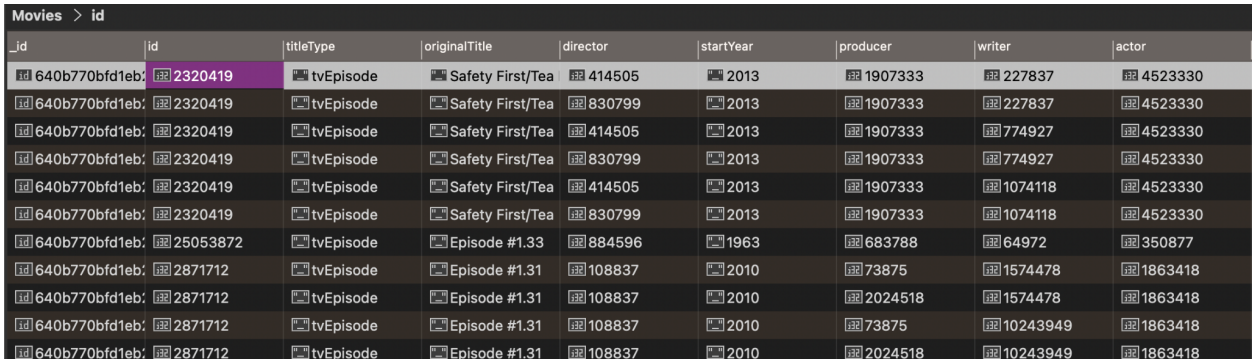
ASSIGNMENT 4

Anikhet Mulky
am9559@g.rit.edu

Q1)

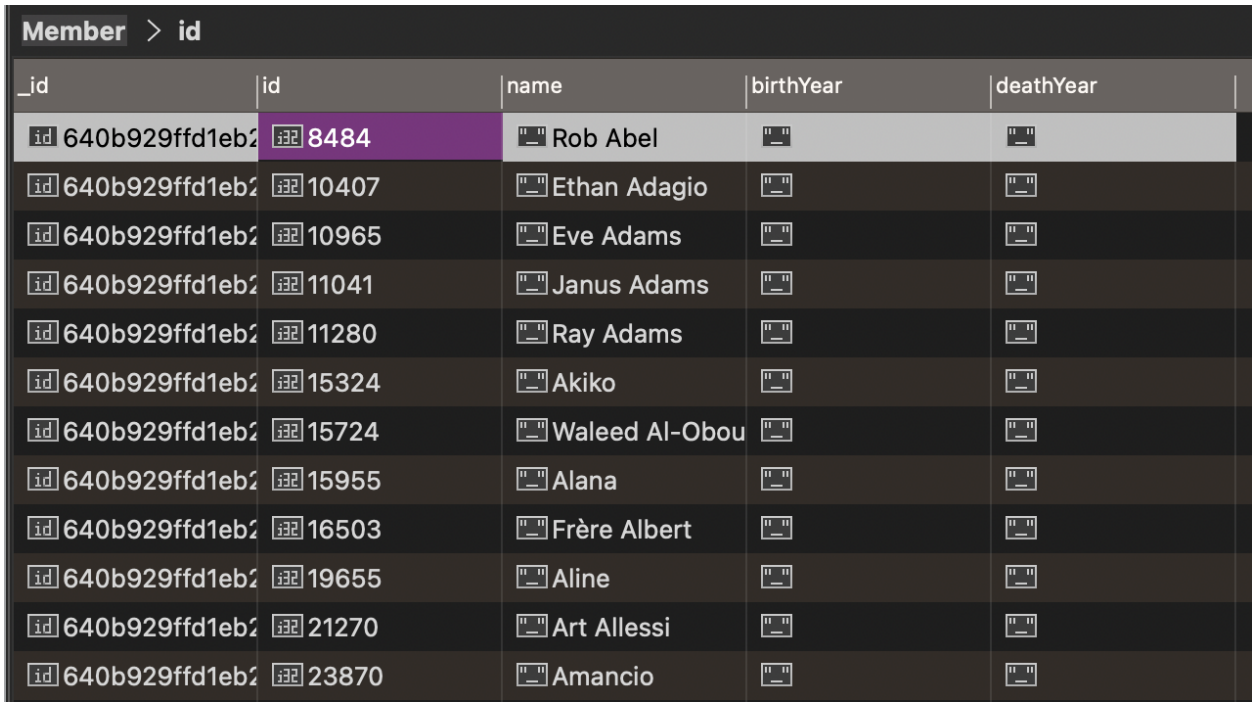
For the import in MongodB, connection was established between Studio3T and PostGres (SQL -> Migration) on my system through which the tables were imported which took around a minute roughly.

The two tables (Member, Movies) in Postgres were converted to collections with the respective fields assigned.



Movies > id

_id	id	titleType	originalTitle	director	startYear	producer	writer	actor
640b770bfd1eb2	2320419	tvEpisode	Safety First/Tea	414505	2013	1907333	227837	4523330
640b770bfd1eb2	2320419	tvEpisode	Safety First/Tea	830799	2013	1907333	227837	4523330
640b770bfd1eb2	2320419	tvEpisode	Safety First/Tea	414505	2013	1907333	774927	4523330
640b770bfd1eb2	2320419	tvEpisode	Safety First/Tea	830799	2013	1907333	774927	4523330
640b770bfd1eb2	2320419	tvEpisode	Safety First/Tea	414505	2013	1907333	1074118	4523330
640b770bfd1eb2	2320419	tvEpisode	Safety First/Tea	830799	2013	1907333	1074118	4523330
640b770bfd1eb2	25053872	tvEpisode	Episode #1.33	884596	1963	683788	64972	350877
640b770bfd1eb2	2871712	tvEpisode	Episode #1.31	108837	2010	73875	1574478	1863418
640b770bfd1eb2	2871712	tvEpisode	Episode #1.31	108837	2010	2024518	1574478	1863418
640b770bfd1eb2	2871712	tvEpisode	Episode #1.31	108837	2010	73875	10243949	1863418
640b770bfd1eb2	2871712	tvEpisode	Episode #1.31	108837	2010	2024518	10243949	1863418



Member > id

_id	id	name	birthYear	deathYear
640b929ffd1eb2	8484	Rob Abel		
640b929ffd1eb2	10407	Ethan Adagio		
640b929ffd1eb2	10965	Eve Adams		
640b929ffd1eb2	11041	Janus Adams		
640b929ffd1eb2	11280	Ray Adams		
640b929ffd1eb2	15324	Akiko		
640b929ffd1eb2	15724	Waleed Al-Obou		
640b929ffd1eb2	15955	Alana		
640b929ffd1eb2	16503	Frère Albert		
640b929ffd1eb2	19655	Aline		
640b929ffd1eb2	21270	Art Allessi		
640b929ffd1eb2	23870	Amancio		

Q2)

2.1) It took 0.4 seconds for its execution.

```
//FIRST
db.Member.find({
  name: /^Phi/,
  deathYear: "",
  _id: {
    $nin: db.Movies
      .find({ startYear: "2014" }, { actor: 1, _id: 0 })
      .flatMap(Movies => Movies.actor.map(actor => actor.actor))
  }


})
```

Raw shell output | Find Query (line 32) | Explain

50 | Documents 1 to 50

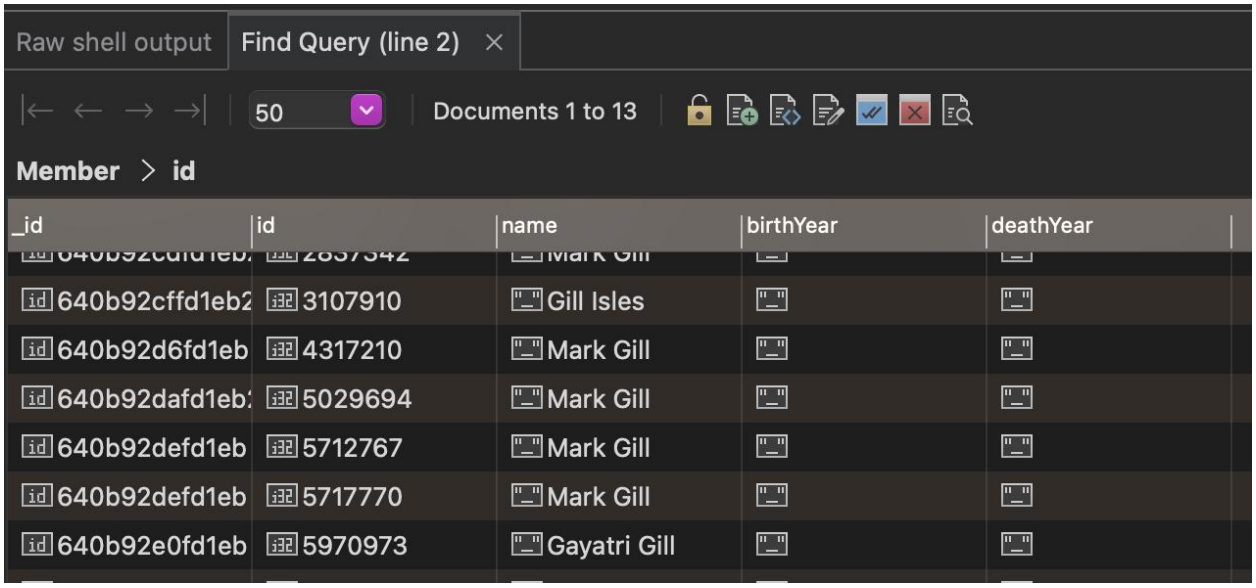
Movies > actor

_id	actor
	3356090
	3356090
	332871
	332871
	332871
	2452431
	2452431
	7149404

2.2)
It took 12 seconds to run.

```
db.Member.aggregate([
  {
    $lookup: {
      from: "Movies",
      localField: "id",
      foreignField: "producer",
      as: "moviess"
    }
  },
  {
    $match: {
      "name": /Gill/,
      "moviess.genre": "Talk-Show",
      "moviess.startYear": "2017"
    }
  },
  {
    $project: {
      _id: 0,
      name: 1
    }
  }
])
```

Output:



2.3)
It took 7 seconds for the query to run.

```
db.Movies.aggregate([
  {
    $lookup: {
      from: "Members",
      localField: "writer",
      foreignField: "id",
      as: "writer"
    }},
  {
    $match: {
      "Member.name": /Bhardwaj/
    }
  },
  {
    $project: {
      _id: 0,
      runtimeMinutes: 1
    }
  }
])
```

Output:

Raw shell output | Find Query (line 1) ×

50 | Documents 1 to 5

Member > id

_id	id	name	birthYear	deathYear
640b92bcfd1eb	13000463	Kripa Shankar Bh		
640b92c3fd1eb	14123797	Anand Bhardwaj		
640b92effd1eb2	8866295	Pradeep Bhardw		
640b92f1fd1eb2	9263917	Anand Bhardwaj		

2.4) It took 11 seconds for this query

```
db.Member.aggregate([
  {
    $lookup: {
      from: "Movies",
      localField: "id",
      foreignField: "producer",
      as: "moviess"
    }
  },
    {
    $unwind: "$moviess"
  },

  {
    $match: {
      "deathYear":"",
      "moviess.runtimeMinutes": {
        $gt: 120
      }
    }
  },
  {
    $project: {
      _id: 0,
      name: 1
    }}])
```

Output:

Raw shell output | Find Query (line 1) ×

50 | Documents 1 to 7

Member > id

_id	id	name	birthYear	deathYear
640b92a1fd1eb2	159147	Aditya Chopra	1971	
640b92b9fd1eb	12577669	Baba Desai		
640b92b9fd1eb	1260472	Vasant Doshi		
640b92c0fd1eb	13637518	Aditya Chopra		

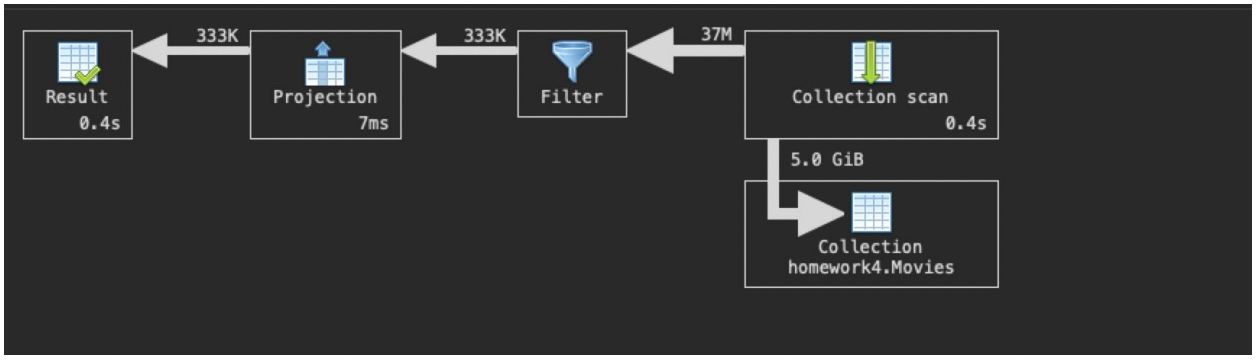
2.5
It took 8 seconds.

```
db.Movies.aggregate([
  {
    $match: {
      genres: "Sci-Fi"
    }
  },
  {
    $lookup: {
      from: "Members",
      localField: "writer",
      foreignField: "id",
      as: "writer_info"
    }
  },
  {
      $unwind : "writer_info"
  },
  {
    $match: {
      "writer_info.name": "James Cameron"
    }

  },
  {
    $match: {
      "writer_info.name": "Sigourney Weaver"
    }
  },
  {
    $project: {
      _id: 0,
      title: 1,
      year: 1,
```

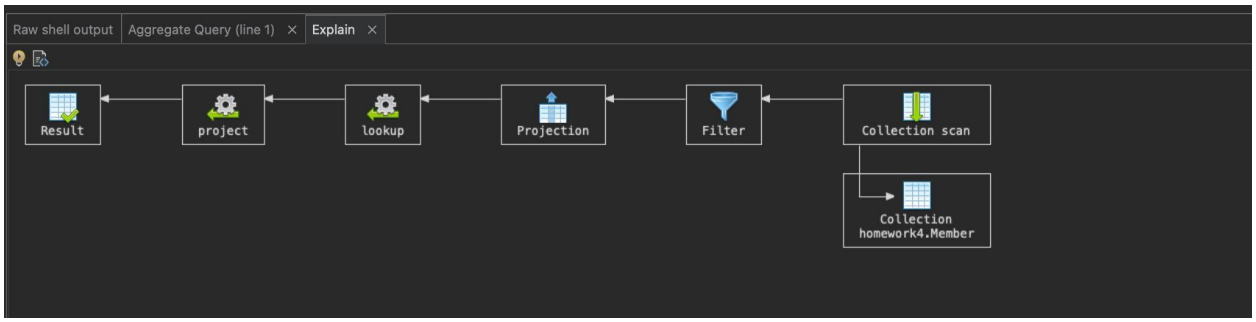
Q3)

3.1



It took MongodB 0.4 seconds to run the collection scan.
37M documents were passed from collections to Filter.
333k documents were passed after that.
It took mongo 7ms to run Projection.
The whole query took 0.4 seconds to run.

3.4



Collection Scan took 4.5 Seconds
Projection took 75 milliseconds.
Lookup took 2.5 seconds.
Results took 7 seconds for the entire executions
12M documents were transferred from collections to filter.

Q4.

4.2 For this query where it is required to have optimization, we can implement
For Movies Collection:

a) **Index on “startYear”**
b) **Index on “genres”**
c) **Index on “producers.name”**

4.3 For this query where it were written by members who had “Bhardwaj” in their names and are still alive
For Members Collection:

a) **Index on “name” field**

4.4 So for this query "Alive producers with the most long-run films created (runtime more than 120 minutes)" optimization can be done by:
For Movies Collection:

a) **Index on “runtimeMinutes”**
b) **Index on “producers**

For Members Collection:

a) **Index on “deathYear”**

4.5 The query here which is Sci-Fi movies directed by James Cameron and starring Sigourney Weaver can be optimized by,
For Movies Collection:

a) **Index on “genres”**
b) **Index on “directors”**
c) **Index on “actor”**