# HOMEWORK 4:

- Anirudh Narayanan(an9425)

Q1.

Create Json files of the data from HW2 as follows:

# pre-process data to remove the format to avoid error while uploading to mongodb :

```
--Replacing double quote with single quotes
CREATE TABLE Memberm AS SELECT * FROM member;
update Memberm set name = replace(name, '"', '''');

CREATE TABLE Genrem AS SELECT * FROM genre;
update Genrem set genres = replace(genres, '"', '''');

CREATE TABLE Characterm AS SELECT * FROM characters;
update Characterm set characters = replace(characters, '"', '''');

CREATE TABLE Titlem AS SELECT * FROM title;
update Titlem set primarytitle = replace(primarytitle, '"', ''''),
original_title = replace(original_title, '"', ''''), titletype =
replace(titletype, '"', '''');
```

# load data as Json: (MOVIES COLLECTION)

```
COPY (SELECT json_strip_nulls(row_to_json(t)) FROM
(SELECT tid as _id, titletype, primarytitle, original_title, startYear,
endYear, runtimeminutes,
      averagerating, numvotes, genres, actors, directorid as directors,
producerid as producers, writerid as writers FROM Titlem T
   LEFT JOIN (SELECT title as titleid, array_agg(g.genres) as genres
           FROM title_genre JOIN Genrem g
   ON title_genre.genre = g.id
           GROUP BY titleid ORDER BY titleid) as t1 ON T.tid = t1.titleid
   LEFT JOIN (SELECT title as titleid, array_agg(director) as directorid
           FROM title_director GROUP BY titleid) as t2 ON T.tid = t2.titleid
   LEFT JOIN (SELECT title as titleid, array_agg(producer) as producerid
           FROM title_producer GROUP BY titleid) as t3 ON T.tid = t3.titleid
   LEFT JOIN (SELECT title as titleid, array_agg(writer) as writerid
           FROM title_writer GROUP BY titleid) as t4 ON T.tid = t4.titleid
   LEFT JOIN (SELECT titleid, json_agg(actors) as actors FROM
            (SELECT t5.titleid, json_build_object('actor', t5.actorid, 'roles',
t5.roles) as actors FROM
```

```
           (SELECT atc.title as titleid, atc.actor as actorid,
array_agg(characterm.characters) as roles
           FROM characterm JOIN actor_title_character atc
   ON characterm.id = atc.characters
           GROUP BY (atc.title, atc.actor) ORDER BY (atc.title, atc.actor)) as
t5) AS doo
           GROUP BY titleid) as t6 ON t6.titleid = T.tid) t) to
   '/Users/anirudhramesh/Desktop/INTRO_TO_BIGDATA/ASSN4/mem_j2.json';
```

MEMBER COLLECTION:

```
COPY (
   SELECT array_to_json(array_agg(json_strip_nulls(row_to_json(t))))
   FROM (SELECT id as "_id", name , birthyear, deathyear
         FROM Memberm) t) to
   '/Users/anirudhramesh/Desktop/INTRO_TO_BIGDATA/ASSN4/mem_j.json';
```

USE COMMAND LINE TO LOAD THESE JSON FILES TO MONGODB DATABASE:
MOVIES:

```
mongoimport --db HW4 --collection movies --file
"/Users/anirudhramesh/Desktop/INTRO_TO_BIGDATA/ASSN4/mem_j2.json" --drop
```

MEMBERS:

```
mongoimport --db HW4 --collection members --file
"/Users/anirudhramesh/Desktop/INTRO_TO_BIGDATA/ASSN4/mem_j.json" --drop --jsonArray
```

**TOTAL TIME TAKEN: 20 minutes**

Q2.

1.

```
db.movies.aggregate([{"$match": {
            "startYear": {"$ne": '2014'}
        }
    }, {
        "$lookup": {
            "from": "members",
            "localField": "actors.actor",
            "foreignField": "_id",
            "as": "Actor"
        }
    }, {
        "$unwind": "$Actor"
    }, {
        "$match": {
            "Actor.name": {"$regex": "^Phi*"},
            "Actor.deathYear": {"$exists": false},
        }
    }, {
        "$project": {
            "_id": 0,
            "Actor.name": 1,
        }
    }
]);
```

2.

```
db.movies.aggregate([
   {
       "$match": {"genres": {"$eq" : "Talk-Show"}, "startYear": {"$eq":
'2017'}}
   },
   {
       $lookup: {
           from: "members",
           localField: "producers",
           foreignField: "_id",
           as : "producer"
       }
   },
   {
       $unwind : "$producer"
   },
   {
       $match: {
       "producer.name": {"$regex": /Gill/}
       }
   },
   {
   $group: {_id: "$producer.name", count: {$sum: 1}}
   },
   {
       $match: {
           "count": {
           "$gte": 50
           }
       }
   }
]);
```

3.

```

db.movies.aggregate([
    {$unwind: "$writers"},
    {"$addFields":{
    "cruntimeminuntes": {"$toInt": "$runtimeminutes"}}},
    {$group: {
        _id: "$writers", runtimeminutes: {$push: "$cruntimeminutes"}
    }},
    {$lookup: {
            from: "members",
            localField: "_id",
            foreignField: "_id",
            as : "writer"
    }},
    {
        $match: {"writer.name": /Bhardwaj/, "writer.deathyear": {$exists: false}}
    },
    {$unwind: "$cruntimeminutes"},
    {$group: {_id: null, avgruntime: {$avg: "cruntimeminutes"}}},
    {$project: {_id: 0}}
]);
```

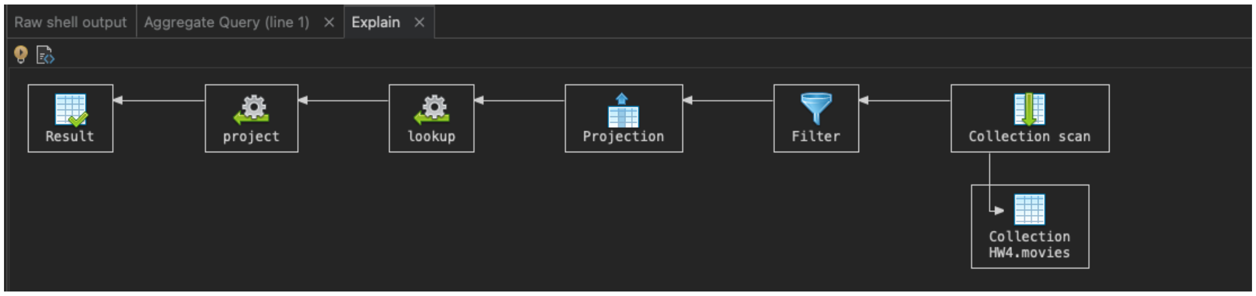
4.
```
db.movies.aggregate([
   {"$addFields":{
   "cruntimeminuntes": {"$toInt": "$runtimeminutes"}}},
   {
       $match: {"cruntimeminutes": {"$gte": 120}, "titletype": {"$eq":
"movie"}}
   },
   {$lookup: {
           from: "members",
           localField: "producers",
           foreignField: "_id",
           as : "producer"
   }},
   {
      $match: {"producer.deathYear": {"$exists": false}}
   },
   {$unwind: "$producer"},

   {$group: {
       _id: "$producer.name", numTitles: {$sum: 1}
   }},
]);
```

5.

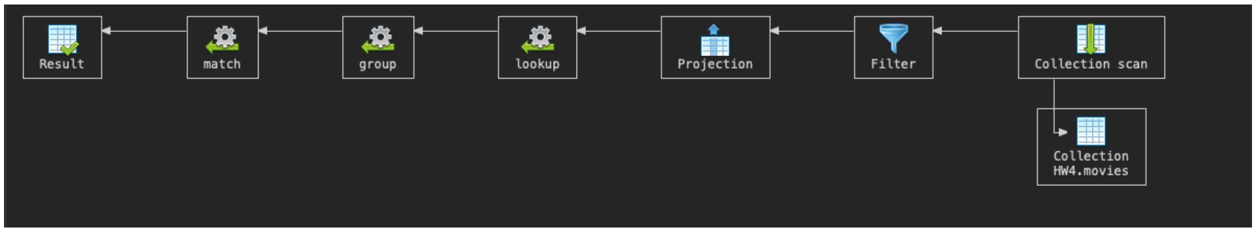
```
db.movies.aggregate([
   {$match: { "genres": {"$eq": "Sci-Fi"}}},
   {$unwind: "$directors"},
   {$lookup: {
           from: "members",
           localField: "directors",
           foreignField: "_id",
           as : "director"
   }},
   {$match: { "director_name": {"$eq": "James Cameron"}}},
   {"$unwind": "$actors"},
   {
   $lookup:{
           from: "members",
           localField: "actors.actors",
           foreignField: "_id",
           as : "acters"
   }},
   {$match: {"acters.name": "Sigourney Weaver"}},
   {$project: {
       _id: 1,
       title: 1,
       director: 1,
       actor: 1,

   }}
]
);
```

Q3.

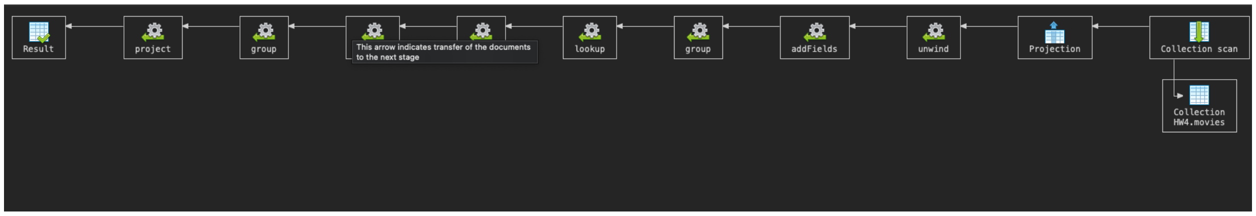
1. Here, we can see we first scan the collection, then filter, project the data, then lookup and project the final output



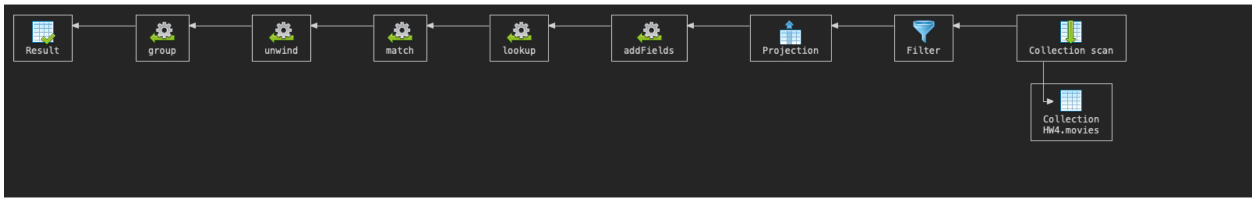
2. Here, we can see we scan the collection, we filter out, then project, then group together and match the corresponding value and output the result.



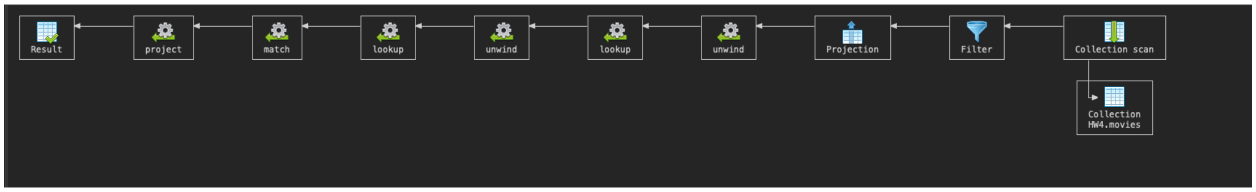
3. Here ,we first scan, then unwind and add new field of changing the type of field, then group together and lookup, and project the result.



4. Here, we first scan, then add new field for changing type of runtime to integer, then match, unwind the producer and match with title by grouping and output the result.



5. Here, we first scan the collection, then unwind the directors, actors, and match with the required actor name and director name and project the result after matching them .



Q4.

1. Here we index the startYear and actor as we are performing string search on both of them.
   SCRIPT:
   db.movies.createIndex({startYear:1, actors: 1})

   Old_time: 15s
   New_time : 8s

2. Here, we index genres and producers, since it is a string search and takes a lot of time,
   SCRIPT:
   db.movies.createIndex({genres: 1, producers: 1})

   OLD time: 10s
   New time: 5s
3. Here, we index and takes a lot of time,
   SCRIPT:
   db.movies.createIndex({runtimeminutes: 1})

   OLD time: 12s
   New time: 4s

4. Here, we index runtime as we are converting it to integers and deathYear, since it is a string search and takes a lot of time,
   SCRIPT:
   db.movies.createIndex({runtimeminutes: 1})
   db.members.createIndex({name: 1})

   OLD time: 19s
   New time: 12s

5. Here, we index name in the members collection and genres in the movies collection
   SCRIPT:
   db.movies.createIndex({genres: 1})

```
db.members.createIndex({name: 1})
```