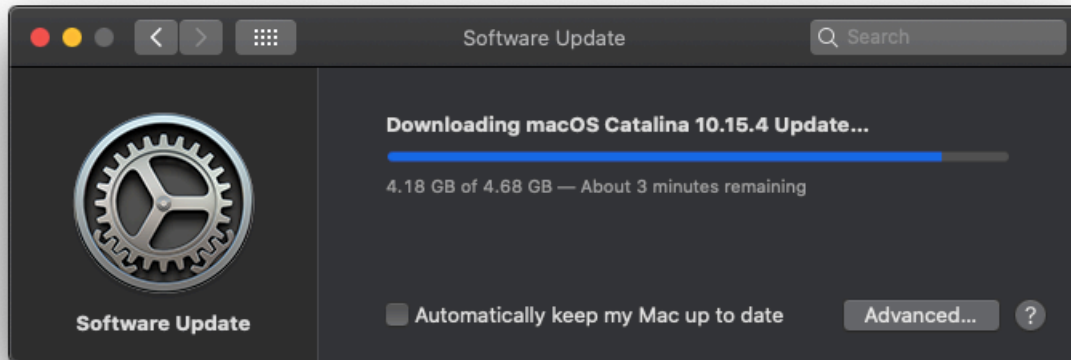


## MSDE 631: Lab 5—Using and Learning the MongoDB Environment

Etana Disasa

### Installing MongoDB on a Mac OS

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



It was that XCode was installed.

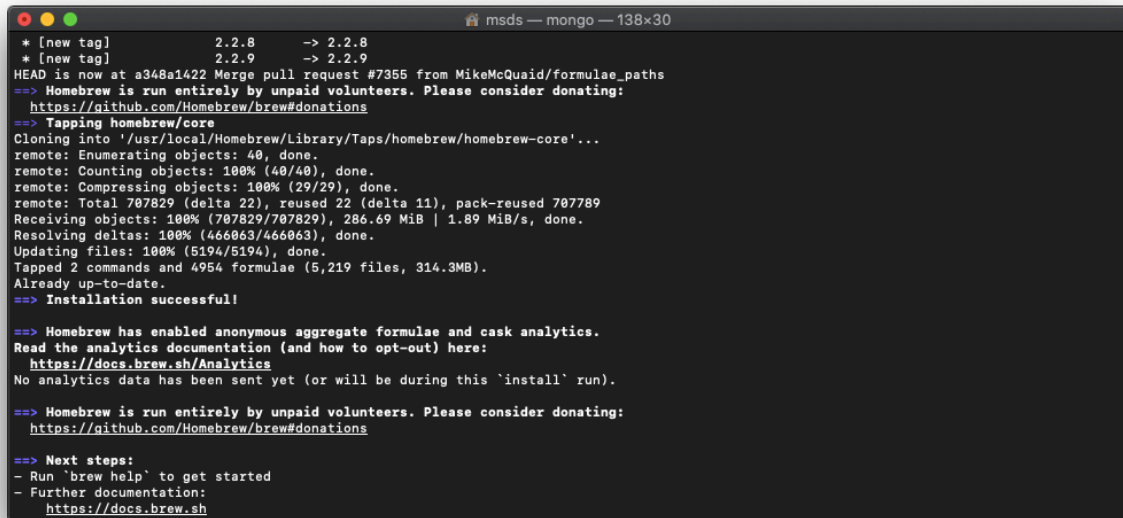


## MSDE 631: Lab 5—Using and Learning the MongoDB Environment

### Etana Disasa

Installing Homebrew

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



```
msds — mongo — 138x30
* [new tag] 2.2.8 -> 2.2.8
* [new tag] 2.2.9 -> 2.2.9
HEAD is now at a348a1422 Merge pull request #7355 from MikeMcQuaid/formulae_paths
==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
    https://github.com/Homebrew/brew#donations
==> Tapping homebrew/core
Cloning into '/usr/local/Homebrew/Library/Taps/homebrew/homebrew-core'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 707829 (delta 22), reused 22 (delta 11), pack-reused 707789
Receiving objects: 100% (707829/707829), 286.69 MiB | 1.89 MiB/s, done.
Resolving deltas: 100% (466063/466063), done.
Updating files: 100% (5194/5194), done.
Tapped 2 commands and 4954 formulae (5,219 files, 314.3MB).
Already up-to-date.
==> Installation successful!

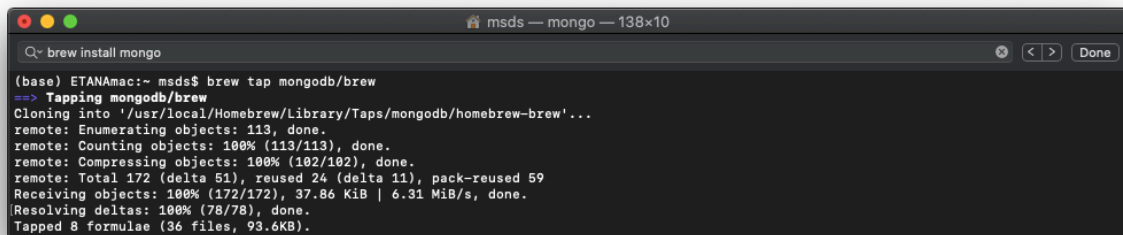
==> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
    https://docs.brew.sh/Analytics
No analytics data has been sent yet (or will be during this 'install' run).

==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
    https://github.com/Homebrew/brew#donations

==> Next steps:
- Run 'brew help' to get started
- Further documentation:
    https://docs.brew.sh
```

Tapping Homebrew services in order to start the MongoDB server

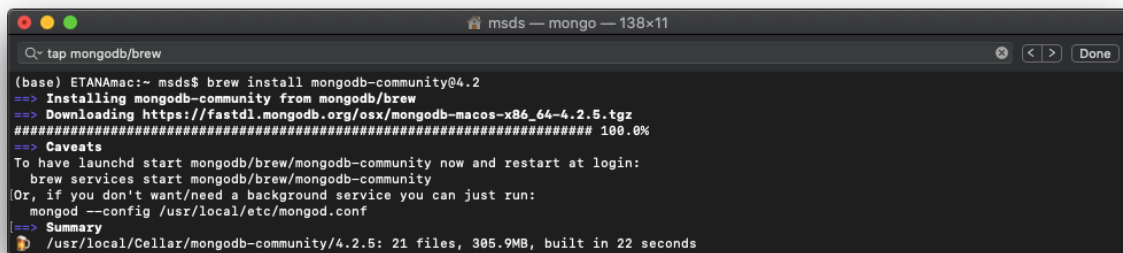
```
brew tap mongodb/brew
```



```
msds — mongo — 138x10
Q- brew install mongo
(base) ETANAmac:~ msds$ brew tap mongodb/brew
==> Tapping mongodb/brew
Cloning into '/usr/local/Homebrew/Library/Taps/mongodb/homebrew-brew'...
remote: Enumerating objects: 113, done.
remote: Counting objects: 100% (113/113), done.
remote: Compressing objects: 100% (102/102), done.
remote: Total 172 (delta 51), reused 24 (delta 11), pack-reused 59
Receiving objects: 100% (172/172), 37.86 KiB | 6.31 MiB/s, done.
Resolving deltas: 100% (78/78), done.
Tapped 8 formulae (36 files, 93.6KB).
```

Install MongoDB Community Edition 4.2

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



```
msds — mongo — 138x11
Q- tap mongodb/brew
(base) ETANAmac:~ msds$ brew install mongodb-community@4.2
==> Installing mongodb-community from mongodb/brew
==> Downloading https://fastdl.mongodb.org/osx/mongodb-macos-x86_64-4.2.5.tgz
##### 100.0%
==> Caveats
To have launchd start mongodb/brew/mongodb-community now and restart at login:
  brew services start mongodb/brew/mongodb-community
Or, if you don't want/need a background service you can just run:
  mongod --config /usr/local/etc/mongod.conf
[=> Summary
/usr/local/Cellar/mongodb-community/4.2.5: 21 files, 305.9MB, built in 22 seconds
```

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

### Etana Disasa

Start Brew Services

brew services start mongodb-community@4.2

```
msds — mongo — 138x11
Q~ 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 — 138x30
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 — 138x18
Q~ 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 mongod

```
msds — -bash — 138x5
brew install mongo
---
> exit
bye
(base) ETANAmac:~ msds$
```

Mongo book

```
msds — mongo book — 138x25
brew install mongo
---
(base) ETANAmac:~ msds$ mongo book
MongoDB shell version v4.2.5
connecting to: mongodb://127.0.0.1:27017/book?compressors=disabled&gssapiServiceName=mongod
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 — 138x13
> 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 — 138x8
> 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 — 157x14
> 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 — 157x14
["./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 — 157x5
]
> 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 — 157x19
],
{
  "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

```
msds — mongo book — 157x5
]
> db.towns.find()
{ "_id" : ObjectId("5e963346785ca225b6043115"), "name" : "New York", "population" : 22200000, "last_census" : ISODate("2009-07-31T00:00:00Z"), "famous_for" :
[ "Statue of Liberty", "Food" ], "mayor" : { "name" : "Bill de Blasio", "party" : "I" } }
>
```

### 2. Working with data in MongoDB

- Go ahead and enter: `db.towns.insert<return>`
- Now lets turn the page and enter the **InsertCity** 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.

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

```
msds — mongo book — 157x12
> 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}); }
> insertCity("Punxsutawney", 6200, '2008-01-31', ["phil the groundhog"], { name: "Jim Wehrle" })
> insertCity("Portland", 582000, '2007-09-20', ["beer", "food"], {name: "Sam Adams", party: "D" })
> db.towns.find()
{ "_id" : ObjectId("5e963346785ca225b6043115"), "name" : "New York", "population" : 22200000, "last_census" : ISODate("2009-07-31T00:00:00Z"), "famous_for" :
[ "Statue of Liberty", "Food" ], "mayor" : { "name" : "Bill de Blasio", "party" : "I" } }
{ "_id" : ObjectId("5e9638fe785ca225b6043116"), "name" : "Punxsutawney", "population" : 6200, "last_census" : ISODate("2008-01-31T00:00:00Z"), "famous_for" :
[ "phil the groundhog" ], "mayor" : { "name" : "Jim Wehrle" } }
{ "_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" } }
>
```

### 3. Now let's look at some queries:

- Enter: `db.towns.find({population: {$lt:10000}})`

```
msds — mongo book — 157x5
"beer", "food" ], "mayor" : { "name" : "Sam Adams", "party" : "D" } }
> db.towns.find({population: {$lt:10000}})
{ "_id" : ObjectId("5e9638fe785ca225b6043116"), "name" : "Punxsutawney", "population" : 6200, "last_census" : ISODate("2008-01-31T00:00:00Z"), "famous_for" :
[ "phil the groundhog" ], "mayor" : { "name" : "Jim Wehrle" } }
>
```



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

Etana Disasa

**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 — 157x5
{ "_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 — 157x10
> var population_range = {}
> 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 — 157x5
... {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 : [
```

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

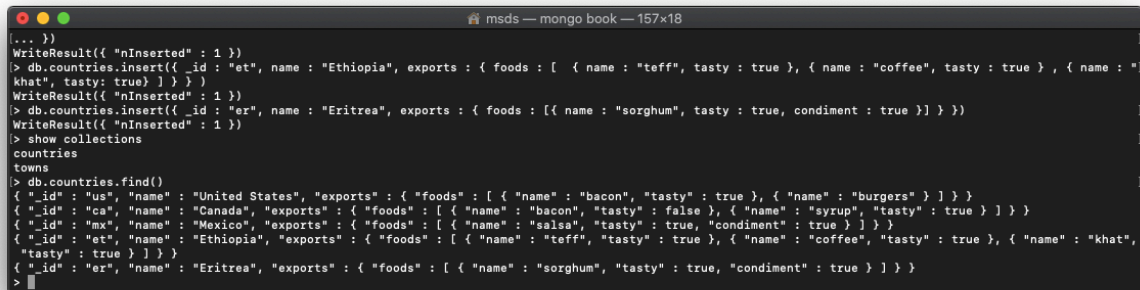
Etana Disasa

```
{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.

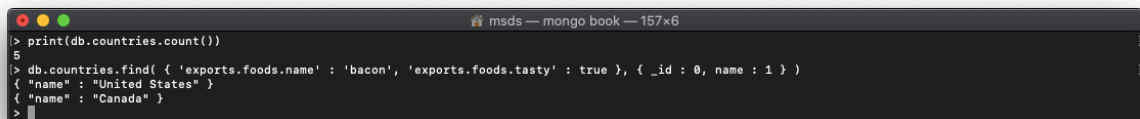


```
msds — mongo book — 157x18  
[... }]  
WriteResult({ "nInserted" : 1 })  
> db.countries.insert({ _id : "et", name : "Ethiopia", exports : { foods : [ { name : "teff", tasty : true }, { name : "coffee", tasty : true }, { name : "khat", tasty : true } ] } })  
WriteResult({ "nInserted" : 1 })  
> db.countries.insert({ _id : "er", name : "Eritrea", exports : { foods : [ { name : "sorghum", tasty : true, condiment : true } ] } })  
WriteResult({ "nInserted" : 1 })  
> show collections  
countries  
towns  
> db.countries.find()  
{ "_id" : "us", "name" : "United States", "exports" : { "foods" : [ { "name" : "bacon", "tasty" : true }, { "name" : "burgers" } ] } }  
{ "_id" : "ca", "name" : "Canada", "exports" : { "foods" : [ { "name" : "bacon", "tasty" : false }, { "name" : "syrup", "tasty" : true } ] } }  
{ "_id" : "mx", "name" : "Mexico", "exports" : { "foods" : [ { "name" : "salsa", "tasty" : true, "condiment" : true } ] } }  
{ "_id" : "et", "name" : "Ethiopia", "exports" : { "foods" : [ { "name" : "teff", "tasty" : true }, { "name" : "coffee", "tasty" : true }, { "name" : "khat", "tasty" : true } ] } }  
{ "_id" : "er", "name" : "Eritrea", "exports" : { "foods" : [ { "name" : "sorghum", "tasty" : true, "condiment" : true } ] } }  
>
```

5. Now, let's 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}`  
    `)`



```
msds — mongo book — 157x6  
> print(db.countries.count())  
5  
> db.countries.find( { 'exports.foods.name' : 'bacon', 'exports.foods.tasty' : true }, { _id : 0, name : 1 } )  
{ "name" : "United States" }  
{ "name" : "Canada" }  
>
```

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

```
msds — mongo book — 157x13
> db.categories.insert( { _id: "MongoDB", parent: "Databases" } )
WriteResult({ "nInserted" : 1 })
> db.categories.insert( { _id: "dbm", parent: "Databases" } )
WriteResult({ "nInserted" : 1 })
> db.categories.insert( { _id: "Databases", parent: "Programming" } )
WriteResult({ "nInserted" : 1 })
> db.categories.insert( { _id: "Languages", parent: "Programming" } )
WriteResult({ "nInserted" : 1 })
> db.categories.insert( { _id: "Programming", parent: "Books" } )
WriteResult({ "nInserted" : 1 })
> db.categories.insert( { _id: "Books", parent: null } )
WriteResult({ "nInserted" : 1 })
>
```

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.

```
msds — mongo book — 157x6
Databases
> db.categories.findOne( { _id: "dbm" } ).parent
Databases
> db.categories.findOne( { _id: "Languages" } ).parent
Programming
>
```

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"} )
```

```
msds — mongo book — 157x9
> db.categories.find( {parent: "Databases"} )
{ "_id" : "MongoDB", "parent" : "Databases" }
{ "_id" : "dbm", "parent" : "Databases" }
> db.categories.find( {parent: "Programming"} )
{ "_id" : "Databases", "parent" : "Programming" }
{ "_id" : "Languages", "parent" : "Programming" }
> db.categories.find( {parent: "Books"} )
{ "_id" : "Programming", "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.*

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

Etana Disasa

10. Now...

- What is the child of the Databases node:  
`db.categories.findOne( { _id: "Databases" } ).children`

```
msds — mongo book — 157x5
> db.categories.insert( { _id: "Books", children: [ "Programming" ] } )
WriteResult({ "nInserted" : 1 })
> db.categories.findOne( { _id: "Databases" } ).children
[ "MongoDB", "dbm" ]
>
```

- 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.*

```
msds — mongo book — 157x5
> db.categories.findOne( { _id: "Databases" } ).children
[ "MongoDB", "dbm" ]
> db.categories.find( { children: "MongoDB" } )
{ "_id" : "Databases", "children" : [ "MongoDB", "dbm" ] }
>
```

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`

```
msds — -bash — 157x6
> exit
bye
(base) ETANAmac:~ msds$ brew services stop mongodb-community@4.2
Stopping 'mongodb-community'... (might take a while)
==> Successfully stopped 'mongodb-community' (label: homebrew.mxcl.mongodb-community)
(base) ETANAmac:~ msds$
```

End of Assignment