week_9_quiz

Adejare Windokun

Friday, October 24, 2014

```
if (!require(rmongodb)) install.packages('rmongodb')
```

```
## Loading required package: rmongodb
```

```
library(rmongodb)
if (!require(jsonlite)) install.packages('jsonlite')
```

```
## Loading required package: jsonlite
##
## Attaching package: 'jsonlite'
##
## The following object is masked from 'package:utils':
##
##     View
```

```
library(jsonlite)

# install rmongodbHelper package from GitHub
# install.packages("devtools")
if (!require(devtools)) install.packages('devtools')
```

```
## Loading required package: devtools
## WARNING: Rtools is required to build R packages, but is not currently
installed.
##
## Please download and install Rtools 3.1 from http://cran.r-
project.org/bin/windows/Rtools/ and then run find_rtools().
```

```
library(devtools)

if (!require(rmongodbHelper))
devtools::install_github("joyofdata/rmongodbHelper")
```

```
## Loading required package: rmongodbHelper
```

```
library(rmongodbHelper)

# Example of the rmongodbHelper
# json_qry <-
# '{
#   "$or": [
#     {"a":1},
#     {"a":3}
#   ]
```

```
# }'
#
# bson <- rmongodbHelper::json_to_bson(json_qry)
```

connect to the mongodb database

```
mongo = mongo.create()
mongo.is.connected(mongo)
```

```
## [1] TRUE
```

Code to destroy the database and collection if neccessary

```
if (mongo.is.connected(mongo) == TRUE) {
     mongo.drop(mongo, "employment.employees")
     mongo.drop.database(mongo, "employment")
#     #res <- mongo.get.database.collections(mongo, "employment")
#     #print(res)
#     #close connection
#     mongo.destroy(mongo)
 }
```

```
## [1] TRUE
```

1. Create a new MongoDB database called employment.

```
db <- "employment"
mongo <- mongo.create(db=db)
```

2. Insert a new record for Wendy Yasquez into the database and into a collection called employees.

```
ins = '{"Name":"Wendy Yasquez", "Title":"Assistant Professor",
"Salary":86000, "Department":"Computer Science", "Hire_Year":1998}'

#This shows you what the JSON statement looks like before you insert it into
the database
cat(prettify(ins))
```

```
## {
##     "Name": "Wendy Yasquez",
##     "Title": "Assistant Professor",
##     "Salary": 86000,
##     "Department": "Computer Science",
##     "Hire_Year": 1998
## }
```

```
if (mongo.is.connected(mongo)) {
        mongo.insert(mongo, "employment.employees", ins)
}
```

```
## [1] TRUE
```

See what records you have

```
#No query criteria
q = "{}"
if (mongo.is.connected(mongo)) {
    cursor <- mongo.find(mongo, "employment.employees",query = q)
    while (mongo.cursor.next(cursor))
        print(mongo.cursor.value(cursor))
    mongo.cursor.destroy(cursor)
}
```

```
##  _id : 7      544e79c12fbaedb5228805c5
##  Name : 2     Wendy Yasquez
##  Title : 2    Assistant Professor
##  Salary : 16      86000
##  Department : 2   Computer Science
##  Hire_Year : 16   1998

## [1] FALSE
```

3. Write a JavaScript function to insert new professors into the employees collection. Could not do this in R - used Robomongo

db.system.js.save({ _id: "insertProf", value : function (name, title, salary, department, hire_year){ db.employees.insert({Name:name, Title:title, Salary:salary, Department:department, Hire_Year:hire_year})} })

4. Use this function to insert the records for Raoul Dewan, Isabelle Winters, and Jack McDunn. First you have to load the scripts db.loadServerScripts();

   Did this using the function I created:

   insertProf('Raoul Dewan', 'Assistant Professor', 78000, ['Physics', 'Biology'], 2009)
   insertProf('Isabelle Winters', 'Associate Professor', 92000, 'Physics', 1995)
   insertProf('Jack MuDunn', 'Associate Professor', 101000, 'Physics', 1993)

```
ins1 = '{"Name":"Raoul Dewan", "Title":"Assistant Professor", "Salary":78000,
"Department": "[Physics, Biology]", "Hire_Year":2009}'

ins2 = '{"Name":"Isabelle Winters", "Title":"Associate Professor",
"Salary":92000, "Department":"Physics", "Hire_Year":1995}'

ins3 = '{"Name":"Jack MuDunn", "Title":"Associate Professor",
"Salary":101000, "Department":"Physics", "Hire_Year":1993}'

        mongo.insert(mongo, "employment.employees", ins1)
```

```
## [1] TRUE
```

```
        mongo.insert(mongo, "employment.employees", ins2)
```

```
## [1] TRUE
```

```
        mongo.insert(mongo, "employment.employees", ins3)
```

```
## [1] TRUE
```

5. Write a JavaScript function to insert new administrative employees into the employees collection.

db.system.js.save({ _id: "insertAdmins", value : function (name, title, salary, division, location, hire_year){ db.employees.insert({Name:name, Title:title, Salary:salary, Division:division, Location:location, Hire_Year:hire_year}) }})

6. Use this function to insert the records for Tonja Baldner and Dennis Bohnet. db.loadServerScripts(); insertAdmins('Tonja Baldner', 'Assistant to the Dean', 42000, 'Ats and Sciences', '', 2001) insertAdmins('Dennis Bohnet', 'Vice President', 106000, 'Academic Affairs', 'Main Campus', 1997)

```
ins4 = '{"Name":"Tonja Baldner", "Title":"Assistant to the Dean",
"Salary":42000, "Division":"Ats and Sciences", "Location": "",
"Hire_Year":1995}'

ins5 = '{"Name":"Dennis Bohnet", "Title":"Vice President", "Salary":106000,
"Division":"Academic Affairs", "Location": "Main Campus",  "Hire_Year":1993}'

        mongo.insert(mongo, "employment.employees", ins4)
```

```
## [1] TRUE
```

```
        mongo.insert(mongo, "employment.employees", ins5)
```

```
## [1] TRUE
```

7. Show the code that will return all employees with salaries less than $90,000.

db.employees.find({Salary:{$lt:90000}})

```
q = '{"Salary":{"$lt":90000}}'
if (mongo.is.connected(mongo)) {
    cursor <- mongo.find(mongo, "employment.employees",query = q)
    while (mongo.cursor.next(cursor))
        print(mongo.cursor.value(cursor))
    mongo.cursor.destroy(cursor)
}
```

```
##  _id : 7      544e79c12fbaedb5228805c5
##  Name : 2     Wendy Yasquez
##  Title : 2    Assistant Professor
##  Salary : 16      86000
##  Department : 2   Computer Science
##  Hire_Year : 16   1998
##  _id : 7      544e79c32fbaedb5228805c6
##  Name : 2     Raoul Dewan
##  Title : 2    Assistant Professor
```

```
##  Salary : 16      78000
##  Department : 2   [Physics, Biology]
##  Hire_Year : 16   2009
##  _id : 7      544e79c32fbaedb5228805c9
##  Name : 2     Tonja Baldner
##  Title : 2    Assistant to the Dean
##  Salary : 16      42000
##  Division : 2     Ats and Sciences
##  Location : 2
##  Hire_Year : 16   1995

## [1] FALSE
```

8. Show the code that will return all professors with salaries less than $90,000.

db.employees.find({Title: { $regex: /PROFESSOR/i }, Salary:{$lt:90000}})

9. Show the code that will return all Physics professors hired before 2001.

db.employees.find({Department:'Physics', Hire_Year:{$lt:2001}})

```
json_qry <-
 '{
   "Department": "Physics",
     "Hire_Year":
        {"$lt":2001}

 }'

cat(prettify(json_qry))
```

```
## {
##     "Department": "Physics",
##     "Hire_Year": {
##         "$lt": 2001
##     }
## }
```

```
bson <- rmongodbHelper::json_to_bson(json_qry)

if (mongo.is.connected(mongo)) {
    cursor <- mongo.find(mongo, "employment.employees", query = bson)
    while (mongo.cursor.next(cursor))
        print(mongo.cursor.value(cursor))
    mongo.cursor.destroy(cursor)
}
```

```
##  _id : 7      544e79c32fbaedb5228805c7
##  Name : 2     Isabelle Winters
##  Title : 2    Associate Professor
##  Salary : 16      92000
##  Department : 2   Physics
```

```
##  Hire_Year : 16   1995
##  _id : 7      544e79c32fbaedb5228805c8
##  Name : 2     Jack MuDunn
##  Title : 2    Associate Professor
##  Salary : 16      101000
##  Department : 2   Physics
##  Hire_Year : 16   1993

## [1] FALSE
```

10. Show the code that will return all professors who teach for departments other than Physics. (This should include professors who teach for Physics and also other departments.)

db.employees.find({ $and: [{$or: [{Department:{$ne:'Physics'}},

```
        {'Department.1': {$exists: true}}

    ]},

 {Title: { $regex: /PROFESSOR/i }}
```

]})

11. Show the code that will return all employees who were either hired before 1997 or who have salaries greater than $100,000.

db.employees.find({$or: [{Salary:{$gt:100000}}, {Hire_Year:{$lt:1997}}] })

```
json_qry <-
 '{
    "$or": [{
        "Salary":
            {"$gt":100000}},
        {"Hire_Year":
            {"$lt":1997}} ]

 }'

cat(prettify(json_qry))

## {
##     "$or": [
##         {
##             "Salary": {
##                 "$gt": 100000
##             }
##         },
##         {
##             "Hire_Year": {
##                 "$lt": 1997
```

```
##             }
##         }
##     ]
## }
```

```
bson <- rmongodbHelper::json_to_bson(json_qry)

if (mongo.is.connected(mongo)) {
    cursor <- mongo.find(mongo, "employment.employees", query = bson)
    while (mongo.cursor.next(cursor))
        print(mongo.cursor.value(cursor))
    mongo.cursor.destroy(cursor)
}
```

```
##  _id : 7      544e79c32fbaedb5228805c7
##  Name : 2     Isabelle Winters
##  Title : 2    Associate Professor
##  Salary : 16      92000
##  Department : 2   Physics
##  Hire_Year : 16   1995
##  _id : 7      544e79c32fbaedb5228805c8
##  Name : 2     Jack MuDunn
##  Title : 2    Associate Professor
##  Salary : 16      101000
##  Department : 2   Physics
##  Hire_Year : 16   1993
##  _id : 7      544e79c32fbaedb5228805c9
##  Name : 2     Tonja Baldner
##  Title : 2    Assistant to the Dean
##  Salary : 16      42000
##  Division : 2     Ats and Sciences
##  Location : 2
##  Hire_Year : 16   1995
##  _id : 7      544e79c32fbaedb5228805ca
##  Name : 2     Dennis Bohnet
##  Title : 2    Vice President
##  Salary : 16      106000
##  Division : 2     Academic Affairs
##  Location : 2     Main Campus
##  Hire_Year : 16   1993

## [1] FALSE
```

12. Suppose Tonja Baldner has been given a 10% raise. Show the code that will update her salary correctly. db.employees.update({ Name: 'Tonja Baldner'}, { $mul: { Salary: 1.1 } })

```
cursor <- mongo.find.one(mongo, "employment.employees", query =
'{"Name":"Tonja Baldner"}')
print("Before Update")
```

```
## [1] "Before Update"
```

```
print(cursor)
```

```
##  _id : 7      544e79c32fbaedb5228805c9
##  Name : 2     Tonja Baldner
##  Title : 2    Assistant to the Dean
##  Salary : 16      42000
##  Division : 2     Ats and Sciences
##  Location : 2
##  Hire_Year : 16   1995
```

```
crit = '{ "Name": "Tonja Baldner"}'
obj = '{ "$mul": { "Salary": 1.1 } }'

mongo.update(mongo, "employment.employees", criteria = crit, objNew= obj)
```

```
## [1] TRUE
```

```
cursor <- mongo.find.one(mongo, "employment.employees", query =
'{"Name":"Tonja Baldner"}')
print("After Update")
```

```
## [1] "After Update"
```

```
print(cursor)
```

```
##  _id : 7      544e79c32fbaedb5228805c9
##  Name : 2     Tonja Baldner
##  Title : 2    Assistant to the Dean
##  Salary : 1   46200.000000
##  Division : 2     Ats and Sciences
##  Location : 2
##  Hire_Year : 16   1995
```

Have to do question 14 first, otherwise, Prof Dewan is already deleted from the database 14. Instead of removing Professor Dewan□s record, we might prefer to create a new collection called pastemployees and move his record there. Show the code that will move his record to the new collection and add a departyear value of 2014 to his record. (You can do it in two steps.)

Will first copy over the document from the employees collection to the newly created pastemployees collection Will then insert the new field = departyear and update it to 2014

Do everything in one step:

db.employees.find({Name: 'Raoul Dewan'}).forEach(function(doc){
db.pastemployees.insert(doc) db.pastemployees.update(doc,{$set : {"departyear":2014}})
db.employees.remove(doc)

});

```
cursor <- mongo.find.one(mongo, "employment.employees",query =
'{"Name":"Raoul Dewan"}')

print (cursor)

##  _id : 7      544e79c32fbaedb5228805c6
##  Name : 2     Raoul Dewan
##  Title : 2    Assistant Professor
##  Salary : 16      78000
##  Department : 2   [Physics, Biology]
##  Hire_Year : 16   2009

mongo.insert(mongo, "employment.pastemployees", cursor)

## [1] TRUE

crit = '{"Name":"Raoul Dewan"}'
objNew = '{"$set" : {"departyear":2014}}'
mongo.update(mongo, "employment.pastemployees", criteria = crit, objNew)

## [1] TRUE
```

13. Professor Dewan has been offered a job at another university. Show the code that would remove his record from the database. db.employees.remove({ Name : "Raoul Dewan" }, 1)

```
crit = '{ "Name": "Raoul Dewan"}'
mongo.remove(mongo, "employment.employees", criteria = crit)

## [1] TRUE
```