Family Movies", "counter" : 1 }
{ "_id" : { "actor" : "Aarif Rahman" }, "genre" : "Action & Adventure", "counter" : 1 }
{ "_id" : { "actor" : "Aarjav Trivedi" }, "genre" : "Dramas", "counter" : 1 }
Type "it" for more
```

5. Export results to csv

Subsequently, we need to export the results from the queries to upload them in Jupyter Notebook, that we use to visualize them. Initially, we created a directory in desktop with name csv_tasks to add the csv files from the queries. Then, we create a javascript script file for each task to export data. Below, we can see the code for task1.

```
print("show_id^type^title")
cursor = db.netflix.find({date_added:/2019/}, {show_id:1, type:1, title:1, _id:0})
while (cursor.hasNext()) {
   jsonobject = cursor.next();
   print(jsonobject.show_id.valueOf() + "^" + jsonobject.type + "^" + jsonobject.title)
}
```

We used print with the column names. Then, we used the query that we run with name as cursor and we created a while loop for each row and we print the values from each 3 columns separated with "^", because some values used comma and other most common symbols. And we created this kind of script in each task, we changed only the values and the cursor name. As we can see below for all tasks.

Task2

```
print("country^series")
cursor = db.netflix.aggregate([{$match:{"type":"TV
Show"}},{$unwind:"$country"},{$sortByCount:"$country"}])
while (cursor.hasNext()) {
   jsonobject = cursor.next();
   print(jsonobject._id.valueOf() + "^" + jsonobject.count)
}
```

Task3

```
print("genre^productions")
cursor = db.netflix.aggregate([{$unwind:"$listed_in"},{$sortByCount:"$listed_in"}])
while (cursor.hasNext()) {
   jsonobject = cursor.next();
   print(jsonobject._id.valueOf() + "^" + jsonobject.count)
}
```

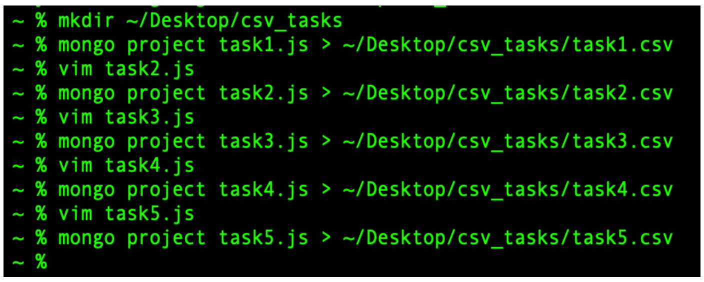
Task4

```
print("actor^productions")
cursor = db.netflix.aggregate([{$unwind:"$cast"},{$sortByCount:"$cast"},{$limit:20}])
while (cursor.hasNext()) {
   jsonobject = cursor.next();
   print(jsonobject._id.valueOf() + "^" + jsonobject.count)
}
```

Task5

```
print("actor^genre^works")
cursor =
db.netflix.aggregate([{$unwind:"$cast"},{$unwind:"$listed_in"},{$group:{_id:{cast:"$cast", listed_in:"$listed_in"},counter:{$sum:1}}},{$sort:{"_id.cast":1, counter:-1,
"_id.listed_in":1}},{$group:{_id:{actor:"$_id.cast"},genre:{$first:"$_id.listed_in"},counter:{$first:"$counter"}}},{$sort:{"_id.actor":1}}])
while (cursor.hasNext()) {
   jsonobject = cursor.next();
   print(jsonobject._id.actor.valueOf() + "^" + jsonobject.genre + "^" +
jsonobject.counter)
}
```

Below we can see the screenshot from terminal with the commands that we run to export the csv files. In the next step, we transferred the directory in the directory of project.



```
~ % mkdir ~/Desktop/csv_tasks
~ % mongo project task1.js > ~/Desktop/csv_tasks/task1.csv
~ % vim task2.js
~ % mongo project task2.js > ~/Desktop/csv_tasks/task2.csv
~ % vim task3.js
~ % mongo project task3.js > ~/Desktop/csv_tasks/task3.csv
~ % vim task4.js
~ % mongo project task4.js > ~/Desktop/csv_tasks/task4.csv
~ % vim task5.js
~ % mongo project task5.js > ~/Desktop/csv_tasks/task5.csv
~ %
```

6. Graphic illustration of results

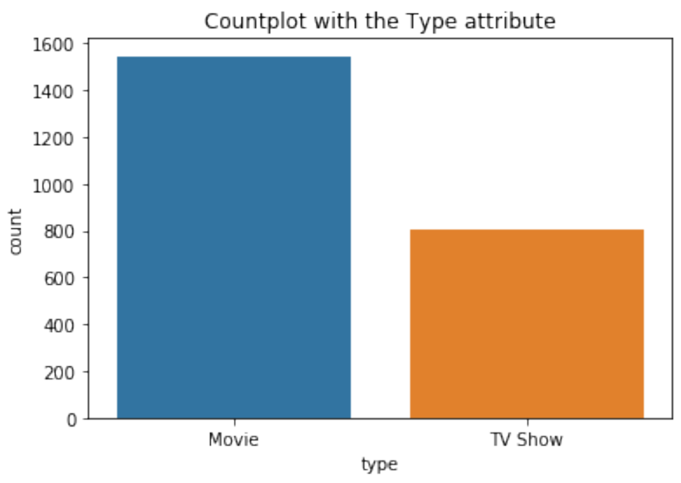
In each task, the csv file in the first 4 rows had some information for mongo, like the cell below, so we removed these 4 rows, when we load the data

Here are the first 4 rows:

```
MongoDB shell version v4.4.5
connecting to:
mongodb://127.0.0.1:27017/project?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("4ba3fc97-7d7b-4b72-8758-8f6e6e0bd7bf") }
MongoDB server version: 4.4.5
```

6.1 Visualization task1

In task1 data, we get 3 columns, show_id and title are nominal attributes and type is a binary attribute. We created a countplot for type attribute.

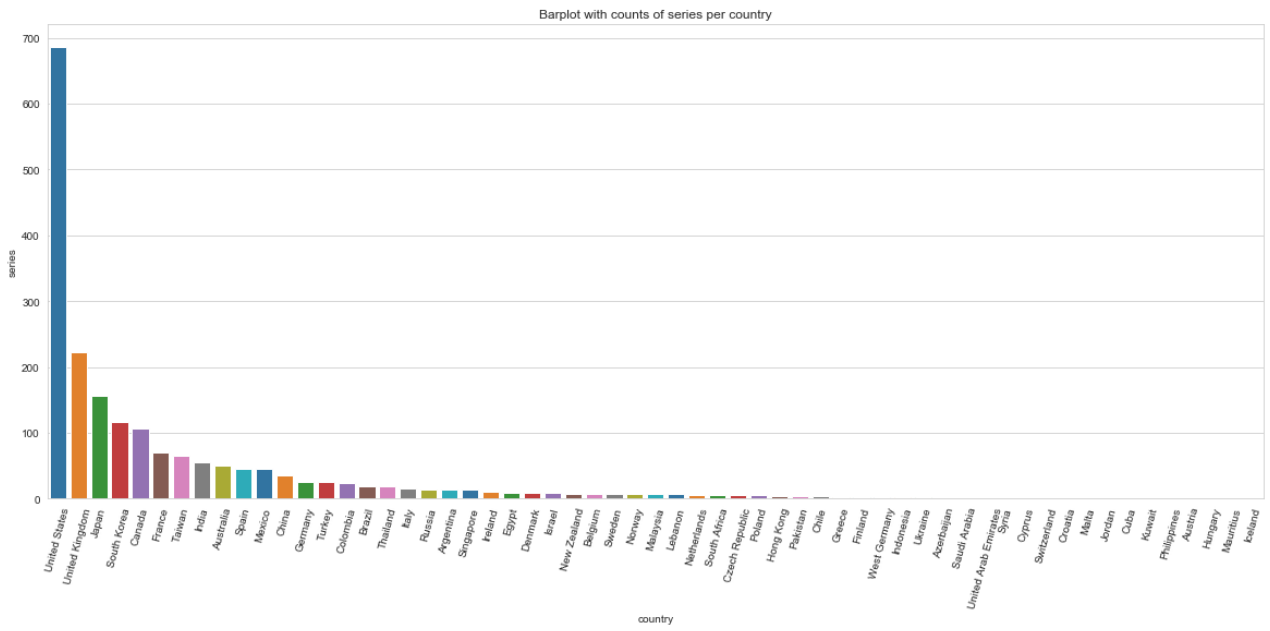


From the plot, we can understand that Movies are double in number than TV Shows.

With the other attributes we did not create visualizations, as they are nominal values. Also, show_id is unique for each row, and we can't visualize anything.

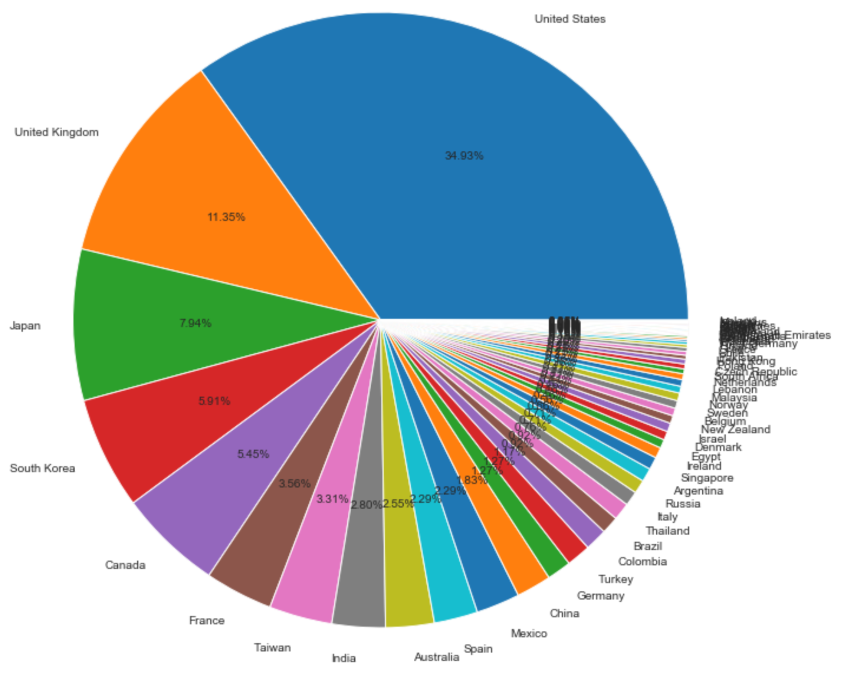
6.2 Visualization task2

In task2 data, we had 2 columns, country is a nominal attribute and series is a Ratio-scaled attribute, because of zero-point, values have order and measured on a scale of equal-sized units. We created a barplot with counts of series per country.

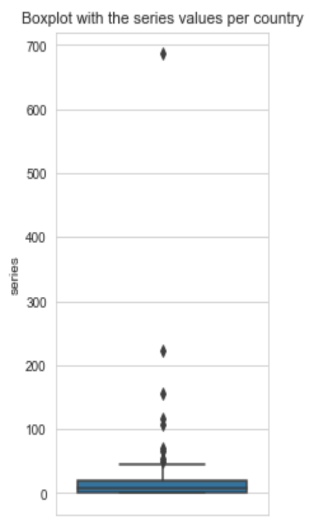


United States have the most series in number than other countries.

Also, we created a pie-plot with the percent of countries in series. We can see, that United States have more than 1/3 of series. Then, we can see United Kingdom with 11.35%.



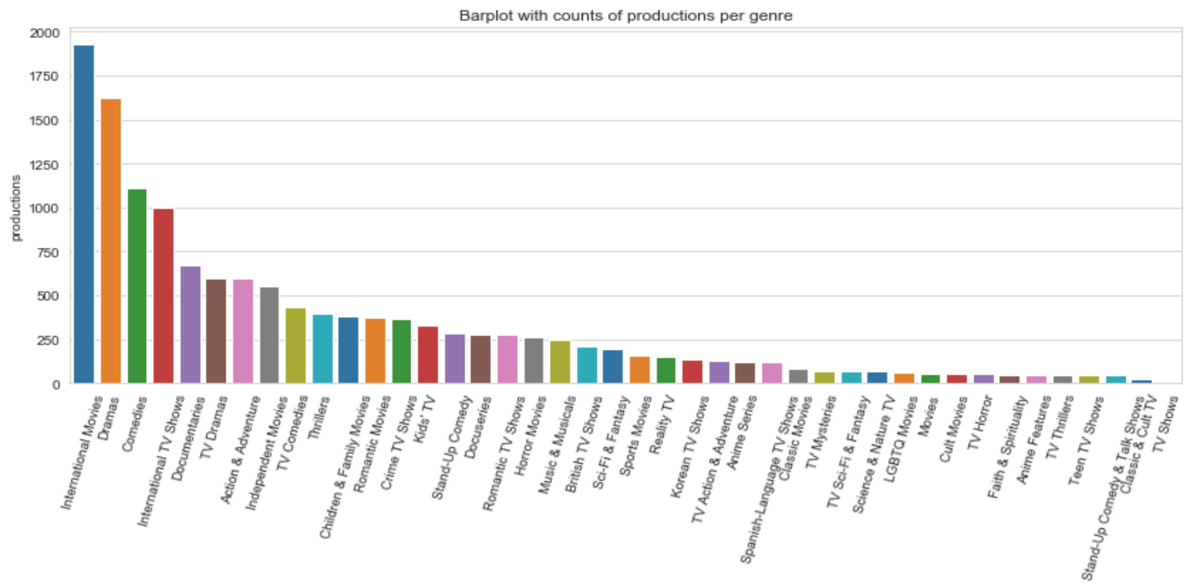
Then, we created a boxplot with series values.



Most countries have series lower than 50, and over 50 there are 7 countries.

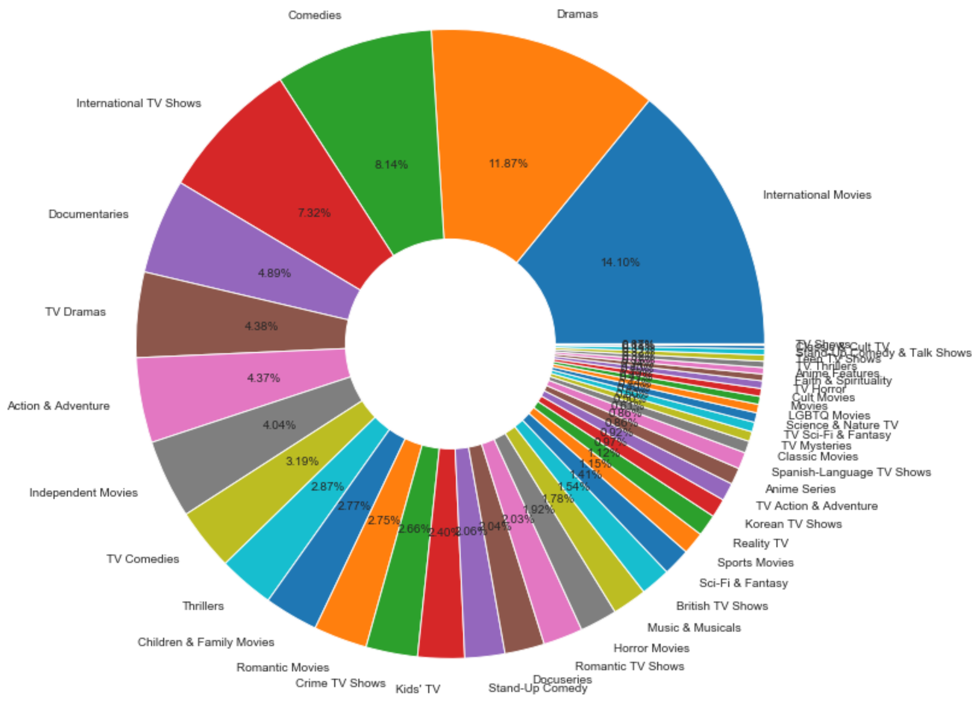
6.3 Visualization task3

In task3 data, we had 2 columns, genre is a nominal attribute and productions is a Ratio-scaled attribute, because of zero-point, values have order and measured on a scale of equal-sized units. We create a barplot with counts of productions per genre.



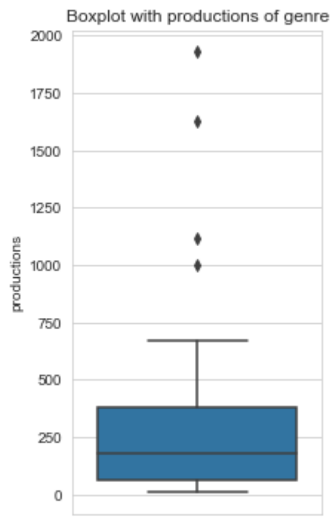
International Movies have the most productions and then we can see dramas, comedies and international TV shows.

Then, we created a donut chart as we can see below:



International Movies have the most productions with 14.1% and then we can see dramas with 11.87%. All the other values are lower than 10%.

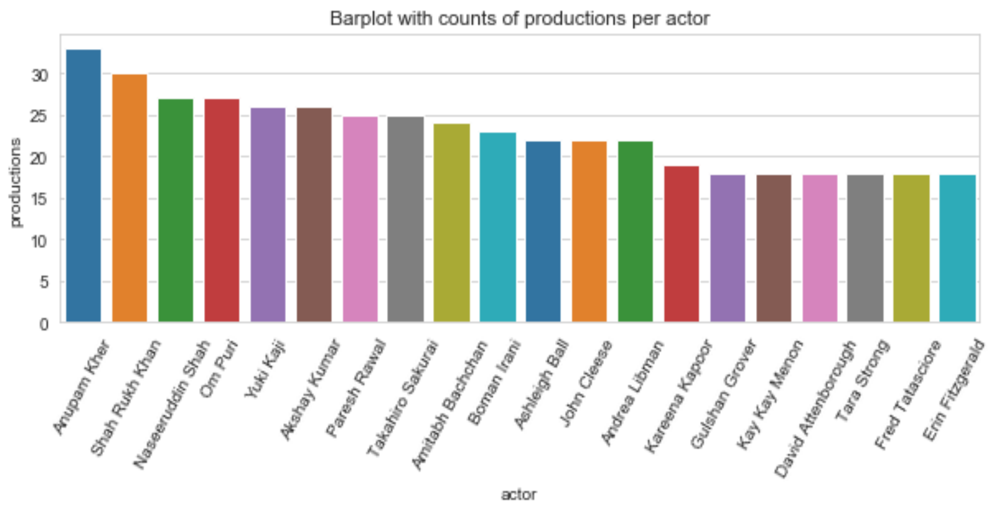
Also, we created a boxplot for productions.



Most of the values are lower than 750, and we can see 4 values over them. Maybe these values are international movies, dramas, comedies and international TV shows.

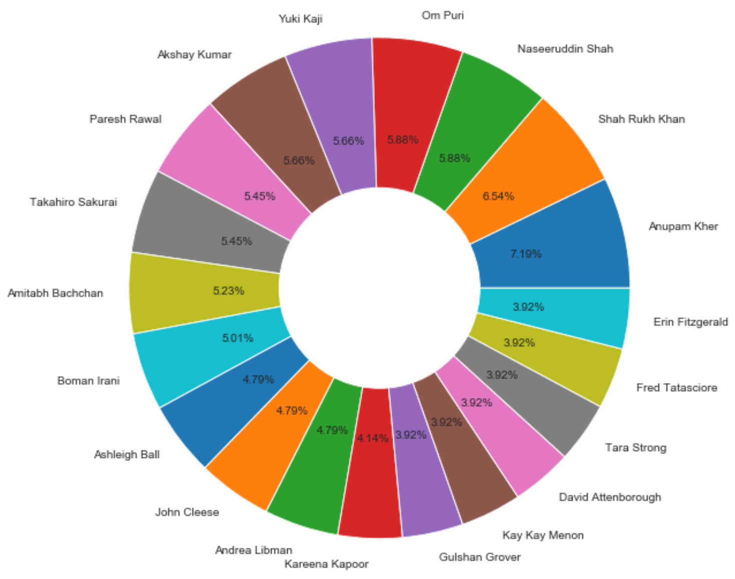
6.4 Visualization task4

In task4 data, we had 2 columns, actor is a nominal attribute and productions is a Ratio-scaled attribute, because of zero-point, values have order and measured on a scale of equal-sized units. We create a barplot with counts of productions per actor.



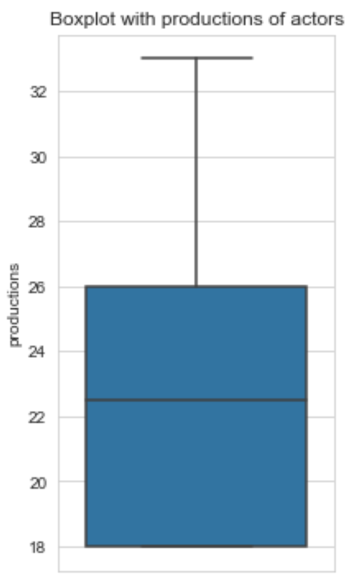
We can see in this plot, that Anupam Kher is the first actor with most productions. Then, there are Shah Rukh Khan, Naseeruddin Shah, Om Puri.

Then, we created a donut chart with the actors.



From the donut chart, we see that all the actors are very close, with the first one with 7.19% and the last one with 3.92%, which difference is near to 3%, which is not too much.

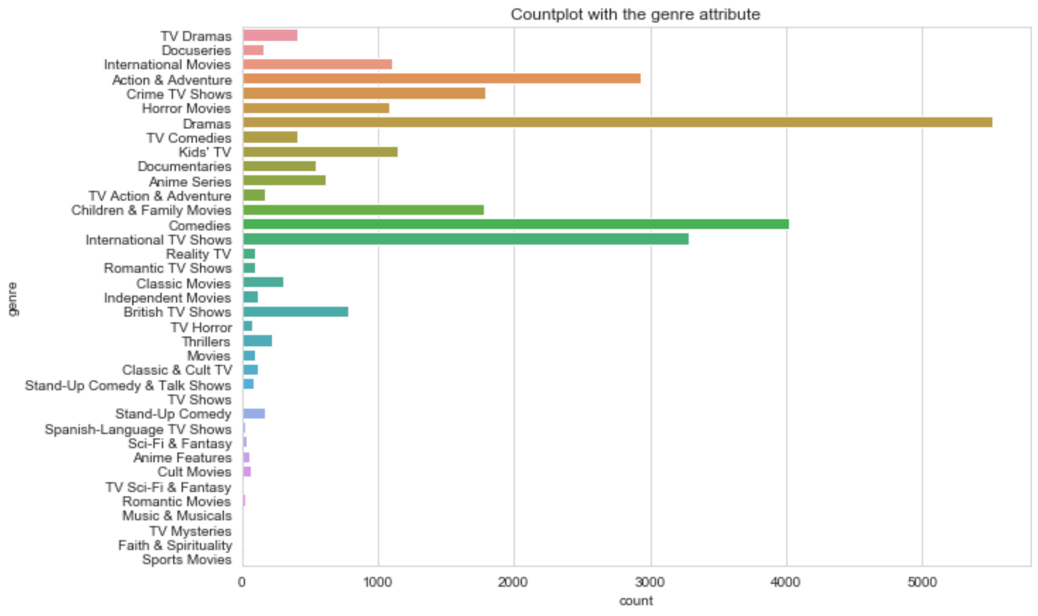
Then, we created a boxplot with productions of actors



We can see that the median is near to 22, and the values are from 18 to 26 from Q1 to Q3, and 33 is the upper extreme.

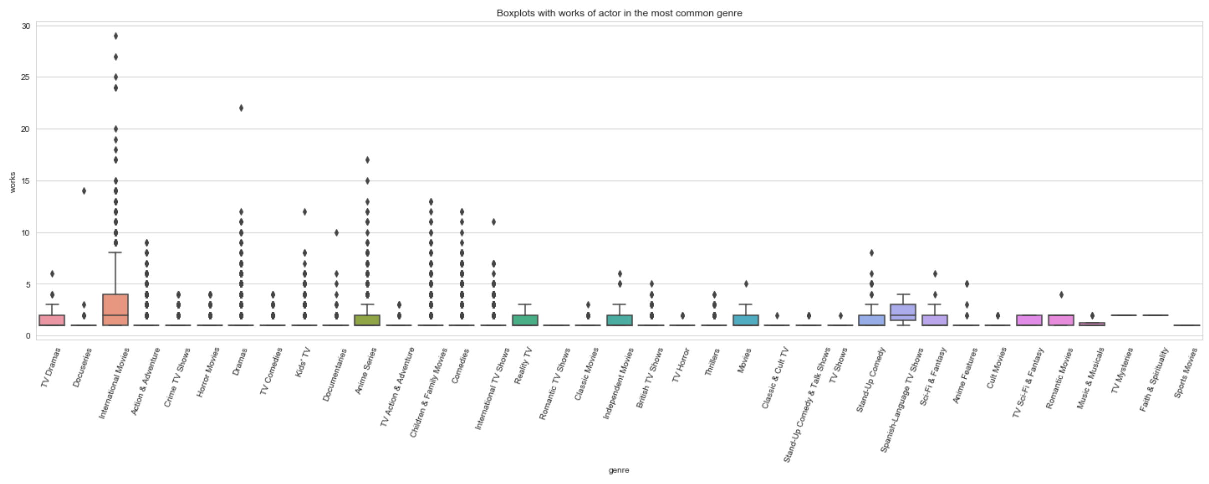
6.5 Visualization task5

In task5 data, we had 3 columns, actor and genre are nominal attributes. Works is a Ratio-scaled attribute, because of zero-point, values have order and measured on a scale of equal-sized units. We create a countplot with the genre attribute.



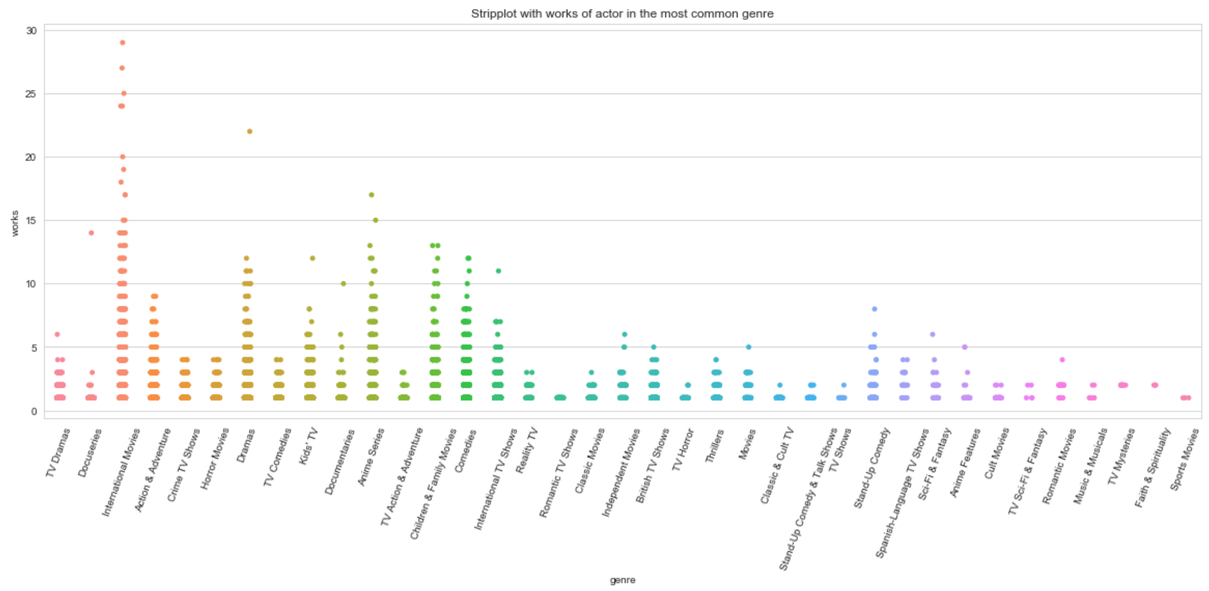
In this plot we can see that, Dramas are the most common genre for more than 5000 actors. Then, we have comedies with 4000 and around to 3000 are international tv shows and action & adventures. All the other genres are lower than 2000.

Then, we created a subplot with boxplot with works of actor in the most common genre of them.



We can see that international movies values are very sparse, and 15 values are over Q3 percentile, which is equal to 8 and extreme upper is equal to 30. Most of the genres have extreme upper values lower than 10 works per actor. Of course, the median in most of the genres are around to 1.

Finally, we created a strip-plot with the works of actor in the most common genre.



We can see that most values of each genre are lower than 5 and 6 genres have lots of values more than 5, which are international movies, action & adventure, dramas, anime series, children & family movies and comedies.

7. About tools that we use

For the implementation of the project, we first used the jupyter notebook to convert csv file to json file with python programming language. Then, after we did the conversion by running the code, we had to install mongodb. In macos operating system for the installation we used the instructions from the following link:
https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/#run-mongodb-community-edition

However, because at some point we encountered some problems we had to reinstall mongodb, after first uninstalling it with instructions from the following link:
https://medium.com/@rajanmaharjan/uninstall-mongodb-macos-completely-d2a6d6c163f9

We also installed the GUI for mongodb where it is called Mongodb Compass, in the following link we found the flow of the installation instructions:
https://docs.mongodb.com/compass/current/

For our preprocessing and visualizations, we used Jupyter Notebook and we code in Python. You can install it from the link of the organization, which is below:
https://jupyter.org/
or you can install Anaconda Navigator, that includes Jupyter Notebook from the link below:
https://www.anaconda.com/