**Objective**

* Learn to use a popular document database MongoDB
* Learn to use the CRUD (create, retrieve, update, delete) methods of Mongo DB
* Code a Map Reduce program for Mongo DB
* Use Map Reduce and sort-join join method to join two collections.

**Setup**

* download mongoDB community edition <https://www.mongodb.com/download-center/community>
* you may want to use homebrew to install on OSX. For windows you can install use the windows msi download
* to start the server use the command: **> mongod**
  + commands are found in the /bin directory where mongoDB was installed.
  + if you use homebrew or windows msi installer, then the server will already be running and you can skip this step
* to start the shell, use the command: **> mongo**
* by default, the shell will create a use a mongoDB database named “test”
* the shell command **use foo** will switch to or create a database “foo”
* to run a script file **> mongo script.js**

**Requirements and Directions**

* download the files from iLearn with sample data: customers\_load.js and orders\_load.js
* You can use the “test” database or create a new database. Run the scripts to create and load collections “customers” and “orders”

**mongo customers\_load.js**

**mongo orders\_load.js**

* Familiarize yourself with the customer and order document schema (see page 5)

**Part 1**

* familiarize yourself with mongodb shell commands to insert, retrieve, update and delete documents.

1. using the insert method, create 3 documents in a collection named **“patients”**. Each patient document has attributes: name, ssn, age, address. Patients ages should be 10, 20 and 30.
2. The patient with age 30 has a list of prescriptions

prescriptions : [

{ id: "RX743009", tradename : "Hydrochlorothiazide" },

{ id : "RX656003", tradename : "LEVAQUIN", formula : "levofloxacin" }

]

1. Retrieve and list all patient data.
2. Retrieve the patient document whose age is equal to 20.
3. Retrieve the patients where age is less than 25.
4. Using the drop method to delete the entire collection.
5. Convert the commands you used in steps 1-6 into a script1.js file. Read the next section “Coding a mongo script file”. Test your script using the command

mongo script1.js

note: if the script1 file is not in the same directory as the mongo executable, you will have to give directory path to the script1.js file.

**Coding a mongo script.js file**

* When using the interactive shell, the shell will automatically print out results of commands and queries. But when used in a script there will be no output.
* To print an object use printjson( ) statement

doca = db.col.findOne({name: "tom"})

print("Document with name tom")

printjson(doca)

* To print out a value such as a string or number

print("hello world")

* The find( ) command returns a cursor over a list of objects. To print the objects use the code

cursor = db.col.find({name: "tom"})

print("Displaying all documents with name tom")

while (cursor.hasNext()){

printjson(cursor.next())

}

**Part 2**

* code a script2.js file that does a map reduce of the customers collections and produces a report that **shows zip code that start with ‘9’ and the count of customers for each zip code.** There should be 51 zipcodes printed.

**Part 3**

* code a script3.js file that does a map reduce that answers this question? What is the average quantity for orders? If an order includes

{ itemNo: 1, qty: 4 }, { itemNo: 2, qty: 1}

the total quantity for this order is 5.

**Your script calculates the average quantity and displays a single number.** You should have an average of 2.675 items per order.

**Extra Credit (+20 points)**

* code a script4.js file that does uses map reduce to do a join of the customers and orders collections and **summarizes the quantity of items sold by zip code. Your output should have for each zip code, the count of items sold to customers in that zip code.**  There are 27 zip codes with items sold and the zip code with the most items sold is 38101.

**What to submit for this assignment?**

The 3 script files for map reduce script1.js, script2.js, script3.js

If you do the extra credit, then you will also submit file script4.js

**Mongo Reference of common comands**

> mongod start the server

> mongo start the shell

> quit() quit the shell

> db displays the current database being used

> use foobar switches database to foobar. Creates foobar if it does not exist.

> db.coll.insert( object )

inserts a document into collection “coll”. If “coll” does not exist, it is created.

> db.coll.find( ) displays all documents in collection “coll”

> db.coll.findOne( ) displays one documents in collection “coll”

> db.coll.update( {attribute: value}, object ) update the document

> db.coll.remove(object ) delete the document

> db.coll.drop( ) drops collection

> db.coll.mapReduce( mapF, reduceF, {out: {reduce: "mr\_coll"}})

performs map reduce operation over collection “coll” using map function “mapF” and reduce function “reduceF”, the output is merged into collection “mr\_coll” using the reduce function to combine entries.

> db.coll.mapReduce( mapF, reduceF, {out: "mr\_coll"})

performs map reduce operation over collection “coll” using map function “mapF” and reduce function “reduceF”. The ouptut if written to collection “mr\_coll”. If this collection already exists, it is replaced.

**A sample customer document**

> db.customers.findOne()

{

"\_id" : ObjectId("5e8233da9c97cf2cbbf58c30"),

"customerId" : 6,

"customer\_name" : "California Chamber Of Commerce",

"address" : {

"street" : "3255 Ramos Cir",

"city" : "CA",

"state" : "Sacramento",

"zip" : "95827"

},

"contact" : {

"last\_name" : "Mauro",

"first\_name" : "Anton"

}

}

**A sample order document**

Most orders have only a single item. But some have multiple items.

{

"\_id" : ObjectId("5e823dc1cc7841f049f2ab76"),

"orderId" : 12,

"customer" : 96,

"date" : "2018-04-26",

"total" : 662,

"items" : [

{

"itemNo" : 5,

"qty" : 1

},

{

"itemNo" : 6,

"qty" : 2

},

{

"itemNo" : 7,

"qty" : 3

},

{

"itemNo" : 8,

"qty" : 1

}

]

}