

MongoDB lab

1. INTRODUCTION

You should follow along with this tutorial by entering the commands into your personal database, and completing the exercises where prompted. Ask the lab helpers if you get stuck.

NOTE: You can use the VM or MACS Linux Server to run MongoDB. If you use the MACS Linux server you must first request a MongoDB account and database to be set up for you. please contact help@macs.hw.ac.uk to set this up. Your personal database and username should be the same.

In this document commands are shown using *fixed width font*, and variables that you need to change in the commands are shown in angle brackets in italics *<variable>* (do **not** include the angle brackets in the command).

2. CONNECTING

Open a terminal window on the VM and connect to MongoDB using the mongo client by typing: `mongo`

If you use ssh to the MACS Linux Servers then you'd need to use this command instead putting your user details (eg ab12) where it has *<username>*:


```
mongo -u <username> -p --authenticationDatabase <username> --host mongo-server-1
```

Enter your password if prompted. Then type:

`use hw`

If you use ssh to the MACS Linux server then it'd be:
`use <username>`

You have connected to your personal database.



```
2. MACS (ssh)
-bash-4.1$ mongo -u ajg33 -p --authenticationDatabase ajg33 --host mongo-server-1
MongoDB shell version: 2.6.5
Enter password:
connecting to: mongo-server-1:27017/test
> use ajg33
switched to db ajg33
> show collections
> exit
bye
-bash-4.1$
```

Figure 1: Screenshot showing connection interaction to MACS Server

Look at the collections that currently exist (nothing should appear as your database is empty):

```
show collections
```

Change your password if using the MACS Server (important for securing your work):

```
db.changeUserPassword("<username>", "<password>")
```

Now exit mongo and return to the terminal's command line:

```
exit
```

3. IMPORTING DATA

Download the data file for this lab from:

```
wget www.macs.hw.ac.uk/~pb56/labData.json
```

Open the file and look at the data.

```
less labData.json
```

Notice that there is one JSON document per line. (Press q to exit *less*.)

Using the Linux command line, insert the example data into your database using `mongoimport` (the file parameter below assumes you are in the directory containing the file and you have not renamed it):

```
mongoimport -db hw --collection hwuPeople --file labData.json
```

On the Macs Linux Server the import command would be like this:

```
mongoimport --db <username> -u <username> --host mongo-server-1
--authenticationDatabase <username> --collection hwuPeople
--file labData.json
```

Note that there is no `-p` parameter in the `mongoimport` command.

Repeat the above steps (from CONNECTING) to logon and view the collections. When you look at the collections you should now see a "hwuPeople" collection.

Inspect the contents of `hwuPeople`:

```
db.hwuPeople.find()
```

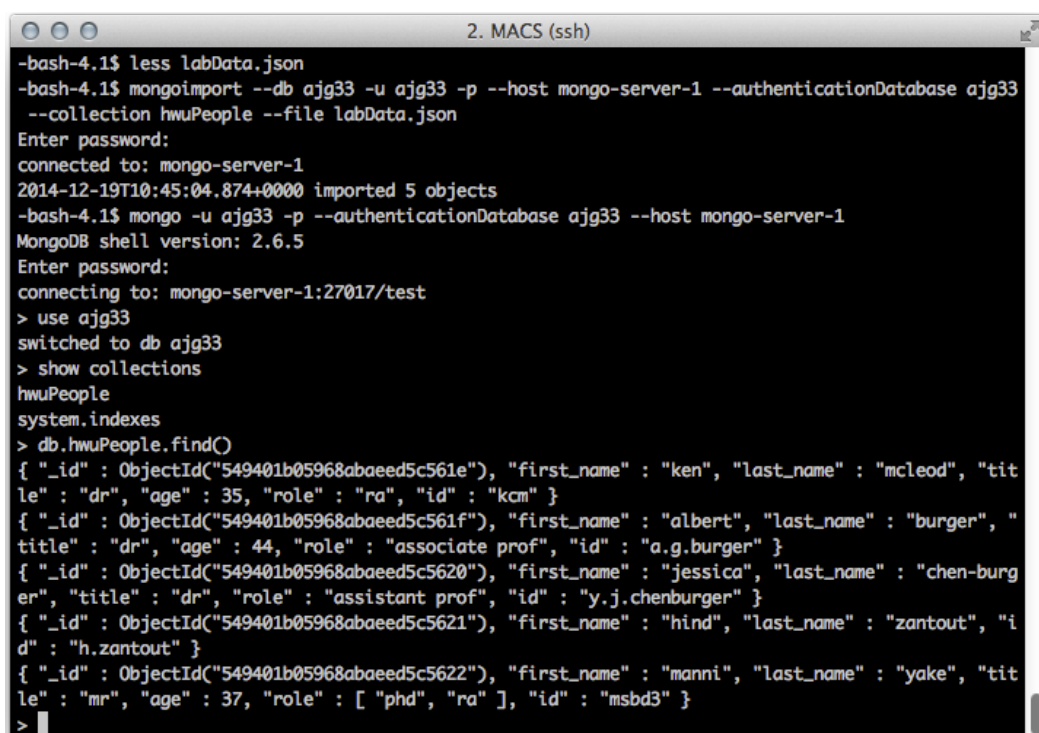
...also try this version with formatting:

```
db.hwuPeople.find().pretty()
```

You will see 5 people listed, as shown in Figure 2. In each person you will see:

```
{ "_id" : ObjectId("541ff6239c494c635a2d32d7"), "first_name" ...
```

This `"_id"` is the internal identifier that is used to uniquely identify documents. It is the only part of the document that is indexed by default, i.e. it is the primary key.



```
2. MACS (ssh)
-bash-4.1$ less labData.json
-bash-4.1$ mongoimport --db ajg33 -u ajg33 -p --host mongo-server-1 --authenticationDatabase ajg33
--collection hwuPeople --file labData.json
Enter password:
connected to: mongo-server-1
2014-12-19T10:45:04.874+0000 imported 5 objects
-bash-4.1$ mongo -u ajg33 -p --authenticationDatabase ajg33 --host mongo-server-1
MongoDB shell version: 2.6.5
Enter password:
connecting to: mongo-server-1:27017/test
> use ajg33
switched to db ajg33
> show collections
hwuPeople
system.indexes
> db.hwuPeople.find()
{ "_id" : ObjectId("549401b05968abaeed5c561e"), "first_name" : "ken", "last_name" : "mcleod", "title" : "dr", "age" : 35, "role" : "ra", "id" : "kam" }
{ "_id" : ObjectId("549401b05968abaeed5c561f"), "first_name" : "albert", "last_name" : "burger", "title" : "dr", "age" : 44, "role" : "associate prof", "id" : "a.g.burger" }
{ "_id" : ObjectId("549401b05968abaeed5c5620"), "first_name" : "jessica", "last_name" : "chen-burger", "title" : "dr", "role" : "assistant prof", "id" : "y.j.chenburger" }
{ "_id" : ObjectId("549401b05968abaeed5c5621"), "first_name" : "hind", "last_name" : "zantout", "id" : "h.zantout" }
{ "_id" : ObjectId("549401b05968abaeed5c5622"), "first_name" : "manni", "last_name" : "yake", "title" : "mr", "age" : 37, "role" : [ "phd", "ra" ], "id" : "msbd3" }
>
```

Figure 2: Importing data

4. QUERYING

Let us find all the people with role "ra":

```
db.hwuPeople.find({role : "ra"})
```

Notice that Manni appears in the output: mongo automatically searches inside an array. To find all people named "Ken McLeod":

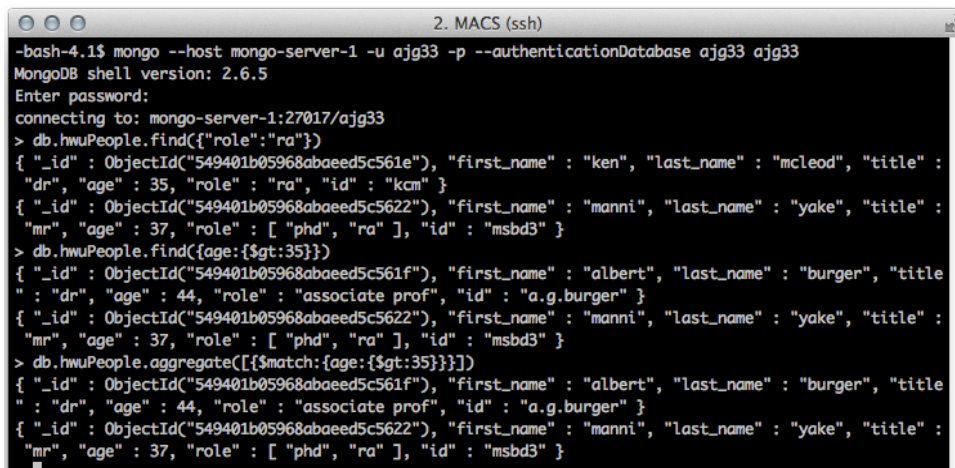
```
db.hwuPeople.find({first_name: "ken", last_name: "mcLeod"})
```

To find all people older than 35:

```
db.hwuPeople.find({age : {$gt: 35}})
```

There are multiple ways to write the same query, e.g.:

```
db.hwuPeople.aggregate([{$match : { age : {$gt: 35}}}])
```



```
2. MACS (ssh)
-bash-4.1$ mongo --host mongo-server-1 -u ajg33 -p --authenticationDatabase ajg33 ajg33
MongoDB shell version: 2.6.5
Enter password:
connecting to: mongo-server-1:27017/ajg33
> db.hwuPeople.find({"role":"ra"})
{ "_id" : ObjectId("549401b05968abaeed5c561e"), "first_name" : "ken", "last_name" : "mcLeod", "title" : "dr", "age" : 35, "role" : "ra", "id" : "kcm" }
{ "_id" : ObjectId("549401b05968abaeed5c5622"), "first_name" : "manni", "last_name" : "yake", "title" : "mr", "age" : 37, "role" : [ "phd", "ra" ], "id" : "msbd3" }
> db.hwuPeople.find({age:{$gt:35}})
{ "_id" : ObjectId("549401b05968abaeed5c561f"), "first_name" : "albert", "last_name" : "burger", "title" : "dr", "age" : 44, "role" : "associate prof", "id" : "a.g.burger" }
{ "_id" : ObjectId("549401b05968abaeed5c5622"), "first_name" : "manni", "last_name" : "yake", "title" : "mr", "age" : 37, "role" : [ "phd", "ra" ], "id" : "msbd3" }
> db.hwuPeople.aggregate([{$match:{age:{$gt:35}}}])
{ "_id" : ObjectId("549401b05968abaeed5c561f"), "first_name" : "albert", "last_name" : "burger", "title" : "dr", "age" : 44, "role" : "associate prof", "id" : "a.g.burger" }
{ "_id" : ObjectId("549401b05968abaeed5c5622"), "first_name" : "manni", "last_name" : "yake", "title" : "mr", "age" : 37, "role" : [ "phd", "ra" ], "id" : "msbd3" }
```

Figure 3: Basic queries

🔑 **TASK:** write a query to find all the RAs under 40 years old.

Hint: use `$lt` for less than.

You can also find the unique values, such as distinct titles within the collection:

```
db.hwuPeople.distinct("title")
```

... which should return a list ["dr", "prof", "mr"] to get the number of items in the list add `.length` to the previous command.

```
db.hwuPeople.distinct("title").length
```

You can choose which fields in a document are returned. To display only names:

```
db.hwuPeople.find({}, {first_name : 1, last_name : 1})
```

Note that the ObjectId is still displayed. To remove that you need:

```
db.hwuPeople.find({}, {_id:0, first_name : 1, last_name : 1})
```

```
> db.hwuPeople.find({}, {_id:0, first_name : 1, last_name : 1})
{ "first_name" : "ken", "last_name" : "mcleod" }
{ "first_name" : "albert", "last_name" : "burger" }
{ "first_name" : "jessica", "last_name" : "chen-burger" }
{ "first_name" : "hind", "last_name" : "zantout" }
{ "first_name" : "manni", "last_name" : "yake" }
>
```

Figure 4: Choosing the columns to return

It is possible to search for strings within text fields – try this:

```
db.hwuPeople.find({$text: {$search: "burger"}})
```

The above query did not work as we have not set up a text index yet. To create an index on the first and last names we use the command:

```
db.hwuPeople.createIndex({first_name: "text", last_name: "text"})
```

Now we can run our search query to find all people who have the string "burger" in their name.

Another way to find 'contains' is using regex as follows (this should also work on non-indexed fields):

```
db.hwuPeople.find({last_name:/burger/i})
```

Note: the *i* makes it a case-insensitive search, remove the *i* to make it case-sensitive

You can use this query approach in conjunction with `distinct` – for example to find the unique titles for people with 'burger' in their surname:

```
db.hwuPeople.distinct("title",{ "last_name": /burger/i})
```

Note: this returns the distinct values for "title" based on the query "last_name" contains the value 'burger'.

You can change this query to find people who have a surname starting ^ with burger or ending \$ in burger too as follows:

```
db.hwuPeople.distinct("title",{ "last_name": /^burger/i})
```

```
db.hwuPeople.distinct("title",{ "last_name": /burger$/i})
```

You can specify multiple conditions using *AND* logic to add more conditions to the query; alternatively you can specify *OR* to permit alternatives - for example find all people that have a role "assistant prof" *or* "associate prof":

```
db.hwuPeople.find({$or: [{role: 'associate prof'}, {role: 'assistant prof'}]})
```

```
> db.hwuPeople.find({$or: [{role: 'associate prof'}, {role: 'assistant prof'}]})
{ "_id" : ObjectId("549401b05968abaeed5c561f"), "first_name" : "albert", "last_name" : "burger", "title" : "dr", "age" : 44, "role" : "associate prof", "id" : "a.g.burger" }
{ "_id" : ObjectId("549401b05968abaeed5c5620"), "first_name" : "jessica", "last_name" : "chen-burger", "title" : "dr", "role" : "assistant prof", "id" : "y.j.chenburger" }
> db.hwuPeople.find({age:{$exists:false}})
{ "_id" : ObjectId("549401b05968abaeed5c5620"), "first_name" : "jessica", "last_name" : "chen-burger", "title" : "dr", "role" : "assistant prof", "id" : "y.j.chenburger" }
{ "_id" : ObjectId("549401b05968abaeed5c5621"), "first_name" : "hind", "last_name" : "zantout", "id" : "h.zantout" }
> db.hwuPeople.count({age:{$exists:false}})
2
```

Figure 5: AND / OR conditions

To find everyone who does not have an "age" specified:

```
db.hwuPeople.find({age : {$exists: false}})
```

Rather than listing the documents without an age, we can simply ask for the number of documents using the *count* function:

```
db.hwuPeople.count({age : {$exists: false}})
```

You could also do this as follows:

```
db.hwuPeople.find({age : {$exists: false}}).count()
```

To sort the results of a "find" query append `.sort({name: 1})` after the `find()` statement. Use 1 for ascending order, and -1 for descending order.

🔗 **TASK:** Sort the entire list of people in ascending order by age.

You can also sort after aggregating, using the `sortByCount` function as follows:

```
db.hwuPeople.aggregate([ {$sortByCount: "$title"},{$limit:3}])
```

note: the *limit 3* is optional, here it demonstrates how you could retrieve just the top 3 groups based on the count

Aggregation also allows you to run calculations per group such as finding the sum, minimum, average etc:

```
db.hwuPeople.aggregate([{$group: {_id:"$title",total: {"$sum": '$age'}}} ])
```

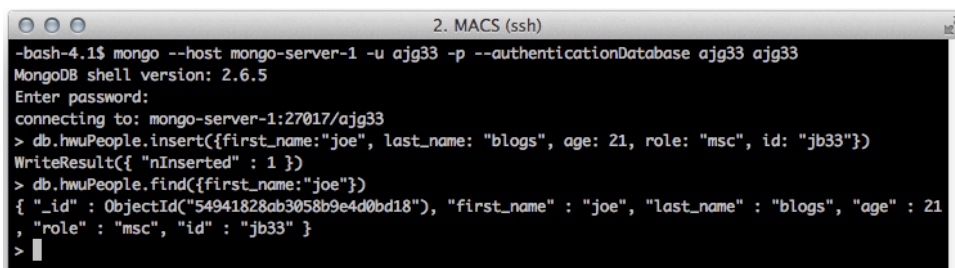
5. INSERTING, UPDATING & REMOVING

To insert someone into the database:

```
db.hwuPeople.insert({first_name : "joe", last_name : "blogs", age : 21,  
role : "msc", id : "jb33"})
```

To prove this worked:

```
db.hwuPeople.find({first_name : "joe"})
```



```
2. MACS (ssh)
-bash-4.1$ mongo --host mongo-server-1 -u ajg33 -p --authenticationDatabase ajg33 ajg33
MongoDB shell version: 2.6.5
Enter password:
connecting to: mongo-server-1:27017/ajg33
> db.hwuPeople.insert({first_name:"joe", last_name: "blogs", age: 21, role: "msc", id: "jb33"})
WriteResult({ "nInserted" : 1 })
> db.hwuPeople.find({first_name:"joe"})
{ "_id" : ObjectId("54941828ab3058b9e4d0bd18"), "first_name" : "joe", "last_name" : "blogs", "age" : 21, "role" : "msc", "id" : "jb33" }
> |
```

Figure 6: Inserting data

- 🔑 **TASK:** add yourself, specifying your `_id` manually.
HINT: treat `"_id"` as just another name/value pair.

Dr Burger has been promoted and so we need to change his `"title"` and his `"role"` to `"prof"`:

```
db.hwuPeople.update({last_name : "burger"}, {$set: {title : "prof", role : "prof"}})
```

- 🔑 **QUESTION:** What are the potential side effects of the previous statement?
HINT: `"burger"` is a rare last name in the UK and this is a small dataset

With a flexible schema you can add information to one document but not to others:
add an email address for Dr McLeod:

```
db.hwuPeople.update({last_name: "mcleod"}, {$set: {email: "kcm1@hw.ac.uk"}})
```

Look at the document for Manni:

```
db.hwuPeople.find({first_name : "manni"})
```

He is listed as being 37, but his birthday was last week; to increase his age:

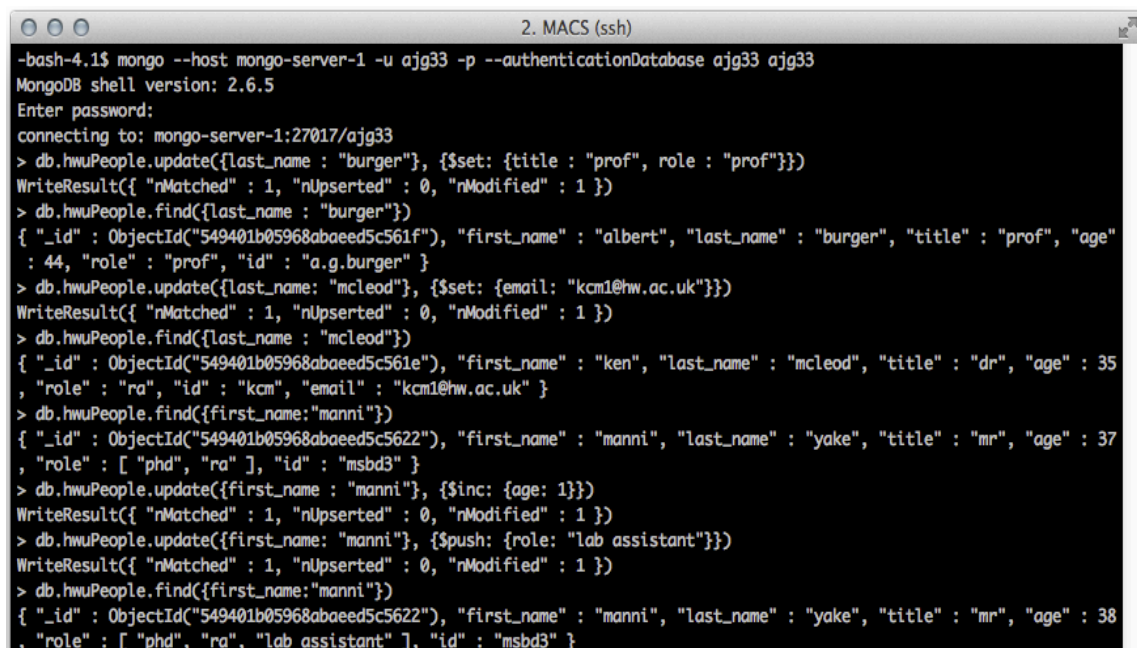
```
db.hwuPeople.update({first_name : "manni"}, {$inc: {age: 1}})
```

Manni has another role (lab assistant), to add this:

```
db.hwuPeople.update({first_name: "manni"}, {$push: {role: "lab assistant"}})
```

NOTE: `$push` only works for arrays.

- 🔑 **TASK:** update your information to provide your email address and your title (e.g., Mr, Ms etc.).



```
2. MACS (ssh)
-bash-4.1$ mongo --host mongo-server-1 -u ajg33 -p --authenticationDatabase ajg33 ajg33
MongoDB shell version: 2.6.5
Enter password:
connecting to: mongo-server-1:27017/ajg33
> db.hwuPeople.update({last_name : "burger"}, {$set: {title : "prof", role : "prof"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.hwuPeople.find({last_name : "burger"})
{ "_id" : ObjectId("549401b05968abaed5c561f"), "first_name" : "albert", "last_name" : "burger", "title" : "prof", "age" : 44, "role" : "prof", "id" : "a.g.burger" }
> db.hwuPeople.update({last_name: "mcleod"}, {$set: {email: "kcm1@hw.ac.uk"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.hwuPeople.find({last_name : "mcleod"})
{ "_id" : ObjectId("549401b05968abaed5c561e"), "first_name" : "ken", "last_name" : "mcleod", "title" : "dr", "age" : 35, "role" : "ra", "id" : "kcm", "email" : "kcm1@hw.ac.uk" }
> db.hwuPeople.find({first_name: "manni"})
{ "_id" : ObjectId("549401b05968abaed5c5622"), "first_name" : "manni", "last_name" : "yake", "title" : "mr", "age" : 37, "role" : [ "phd", "ra" ], "id" : "msbd3" }
> db.hwuPeople.update({first_name : "manni"}, {$inc: {age: 1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.hwuPeople.update({first_name: "manni"}, {$push: {role: "lab assistant"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.hwuPeople.find({first_name: "manni"})
{ "_id" : ObjectId("549401b05968abaed5c5622"), "first_name" : "manni", "last_name" : "yake", "title" : "mr", "age" : 38, "role" : [ "phd", "ra", "lab assistant" ], "id" : "msbd3" }
```

Figure 7: Updating existing entries

If you try to update a document that is not there, nothing happens:

```
db.hwuPeople.update({first_name: "andy", last_name: "proudlove", role:
"ra"}, {age: 47})

db.hwuPeople.find({first_name: "andy"})
```

Andy has not been added. However, mongoDB supports “upserts” (update or insert if there is no document found):

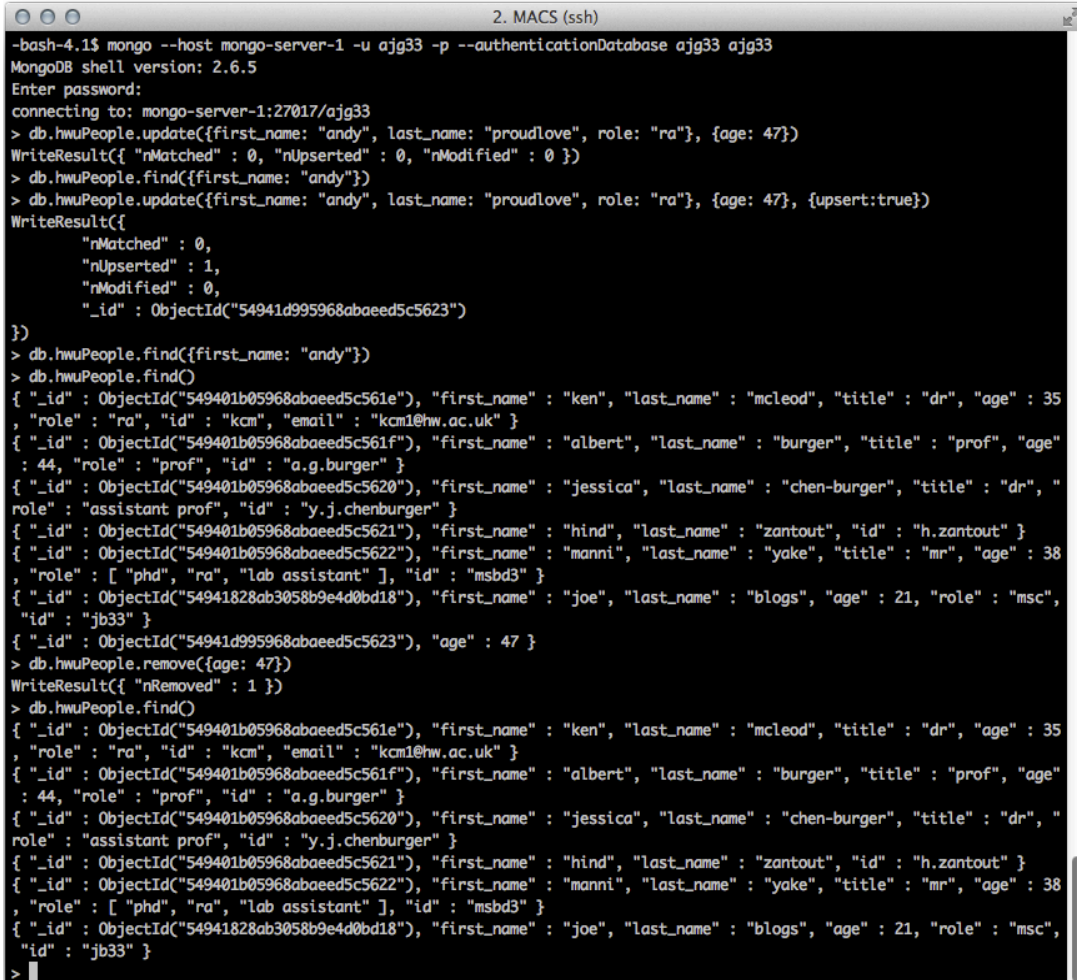
```
db.hwuPeople.update({first_name: "andy", last_name: "proudlove", role:
"ra"}, {age: 47}, {upsert:true})

db.hwuPeople.find({first_name: "andy"})

db.hwuPeople.find()
```

Notice what has happened: a new document has been added, but it only contains “age”. Clearly, this is wrong! To remove this document (assuming your age is not 47):

```
db.hwuPeople.remove({age: 47})
```



```
2. MACS (ssh)
-bash-4.1$ mongo --host mongo-server-1 -u ajg33 -p --authenticationDatabase ajg33 ajg33
MongoDB shell version: 2.6.5
Enter password:
connecting to: mongo-server-1:27017/ajg33
> db.hwuPeople.update({first_name: "andy", last_name: "proudlove", role: "ra"}, {age: 47})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.hwuPeople.find({first_name: "andy"})
> db.hwuPeople.update({first_name: "andy", last_name: "proudlove", role: "ra"}, {age: 47}, {upsert:true})
WriteResult({
  "nMatched" : 0,
  "nUpserted" : 1,
  "nModified" : 0,
  "_id" : ObjectId("54941d995968abaeed5c5623")
})
> db.hwuPeople.find({first_name: "andy"})
> db.hwuPeople.find()
{ "_id" : ObjectId("549401b05968abaeed5c561e"), "first_name" : "ken", "last_name" : "mcleod", "title" : "dr", "age" : 35,
  "role" : "ra", "id" : "kcm", "email" : "kcm1@hw.ac.uk" }
{ "_id" : ObjectId("549401b05968abaeed5c561f"), "first_name" : "albert", "last_name" : "burger", "title" : "prof", "age" : 44,
  "role" : "prof", "id" : "a.g.burger" }
{ "_id" : ObjectId("549401b05968abaeed5c5620"), "first_name" : "jessica", "last_name" : "chen-burger", "title" : "dr", "role" : "assistant prof",
  "id" : "y.j.chenburger" }
{ "_id" : ObjectId("549401b05968abaeed5c5621"), "first_name" : "hind", "last_name" : "zantout", "id" : "h.zantout" }
{ "_id" : ObjectId("549401b05968abaeed5c5622"), "first_name" : "manni", "last_name" : "yake", "title" : "mr", "age" : 38, "role" : [ "phd", "ra", "lab assistant" ],
  "id" : "msbd3" }
{ "_id" : ObjectId("54941828ab3058b9e4d0bd18"), "first_name" : "joe", "last_name" : "blogs", "age" : 21, "role" : "msc", "id" : "jb33" }
{ "_id" : ObjectId("54941d995968abaeed5c5623"), "age" : 47 }
> db.hwuPeople.remove({age: 47})
WriteResult({ "nRemoved" : 1 })
> db.hwuPeople.find()
{ "_id" : ObjectId("549401b05968abaeed5c561e"), "first_name" : "ken", "last_name" : "mcleod", "title" : "dr", "age" : 35,
  "role" : "ra", "id" : "kcm", "email" : "kcm1@hw.ac.uk" }
{ "_id" : ObjectId("549401b05968abaeed5c561f"), "first_name" : "albert", "last_name" : "burger", "title" : "prof", "age" : 44,
  "role" : "prof", "id" : "a.g.burger" }
{ "_id" : ObjectId("549401b05968abaeed5c5620"), "first_name" : "jessica", "last_name" : "chen-burger", "title" : "dr", "role" : "assistant prof",
  "id" : "y.j.chenburger" }
{ "_id" : ObjectId("549401b05968abaeed5c5621"), "first_name" : "hind", "last_name" : "zantout", "id" : "h.zantout" }
{ "_id" : ObjectId("549401b05968abaeed5c5622"), "first_name" : "manni", "last_name" : "yake", "title" : "mr", "age" : 38, "role" : [ "phd", "ra", "lab assistant" ],
  "id" : "msbd3" }
{ "_id" : ObjectId("54941828ab3058b9e4d0bd18"), "first_name" : "joe", "last_name" : "blogs", "age" : 21, "role" : "msc", "id" : "jb33" }
>
```

Figure 8: Upsert interaction

👉 **TASK:** Write an upsert query to properly insert Andy Proudlove into the database.

👉 **TASK:** Use the *remove* command to delete Joe Bloggs from the database.
HINT: most of the operations that work with the *insert* command also work with the *remove* command.

6. OPTIMISATION

For queries to be efficient they must use an index. To check if a query uses an index use the `.explain()` method:

```
db.hwuPeople.find({age : {$gt: 35}}).explain()
```



Figure 9: Explain query plan

We can see that the winning plan is **COLLSCAN** meaning that the collection is being scanned, **i.e. no index was used**.

To see more details of the number of documents scanned add "executionStats" as follows:

```
db.hwuPeople.find({age : {$gt: 35}}).explain("executionStats")
```

To add an index to "age":


```
db.hwuPeople.createIndex({age: 1})
```

Now if we run the `.explain()` method again we see the winning plan is based on an **IXSCAN**, i.e. it is using the index on the age field that we just created.

Also try the executionStats to see how many documents were examined this time.

```
hwuPeople.find({age : {$gt: 35}}).explain("executionStats")
```


7. BACKUP



```
1. ssh
> db.hwuPeople.find({age : {$gt: 35}}).explain()
{
  "queryPlanner" : {
    "plannerVersion" : 1,
    "namespace" : "ajg33.hwuPeople",
    "indexFilterSet" : false,
    "parsedQuery" : {
      "age" : {
        "$gt" : 35
      }
    },
    "winningPlan" : {
      "stage" : "FETCH",
      "inputStage" : {
        "stage" : "IXSCAN",
        "keyPattern" : {
          "age" : 1
        },
        "indexName" : "age_1",
        "isMultiKey" : false,
        "isUnique" : false,
        "isSparse" : false,
        "isPartial" : false,
        "indexVersion" : 1,
        "direction" : "forward",
```

Figure 10: Explain plan after index creation

Exit mongo and use mongodump to backup your current database:

```
mongodump -d hw -c hwuPeople
```

On the Linux Server this would be:

```
mongodump -u <username> --host mongo-server-1
--authenticationDatabase <username> -d <username> -c hwuPeople
```

Note that there is no -p parameter in the mongodump command.

In your current directory you should see a "dump" folder. Your files are in there. You can copy them to another machine and use mongoimport to import them.



```
-bash-4.1$ ls
labData.json
-bash-4.1$ mongodump -u ajg33 -p --host mongo-server-1 --authenticationDatabase ajg33 -d ajg33 -c hwuPeople
Enter password:
connected to: mongo-server-1
2014-12-19T13:02:45.534+0000 DATABASE: ajg33 to dump/ajg33
2014-12-19T13:02:45.537+0000 ajg33.hwuPeople to dump/ajg33/hwuPeople.bson
2014-12-19T13:02:45.540+0000 5 documents
2014-12-19T13:02:45.541+0000 Metadata for ajg33.hwuPeople to dump/ajg33/hwuPeople.metadata.json
-bash-4.1$ ls
dump labData.json
-bash-4.1$ ls dump/
ajg33
-bash-4.1$ ls dump/ajg33/
hwuPeople.bson hwuPeople.metadata.json
-bash-4.1$
```

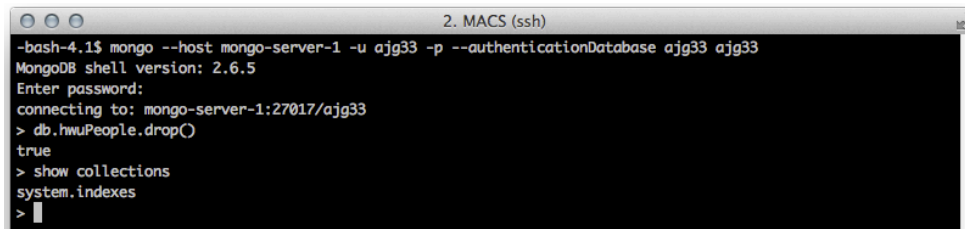
Figure 11: Exporting a collection

8. DELETING A COLLECTION

Log into mongo and switch to your database.

To delete the hwuPeople collection:

```
db.hwuPeople.drop()
show collections
```

A terminal window titled '2. MACS (ssh)' showing a MongoDB shell session. The user runs the command `mongo --host mongo-server-1 -u ajg33 -p --authenticationDatabase ajg33 ajg33`. The prompt changes to `MongoDB shell version: 2.6.5`. The user enters a password. The prompt changes to `connecting to: mongo-server-1:27017/ajg33`. The user runs `> db.hwuPeople.drop()` and the output is `true`. The user then runs `> show collections` and the output is `system.indexes`.

```
-bash-4.1$ mongo --host mongo-server-1 -u ajg33 -p --authenticationDatabase ajg33 ajg33
MongoDB shell version: 2.6.5
Enter password:
connecting to: mongo-server-1:27017/ajg33
> db.hwuPeople.drop()
true
> show collections
system.indexes
>
```

Figure 12: Dropping a collection

9. EXERCISE

TASK: Create a new collection (called "exercise") that includes the following information:

name: albert burger, role: supervisor
name: alasdair gray, role: supervisor
name: iain wiles, role: phd
name: steve smith, role: phd
name: hugh dollar, role: phd

Additionally, include the following relationships:

Alasdair supervises Iain and Steve.

Albert supervises Steve and Hugh.

HINT: use an array to hold the supervisor information against each student.

👉 **TASK:** Now construct an optimized query to list all of Alasdair Gray's students.

👉 **TASK:** Write a query to find all the students with 2 supervisors.
HINT: use the condition `$size: 2`.

👉 **TASK:** Write a query to find those with more than 1 supervisor
HINT: use the condition `$where` and the `javascript length` feature.

10. USING JAVASCRIPT WITH MONGODB

Let's use Javascript to create 50 robots in a collection from the MonogoDB shell. We will use a FOR loop (iteration) and the assign the value to variable *i*

```
for (i=0;i<50;i++)
{
  db.robots1.insert({name:"robot"+i } )
}
```

Count the documents in the collection to make sure it all went as planned.

```
db.robots1.count()
```

...or you can take a look with:

```
db.robots1.find()
```

Note: you will need to type "it" to iterate through the results

Now let's add a unique index to the robot collection based on the name.

```
db.robots1.createIndex({"name":1},{unique:true})
```

Try to run the JAVASCRIPT again to insert the same 50 robots... you should get an error about duplicates.

Comparing Collections

OK now let's make another collection of robots numbered from 25 to 40 and this time put them in robot2 collection.

```
for (i=25;i<40;i++)
{
  db.robots2.insert({name:"robot"+i } )
}
```

If we wanted to carry out a comparison between the collections we can use variables like this – here getting a list of robots in collection robot1 that are not in (\$nin) the robot2 collection.

```
v1 = db.robots2.find().map(function (a) {return a.name;})
db.robots1.find ({name: {$nin:v1}})
```

Try typing **v1** to see what it stores. It's a list of values. To get a COUNT of the difference you just need to add the count function to the output like this:

```
db.robots1.find ({name: {$nin:v1}}).count()
```

To remove the robots from a collection use a blank search filter like this:

```
db.robots1.deleteMany({})
```

To check they were deleted use count() or find ().

```
db.robots1.count()
```

--- END OF THE LAB ---