Department of Computer Science

Big Data Management & Analysis (CSC4023Z) Assignment 1 2024

Group Name: TempName

Name	Student Number
Matthew Fleischman	FLSMAT002
Sam Frost	FRSSAM005
Ariel Levy	LVYARI002
Nicola Sartori	SRTNIC002
Thomas Schroeder	SCHTHO025

Question 1

Data Set Overview[1]

Name: Netflix Movies and TV Shows
Source: Kaggle
Creator: Shivam Bansal
Availability: Publicly Available
Date Accessed: 26 February 2024
File Type: .csv
File Size: 3.4 MB
Number of Values: 8807
Number of Columns: 12

Column Descriptions:

- *_id*: A String that serves as the unique identifier for each Netflix item. The format is an 's' followed by the row number.
- *type*: A String that stores either "Movie" or "TV Show" depending on how the item is displayed on Netflix.
- *title*: A String that stores the name of the Netflix item.
- *director*: A String List that contains the name(s) of the director(s) of the Netflix item (which is usually a movie)
- *cast*: A String List that names every member of the item's performing cast.
- *country*: A String List that lists the country/countries where the piece of content was made or filmed.
- *date_added*: A DateTime (ISO format) that stores the date that the item was placed on Netflix.
- *release_year*: An Integer that stores the year in which the first release of the Netflix item occurred.
- *rating*: A String that stores the age restriction that applies to the viewing of the Netflix item.
- *duration*: A String that specifies the length of a movie (in minutes) or the number of seasons a TV show has.
- *listed_in*: A String List that contains the genres that apply to the piece of content.
- *description*: A String that is a short narrative describing the Netflix item.

See the attached zip file *data.zip* for the dataset used: *netflix.csv*[1]. Otherwise, see the wiki on Vula.

[1] Bansal, S. (2021) Netflix Movies and TV Shows, Kaggle. Available at: https://www.kaggle.com/datasets/shivamb/netflix-shows (Accessed: 26 February 2024).

Question 2

Pre-processing and loading of the dataset

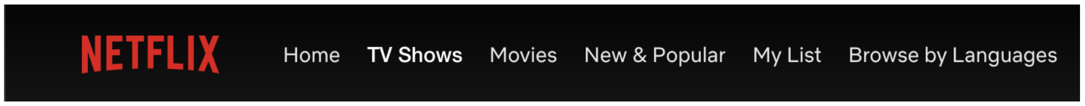
- The original *_id* field which was in the form *s{row_number}*, provided no value once the data was split into collections, as a result it was not loaded.
- A new *_id* field was created using the *type* field to create an incrementing identifier in the form *mov{count}* or *tv{count}* respectively. The values of these fields provide a means to identify tuples (whether a movie or show) with a useful collection-specific id value.
- All the entries of the *date_added* field were adjusted to comply with the standard ISO format. e.g. "September 25, 2021" was converted to "2021-09-25".
- *Cast, listed_in, country* and *director* fields, which were found to contain multiple values, were split on the comma delimiter and converted to lists to allow loading in accordance with the standard JSON format.

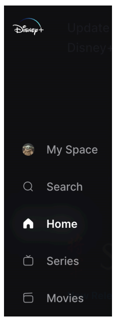
Loading the dataset using PyMongo and our Python program *loadDB.py* provides an easy way to pre-process the data before loading it to the database. Converting the dates to ISO format once loaded – while convenient for date operations – may prove a costly operation when the data is scaled up. Including unnecessary fields such as the original *_id* field would waste valuable storage.

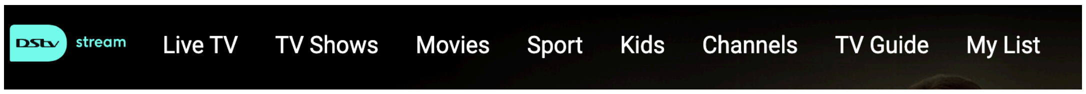
Collection Choice & Use Cases

Two collections were created using the new *_id* fields:
- The *Movies* collection containing all tuples with movies
- The *Shows* collection containing all tuples with TV shows

In the context of streaming platforms (Netflix, DSTV stream and Disney+), separating content into a movie and TV show category is standard practice (as can be seen in the examples provided below).



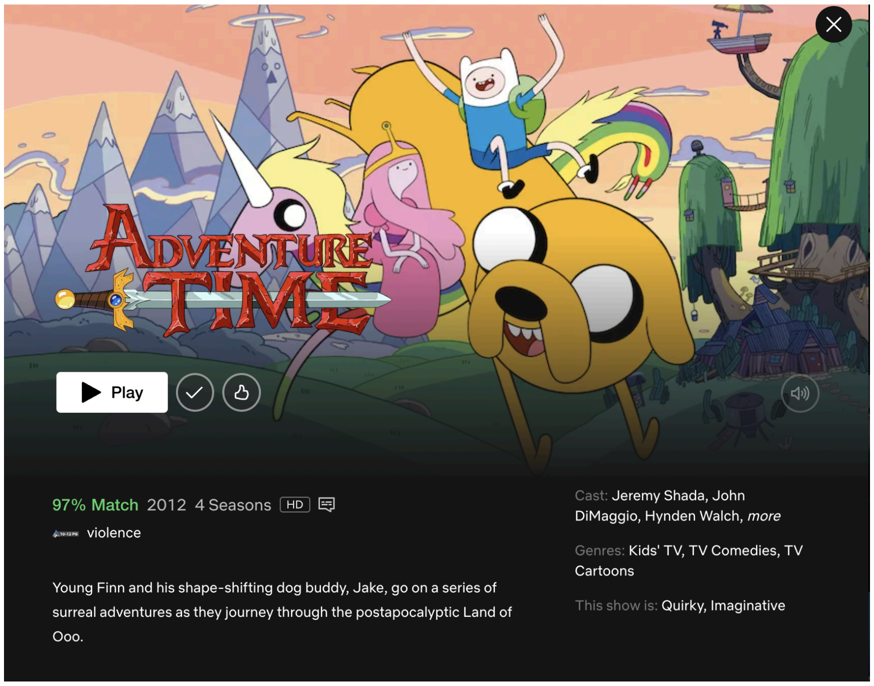




Having separate collections for this means that when a user navigates to one of these pages on a streaming platform, the platform only has to query a single collection instead of the entire dataset; which should help improve performance.

In practice, our dataset includes information that is primarily displayed on movie or show cards. This includes the movie or show name, cast, genres, description, release date, number of seasons or runtime and age rating; this information corresponds directly to the *title, cast, listed_in, description, release_year, duration* and *rating* fields in our dataset.

Example of a TV show card (Netflix):



97% Match 2012 4 Seasons HD

violence

Young Finn and his shape-shifting dog buddy, Jake, go on a series of surreal adventures as they journey through the postapocalyptic Land of Ooo.

Cast: Jeremy Shada, John DiMaggio, Hynden Walch, *more*

Genres: Kids' TV, TV Comedies, TV Cartoons

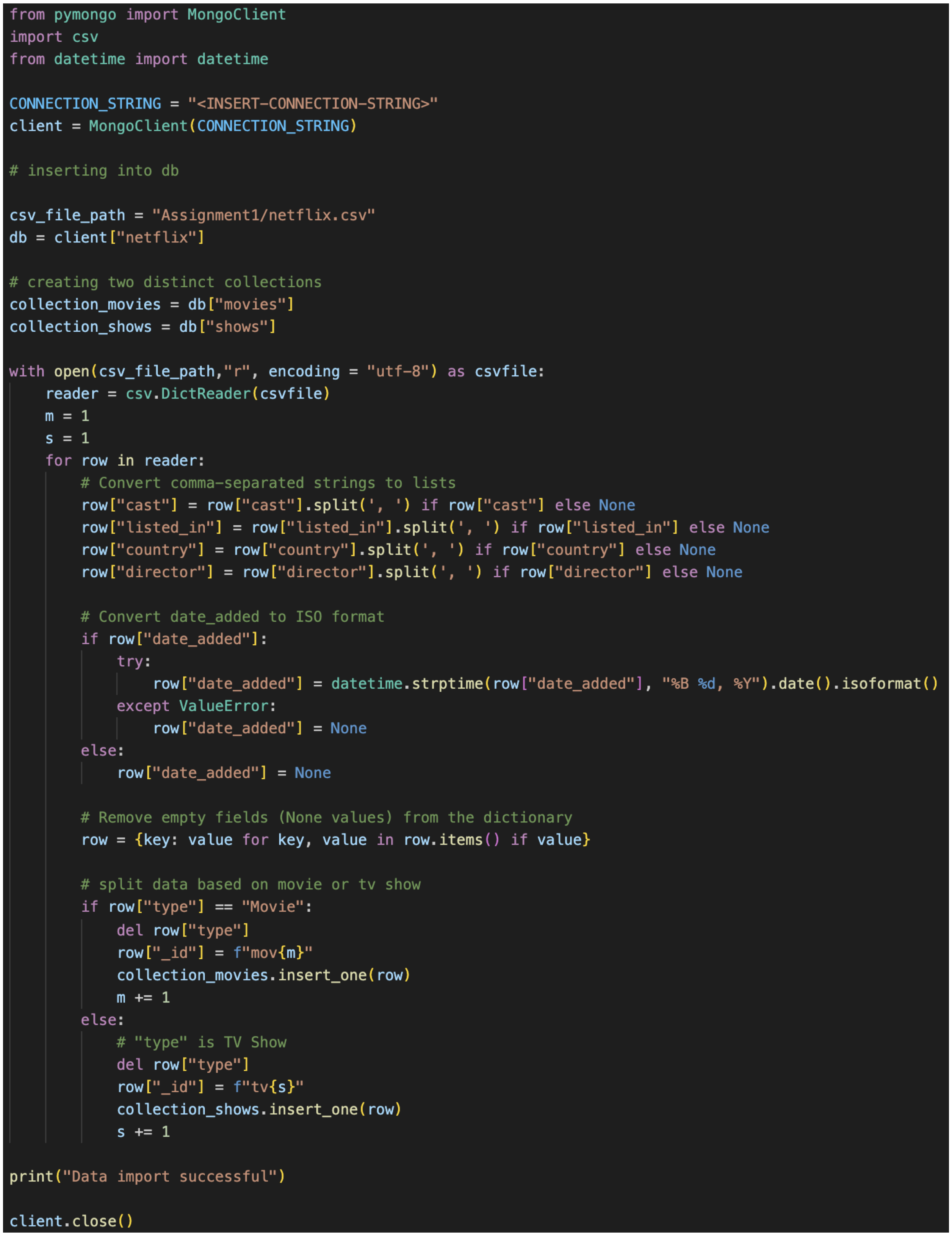
This show is: Quirky, Imaginative

Overall, the separation of the database into movies and shows collection mirrors how the user would primarily interact with the data on a streaming platform (looking for either shows or movies). In the context of big data, this would reduce the amount of data having to be queried by about 50%, significantly reducing query time.

<u>Question 3</u>

Loading the Database

Below is the source code *loadDB.py* used to populate the MongoDB database with our dataset.



```python
from pymongo import MongoClient
import csv
from datetime import datetime

CONNECTION_STRING = "<INSERT-CONNECTION-STRING>"
client = MongoClient(CONNECTION_STRING)

# inserting into db

csv_file_path = "Assignment1/netflix.csv"
db = client["netflix"]

# creating two distinct collections
collection_movies = db["movies"]
collection_shows = db["shows"]

with open(csv_file_path,"r", encoding = "utf-8") as csvfile:
    reader = csv.DictReader(csvfile)
    m = 1
    s = 1
    for row in reader:
        # Convert comma-separated strings to lists
        row["cast"] = row["cast"].split(', ') if row["cast"] else None
        row["listed_in"] = row["listed_in"].split(', ') if row["listed_in"] else None
        row["country"] = row["country"].split(', ') if row["country"] else None
        row["director"] = row["director"].split(', ') if row["director"] else None

        # Convert date_added to ISO format
        if row["date_added"]:
            try:
                row["date_added"] = datetime.strptime(row["date_added"], "%B %d, %Y").date().isoformat()
            except ValueError:
                row["date_added"] = None
        else:
            row["date_added"] = None

        # Remove empty fields (None values) from the dictionary
        row = {key: value for key, value in row.items() if value}

        # split data based on movie or tv show
        if row["type"] == "Movie":
            del row["type"]
            row["_id"] = f"mov{m}"
            collection_movies.insert_one(row)
            m += 1
        else:
            # "type" is TV Show
            del row["type"]
            row["_id"] = f"tv{s}"
            collection_shows.insert_one(row)
            s += 1

print("Data import successful")

client.close()
```

Testing

The following methods were used in order to verify and test if the dataset was populated successfully:

1. Using the l*oadDB.py* program with no runtime errors

```
"/Users/thomas/Library/CloudStorage/#/Other computers/My Computer/University/CSC
Honours/CSC4032Z - Big Data and Analytics/Assignment1/loadDB.py"
Data import successful
thomas@Thomass-MacBook-Pro ~ %
```

2. Running various shell commands

a. Database and collections are showing up and working

```
test> show dbs
admin       40.00 KiB
bookstore  104.00 KiB
config     108.00 KiB
local       72.00 KiB
netflix      2.95 MiB
test> use netflix
switched to db netflix
netflix> show collections
movies
shows
```

b. Documents of collection loaded correctly

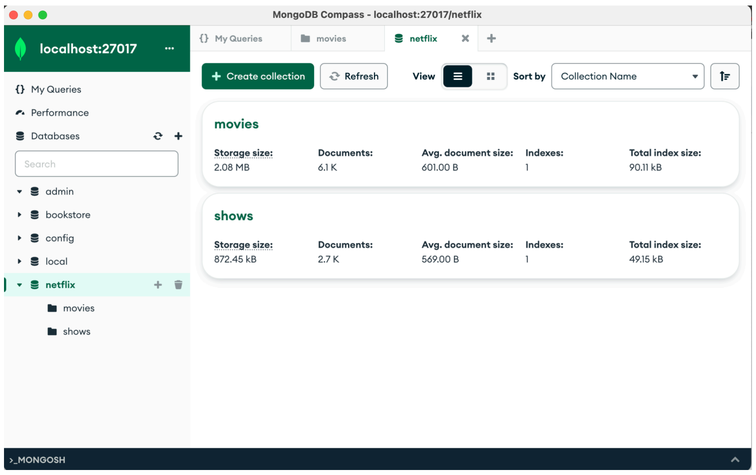
```
netflix> db.movies.find().limit(1)
[
  {
    _id: 'mov1',
    title: 'Dick Johnson Is Dead',
    director: [ 'Kirsten Johnson' ],
    country: [ 'United States' ],
    date_added: '2021-09-25',
    release_year: '2020',
    rating: 'PG-13',
    duration: '90 min',
    listed_in: [ 'Documentaries' ],
    description: 'As her father nears the end of his life, filmmaker Kirsten Johnson stages his death in
inventive and comical ways to help them both face the inevitable.'
  }
]
```

```
netflix> db.shows.find().limit(1)
 {
   _id: 'tv1',
   title: 'Blood & Water',
   cast: [
     'Ama Qamata',          'Khosi Ngema',
     'Gail Mabalane',       'Thabang Molaba',
     'Dillon Windvogel',    'Natasha Thahane',
     'Arno Greeff',         'Xolile Tshabalala',
     'Getmore Sithole',     'Cindy Mahlangu',
     'Ryle De Morny',       'Greteli Fincham',
     'Sello Maake Ka-Ncube', 'Odwa Gwanya',
     'Mekaila Mathys',      'Sandi Schultz',
     'Duane Williams',      'Shamilla Miller',
     'Patrick Mofokeng'
   ],
   country: [ 'South Africa' ],
   date_added: '2021-09-24',
   release_year: '2021',
   rating: 'TV-MA',
   duration: '2 Seasons',
   listed_in: [ 'International TV Shows', 'TV Dramas', 'TV Mysteries' ],
   description: 'After crossing paths at a party, a Cape Town teen sets out to prove whether a private-school
swimming star is her sister who was abducted at birth.'
 }
```

c. **Verifying that all the documents in the collections add up to 8807 (the number of rows from our dataset)**

```
netflix> db.shows.countDocuments()
2676
netflix> db.movies.countDocuments()
6131
ghci> 2676 + 6131
8807
```

3. Viewing database in MongoDB atlas



movies

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
2.08 MB	6.1 K	601.00 B	1	90.11 kB

shows

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
872.45 kB	2.7 K	569.00 B	1	49.15 kB

Question 4

1. Graph Databases

Benefits of MongoDB:

- Graph databases are highly complex, and their models can be intricate. MongoDB offers a simpler database to store the information relating to each title.
- Graph databases are not ideal for tabular or non-relationship-heavy data. The Netflix data has relatively few relationships.
- MongoDB will provide better horizontal scaling when more titles are added to the database.

Disadvantages of MongoDB:

- MongoDB is not relationship-centric like graph databases when dealing with complex data. If, in the future, more interrelated data types are added to the Netflix database, a graph structure may provide more functionality.
- Graph databases are more efficient at traversing interconnected data.
- A graph database offers a low response time when providing real-time analytics on related data.

Polyglot persistence scenario:

- If user data was added as a collection to the Netflix database, a graph structure could enhance Netflix's recommendation engine by determining user preferences through analysis of viewing patterns.

2. Key-Value Databases

Benefits of MongoDB:

- It provides the ability to store multiple data fields relating to each title.
- More complex queries can be made. This is important for our Netflix database as there are various fields that we would want to know information about simultaneously.
- Basic relationships between data entries can be stored.
- More flexibility and diversity in the stored data, such as nested documents.

Disadvantages of MongoDB:

- Key-value databases have better performance and scalability. They are also more suitable for data that does not have relationships.
- Data is often more easily partitioned and fragmented in Key-value databases.
- MongoDB databases are harder to interpret and navigate.
- Key-value databases have a caching layer, which improves read performance.

Polyglot persistence scenario:

- Key-value databases can serve as a caching layer for frequently accessed data, such as the description of a movie/series. The movie/series title would act as the key, and the description would act as the value. This will improve system performance.

3. Column-Family Databases

Benefits of MongoDB:

- It is more optimised for complex queries than a column-family database.
- When queries evolve, the schema does not have to drastically change.
- Has a more flexible schema.

Disadvantages of MongoDB:

- Column-family databases are better at allowing simple schema changes without significant cost. If a new data field is added (such as the net profit of the Netflix title) we simply introduce a new column. This will not affect existing data.
- Column-family databases are designed for fault tolerance. Essential data can be replicated across different nodes. This may also provide better consistency.
- In column-family databases, only the data we need (in the specific column) is fetched. This may provide better query performance relative to a MongoDB database.

Polyglot persistence scenario:

- Like the key-value scenario, frequently accessed data could be stored in a column-family database. This allows for faster query performance and can reduce overall disk I/O requirements.

4. Relational Databases

Benefits of MongoDB:

- Scalability, relational database management systems(RDMS) are typically designed for vertical scaling. To improve data or request capacity new or improved hardware will have to be utilised. MongoDB supports horizontal scaling whereby new nodes can be brought into the network to deal with increased demands. These new nodes will also be more cost effective compared to having to improve or upgrade existing hardware.
- Flexibility, schemas in relational databases have to be defined upfront and any changes made to the schema is complex and may require the database to go offline. MongoDB does not require a fixed schema as a document-store. The schema can be easily modified at any point to better suit the needs without requiring the system to go offline.

Disadvantages of MongoDB:

- Relational databases offer ACID transactions which guarantee consistency for every transaction, this will provide a greater degree of data consistency compared to MongoDB.
- Primary and foreign keys in relational databases ensure that there is no duplicated data improving data accuracy over MongoDB
- Normalisation of data in relational databases reduces the quantity of data that is required to be stored reducing the cost of storage.
- SQL, the primary query language for relational databases uses a syntax that is easily understandable with wide support, tools and resources online. MongoDB as the biggest NoSQL database has good support, but not at the same scale as relational databases.

Polyglot persistence scenario:

- A relational database offers the advantage of consistency, however for storing movie and show information consistency is not a strong requirement and eventual consistency is suitable. However, in the broader Netflix context, a relational database could be used to track the subscription status of users.

5. Hierarchical Databases

Benefits of MongoDB:

- Hierarchical databases typically rely on vertical scalability, which can be limiting in terms of storage capacity. Whereas, MongoDB excels at horizontal scalability through sharding data across multiple servers.
- Hierarchical databases (like XML) have a rigid structure defined by the schema, making it less adaptable to changes, unlike MongoDB.

Disadvantages of MongoDB:

- Hierarchical databases provide a predictable data structure and are efficient for data retrieval and storage.
- Handling complex hierarchical relationships is much easier when using a hierarchical database compared to MongoDB. However, there are few hierarchical relationships in our Netflix data.

Polyglot persistence scenario:

- There is no use of a hierarchical database for our Netflix data, as there are few hierarchical relationships.

Question 5

Ariel Levy (LVYARI002)

1. A query that finds the number of movies released in a given calendar year (e.g. 2020):

```
db.movies.countDocuments({ release_year: '2020'})

> 517
```

2. A query that shows all TV shows made in a specific country (e.g. South Africa) in alphabetical order:

```
db.shows.find(
        { country: 'South Africa' },
{ title: 1, _id: 0 }).sort({ title: 1 })

> { title: 'Agent' },
  { title: 'Blood & Water' },
  { title: 'Brave Miss World' },
  { title: 'Diamond City' },
  { title: 'How To Ruin Christmas' },
  { title: "Kings of Jo'Burg" },
  { title: 'Queen Sono' },
  { title: 'Shaka Zulu' },
  { title: 'The Indian Detective' },
  { title: 'Tjovitjo' },
  { title: 'Troy' }
```

3. A query that shows all Spanish language TV shows that have more than 2 seasons (only the first 5 entries are shown):

```
db.shows.find({
        listed_in: { $regex: /Spanish/i },
        duration: { $regex: /\b\d+\b/, $gte: '2' } },
{ _id : 0, title: 1, duration: 1 }).limit(5)

> { title: 'Falsa identidad', duration: '2 Seasons' },
  { title: 'La casa de papel', duration: '5 Seasons' },
  { title: 'Valeria', duration: '2 Seasons' },
  { title: 'Control Z', duration: '2 Seasons' },
  { title: 'Sky Rojo', duration: '2 Seasons' }
```

4. Update all children's movies released before 2000 with a PG rating to PG-13:

```
db.movies.updateMany({
        listed_in: { $regex: /child/i },
        release_year: { $lt: '2000' },
        rating: 'PG'},
{$set: { rating: 'PG-13' }})
```

<u>BEFORE</u>

```
> { title: 'Labyrinth', release_year: '1986', rating: 'PG' },
  { title: 'Beethoven', release_year: '1992', rating: 'PG' },
  { title: "Beethoven's 2nd", release_year: '1993', rating: 'PG' },
  { title: 'My Girl 2', release_year: '1994', rating: 'PG' },
  { title: 'Dennis the Menace', release_year: '1993', rating: 'PG' }
```

<u>AFTER</u>

```
> { title: 'Labyrinth', release_year: '1986', rating: 'PG-13' },
  { title: 'Beethoven', release_year: '1992', rating: 'PG-13' },
  { title: "Beethoven's 2nd", release_year: '1993', rating: 'PG-13' },
  { title: 'My Girl 2', release_year: '1994', rating: 'PG-13' },
  { title: 'Dennis the Menace', release_year: '1993', rating: 'PG-13' }
```

Thomas Schroeder (SCHTHO025)

1. Querying shows that Justin Roiland, Aaron Paul, Duncan Trussell, Bryan Cranston or Bob Odenkirk are cast members of. Returning the title, release year and the specific cast member who is featured and sorted by the release year:

```
db.shows.find({cast: {$in: ["Justin Roiland", "Aaron Paul", "Duncan Trussell", "Bryan Cranston",
"Bob Odenkirk"]}}, {_id: 0, title:1, release_year: 1, "cast.$": 1}).sort({release_year: 1})
```

```
> [
  {
        title: 'Breaking Bad',
        cast: [ 'Bryan Cranston' ],
        release_year: '2013'
  },
  {
        title: 'W/ Bob & David',
        cast: [ 'Bob Odenkirk' ],
        release_year: '2015'
  },
  {
        title: 'Better Call Saul',
        cast: [ 'Bob Odenkirk' ],
        release_year: '2018'
  },
  {
        title: 'The Midnight Gospel',
        cast: [ 'Duncan Trussell' ],
        release_year: '2020'
  },
  {
        title: 'BoJack Horseman',
        cast: [ 'Aaron Paul' ],
        release_year: '2020'
  },
  {
        title: 'Chicago Party Aunt',
        cast: [ 'Bob Odenkirk' ],
        release_year: '2021'
  }
]
```

2. Querying British adult (TV-MA or R) comedies that were released between 2020-2022. Returning the name, release date and description of the show and sorting by year and age rating for each year:

```
db.movies.find({country: "United Kingdom", release_year: {$gte: "2020"}, listed_in:
"Comedies", rating: {$in: ["TV-MA", "R"]}},{_id:0, title:1, description:1, release_year: 1, rating:
1})
```

```
> [
  {
    title: 'I Care a Lot',
    release_year: '2021',
    rating: 'R',
    description: 'A court-appointed legal guardian defrauds her older clients and traps them under
her care. But her latest mark comes with some unexpected baggage.'
  },
  {
    title: 'Love Wedding Repeat',
    release_year: '2020',
    rating: 'TV-MA',
    description: "Different versions of the same day unfold as Jack juggles difficult guests,
unbridled chaos and potential romance at his sister's wedding."
  }
]
```

3. Updating the movies collection by translating the equivalent 18+ United States age ratings to South African to FPB (Film and Publication Board) ratings (i.e. TV-MA → 18 and NC-17 → 18):

```
db.movies.updateMany({rating: {$in: ["TV-MA", "NC-17"]}}, {$set: {rating: "18"}})
```

```
> {
  acknowledged: true,
  insertedId: null,
  matchedCount: 2065,
  modifiedCount: 2065,
  upsertedCount: 0
}
```

4. Deleting South African shows that were released before 2000:

```
db.shows.deleteMany({country: "South Africa", release_year: {$gte: "2000"}})
```

```
> { acknowledged: true, deletedCount: 10 }
```

Matthew Fleischman (FLSMAT002)

1. Find the titles of movies made during 2017 and 2018 (the last 6 outputs are shown):

```
db.movies.find({ $and: [{ release_year: { $gt:"2016" }}, { release_year: { $lt:"2019" } }] }, {
release_year: 1, title: 1 })
```

```
> { _id: 'mov236', title: 'The Lost Café', release_year: '2018' },
  { _id: 'mov291', title: 'The Beguiled', release_year: '2017' },
  { _id: 'mov292', title: 'The Book of Henry', release_year: '2017' },
  { _id: 'mov324', title: 'Back to Q82', release_year: '2017' },
  { _id: 'mov325', title: 'Home Again', release_year: '2017' },
  { _id: 'mov326', title: 'Midnight Sun', release_year: '2018' }
```

2. Search for movies with "adventure" in the title or description, and limit the results to 3 records:

```
db.movies.find({ $or: [{ title: /adventure/i }, { description: /adventure/i }] }).limit(3)
```

```
> {
    _id: 'mov10',
    title: 'Go! Go! Cory Carson: Chrissy Takes the Wheel',
    director: [ 'Alex Woo', 'Stanley Moore' ],
    cast: [ 'Maisie Benson', 'Paul Killam', 'Kerry Gudjohnsen', 'AC Lim' ],
    date_added: '2021-09-21',
    release_year: '2021',
    rating: 'TV-Y',
    duration: '61 min',
    listed_in: [ 'Children & Family Movies' ],
    description: 'From arcade games to sled days and hiccup cures, Cory Carson's curious little
sister Chrissy speeds off on her own for fun and adventure all over town!'
  },
  {
    _id: 'mov34',
    title: 'Naruto Shippuden: The Movie',
    director: [ 'Hajime Kamegaki' ],
    cast: [
      'Junko Takeuchi',      'Chie Nakamura',
      'Yoichi Masukawa',     'Koichi Tochika',
      'Ayumi Fujimura',      'Keisuke Oda',
      'Daisuke Kishio',      'Fumiko Orikasa',
      'Hidetoshi Nakamura', 'Tetsuya Kakihara',
      'Kisho Taniyama',      'Miyuki Sawashiro',
      'Katsuyuki Konishi',  'Masako Katsuki',
      'Keiko Nemoto',        'Masashi Ebara',
      'Kazuhiko Inoue',     'Showtaro Morikubo',
      'Romi Park',          'Daisuke Ono',
      'Seizo Kato'
    ],
    country: [ 'Japan' ],
    date_added: '2021-09-15',
    release_year: '2007',
    rating: 'TV-PG',
    duration: '95 min',
    listed_in: [ 'Action & Adventure', 'Anime Features', 'International Movies' ],
    description: "The adventures of adolescent ninja Naruto Uzumaki continue as he's tasked with
protecting a priestess from a demon – but to do so, he must die."
  },
  {
    _id: 'mov41',
    title: 'A StoryBots Space Adventure',
    director: [ 'David A. Vargas' ],
    cast: [
      'Evan Spiridellis',
      'Erin Fitzgerald',
      'Jeff Gill',
      'Fred Tatasciore',
      'Evan Michael Lee',
      'Jared Isaacman',
      'Sian Proctor',
      'Chris Sembroski',
      'Hayley Arceneaux'
    ],
    date_added: '2021-09-14',
    release_year: '2021',
    rating: 'TV-Y',
    duration: '13 min',
    listed_in: [ 'Children & Family Movies' ],
    description: "Join the StoryBots and the space travellers of the historic Inspiration4 mission as
they search for answers to kids' questions about space."
  }
```

3. Find Indian shows, and update their listed_in field to include 'Bollywood' (the first 3 outputs are shown):

```
db.shows.updateMany({ country: "India" }, { $addToSet: { listed_in: "Bollywood" } })
```

BEFORE:

```
> {
   _id: 'tv4',
   listed_in: [ 'International TV Shows', 'Romantic TV Shows, 'TV Comedies' ]
},
 {
   _id: 'tv21',
   listed_in: [ 'Kids' TV' ] },
 {
   _id: 'tv25',
   listed_in: [ 'International TV Shows', 'TV Dramas', 'TV Sci-Fi & Fantasy' ] }
```

AFTER:

```
> {
   _id: 'tv4',
   listed_in: [ 'International TV Shows', 'Romantic TV Shows, 'TV Comedies' , 'Bollywood' ] },
 {
   _id: 'tv21',
   listed_in: [ 'Kids' TV', 'Bollywood' ] },
 {
   _id: 'tv25',
   listed_in: [ 'International TV Shows', 'TV Dramas', 'TV Sci-Fi & Fantasy' , 'Bollywood' ] }
```

4. Count the number of Movies that Jack Black acted in during 2006:

```
db.movies.countDocuments({ $and: [{ cast: "Jack Black" }, { release_year: "2006" }] })

> 2
```

Sam Frost (FRSSAM005)

1. Find TV Shows released in 2017 in South Africa:

```
db.shows.find({ release_year: 2017, country: 'South Africa'})
```

```
> {
    _id: 's2612',
    type: 'TV Show',
    title: 'Tjovitjo',
    cast: 'Warren Masemola, Nobulali Dangazele, Rapulana Seiphemo, Sibulele Gcilitshana,
Jabulile Mahlambi, Ntosh Madlingozi, Hlengiwe Lushaba, Kanyi Nokwe, Harriet Manamela,
Soso Rungqu, Fisiwe Kubeka, Victor Mohale',
    country: 'South Africa',
    date_added: 'April 29, 2020',
    release_year: 2017,
    rating: 'TV-MA',
    duration: '2 Seasons',
    listed_in: 'International TV Shows, TV Dramas',
    description: 'Amidst poverty and struggle, a hardened pantsula dance leader enters a dark
space and searches for redemption and salvation in his community.'
  }
```

2. Find Movies released before 1950 and display only their titles and director's name(s):

```
db.movies.find({release_year: {$lt:1950}},{title:true, director: true, _id: false})
```

```
>  { title: 'Know Your Enemy - Japan',  director: 'Frank Capra, Joris Ivens' },
  { title: 'Let There Be Light', director: 'John Huston' },
  { title: 'Nazi Concentration Camps', director: 'George Stevens' },
  { title: 'Prelude to War', director: 'Frank Capra' },
  { title: 'San Pietro', director: 'John Huston' },
  { title: 'The Battle of Midway', director: 'John Ford' },
  { title: 'The Memphis Belle: A Story of a\\nFlying Fortress', director: 'William Wyler'},
  { title: 'The Negro Soldier', director: 'Stuart Heisler' },
  { title: 'Thunderbolt', director: 'William Wyler, John Sturges' },
  { title: 'Tunisian Victory', director: 'Frank Capra, John Huston, Hugh Stewart, Roy Boulting,
Anthony Veiller' },
  { title: 'Undercover: How to Operate Behind Enemy Lines', director: 'John Ford' },
  { title: 'Why We Fight: The Battle of Russia', director: 'Frank Capra, Anatole Litvak' },
  { title: 'WWII: Report from the Aleutians', director: 'John Huston' }
```

3. Delete all movies with the word 'Jaws' in its title:

```
db.movies.deleteMany({title: {$regex: 'Jaws'}})

> { acknowledged: true, deletedCount: 4 }
```

4. Update all the TV Shows with the word 'sex' in their description to a rating of 'TV-MA' (which is the correct adult designation) only the title, rating, and description are shown:

```
db.movies.updateMany({description: {$regex: 'sex'}}, {$set: {rating: 'TV-MA'}})
```

<u>BEFORE (the last 3 outputs):</u>

```
> {
   title: 'Lost Girl',
   rating: 'TV-14',
   description: "Discovering she's a succubus who sustains herself by feeding on the sexual
energy of humans, seductive Bo sets out on a journey to understand herself."
 },
 {
   title: 'My Hotter Half',
   rating: 'TV-PG',
   description: 'Couples compete to see who can take the sexiest selfie, with the loser getting a
much-needed makeover.'
 },
 {
   title: 'Queens vs. Kings',
   rating: 'TV-14',
   description: 'No "ism" is left unturned in this celebrity game show that battles stereotypes and
sexism by pitting male and female comedians against each other.'
 }
```

<u>AFTER (the last 3 outputs):</u>

```
> {
   title: 'Lost Girl',
   rating: 'TV-MA',
   description: "Discovering she's a succubus who sustains herself by feeding on the sexual
energy of humans, seductive Bo sets out on a journey to understand herself."
 },
 {
   title: 'My Hotter Half',
   rating: 'TV-MA',
   description: 'Couples compete to see who can take the sexiest selfie, with the loser getting a
much-needed makeover.'
 },
 {
   title: 'Queens vs. Kings',
   rating: 'TV-MA',
   description: 'No "ism" is left unturned in this celebrity game show that battles stereotypes and
sexism by pitting male and female comedians against each other.'
 }
```

Nicola Sartori (SRTNIC002)

1. **Count the number of shows in each ratings category and present the top 3. Only the category and total are shown:**

```
db.shows.aggregate( [ {
        $group: {
                _id: "$rating",
                total: { $sum: 1 }
                 }
        },{ $sort: { total: -1 } },{$limit: 3}] )
```

```
> { _id: 'TV-MA', total: 1024 },
  { _id: 'TV-14', total: 666 },
  { _id: 'TV-PG', total: 262 }
```

2. **Find the movie with the longest runtime for each genre in the listed_in field. Only the genre, title and runtime are shown (the time is in minutes) – only the first 5 results are shown.**

```
convertToIntStage = { $addFields:{
                convertedDuration : {$toInt:  { $trim: { input: "$duration",  chars: " min"  } }}}}

db.movies.aggregate([
        { $unwind: "$listed_in" },
        convertToIntStage,
        {$sort:{ convertedDuration: -1 }},
                {$group: {
                        _id: "$listed_in",
                         longestRunTime: { $max: "$convertedDuration" },
                         title:{$first:"$title"}, } },
                { $sort: { longestRunTime: -1 }  },{
$project: {_id:1,title:1,longestRunTime : 1}} ])
```

```
> {
    _id: 'Dramas',
    longestRunTime: 312,
    title: 'Black Mirror: Bandersnatch'
  },
  {
    _id: 'Sci-Fi & Fantasy',
    longestRunTime: 312,
    title: 'Black Mirror: Bandersnatch'
  },
  {
    _id: 'International Movies',
    longestRunTime: 312,
    title: 'Black Mirror: Bandersnatch'
  },,
  {
    _id: 'Documentaries',
    longestRunTime: 273,
    title: 'Headspace: Unwind Your Mind'
  },
  {
    _id: 'Comedies',
    longestRunTime: 253,
    title: 'The School of Mischief'
  },
  {
    _id: 'Romantic Movies',
    longestRunTime: 233,
    title: 'Lock Your Girls In'
  },
  {
    _id: 'Classic Movies',
    longestRunTime: 229,
    title: 'Once Upon a Time in America'
  },
  { _id: 'Music & Musicals', longestRunTime: 224, title: 'Lagaan' },
  {
    _id: 'Action & Adventure',
    longestRunTime: 214,
    title: 'Jodhaa Akbar'
  },
  {
    _id: 'Faith & Spirituality',
    longestRunTime: 205,
    title: 'The Gospel of Luke'
  },
  {
    _id: 'Independent Movies',
    longestRunTime: 189,
    title: 'Magnolia'
  },
  { _id: 'Cult Movies', longestRunTime: 172, title: 'Cloud Atlas' },
  { _id: 'Thrillers', longestRunTime: 171, title: 'Andhaghaaram' },
  {
    _id: 'Horror Movies',
    longestRunTime: 171,
    title: 'Andhaghaaram'
  },
  { _id: 'Sports Movies', longestRunTime: 161, title: 'Dangal' },
  {
    _id: 'Children & Family Movies',
    longestRunTime: 152,
    title: 'Star Wars: Episode VIII: The Last Jedi'
  },
  {
    _id: 'Stand-Up Comedy',
    longestRunTime: 146,
    title: 'Tim Minchin And The Heritage Orchestra Live'
  },
  { _id: 'LGBTQ Movies', longestRunTime: 143, title: 'Ride or Die' },
  {
    _id: 'Anime Features',
    longestRunTime: 140,
    title: 'Mobile Suit Gundam III: Encounters in Space'
  },
  {
    _id: 'Movies',
    longestRunTime: 115,
    title: 'American Masters: Inventing David Geffen'
  }
```

3. Delete all shows that are from India or the United Kingdom that have more than one season.

```
                db.shows.deleteMany({ $and:
                        [{ country: {$in:["India","United Kingdom"]} },
                        { duration: { $nin: [   "1 Seasons" ]  }}]
                        })
```

```
        > { acknowledged: true, deletedCount: 56 }
```

4. Find the movie "Naruto Shippûden the Movie: The Will of Fire" and set it to be from China by changing the country field value.

```
        db.movies.findAndModify({
                query: { title:  "Naruto Shippûden the Movie: The Will of Fire" },
                update: { $set: { country: "china" } } })
```

<u>BEFORE:</u>

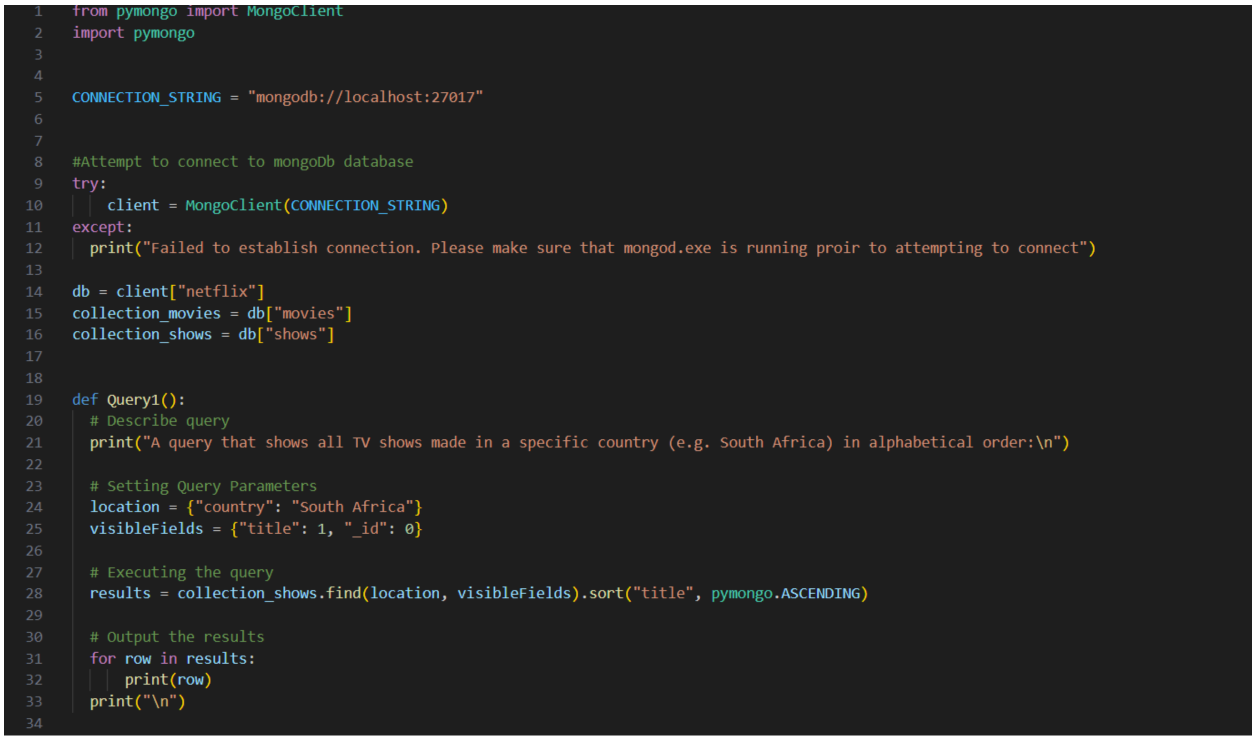
```
        > {
         _id: 'mov33',
         title: 'Naruto Shippûden the Movie: The Will of Fire',
         director: [ 'Masahiko Murata' ],
         cast: [
           'Junko Takeuchi',
           'Chie Nakamura',
           'Kazuhiko Inoue',
           'Satoshi Hino',
           'Showtaro Morikubo',
           'Kentaro Ito',
           'Ryoka Yuzuki',
           'Kohsuke Toriumi',
           'Nana Mizuki',
           'Shinji Kawada',
           'Yoichi Masukawa',
           'Koichi Tochika',
           'Yukari Tamura'
         ],
         country: 'japan',
         date_added: '2021-09-15',
         release_year: '2009',
         rating: 'TV-PG',
         duration: '96 min',
         listed_in: [ 'Action & Adventure', 'Anime Features', 'International Movies' ],
  description: 'When four out of five ninja villages are destroyed, the leader of the one spared tries to find the true culprit and protect his land.'
}
```

After:

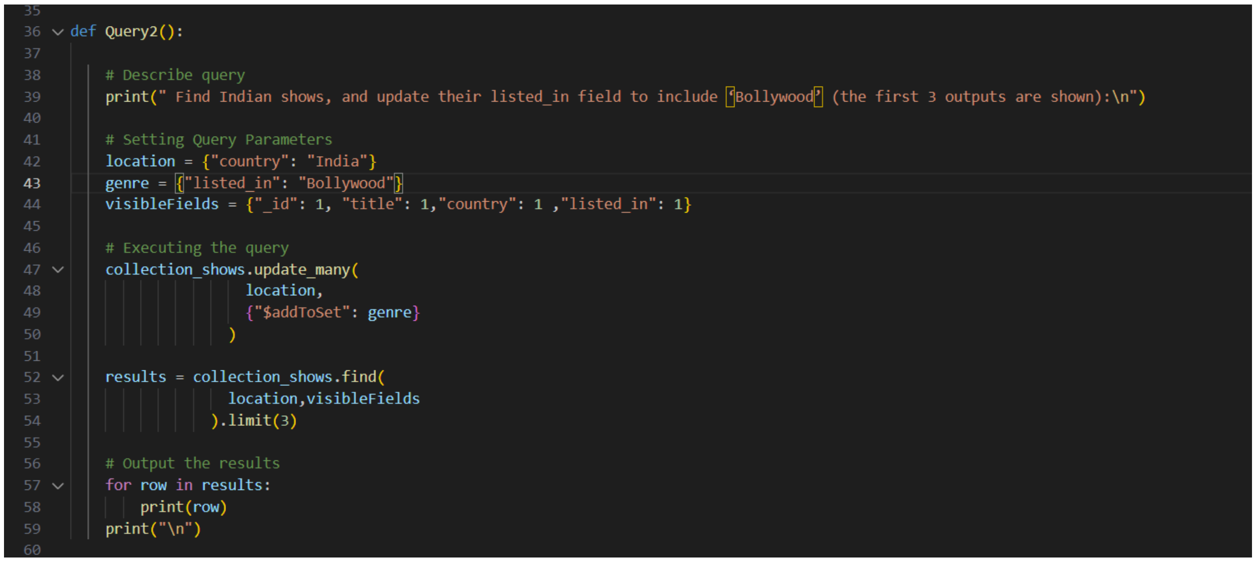
```
> {
  _id: 'mov33',
  title: 'Naruto Shippûden the Movie: The Will of Fire',
  director: [ 'Masahiko Murata' ],
  cast: [
    'Junko Takeuchi',
    'Chie Nakamura',
    'Kazuhiko Inoue',
    'Satoshi Hino',
    'Showtaro Morikubo',
    'Kentaro Ito',
    'Ryoka Yuzuki',
    'Kohsuke Toriumi',
    'Nana Mizuki',
    'Shinji Kawada',
    'Yoichi Masukawa',
    'Koichi Tochika',
    'Yukari Tamura'
  ],
  country: 'china',
  date_added: '2021-09-15',
  release_year: '2009',
  rating: 'TV-PG',
  duration: '96 min',
  listed_in: [ 'Action & Adventure', 'Anime Features', 'International Movies' ],
  description: 'When four out of five ninja villages are destroyed, the leader of the one spared tries to find the true culprit and protect his land.'
}
```

Question 6

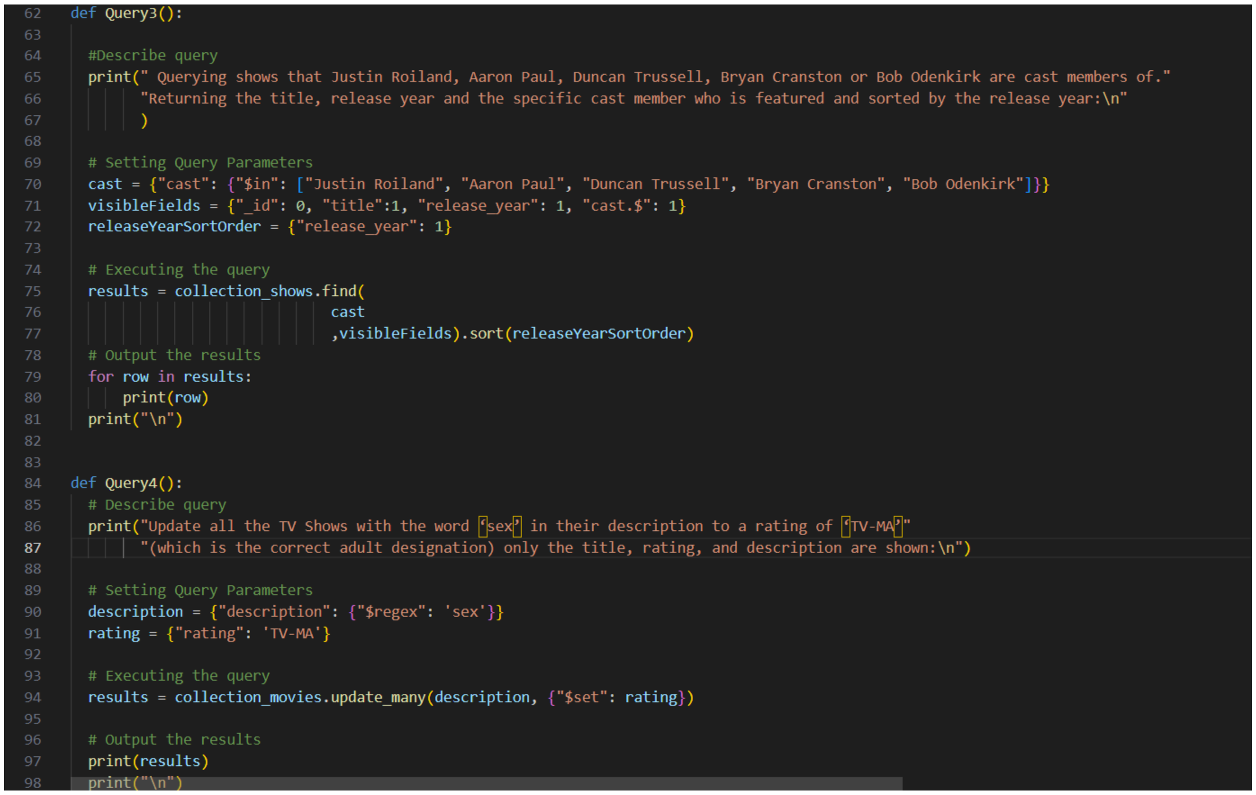
Below is the source code *QueryProgram.py* used to run 5 queries on the database and print the results to the terminal.



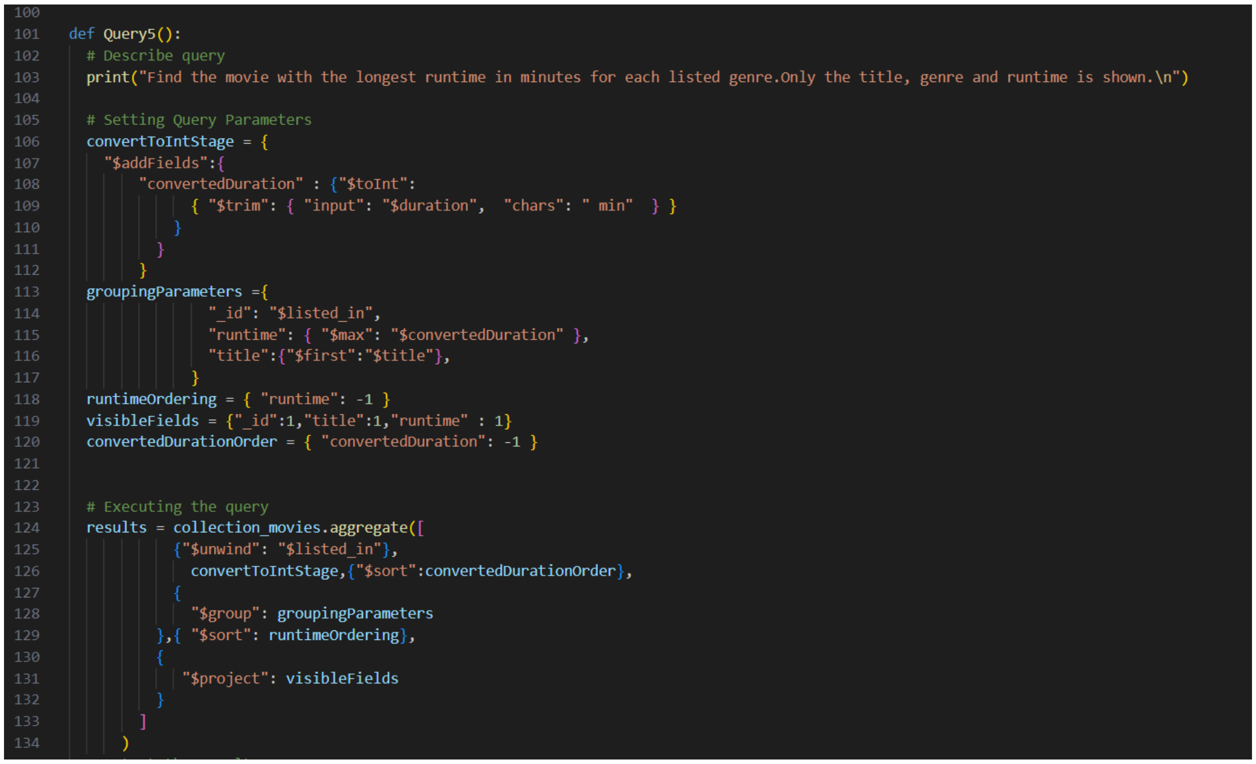
```python
from pymongo import MongoClient
import pymongo


CONNECTION_STRING = "mongodb://localhost:27017"


#Attempt to connect to mongoDb database
try:
    client = MongoClient(CONNECTION_STRING)
except:
  print("Failed to establish connection. Please make sure that mongod.exe is running proir to attempting to connect")

db = client["netflix"]
collection_movies = db["movies"]
collection_shows = db["shows"]


def Query1():
  # Describe query
  print("A query that shows all TV shows made in a specific country (e.g. South Africa) in alphabetical order:\n")

  # Setting Query Parameters
  location = {"country": "South Africa"}
  visibleFields = {"title": 1, "_id": 0}

  # Executing the query
  results = collection_shows.find(location, visibleFields).sort("title", pymongo.ASCENDING)

  # Output the results
  for row in results:
      print(row)
  print("\n")

```



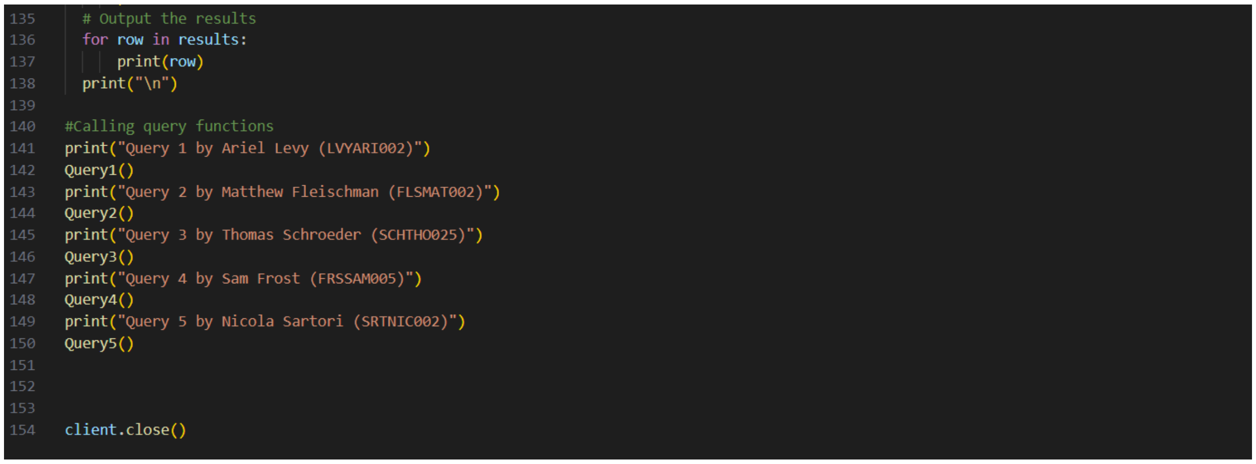
```python

def Query2():

    # Describe query
    print(" Find Indian shows, and update their listed_in field to include 'Bollywood' (the first 3 outputs are shown):\n")

    # Setting Query Parameters
    location = {"country": "India"}
    genre = {"listed_in": "Bollywood"}
    visibleFields = {"_id": 1, "title": 1,"country": 1 ,"listed_in": 1}

    # Executing the query
    collection_shows.update_many(
                    location,
                    {"$addToSet": genre}
                  )

    results = collection_shows.find(
                  location,visibleFields
                ).limit(3)

    # Output the results
    for row in results:
        print(row)
    print("\n")

```



```python
def Query3():

    #Describe query
    print(" Querying shows that Justin Roiland, Aaron Paul, Duncan Trussell, Bryan Cranston or Bob Odenkirk are cast members of."
          "Returning the title, release year and the specific cast member who is featured and sorted by the release year:\n"
          )

    # Setting Query Parameters
    cast = {"cast": {"$in": ["Justin Roiland", "Aaron Paul", "Duncan Trussell", "Bryan Cranston", "Bob Odenkirk"]}}
    visibleFields = {"_id": 0, "title":1, "release_year": 1, "cast.$": 1}
    releaseYearSortOrder = {"release_year": 1}

    # Executing the query
    results = collection_shows.find(
                                    cast
                                    ,visibleFields).sort(releaseYearSortOrder)
    # Output the results
    for row in results:
        print(row)
    print("\n")


def Query4():
    # Describe query
    print("Update all the TV Shows with the word ‘sex’ in their description to a rating of ‘TV-MA’"
          "(which is the correct adult designation) only the title, rating, and description are shown:\n")

    # Setting Query Parameters
    description = {"description": {"$regex": 'sex'}}
    rating = {"rating": 'TV-MA'}

    # Executing the query
    results = collection_movies.update_many(description, {"$set": rating})

    # Output the results
    print(results)
    print("\n")
```



```python

def Query5():
    # Describe query
    print("Find the movie with the longest runtime in minutes for each listed genre.Only the title, genre and runtime is shown.\n")

    # Setting Query Parameters
    convertToIntStage = {
      "$addFields":{
          "convertedDuration" : {"$toInt":
                { "$trim": { "input": "$duration",  "chars": " min"  } }
              }
            }
          }
    groupingParameters ={
                "_id": "$listed_in",
                "runtime": { "$max": "$convertedDuration" },
                "title":{"$first":"$title"},
              }
    runtimeOrdering = { "runtime": -1 }
    visibleFields = {"_id":1,"title":1,"runtime" : 1}
    convertedDurationOrder = { "convertedDuration": -1 }


    # Executing the query
    results = collection_movies.aggregate([
              {"$unwind": "$listed_in"},
                convertToIntStage,{"$sort":convertedDurationOrder},
              {
                "$group": groupingParameters
            },{ "$sort": runtimeOrdering},
            {
              "$project": visibleFields
            }
          ]
        )
```



```
    # Output the results
    for row in results:
        print(row)
    print("\n")

#Calling query functions
print("Query 1 by Ariel Levy (LVYARI002)")
Query1()
print("Query 2 by Matthew Fleischman (FLSMAT002)")
Query2()
print("Query 3 by Thomas Schroeder (SCHTHO025)")
Query3()
print("Query 4 by Sam Frost (FRSSAM005)")
Query4()
print("Query 5 by Nicola Sartori (SRTNIC002)")
Query5()



client.close()
```

Below are the results (these match the respective results generated in Q5):

> Query 1 by Ariel Levy (LVYARI002)
A query that shows all TV shows made in a specific country (e.g. South Africa) in alphabetical order:

```
{'title': 'Agent'}
{'title': 'Blood & Water'}
{'title': 'Brave Miss World'}
{'title': 'Diamond City'}
{'title': 'How To Ruin Christmas'}
{'title': "Kings of Jo'Burg"}
{'title': 'Queen Sono'}
{'title': 'Shaka Zulu'}
{'title': 'The Indian Detective'}
{'title': 'Tjovitjo'}
```

> Query 2 by Matthew Fleischman (FLSMAT002)
Find Indian shows, and update their listed_in field to include 'Bollywood' (the first 3 outputs are shown):

```
{'_id': 'tv4', 'title': 'Kota Factory', 'country': ['India'], 'listed_in': ['International TV Shows', 'Romantic TV Shows', 'TV Comedies', 'Bollywood']}
{'_id': 'tv21', 'title': 'Chhota Bheem', 'country': ['India'], 'listed_in': ["Kids' TV", 'Bollywood']}
{'_id': 'tv25', 'title': 'Dharmakshetra', 'country': ['India'], 'listed_in': ['International TV Shows', 'TV Dramas', 'TV Sci-Fi & Fantasy', 'Bollywood']}
```

> Query 3 by Thomas Schroeder (SCHTHO025)

Querying shows that Justin Roiland, Aaron Paul, Duncan Trussell, Bryan Cranston or Bob Odenkirk are cast members of. Returning the title, release year and the specific cast member who is featured and sorted by the release year:

```
{'title': 'Breaking Bad', 'cast': ['Bryan Cranston'], 'release_year': '2013'}
{'title': 'W/ Bob & David', 'cast': ['Bob Odenkirk'], 'release_year': '2015'}
{'title': 'Better Call Saul', 'cast': ['Bob Odenkirk'], 'release_year': '2018'}
{'title': 'The Midnight Gospel', 'cast': ['Duncan Trussell'], 'release_year': '2020'}
{'title': 'BoJack Horseman', 'cast': ['Aaron Paul'], 'release_year': '2020'}
{'title': 'Chicago Party Aunt', 'cast': ['Bob Odenkirk'], 'release_year': '2021'}
```

> Query 4 by Sam Frost (FRSSAM005)
Update all the TV Shows with the word 'sex' in their description to a rating of 'TV-MA' (which is the correct adult designation) only the title, rating, and description are shown:

```
UpdateResult({'n': 126, 'nModified': 0, 'ok': 1.0, 'updatedExisting': True}, acknowledged=True)
```

> Query 5 by Nicola Sartori (SRTNIC002)
Find the movie with the longest runtime in minutes for each listed genre.Only the title, genre and runtime are shown:

```
{'_id': 'Dramas', 'runtime': 312, 'title': 'Black Mirror: Bandersnatch'}
{'_id': 'International Movies', 'runtime': 312, 'title': 'Black Mirror: Bandersnatch'}
{'_id': 'Sci-Fi & Fantasy', 'runtime': 312, 'title': 'Black Mirror: Bandersnatch'}
{'_id': 'Documentaries', 'runtime': 273, 'title': 'Headspace: Unwind Your Mind'}
{'_id': 'Comedies', 'runtime': 253, 'title': 'The School of Mischief'}
{'_id': 'Romantic Movies', 'runtime': 233, 'title': 'Lock Your Girls In'}
{'_id': 'Classic Movies', 'runtime': 229, 'title': 'Once Upon a Time in America'}
{'_id': 'Music & Musicals', 'runtime': 224, 'title': 'Lagaan'}
{'_id': 'Action & Adventure', 'runtime': 214, 'title': 'Jodhaa Akbar'}
{'_id': 'Faith & Spirituality', 'runtime': 205, 'title': 'The Gospel of Luke'}
{'_id': 'Independent Movies', 'runtime': 189, 'title': 'Magnolia'}
{'_id': 'Cult Movies', 'runtime': 172, 'title': 'Cloud Atlas'}
{'_id': 'Thrillers', 'runtime': 171, 'title': 'Andhaghaaram'}
{'_id': 'Horror Movies', 'runtime': 171, 'title': 'Andhaghaaram'}
{'_id': 'Sports Movies', 'runtime': 161, 'title': 'Dangal'}
{'_id': 'Children & Family Movies', 'runtime': 152, 'title': 'Star Wars: Episode VIII: The Last Jedi'}
{'_id': 'Stand-Up Comedy', 'runtime': 146, 'title': 'Tim Minchin And The Heritage Orchestra Live'}
{'_id': 'LGBTQ Movies', 'runtime': 143, 'title': 'Ride or Die'}
{'_id': 'Anime Features', 'runtime': 140, 'title': 'Mobile Suit Gundam III: Encounters in Space'}
{'_id': 'Movies', 'runtime': 115, 'title': 'American Masters: Inventing David Geffen'}
```

Question 7

Name - Student Number	**Contribution**
Matthew Fleischman - FLSMAT002	Q1 - Helped to decide on *netflix.csv* dataset Q4 - Key-value databases & formatted and improved coherency of answers. Q5 - 4 Queries Q6 - Single Python query
Sam Frost - FRSSAM005	Q1 - Dataset overview and column description Q4 - Graph databases Q5 - 4 Queries Q6 - Single Python query General - Formatting the document
Ariel Levy - LVYARI002	Q1 - Finding the *netflix.csv* dataset Q3 - Creating l*oadDB.py* program Q4 - Hierarchical databases Q5 - 4 Queries Q6 - Single Python query
Nicola Sartori - SRTNIC002	Q4 - Column-Family databases Q5 - 4 Queries Q6 - Single Python query & creating program to run selected queries
Thomas Schroeder - SCHTHO025	Q2 - Design motivation explanation Q3 - Database tests Q4 - Relational databases Q5 - 4 Queries Q6 - Single Python query General - Formatting the document