MSDE 631: Lab 5—Using and Learning the MongoDB Environment
Etana Disasa

Installing MongoDB on a Mac OS

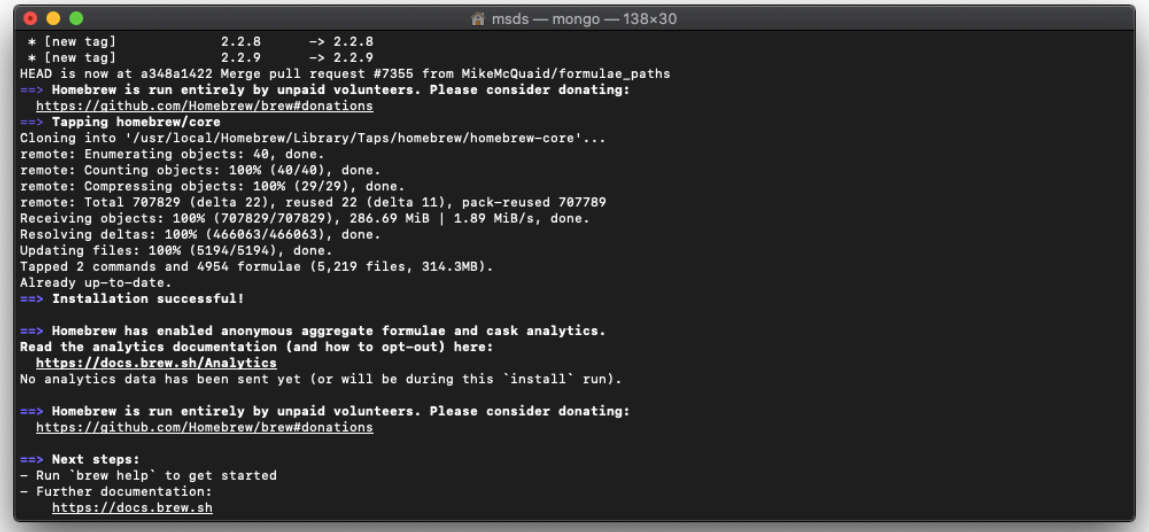
First, I needed to upgrade the operating system on my computer to install the MongoDB envornment and all the tools it needs.

It was that XCode was installed.

MSDE 631: Lab 5—Using and Learning the MongoDB Environment
Etana Disasa

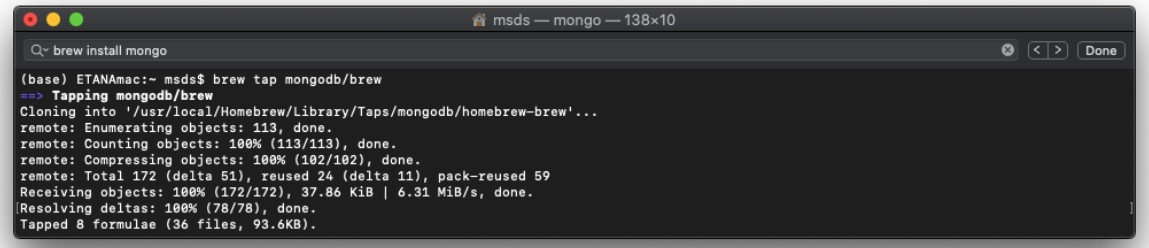
Installing Homebrew

```
ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```



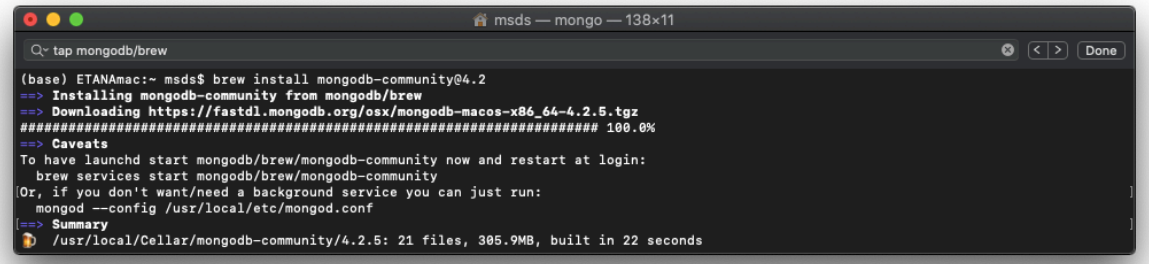
Tapping Homebrew services in order to start the MongoDB server

```
brew tap mongodb/brew
```



Install MongoDB Community Edition 4.2

```
brew install mongodb-community@4.2
```



MSDE 631: Lab 5—Using and Learning the MongoDB Environment
Etana Disasa

Start Brew Services
`brew services start mongodb-community@4.2`

```
msds — mongo — 138×11
brew install
(base) ETANAmac:~ msds$ brew services start mongodb-community@4.2
==> Tapping homebrew/services
Cloning into '/usr/local/Homebrew/Library/Taps/homebrew/homebrew-services'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 698 (delta 0), reused 3 (delta 0), pack-reused 691
Receiving objects: 100% (698/698), 193.18 KiB | 928.00 KiB/s, done.
Resolving deltas: 100% (272/272), done.
Tapped 1 command (40 files, 267.3KB).
==> Successfully started `mongodb-community` (label: homebrew.mxcl.mongodb-commu
```

Start mongo shell
`mongo <return>`

```
msds — mongo — 138×30
MongoDB shell version v4.2.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("1e41d373-1c57-42e8-8202-9e8db307f249") }
MongoDB server version: 4.2.5
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten]
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten]
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten]
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should be at least 1000
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

>
```

The help command: `help <return>`

```
msds — mongo — 138×18
brew install mongo
        help connect                connecting to a db help
        help keys                   key shortcuts
        help misc                   misc things to know
        help mr                     mapreduce

        show dbs                    show database names
        show collections            show collections in current database
        show users                  show users in current database
        show profile                show most recent system.profile entries with time >= 1ms
        show logs                   show the accessible logger names
        show log [name]             prints out the last segment of log in memory, 'global' is default
        use <db_name>               set current database
        db.foo.find()               list objects in collection foo
        db.foo.find( { a : 1 } )    list objects in foo where a == 1
        it                          result of the last line evaluated; use to further iterate
        DBQuery.shellBatchSize = x  set default number of items to display on shell
        exit                        quit the mongo shell
>
```

MSDE 631: Lab 5—Using and Learning the MongoDB Environment
Etana Disasa

Exiting mongodb

```
msds — -bash — 138×5
brew install mongo                                                    Done
---

> exit
bye
(base) ETANAmac:~ msds$
```

Mongo book

```
msds — mongo book — 138×25
brew install mongo                                                    Done
(base) ETANAmac:~ msds$ mongo book
MongoDB shell version v4.2.5
connecting to: mongodb://127.0.0.1:27017/book?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("72866926-f4bd-4d52-9ec0-fd1aa9a1a7f3") }
MongoDB server version: 4.2.5
Server has startup warnings:
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten]
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten]
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten]
2020-04-14T15:31:17.706-0600 I  CONTROL  [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should be at least 1000
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

>
```

```
db.towns.insert({ name: "New York", population: 22200000, last_census:
ISODate("2009-07-31"), famous_for: [ "Statue of Liberty", "Food" ], mayor: {
name: "Bill de Blasio", party: "I" }})
```

```
msds — mongo book — 138×13

> db.towns.insert({
... name: "New York",
... population: 22200000,
... last_census: ISODate("2009-07-31"),
... famous_for: [ "Statue of Liberty", "Food" ],
... mayor: {
... name: "Bill de Blasio",
... party: "I"
... }
... })
WriteResult({ "nInserted" : 1 })
>
```

```
msds — mongo book — 138×8
> show dbs
admin   0.000GB
book    0.000GB
config  0.000GB
local   0.000GB
> show collections
towns
>
```

MSDE 631: Lab 5—Using and Learning the MongoDB Environment
Etana Disasa

1. Lets continue our exploration
    a. Enter `Help admin <return>`
    b. Now look at the list and enter the first command: `ls <return>`

```
msds — mongo book — 157×14
> help admin
        ls([path])                      list files
        pwd()                           returns current directory
        listFiles([path])               returns file list
        hostname()                      returns name of this host
        cat(fname)                      returns contents of text file as a string
        removeFile(f)                   delete a file or directory
        load(jsfilename)                load and execute a .js file
        run(program[, args...])         spawn a program and wait for its completion
        runProgram(program[, args...])  same as run(), above
        sleep(m)                        sleep m milliseconds
        getMemInfo()                    diagnostic
> ls
[native code]
```

    c. Now `try ls() <return>`
    What happens?

```
msds — mongo book — 157×14
        "./Documents/",
        "./.rstudio-desktop/",
        "./.pgAdmin4.5976741433536694823.addr",
        "./.bash_profile",
        "./anaconda3/",
        "./Downloads/",
        "./.continuum/",
        "./.mongorc.js",
        "./.bash_history",
        "./.viminfo",
        "./anaconda2/",
        "./.conda/"
]
>
```

    d. Try: `listfile() <return>`

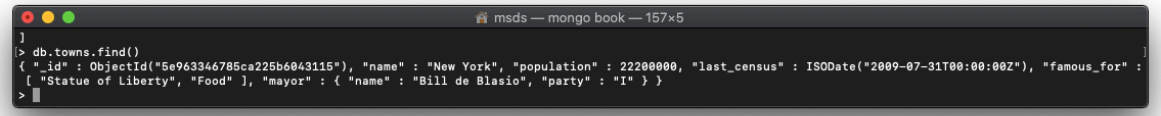
```
msds — mongo book — 157×5
]
> listfiles()
2020-04-14T16:09:41.523-0600 E  QUERY    [js] uncaught exception: ReferenceError: listfiles is not defined :
@(shell):1:1
>
```

    e. Now try: `listFiles() <return>`

```
msds — mongo book — 157×19
        },
        {
                "name" : "./.viminfo",
                "baseName" : ".viminfo",
                "isDirectory" : false,
                "size" : 1799
        },
        {
                "name" : "./anaconda2",
                "baseName" : "anaconda2",
                "isDirectory" : true
        },
        {
                "name" : "./.conda",
                "baseName" : ".conda",
                "isDirectory" : true
        }
]
>
```

Let's list your collection.
    f. Enter: `db.towns.find()<return>`

MSDE 631: Lab 5—Using and Learning the MongoDB Environment
Etana Disasa

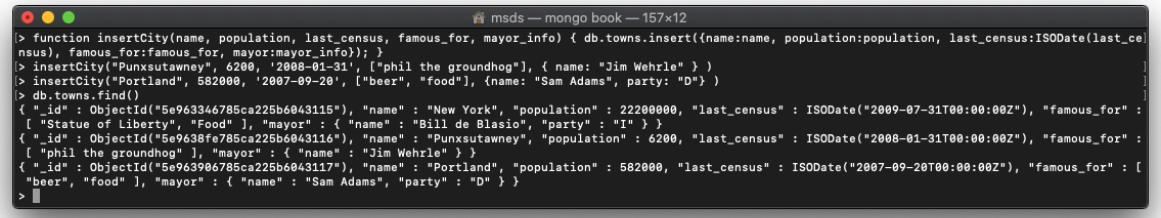


2. Working with data in MongoDB
   a. Go ahead and enter: `db.towns.insert<return>`
   b. Now lets turn the page and enter the **`InserCity`** function:

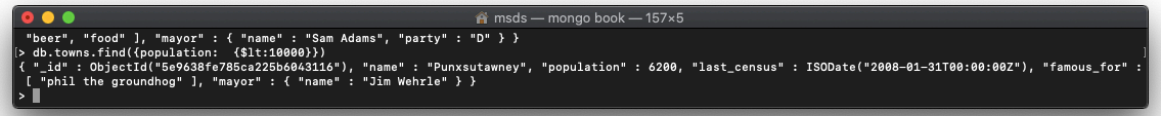
```
function insertCity(
name, population, last_census,
 famous_for, mayor_info
) {
db.towns.insert({
   name:name,
   population:population,
   last_census: ISODate(last_census),
   famous_for:famous_for,
   mayor: mayor_info});
}
<return>
```

   Question: What did you really just do above?
   Answer: *We defined a function `insertCity()` by first declaring what values it takes and using the MongoDB `insert()` function, we are telling our custom function where to insert the information.*

   c. Go ahead and enter the two additional cities found on page 140 into the database.
   d. Verify their existence: `db.towns.find() <return>`



3. Now let's look at some queries:
   a. Enter: `db.towns.find({population: {$lt:10000}})`



Question: What happened this time?
Answer: *This returns all the towns with population less than 10,000 with all the fields in the data.*

b. Now try limiting your results: `db.towns.find({population: {$lt:10000}},{name: 1, population:1})`

```
msds — mongo book — 157×5
{ "_id" : ObjectId("5e9638fe785ca225b6043116"), "name" : "Punxsutawney", "population" : 6200, "last_census" : ISODate("2008-01-31T00:00:00Z"), "famous_for" :
 [ "phil the groundhog" ], "mayor" : { "name" : "Jim Wehrle" } }
> db.towns.find({population: {$lt:10000}},{name: 1, population:1})
{ "_id" : ObjectId("5e9638fe785ca225b6043116"), "name" : "Punxsutawney", "population" : 6200 }
>
```

Question: What happened this time?
Answer: *The result only populates the id, name and populations of a town with population less than 10,000.*

c. Let's try the range query. Enter the following:

```
var population_range = {}
population_range['$lt']=1000000
population_range['$gt']=10000
db.towns.find(
{ population : population_range},
{name : 1})
```

```
msds — mongo book — 157×10
> var population_rante = {}
> population_range['$lt']=1000000
1000000
> population_range['$gt']=10000
10000
> db.towns.find(
... {population : population_range},
... {name : 1})
{ "_id" : ObjectId("5e963906785ca225b6043117"), "name" : "Portland" }
>
```

d. **Question**: Now you try it. Develop a query that will find all towns with the string 'and' in their name.
Answer:
`db.towns.find( { name: { $regex: /and$/ } } )`

```
msds — mongo book — 157×5
... {name : 1})
{ "_id" : ObjectId("5e963906785ca225b6043117"), "name" : "Portland" }
> db.towns.find( { name: { $regex: /and$/ } } )
{ "_id" : ObjectId("5e963906785ca225b6043117"), "name" : "Portland", "population" : 582000, "last_census" : ISODate("2007-09-20T00:00:00Z"), "famous_for" : [
 "beer", "food" ], "mayor" : { "name" : "Sam Adams", "party" : "D" } }
```

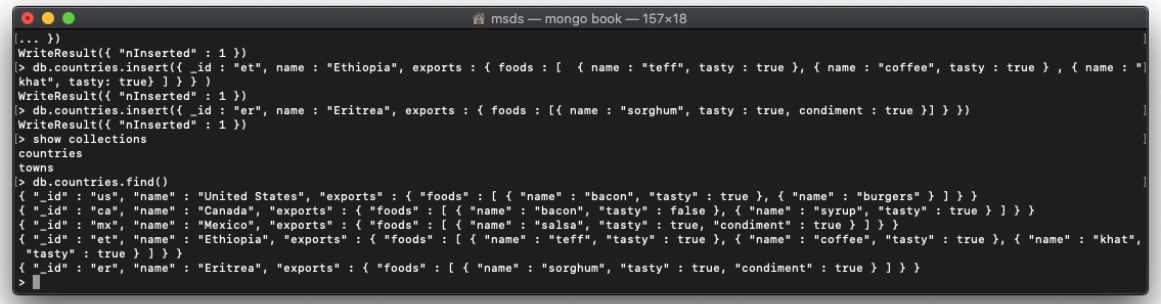
4. On page 143 of the Seven Weeks text, we see how to override the `_id` field with a value of our own. Insert the three countries and two of your own so you have five altogether into your MongoDB database.
   a. Here is the first one:

```
db.countries.insert({
_id: "us",
name : "United States",
exports : {
foods : [
```

```
{name : "bacon", tasty : true},
{name: "burgers"}
]
}
})
```

b. Now execute: `show collections <return>`

c. Notice you've created a new collection? Now enter the remaining four countries on your own.
Execute a `db.countries.find()` statement and post a screenshot of your results below.

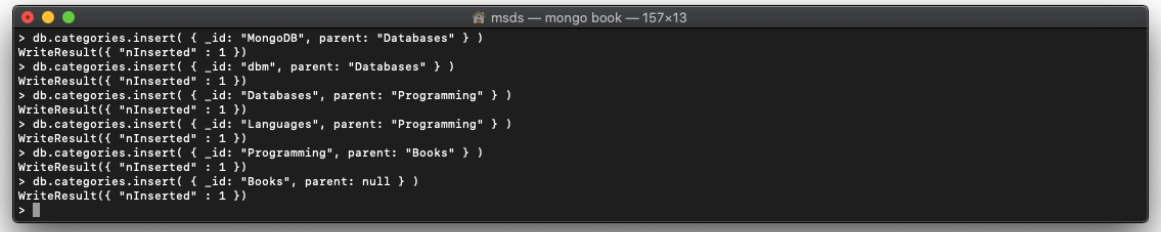


5. Now, lets look at what we did:
   a. Enter: `print(db.countries.count()) <return>`
   b. Now try: `db.countries.find(`
   `{ 'exports.foods.name' : 'bacon', 'exports.foods.tasty' : true},`
   `{_id: 0, name: 1}`
   `)`

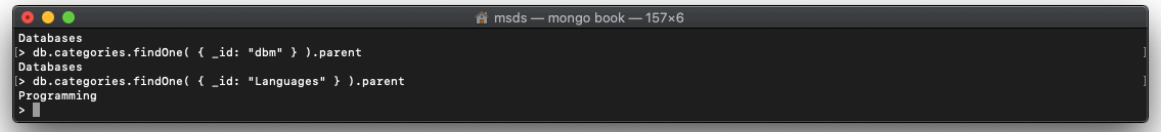
6. Let's create the database:

```
db.categories.insert( { _id: "MongoDB", parent: "Databases" } )
db.categories.insert( { _id: "dbm", parent: "Databases" } )
db.categories.insert( { _id: "Databases", parent: "Programming" } )
db.categories.insert( { _id: "Languages", parent: "Programming" } )
db.categories.insert( { _id: "Programming", parent: "Books" } )
db.categories.insert( { _id: "Books", parent: null } )
```

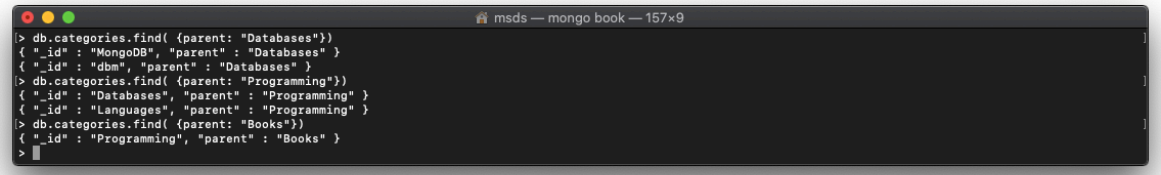
MSDE 631: Lab 5—Using and Learning the MongoDB Environment
Etana Disasa



7. Find the MongoDB document parent:
    a. Enter:
    b. Find the parent of the dbm document:
    c. Now find the parent of the Languages document.



8. **Question:** Now try to write a query to find the children of the Databases document. (Look carefully at the results of your query. It should produce the proper answer. This requires some thought and/or investigation to get the correct answer.)
   Answer:
   db.categories.find({parent: "Databases"})
   db.categories.find({parent: "Programming"})
   db.categories.find({parent: "Books"})

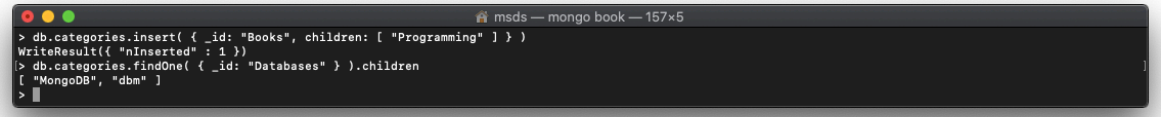


9. Now lets look at the reverse type of pattern, where we insert child references:
    a. First, lets clean up our collection: `db.categories.drop()<return>`
    b. Now lets add the same data using the child pattern:

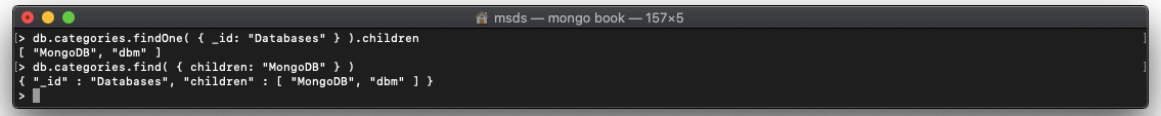
```
db.categories.insert( { _id: "MongoDB", children: [] } )
db.categories.insert( { _id: "dbm", children: [] } )
db.categories.insert( { _id: "Databases", children: [ "MongoDB", "dbm" ] } )
db.categories.insert( { _id: "Languages", children: [] } )
db.categories.insert( { _id: "Programming", children: [ "Databases", "Languages" ] } )
db.categories.insert( { _id: "Books", children: [ "Programming" ] } )
```

Question: What does the empty [ ] mean?
Answer: *The document does not have a child.*

10. Now…
    a. What is the child of the Databases node:
       `db.categories.findOne( { _id: “Databases” } ).children`



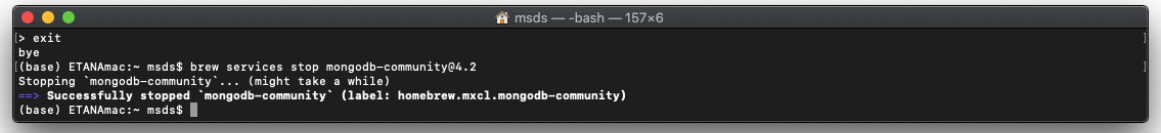
    b. Now enter: `db.categories.find( { children: “MongoDB” } )`
       Question: What happens?
       Answer: *It returns the id and children of the document entry whose children includes MongoDB; in this case Databases.*



To stop your mongo client, enter: `exit<return>`

To stop your mongod server follow the instructions in the "How to access the Databases" document on your desktop.

`brew services stop mongodb-community@4.2`



End of Assignment