**Lab 10**

**MongoDB Tutorial for Beginners:** [**https://beginnersbook.com/2017/09/mongodb-tutorial/**](https://beginnersbook.com/2017/09/mongodb-tutorial/)

**MongoDB Online Terminal:**

[**https://www.jdatalab.com/mongodb/**](https://www.jdatalab.com/mongodb/)

[**https://www.jdoodle.com/online-mongodb-terminal**](https://www.jdoodle.com/online-mongodb-terminal)

1. **Create a database called students and load the data either via a file (provided) or insertMany(). (2 pt)**

**use students**

****

**var file = cat(“./students(2).json”);**

**var o = JSON.parse(file);**

****

**db.studentsrecords.insert(o);**

**A picture containing black

Description automatically generated**

**db.studentsrecords.find().pretty()**

**A screen shot of a computer

Description automatically generated**

1. **Update a document in your collection. For student\_id 1 and class\_id 13, change the class\_id to 3. After updating, show the updated document. (2 pts)**

**db.studentsrecords.find({"student\_id":1,"class\_id":13}).pretty()**

**A screenshot of a cell phone

Description automatically generated**

**db.studentsrecords.update({"student\_id":1,"class\_id":13},{$set:{"class\_id":3}})**

**db.studentsrecords.find({"student\_id":1,"class\_id":3}).pretty()**

**A screenshot of a cell phone

Description automatically generated**

1. **Using the project command, display only select field names. Display the student\_id and class\_id. Do not display the document id field. (2 pts)**

**db.studentsrecords.find({},{“id”:0, “student\_id”:1, “class\_id”:1})**

**A picture containing black, table, large, man

Description automatically generated**

1. **Display the student\_id and class\_id and sort them in ascending order by date. (2 pts)**

**db.studentsrecords.find({}, { "\_id":0, "student\_id":1, "class\_id":1 }).sort({"class\_id":1})**

**A screenshot of a computer

Description automatically generated**

1. **Aggregate the results to provide a count of how many classes each student is taking. (3 pts)**

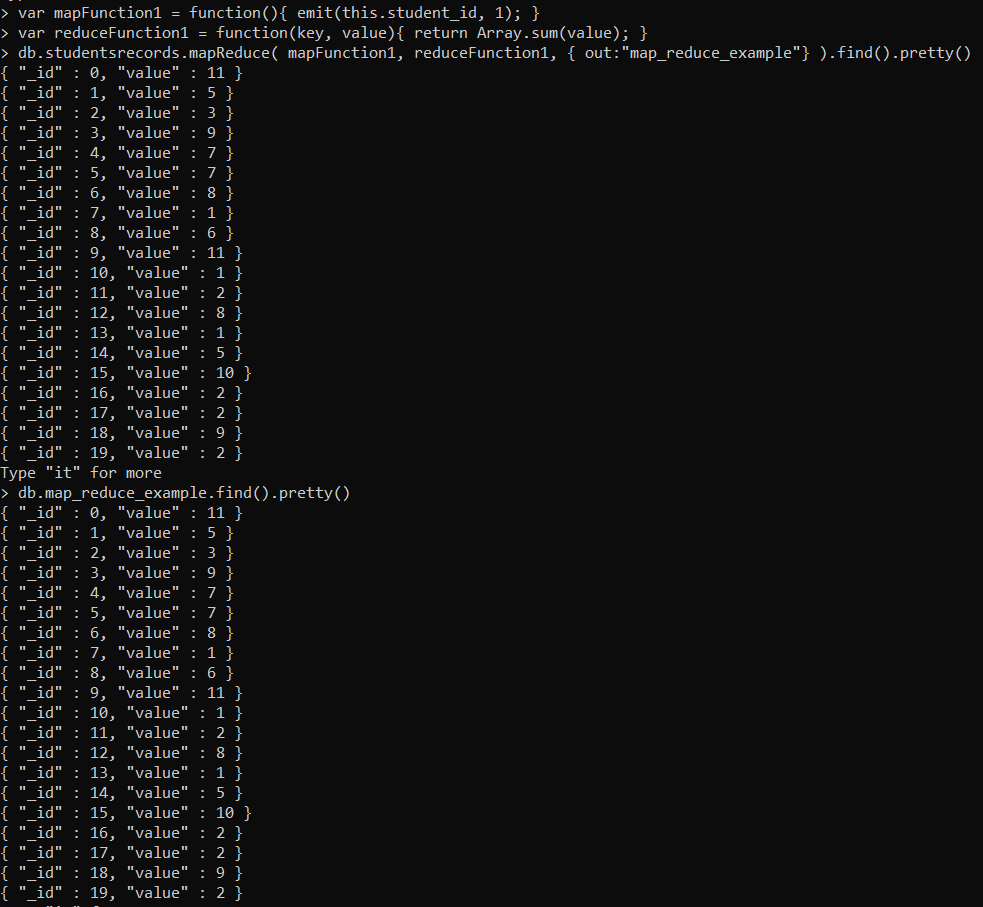
**var mapFunction1 = function(){emit(this.student\_id, 1);}**

**var reduceFunction1 = function(key, value){return Array.sum(value);}**

**db.studentsrecords.mapReduce(mapFunction1, reduceFunction1, {out: “map\_reduce\_example”}.find().pretty()**

***Alternative way to get same output*:**

**db.map\_reduce\_example.find().pretty()**



1. **Aggregate the results to provide a count of how many students are in class\_id 3. (3 pts)**

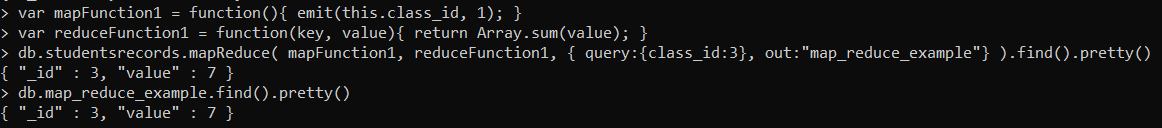
**var mapFunction1 = function(){emit(this.class\_id, 1);}**

**var reduceFunction1 = function(key, value){return Array.sum(value);}**

**db.studentsrecords.mapReduce(mapFunction1, reduceFunction1, {query:{class\_id:3}, out: “map\_reduce\_example”}.find().pretty()**

***Alternative way to get same output*:**

**db.map\_reduce\_example.find().pretty()**



1. **Run the following commands and explain what each command is doing. (3 pts)**

**COMMAND 1:**

db.students.mapReduce (function ()

{ for (var i in this.scores)

{emit( {student\_id:this.student\_id, class\_id:this.class\_id, type:this.scores[i].type},this.scores[i].score); }},

function (key, values) { return Array.avg(values); },

{ out:"test"}).find().pretty()

* **The command is first grouping by the “student\_id”, “class\_id”, and value of “type” of an element at position “i” in the scores array.**
* **It is then calculating the average of values of “score” of an element at position “i” in the scores array based on those variables.**
* **For each combination of “student\_id”, “class\_id”, and “type”, the command is emitting or outputting the “student\_id”, “class\_id”, “type”, and the average “score” based on the previous variables, and saving it in a collection named “test”.**
* **The command is displaying the results based on the “.find().pretty()” extension on the end of the function, but if you do “db.test.find().pretty()”, you can get the same results since the results are being saved in the “test” collection.**

**COMMAND 2:**

db.students.mapReduce (function ()

{ for (var i in this.scores)

{emit( {student\_id:this.student\_id, class\_id:this.class\_id},this.scores[i].score); }},

function (key, values) { return Array.avg(values); },

{ query: {student\_id:4},

out:"test"}).find().pretty()

* **The command is first filtering and using the list of records with “student\_id” as 4.**
* **The command is grouping by the “student\_id” and “class\_id”.**
* **It is then calculating the average of values of “score” of an element at position “i” in the scores array based on those variables.**
* **For each combination of “student\_id” and “class\_id”, the command is emitting or outputting the “student\_id”, “class\_id”, and the average “score” based on the previous variables, and saving it in a collection named “test”.**
* **The command is displaying the results based on the “.find().pretty()” extension on the end of the function, but if you do “db.test.find().pretty()”, you can get the same results since the results are being saved in the “test” collection.**

**COMMAND 3:**

db.students.mapReduce (function ()

{ for (var i in this.scores)

{emit( {student\_id:this.student\_id, class\_id:this.class\_id},this.scores[i].score); }},

function (key, values) { return Array.avg(values); },

{ query: {student\_id:4, class\_id:26},

out:"test"}).find().pretty()

* **The command is first filtering and using the list of records with “student\_id” as 4 and “class\_id” as 26.**
* **The command is grouping by the “student\_id” and “class\_id”.**
* **It is then calculating the average of values of “score” of an element at position “i” in the scores array based on those variables.**
* **For each combination of “student\_id” and “class\_id”, the command is emitting or outputting the “student\_id”, “class\_id”, and the average “score” based on the previous variables, and saving it in a collection named “test”.**
* **The command is displaying the results based on the “.find().pretty()” extension on the end of the function, but if you do “db.test.find().pretty()”, you can get the same results since the results are being saved in the “test” collection.**

1. **Write a Map Reduce command to calculate the average score in each class. (3 pts)**

**var mapFunction1 = function(){for (var i in this.scores){emit({class\_id:this.class\_id}, this.scores[i].score);}}**

**A screenshot of a computer

Description automatically generated**

**var reduceFunction1 = function(key, values){ return Array.avg(values);}**

**A screenshot of a computer

Description automatically generated**

**db.studentsrecords.mapReduce(mapFunction1, reduceFunction1, {out:"test"}).find().pretty()**

**A screenshot of a computer

Description automatically generated**