**Big Data Spring 2018**

**Project 1**

Nicola Eldering

Yifan Chen

**Member 1: Nicolas Eldering**  
My teammate and I agree that I handled 50% of the overall project. My specific tasks included:

* Task 1: I wrote the JavaScript and Java for question 3A
* Task 2: I wrote the JavaScript and Java for question 3C
* Task 3: I wrote the JavaScript and Java for question 3E
* Task 4: Ran and took screenshots for JavaScript

**Member 2: Yifan Chen**  
My teammate and I agree that I handled 50% of the overall project. My specific tasks included:

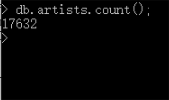
* Task 1: I wrote the JavaScript and Java for question B. I designed the RegExr for this problem to find the specific artists. I wrote a function in mongodb to print the information we need. In java, I researched how to use Pattern class to do the RegExr, like this: Pattern pattern=Pattern.compile(".\*punk.\*", Pattern.CASE\_INSENSITIVE). And I use a cursor to iterate the whole collection.
* Task 2: I wrote the JavaScript and Java for question D. I wrote map and reduce method to emit {artistID, weight} and aggregate their weight. After I doing Mapreducer, to get the top 10 artists from my collection, firstly I used sort({“weight”: -1}) in DESC order of weight. Then I used limit(10) to get the top 10 artists. In java, I implemented the program module of using Mapreducer, I found the way we define map and reduce is using String. Then use both of them as parameters: MapReduceCommand cmd = new MapReduceCommand(collection, map, reduce, "output", MapReduceCommand.OutputType.REPLACE, null); , which is really simple. Finally I used “MapReduceOutput” and “DBCollection” to store my temp answer. In java, the way to sort and limit is “DBObject sort = new BasicDBObject(); sort.put("value.weight",-1); DBCursor cursor= resultColl.find().sort(sort).limit(10);
* Task 3: I wrote the JavaScript and Java for question F. I use two Mapreducer to calculate the total tags of each user used and total artists for each user. To calculate total tags is simple, I just emitted userID paired with count:1. To calculate total artists for each user, firstly I emitted {userID:this.userID, artistID:this.artistID},{count:1}. After reducing it, I emitted {userID:this.\_id.userID},{count:1} to get total artists number for each user. And the result is total tags divided by total artists number. In java, it’s much similar with previous question. I used two arrays to store the tags number and artists number to do the division later.

This section contains the question, the JavaScript solutions and a screenshot of the JavaScript running in a live environment.

**Result Screenshots**

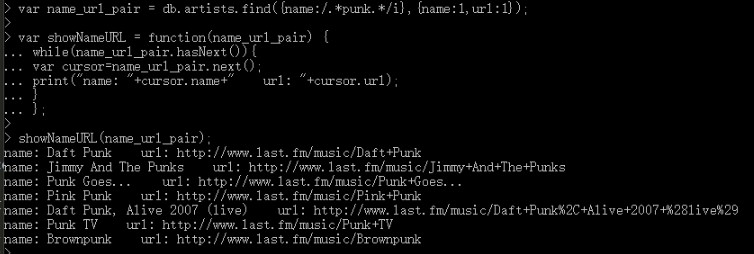
Question a) Find the total number of artists in the database.

To solve this we can ask the collection how many documents it has with the count command.



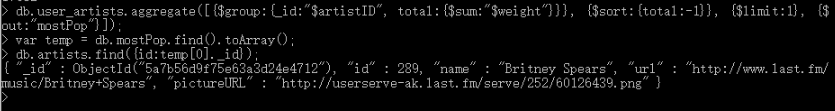
Question b) Find URLs of artists whose names contain the word punk.

We mapReduce the artists collection to emit band name and URL then take only the bands who’s name contains the word punk. Then we call the find command to have the results printed to our console.



Question c) Find the most listened artist(s).

We run an aggregate command on the user\_artists collection to sort it by popularity and limit 1 to grab the most popular artist ID. Then we take that result and store it’s find command’s results in an array. Use the array to grab the artists ID of the most popular artists and use it as a find argument to get the rest of the information.



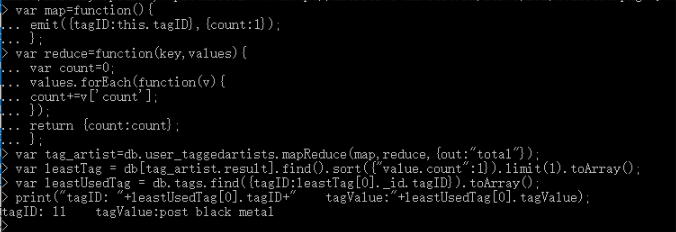
Question d) Write a mapReducer to compute the top 10 popular artists.

We write a map reducer to emit and artists ID and their count. We take our results sort it by weight and limit it so we only get the top 10. Finally we iterate through our top 10 using the artistsID as a find argument on the artists collection and print out our results.



Question e) Write a mapReducer to find the least used tag.

We write a map to emit the tagID and a count of 1 and our reducer sums up the total for each tag. We then sort and limit our result and store it in an array. We then store the relevant columns in an array and finally print out our least used tag.



Question (f) Write a mapReducer to compute the average number of tags used by each user for all artists.

We calculate the number of users and store it in a variable. Then we find the number of artists and store it in a variable. Our map reducer sums up the number of tags. Then all we must do is divide our tag count by our user amount by our artists amount.



Question (f) -- only a few

**Java-code and sample output**

a) Find the total number of artists in the database.

Java:

public int countArtists(){

if(m!=null){

db=m.getDB(localDbName); //connect to database

DBCollection collection = db.getCollection("artists");

return (int)collection.count(); //count() returns a long, cast down to int

}

return 0; //if connect fails

}

Sample Output:

The total number of artists is 17632

b) Find URLs of artists whose names contain the word punk.

public void listURLsNameContainPunk(){

if (m!=null){

db=m.getDB("lastFMdb");

DBCollection collection = db.getCollection("artists");

BasicDBObject query = new BasicDBObject();

BasicDBObject fields = new BasicDBObject();

Pattern pattern=Pattern.compile(".\*punk.\*", Pattern.CASE\_INSENSITIVE);

query.put("name",pattern);

fields.put("name",1);

fields.put("url", 1);

DBCursor cur = collection.find(query, fields);

while(cur.hasNext()) {

DBObject cursor = cur.next();

System.out.println("name: " + cursor.get("name")+" url: "+cursor.get("url"));

}

}

}

Sample Output:

The URLs of artists whose names contain the word "punk" are

name: Daft Punk url: http://www.last.fm/music/Daft+Punk

name: Jimmy And The Punks url: http://www.last.fm/music/Jimmy+And+The+Punks

name: Punk Goes... url: http://www.last.fm/music/Punk+Goes...

name: Pink Punk url: http://www.last.fm/music/Pink+Punk

name: Daft Punk, Alive 2007 (live) url: http://www.last.fm/music/Daft+Punk%2C+Alive+2007+%28live%29

name: Punk TV url: http://www.last.fm/music/Punk+TV

name: Brownpunk url: <http://www.last.fm/music/Brownpunk>

c) Find the most listened artist(s).

Java:

public void listMostListenedArtist(){

if (m!=null) {

db = m.getDB(localDbName); //connect to database

DBCollection collection = db.getCollection("user\_artists");

List<DBObject> operations=new ArrayList<>(); //array for aggregate

DBObject group = new BasicDBObject("$group", new BasicDBObject("\_id","$artistID").append("total", new BasicDBObject("$sum","$weight")));

DBObject sort=new BasicDBObject("$sort",new BasicDBObject("total",-1));

DBObject limit=new BasicDBObject("$limit",1); // so we grab only one (our top artist)

operations.add(group); //add group operations to our array

operations.add(sort); // add sort operations to our array

operations.add(limit); // add limit restriction

AggregationOutput output=collection.aggregate(operations); //aggregation output becomes an iterable

Iterable<DBObject> iterable = output.results(); // grab the iterable

Iterator<DBObject> iterator = iterable.iterator(); // make it so we can go through the results of aggregation

// select details of the most popular artists from artists collection

DBCollection collection\_artists = db.getCollection("artists");

BasicDBObject query = new BasicDBObject();

BasicDBObject fields = new BasicDBObject();

while (iterator.hasNext()) { //parse the results

query.put("id",iterator.next().get("\_id"));

fields.put("id",1);

fields.put("name",1);

fields.put("url",1);

fields.put("pictureURL",1);

DBCursor cursor = collection\_artists.find(query, fields);

while(cursor.hasNext()) { // grab our objects

BasicDBObject details = (BasicDBObject) cursor.next();

System.out.println("id:"+details.get("id")+" name:"+details.get("name")+" URL:"+details.get("url")+" pictureURL:"+details.get("pictureURL"));

}

}

}

}

Sample Output:

The most listened artist is

id:289 name:Britney Spears url:http://www.last.fm/music/Britney+Spears pictureURL:http://userserve-ak.last.fm/serve/252/60126439.png

d) Write a mapReducer to compute the top 10 popular artists.

public void listTop10Artists(){

if (m!=null) {

db = m.getDB("lastFMdb");

DBCollection collection = db.getCollection("user\_artists");

String map = "function(){"+

"emit({artistID:this.artistID},{weight:this.weight});"+

"};";

String reduce = "function(key,values){"+

"var weight=0;"+

"values.forEach(function(v){"+

"weight+=v['weight'];"+

"});"+

"return {weight:weight};"+

"};";

MapReduceCommand cmd = new MapReduceCommand(collection, map, reduce,

"output", MapReduceCommand.OutputType.REPLACE, null);

MapReduceOutput out = collection.mapReduce(cmd);

DBCollection resultColl = out.getOutputCollection();

DBObject sort = new BasicDBObject();

sort.put("value.weight",-1);

DBCursor cursor= resultColl.find().sort(sort).limit(10);

DBCollection collection\_artists = db.getCollection("artists");

while(cursor.hasNext()) {

BasicDBObject details = (BasicDBObject) cursor.next();

BasicDBObject query = new BasicDBObject();

query.put("id",((BasicDBObject)details.get("\_id")).get("artistID"));

DBCursor cur = collection\_artists.find(query);

BasicDBObject specificArtist= (BasicDBObject)cur.next();

System.out.println("id: "+specificArtist.get("id")+" name: "+specificArtist.get("name")+" url: "+specificArtist.get("url")+" pictureURL: "+specificArtist.get("pictureURL"));

}

}

}

Sample Output:

Top 10 popular artists are

id: 289 name: Britney Spears url: http://www.last.fm/music/Britney+Spears pictureURL: http://userserve-ak.last.fm/serve/252/60126439.png

id: 72 name: Depeche Mode url: http://www.last.fm/music/Depeche+Mode pictureURL: http://userserve-ak.last.fm/serve/252/75022.jpg

id: 89 name: Lady Gaga url: http://www.last.fm/music/Lady+Gaga pictureURL: http://userserve-ak.last.fm/serve/252/47390093.png

id: 292 name: Christina Aguilera url: http://www.last.fm/music/Christina+Aguilera pictureURL: http://userserve-ak.last.fm/serve/252/47363849.png

id: 498 name: Paramore url: http://www.last.fm/music/Paramore pictureURL: http://userserve-ak.last.fm/serve/252/35837991.png

id: 67 name: Madonna url: http://www.last.fm/music/Madonna pictureURL: http://userserve-ak.last.fm/serve/252/340387.jpg

id: 288 name: Rihanna url: http://www.last.fm/music/Rihanna pictureURL: http://userserve-ak.last.fm/serve/252/53023109.png

id: 701 name: Shakira url: http://www.last.fm/music/Shakira pictureURL: http://userserve-ak.last.fm/serve/252/52116105.png

id: 227 name: The Beatles url: http://www.last.fm/music/The+Beatles pictureURL: http://userserve-ak.last.fm/serve/252/2588646.jpg

id: 300 name: Katy Perry url: http://www.last.fm/music/Katy+Perry pictureURL: <http://userserve-ak.last.fm/serve/252/42128121.png>

e) Write a mapReducer to find the least used tag.

Java:

public void listLeastUsedTag(){

if (m!=null) {

db = m.getDB(localDbName); // connect to database

DBCollection collection = db.getCollection("user\_taggedartists");

DBCollection collection\_tags = db.getCollection("tags");

String map = "function(){" + // map function to emit tags

"emit({tagID:this.tagID},{count:1});"+

"};";

String reduce = "function(key,values){"+ //reduce to sum our tages

"var count=0;"+

"values.forEach(function(v){"+

"count+=v['count'];"+

"});"+

"return {count:count};"+

"};";

MapReduceCommand cmd = new MapReduceCommand(collection, map, reduce,

"output", MapReduceCommand.OutputType.REPLACE, null); // build command

MapReduceOutput out = collection.mapReduce(cmd); // run mongodb command and catch

DBCollection resultColl = out.getOutputCollection(); // collect our result

DBObject sort = new BasicDBObject();

sort.put("value.count",1); //sort

DBCursor cursor= resultColl.find().sort(sort).limit(1); // grab the least used tag

BasicDBObject query = new BasicDBObject();

while(cursor.hasNext()) { //list our tag

query.put("tagID", ((BasicDBObject) cursor.next().get("\_id")).get("tagID"));

DBCursor cursor = collection\_tags.find(query);

while (cursor.hasNext()) {

System.out.println(cursor.next());

}

}

}

}

Sample Output:

The least used tag is { "\_id" : { "$oid" : "5a7b2ad9f718bf55171dbaa4"} , "tagID" : 11 , "tagValue" : "post black metal"}

f) Write a mapReducer to compute the average number of tags used by each user for all artists.

public void averageTagsbyEachUser(){

if (m!=null) {

db = m.getDB("lastFMdb");

DBCollection collection = db.getCollection("user\_taggedartists");

//calculate the tags' number users used

String map\_tagsNum = "function(){"+

"emit({userID:this.userID},{count:1});"+

"};";

String reduce\_tagsNum="function(key,values){"+

"var cnt=0;"+

"values.forEach(function(v){"+

"cnt+=v['count'];"+

"});"+

"return {count:cnt};"+

"};";

MapReduceCommand cmd = new MapReduceCommand(collection, map\_tagsNum, reduce\_tagsNum,

"output", MapReduceCommand.OutputType.REPLACE, null);

MapReduceOutput out = collection.mapReduce(cmd);

DBCollection resultColl = out.getOutputCollection();

DBCursor cursor = resultColl.find();

ArrayList<Double> tags\_number\_array=new ArrayList<>();

ArrayList<String> userID\_array=new ArrayList<>();

while(cursor.hasNext()) {

DBObject iter=cursor.next();

tags\_number\_array.add((double)((BasicDBObject) iter.get("value")).get("count"));

userID\_array.add(((BasicDBObject) iter.get("\_id")).get("userID").toString());

}

// calculate the artists number the user tagged

String map\_artistsNum\_temp="function(){"+

"emit({userID:this.userID, artistID:this.artistID},{count:1});"+

"};";

String reduce\_artistsNum\_temp="function(key,values){"+

"var cnt=0;"+

"values.forEach(function(v){"+

"cnt+=v['count'];"+

"});"+

"return {count:cnt};"+

"};";

cmd = new MapReduceCommand(collection, map\_artistsNum\_temp, reduce\_artistsNum\_temp,

"output", MapReduceCommand.OutputType.REPLACE, null);

out = collection.mapReduce(cmd);

DBCollection resultColl\_artistsNum\_temp = out.getOutputCollection();

String map\_artistsNum="function(){"+

"emit({userID:this.\_id.userID},{count:1});"+

"};";

String reduce\_artistsNum="function(key,values){"+

"var cnt=0;"+

"values.forEach(function(v){"+

"cnt+=v['count'];"+

"});"+

"return {count:cnt};"+

"};";

cmd = new MapReduceCommand(resultColl\_artistsNum\_temp, map\_artistsNum, reduce\_artistsNum,

"output", MapReduceCommand.OutputType.REPLACE, null);

out = resultColl\_artistsNum\_temp.mapReduce(cmd);

DBCollection resultColl\_artistsNum = out.getOutputCollection();

cursor = resultColl\_artistsNum.find();

ArrayList<Double> artists\_number\_array=new ArrayList<>();

while(cursor.hasNext()) {

artists\_number\_array.add((double)((BasicDBObject) cursor.next().get("value")).get("count"));

}

ArrayList<Double> average=new ArrayList<>();

for (int i=0;i<artists\_number\_array.size();i++){

average.add((tags\_number\_array.get(i))/(artists\_number\_array.get(i)));

System.out.println("userID: "+userID\_array.get(i)+" average: "+average.get(i));

}

}

}

Sample Output(only list a few):

The average number of tags used by each user for all artists is:

userID: 2.0 average: 4.5

userID: 3.0 average: 6.222222222222222

userID: 4.0 average: 1.0701754385964912

userID: 5.0 average: 1.2647058823529411

userID: 6.0 average: 3.5

userID: 7.0 average: 1.5

userID: 8.0 average: 2.857142857142857

userID: 9.0 average: 1.2772277227722773

userID: 10.0 average: 1.8461538461538463

userID: 11.0 average: 1.1

userID: 12.0 average: 3.066147859922179

userID: 13.0 average: 10.5

userID: 14.0 average: 4.036363636363636

userID: 15.0 average: 14.0

userID: 16.0 average: 3.357142857142857

userID: 17.0 average: 5.0

userID: 18.0 average: 1.5

**Mongo Code**

**Question (a) Find the total number of artists in the database.**

db.artists.count();

-----------------------------------------------------------------------------------------------

**Question (b) Find URLs of artists whose names contain the word punk.**

var name\_url\_pair = db.artists.find({name:/.\*punk.\*/i},{name:1,url:1});

var showNameURL = function(name\_url\_pair) {

while(name\_url\_pair.hasNext()){

var cursor=name\_url\_pair.next();

print("name: "+cursor.name+" url: "+cursor.url);

}

};

showNameURL(name\_url\_pair);

-----------------------------------------------------------------------------------------------

**Question (c) Find the most listened artist(s).**

db.user\_artists.aggregate([{$group:{\_id:"$artistID", total:{$sum:"$weight"}}}, {$sort:{total:-1}}, {$limit:1}, {$out:"mostPop"}]);

var temp = db.mostPop.find().toArray();

db.artists.find({id:temp[0].\_id});

-----------------------------------------------------------------------------------------------

**Question (d) Write a mapReducer to compute the top 10 popular artists.**

var map=function(){

emit({artistID:this.artistID},{weight:this.weight});

};

var reduce=function(key,values){

var weight=0;

values.forEach(function(v){

weight+=v['weight'];

});

return {weight:weight};

};

var weight\_artist=db.user\_artists.mapReduce(map,reduce,{out:"total"});

var top10Artists = db[weight\_artist.result].find().sort({"value.weight":-1}).limit(10);

var showDetails = function(top10Artists) {

while(top10Artists.hasNext()){

var cursor=top10Artists.next();

var specificArtist=db.artists.find({id:cursor.\_id.artistID}).toArray();

print("id: "+specificArtist[0].id+" name: "+specificArtist[0].name+" url: "+specificArtist[0].url+" pictureURL: "+specificArtist[0].pictureURL);

}

};

showDetails(top10Artists);

------------------------------------------------------------------------------------------------

**Question (e) Write a mapReducer to find the least used tag.**

var map=function(){

emit({tagID:this.tagID},{count:1});

};

var reduce=function(key,values){

var count=0;

values.forEach(function(v){

count+=v['count'];

});

return {count:count};

};

var tag\_artist=db.user\_taggedartists.mapReduce(map,reduce,{out:"total"});

var leastTag = db[tag\_artist.result].find().sort({"value.count":1}).limit(1).toArray();

var leastUsedTag = db.tags.find({tagID:leastTag[0].\_id.tagID}).toArray();

print("tagID: "+leastUsedTag[0].tagID+" tagValue:"+leastUsedTag[0].tagValue);

-----------------------------------------------------------------------------------------------

**Question (f) Write a mapReducer to compute the average number of tags used by each user for all artists.**

//calculate the number of each user's tags

var map\_tagsNum=function(){

emit({userID:this.userID},{count:1});

};

var reduce\_tagsNum=function(key,values){

var cnt=0;

values.forEach(function(v){

cnt+=v['count'];

});

return {count:cnt};

};

var each\_user\_tags\_number=db.user\_taggedartists.mapReduce(map\_tagsNum,reduce\_tagsNum,{out:"total"});

var tags\_number\_array = db[each\_user\_tags\_number.result].find().toArray();

//calculate the number of artists of each user

var map\_artistsNum\_temp=function(){

emit({userID:this.userID, artistID:this.artistID},{count:1});

};

var reduce\_artistsNum\_temp=function(key,values){

var cnt=0;

values.forEach(function(v){

cnt+=v['count'];

});

return {count:cnt};

};

var total\_each\_user\_artistNum\_temp=db.user\_taggedartists.mapReduce(map\_artistsNum\_temp,reduce\_artistsNum\_temp,{out:"total"});

var total\_each\_user\_artistNum\_temp\_result=db[total\_each\_user\_artistNum\_temp.result];

var map\_artistsNum=function(){

emit({userID:this.\_id.userID},{count:1});

};

var reduce\_artistsNum=function(key,values){

var cnt=0;

values.forEach(function(v){

cnt+=v['count'];

});

return {count:cnt};

};

var total\_each\_user\_artistNum=total\_each\_user\_artistNum\_temp\_result.mapReduce(map\_artistsNum,reduce\_artistsNum,{out:"total"});

var artist\_number\_array = db[total\_each\_user\_artistNum.result].find().toArray();

var storeEachUsertags = function(tags\_number\_array, artist\_number\_array) {

var averageTagsNumbers=[];

for (var i=0;i<tags\_number\_array.length;i++)

averageTagsNumbers[i]=(tags\_number\_array[i].value.count)/(artist\_number\_array[i].value.count);

return averageTagsNumbers;

};

var storeEachUsertagsArray = storeEachUsertags(tags\_number\_array, artist\_number\_array);

for (var i=0;i<storeEachUsertagsArray.length;i++){

print("userID:"+tags\_number\_array[i].\_id.userID +" average:"+storeEachUsertagsArray[i]);

}