CS 411: Database Systems

Fall 2019

Homework 2 (Due by 23:59 CT on Sep 28th)

Logistics

1. The homework is due on Sep 28th, 23:59. **We DO NOT accept late homework submissions.**.
2. Please write down your **intermediate steps** in order to get the full score.
3. Feel free to discuss homework problems with your classmates. But you should write your own solutions.
4. Keep your solutions brief and clear.
5. Please use Piazza if you have questions about the homework but do not post answers. Feel free to use private posts or come to office hours.

MongoDB

Give a MongoDB database with two collections, Movies and Actors, write a mongodb query for each of the following questions.

Movie collections: each Movie has its unique id($movie_id), name($movie_name), country($country), director($director), releasing date($release_year), ratings($ratings), genre($genre) and actors, which is an array of actor id($actor_id).
Every movie is associated with only one genre and one director.

Actor collections: each Actor has his/her unique id($actor_id), name($actor_name), and his/her birth country ($birth_country)

1. Find the number of movies made in Australia

```
db.Movies.find({country:'Australia'}).count()
```

2.Find the name and director of action movies which have ratings better than 3(larger than) or whose name start with "ba" (case insensitive). Your output field should only include the name and director of movies(output it as "movie_name", "director").

```
db.Movies.find( {
    genre: "Action",
    $or: [ { ratings: { $gt: 3 } }, { movie_name: /^ba/ } ]
},{ "movie_name": 1, "director": 1,"_id":0})
```

3. For each country, display its name and average ratings of movies produced in this country. Your output field should only the name of the country(rename it as “country”) and average ratings of movies(output it as “ave_ratings”)

```
db.Movies.aggregate([
        {$group: {“_id”: “$country”, “ave_ratings”: {$avg: “$ratings”}}},
        {$project:{_id:0, country:"$_id",ave_ratings:1}}
])
```

4. Find five candidates for Director of the Decade for “Golden Raspberry Awards”. (Find five directors whose movies from 2009 to 2019 received the lowest ratings on average, sort them in ascending order. Your output field should only the name of the director(output it as “director”) and average ratings of his/her work(output it as “ave_ratings”)

```
db.Movies.aggregate([
```

```
        {$match: {"release_year": {$gte: 2009}}, {"release_year": {$lte:
2019}}},
        {$group: {"_id": "$director", "ave_ratings": {$avg: "$ratings"}}},
        {$project:{_id:0, country:"$_id",ave_ratings:1}},
        {$sort:{ave_ratings:1}},
        {$limit : 5 }

    ])
```

5. Find the actors with at least one movie in genre "Romantic" and released after 1999. Display their id and number of movies they acted in. Your output should only contain actor_id(output it as "actorId") and number of movies(output it as "number")

```
db.Movies.aggregate([
        {$match:{$and:[{release_year:{$gt:1999}},{genre:"Romantic"}]}},
        {$unwind:"$actors"},
        {$group:{"_id":"$actors", number:{$sum:1}}},
        {$project:{_id:0, actorId:"$_id",number:1}}
])
```

6. Find Huge Production before millennium(Find the movies produced by 2000 with 10 actors). Display the names, release year and actors.

```
db.Movies.find(
   {release_year:{$lt:2000}, actors:{$size:10}},
   {_id:0, movie_name:1, release_year:1, actors:1}
   )
```

7. Find the movies produced by a Mexican or Brazilian company which started no earlier than 1995. Your output should only include the name and director of the movies.

```
db.Movies.find(
{'company.start_year':{$gte: 1995},'company.country':{$in:['Mexico',
'Brazil']} },
{director:1, '_id':0,movie_name:1})
```

8. Find the actors with at least one movie in genre "Romantic" and released after 1999. Display their name and number of movies they acted in. Your output should only contain their names and birth country.

```
var cursorList = db.Movies.aggregate([
  {$match:{$and:[{release_year:{$gt:1999}},{genre:"Romantic"}]}},
```

```
  {$unwind: "$actors"},
  {$project:{_id:0, actors:1}}
  ]);
var actorlist = new Array();
while (cursorList.hasNext()){
  var tmp = actorlist.push(cursorList.next().actors)
}
db.Actors.find({actor_id:{$in:actorlist}},{_id:0, actor_name:1,
birth_country:1})
```

9.Find the number of movies that each actor has starred in. You can output the Actors' ID as _id and the number of his/her movies as "number".

```
var mapFunc = function() {
            for (var idx = 0; idx < this.actors.length; idx++) {
              emit(this.actors[idx], 1);
            }
          };
var reduceFunc = function(keyActorId, times) {
              return Array.sum(times);
            };
var mapReduceResult = db.Movies.mapReduce(
                mapFunc,
                reduceFunc,
                {out: "times"}
              );
var updateRes = db.times.update({}, { $rename : {"value": "numMovies"
}}, false, true)

db.times.find();
```

10. Given a MongoDB database with two collections, <code>Movies</code> and <code>Actors</code>.
  For each actor, find the actor's best friend. Best friend refers to who collaborates most with him/her, i.e. they act in most movies together(if there are ties, find the actor with smallest id). You need to output the Actors' ID as "_id" and the ID of his/her best friend as "friendID".

```
var mapFunc = function() {
                for (var idx1 = 0; idx1 < this.actors.length; idx1++) {
                  for(var idx2 = idx1 +1; idx2 < this.actors.length;
idx2++){
                    emit(this.actors[idx1], this.actors[idx2]);
                    emit(this.actors[idx2], this.actors[idx1]);
                  }
```

```
                }
            };

var reduceFunc = function(keyActorId, actors) {
  if(actors.length === 0)
     return -1;
  var modeMap = {};
  var maxEl = actors[0], maxCount = 1;
  for(var i = 0; i < actors.length; i++){
     var el = actors[i];
     if(modeMap[el] == null)
        modeMap[el] = 1;
     else
        modeMap[el] +=1;
     if(modeMap[el] === maxCount){
          if(el < maxEl){
                maxEl = el;
          }
     }
     if(modeMap[el] > maxCount){
                maxEl = el;
                maxCount = modeMap[el];
     }
  }
  return maxEl;
};

var mapReduceRes = db.Movies.mapReduce(
                mapFunc,
                reduceFunc,
                {out: "times"}
               );
var updateRes = db.times.update ( {}, { $rename : {"value": "friendID"
}}, false,true)
db.times.find();
```