# MongoDB wk3

Install curl and follow the instructions to download the file zips.json. Import the file into mongo and name the database zips

**$ mongoimport --host localhost --db zips –collection zips < zips.json**

**$ Mongo**

**> Use zips**

**Question 3.1:**

db.zips.aggregate( [ {$group:{\_id:{state:"$state"}, n:{$sum:1}}} , {$sort:{n:-1}} , {$limit:4} ] )

{

"result" : [

{

"\_id" : {

"state" : "TX"

},

"n" : 1676

},

{

"\_id" : {

"state" : "NY"

},

"n" : 1596

},

{

"\_id" : {

"state" : "CA"

},

"n" : 1523

},

{

"\_id" : {

"state" : "PA"

},

"n" : 1458

}

],

"ok" : 1

}

**Question 3.2**

db.zips.aggregate(

... [

... { $project : { \_id : { $substr : ["$\_id",0,1] } } } ,

... { $group : { \_id : "$\_id", n : {$sum:1} } }

... ]

... )

Becomes

db.zips.aggregate(

... [

... { $project : { \_id : { $substr : ["$city",0,1] } } } ,

... { $group : { \_id : "$\_id", n : {$sum:1} } }

... ]

... )

$substr takes a string and two numbers. The first number represents the number of bytes in the string to skip, and the second number specifies the number of bytes to return from the string.

**Run this again to see what the first characters are:**

db.zips.aggregate(

... [

... { $project : { \_id : { $substr : ["$city",0,1] } } } ,

... { $group : { \_id : "$\_id", n : {$sum:1} } }

... ]

... )

**Then run this to remove the items that returned #s as their first character:**

db.zips.remove ( {city:/^[0-9]/})

**Then run this to find out how many docs are left:**

Db.zips.count()

**Question 3.3**

*Download*[*week3.js*](https://education.10gen.com/static/m102-july-2013/handouts/week3.66b3090f84e3.js)*from the education.10gen.com site. Run this file in the shell to populate in the database of your choice a few documents in collections called:*

*- policies*

*- customers*

*- staffers*

In order to do this, download the file to your local system (I created a folder called mongotraining and placed it inside). ls into the directory in linux and then perform the following

vagrant@precise64:~$ cd mongotraining/

vagrant@precise64:~/mongotraining$ ls

homework2.js week3.js zips.json

vagrant@precise64:~/mongotraining$ mongo localhost/week3 week3.js

MongoDB shell version: 2.4.5

connecting to: localhost/week3

vagrant@precise64:~/mongotraining$ mongo

MongoDB shell version: 2.4.5

connecting to: test

> show dbs

local 0.078125GB

pcat 0.203125GB

test (empty)

**week3 0.203125GB**

zips 0.203125GB

> use week3

switched to db week3

> show collections

customers

policies

staffers

system.indexes

As requested, check to ensure there is data in each collection be entering

db.policies.find().pretty()

db.customers.find().pretty()

db.staffers.find().pretty()

**Test against the collection and find that the only option that returns results is:**

db.policies.find( { status : { $ne : "expired" }, coverages : { $elemMatch : { type : "liability", rates : { $elemMatch : { rate : { $gte : 100 }, current : true } } } } } )

**Question 3.4**

Question: Given the map function above, how many zip codes in PA are closer to Philadelphia? Use map/reduce to find the answer.

The method map\_closest() will already be defined. You can type it at the shell prompt without parameters to see its source code:

**Map:**

> map\_closest

function map\_closest() {

var pitt = [-80.064879, 40.612044];

var phil = [-74.978052, 40.089738];

function distance(a, b) {

var dx = a[0] - b[0];

var dy = a[1] - b[1];

return Math.sqrt(dx \* dx + dy \* dy);

}

if (distance(this.loc, pitt) < distance(this.loc, phil)) {

emit("pitt", 1);

} else {

emit("phil", 1);

}

}

>

**Reduce:**

---- 3.4 (this is someone else’s work I found online as a reference for myself)

// Tous les zip de l'état PA

**> db.zips.find({state: /^PA$/i},{state:1}).pretty()**

// Map (donné dans week3.js)

**> map\_closest**

**function map\_closest() {**

**var pitt = [-80.064879, 40.612044];**

**var phil = [-74.978052, 40.089738];**

**function distance(a, b) {**

**var dx = a[0] - b[0];**

**var dy = a[1] - b[1];**

**return Math.sqrt(dx \* dx + dy \* dy);**

**}**

**if (distance(this.loc, pitt) < distance(this.loc, phil)) {**

**emit("pitt", 1);**

**} else {**

**emit("phil", 1);**

**}**

**}**

// Faire le reduce

**r = function( key , values ){**

**var total = 0;**

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

**total += parseInt(values[i]);**

**print("Total : "+total);**

**}**

**return { count : total };**

**};**

\*\*This reduce yields NaN in the “count” field and cannot be used. Something is wrong with it – It appears that this requires you to work straight through from 3.1 and 3.2 without the changeover from 3.3. Bad homework assignment skipped around and caused issues with our work

// Bad resultat

**> db.zips.mapReduce(map\_closest, r, {out :{inline:1}, query : {state : /^PA$/i}})**

{

"results" : [

{

"\_id" : "phil",

"value" : {

"count" : 732

}

},

{

"\_id" : "pitt",

"value" : {

"count" : 726

}

}

],

"timeMillis" : 142,

"counts" : {

"input" : 1458,

"emit" : 1458,

"reduce" : 19,

"output" : 2

},

"ok" : 1,

}

Résultat : 732

Db.zips.mapReduce called the Map Function **map\_closest**, the reduce function **“r”** (which was created and saved above), exports the data inline via the **out :{inline:1}** portion and runs the query **{state : /^PA$/i}**

**Alternative Call:**

**function** r (key, values) {

**var** result = 0;

**for**(**var** i = 0; i < values.length; i++)

{

result += values[i];

}

**return** result;

}

> db.zips.mapReduce(

... map\_closest,

... r,

... {

... out: "map\_reduce\_example",

... query: { "state" : "PA" }

... }

... )

{

"result" : "map\_reduce\_example",

"timeMillis" : 262,

"counts" : {

"input" : 1458,

"emit" : 1458,

"reduce" : 19,

"output" : 2

},

"ok" : 1,

}

In this case, you create the collection **“map\_reduce\_example”** (instead of keeping the data inline, as above) and must locate the information after this completes via a call to **db.map\_reduce\_example.find()**.

> db.map\_reduce\_example.find()

{ "\_id" : "phil", "value" : 732 }

{ "\_id" : "pitt", "value" : 726 }