**Portfolio Part 2: MongoDB**

**T00155775**

**Joe O Flaherty**

**BSc (Hons) Computing with Multimedia**

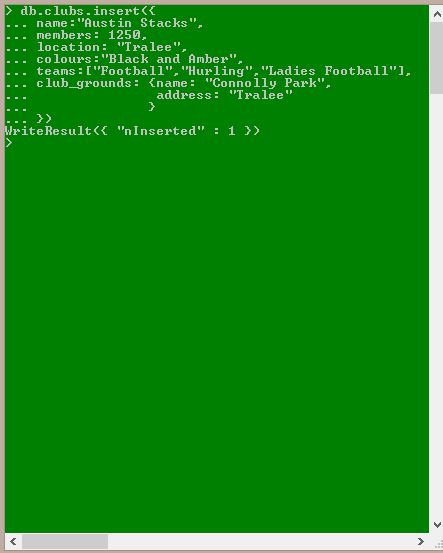
## Implement your own MongoDB database (showing examples of CRUD)

Initial creation of a new database called ‘gaa’ and performance of CRUD operations. All operations were completed with little difficulty.

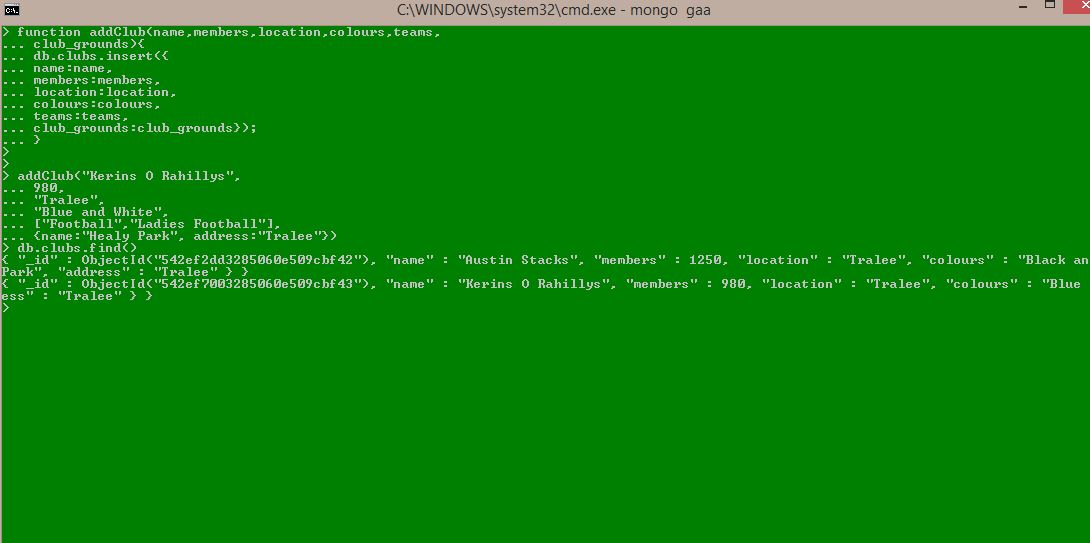
H:\4thYear\Advanced Database Programming\Portfolio\Portfolio_2\Images\New_DB_Gaa.JPG

Inserted a clubs collection, clubs are comprised of:

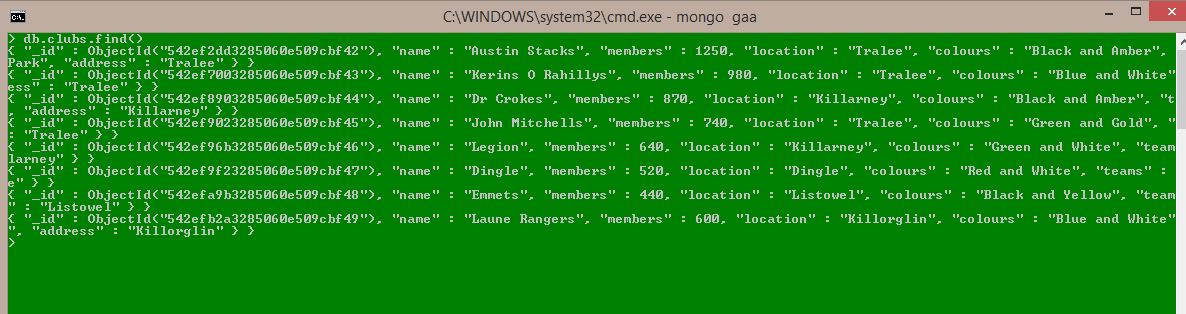
* Name
* Members
* Location
* Colours
* Teams – an array of teams the club fields
* Club\_grounds – which is comprised of a name and an address



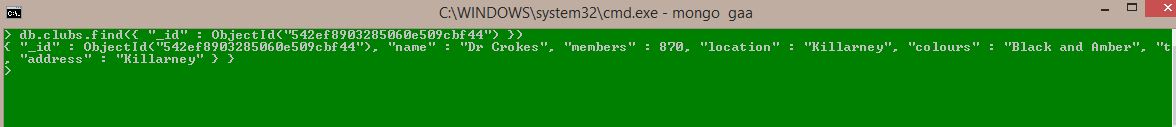
Proceeded to populate the database using a JavaScript function called **addClub (** **)**. The function greatly increased the speed at which new objects could be added to the collection.



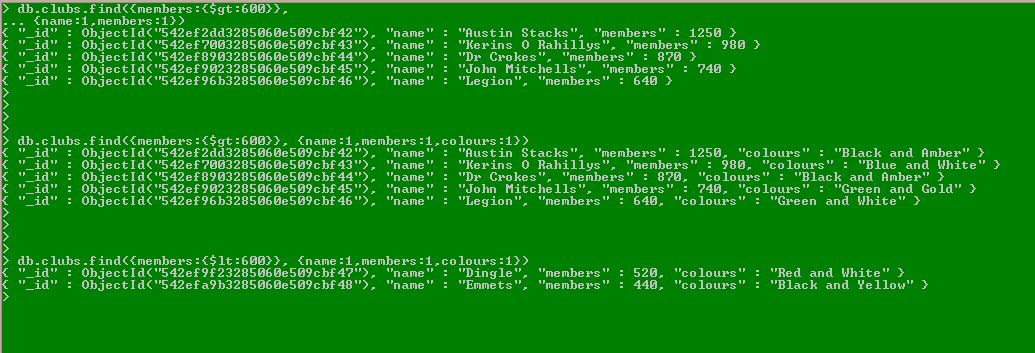
Reading the contents of the collection using db.clubs.find ( ):



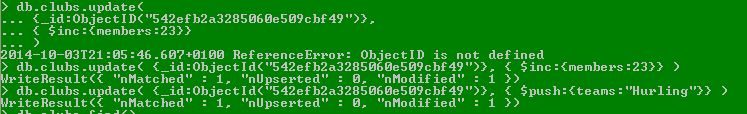
Finding a specific club using db.clubs.find ( ) but giving the object id as a parameter to the function:



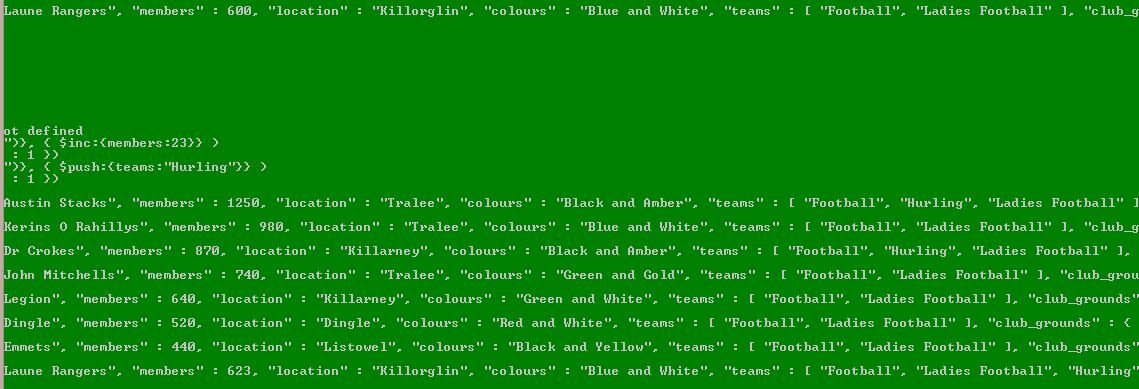
Final read operations using db.clubs.find with a selection of conditions:



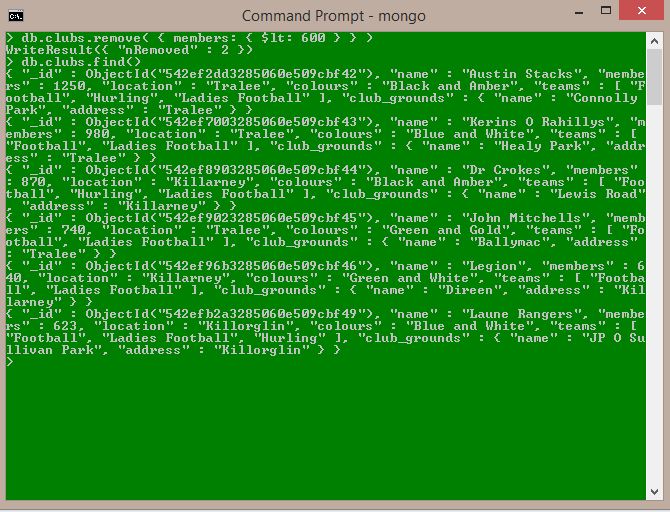
Updating the collection by increasing the number of members Laune Rangers have by 23 using db.clubs.update ( ).



Proof that members has been updated from 600 t0 623 also used $push to add hurling to the teams that Laune Rangers field:

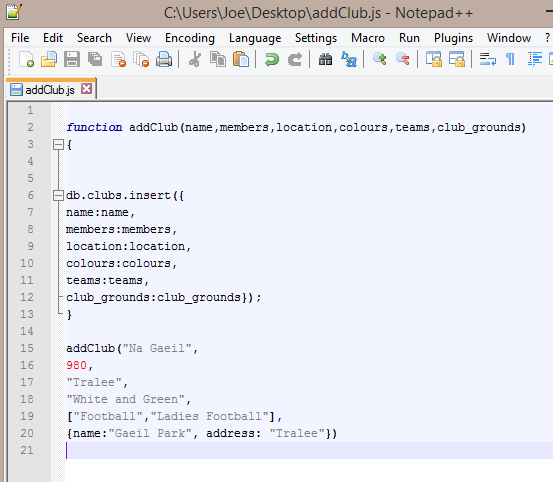


Removed (Deleted) 2 clubs from the collection with a membership of less than 600, Dingle and Emmets.

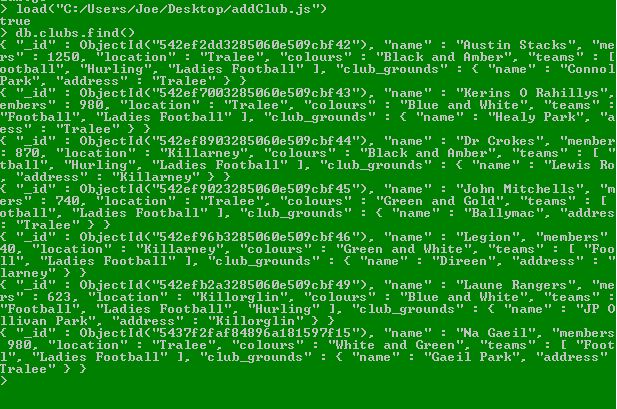


## Run a query from an external JavaScript file

Inserted new clubs into the collection for this question using the same addClub ( ) function but as the question states running the JavaScript as an external file.



Called from Mongo shell using **load (path/to/file.js)** and db.clubs.find ( ) called to confirm Na Gaeil have been added to the database:



## Investigate and upload and show the use of media objects (like images or music files) to your database

Uploading and storing an image in a MongoDB collection initially appears a non-trivial operation. Following extensive investigation it appears that using GridFS is one less complex way of achieving the task. Having discovered this a number of tutorials were still needed to acquire a reasonable understanding of the operation. Another much less attractive option is converting the image to a byte array and the storing it in a collection as a Binary Object.

Eventually mongofiles.exe which comes with MongoDB installation but is used from a regular CLI was used to upload 2 images of club crests to a collection in the ‘gaa’ database. Once usage of mongofiles was researched image upload proved to be quite simple. The major stumbling block came from attempting to use mongofiles from the mongo shell instead of a regular CLI. The following commands were used for experimentation purposes.

mongofiles -d gaa -l 'C:\Crests\AustinStacks.jpg' put 'AustinStacks.jpg'

mongofiles -d gaa -l 'C:\Crests\Dr Crokes.jpg' put 'Dr Crokes.jpg'

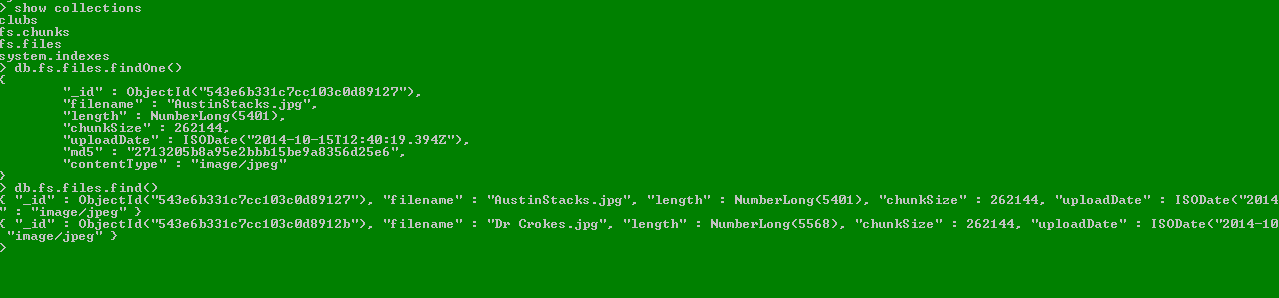
mongofiles -d gaa delete 'Dr Crokes.jpg'

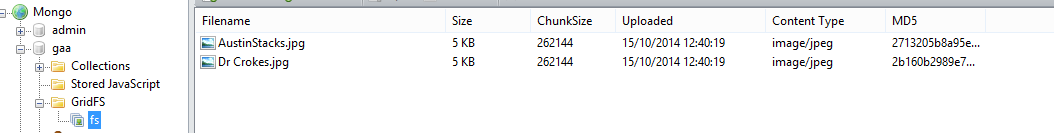
mongofiles -d gaa delete 'AustinStacks.jpg'

mongofiles -d gaa -l 'C:\Users\Joe\AppData\Local\Temp\AustinStacks.jpg' get 'AustinStacks.jpg'

mongofiles -d gaa -l 'C:\Crests\AustinStacks.jpg' put 'AustinStacks.jpg'

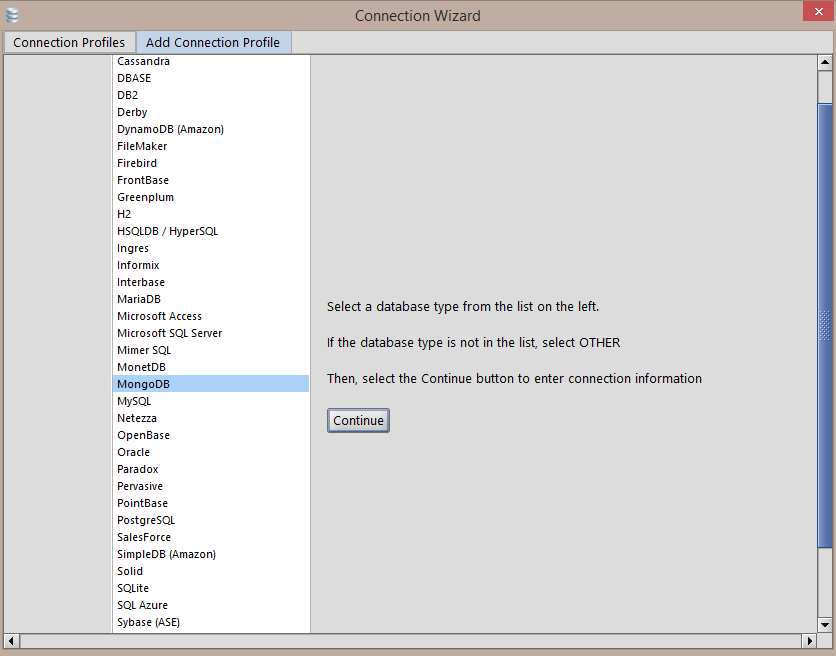
mongofiles -d gaa -l 'C:\Crests\Dr Crokes.jpg' put 'Dr Crokes.jpg'



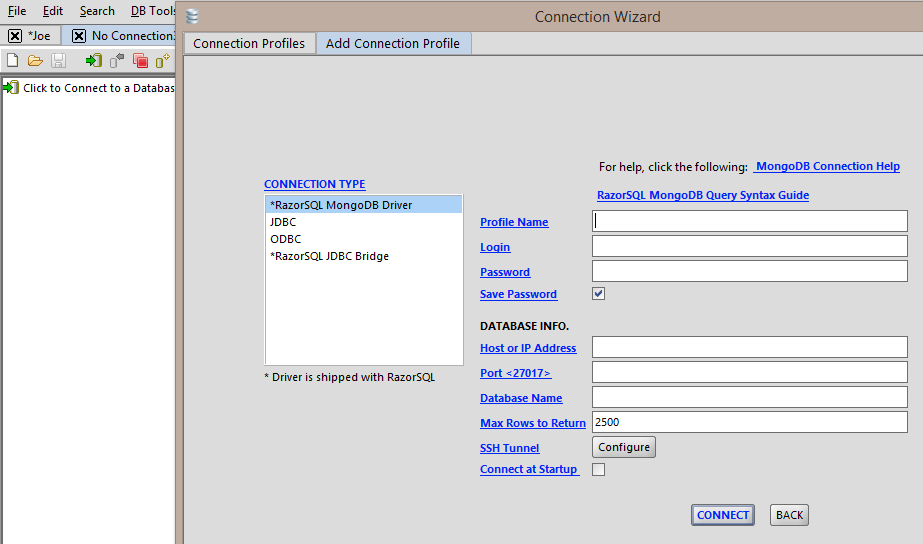


## Download, install and evaluate a tool to use with MongoDB

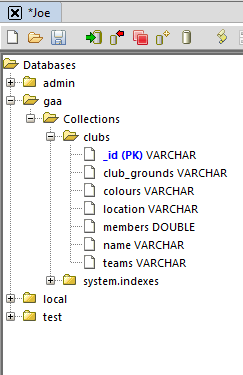
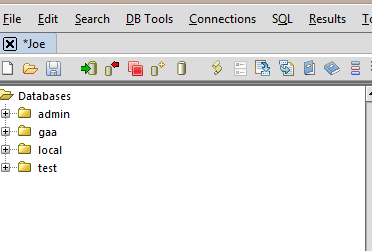
The tool is RazorSQL downloaded from <http://razorsql.com/features/mongodb_gui_tools.html> A GUI that enables connection with an enormous selection of databases including MongoDB, RazorSQL can be evaluated for free over a 30 day period and can be purchased for a maximum price of $99.95.



Once RazorSQL has been installed set up is extremely easy without the need to consult any documentation. Simply select the database required and fill in the fields as necessary to establish a connection.

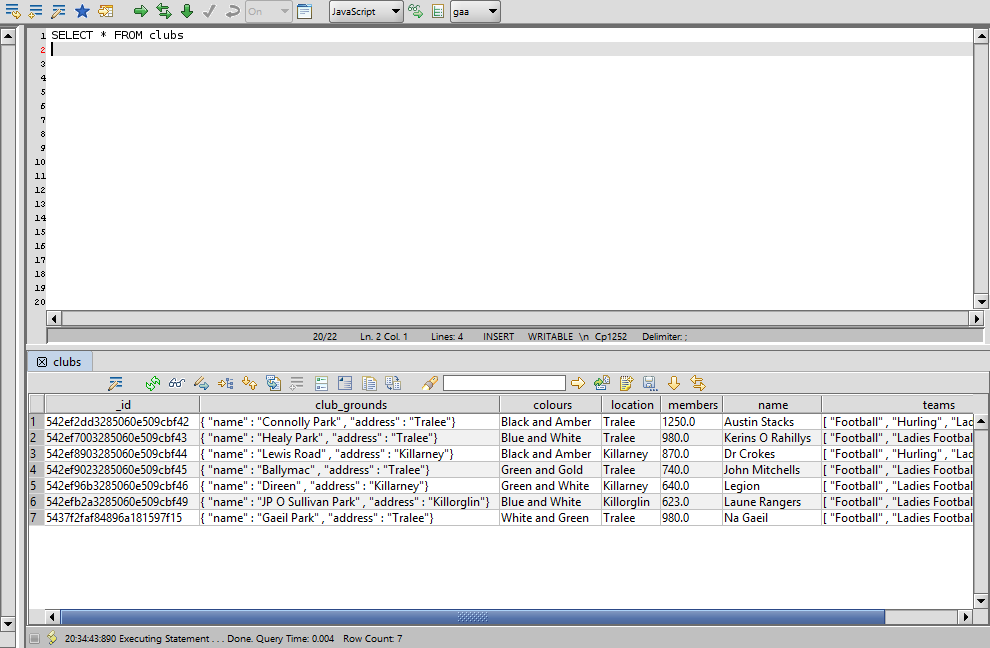


Connected to MongoDB and displays all databases, each database can be expanded to show the collections they contain and expanded again to show the objects in a collection.

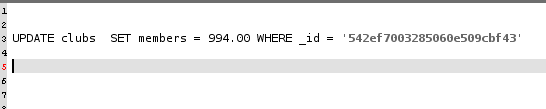


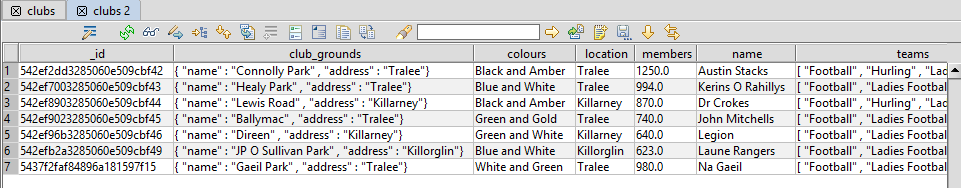
The name of the application gives a clue to the real and amazingly useful purpose of RazorSQL. The screenshot above shows a MongoDB collection ‘clubs’ displayed very much as a traditional relational database would be including Primary Key. This leads to the feature of the tool I found the most useful and exciting, the ability to interact with a MongoDB database using SQL.

The following example shows a SELECT \* FROM clubs SQL query run against a MongoDB instance, the results of the query are displayed ala sqldeveloper in the lower pane.

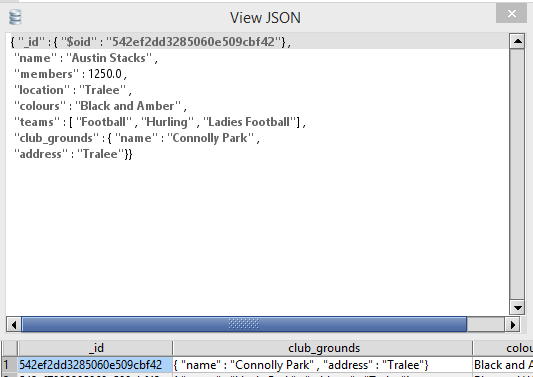


All queries are performed exactly as they would in SQL, below is an example of an UPDATE query.





Razor also allows a user right click on any row returned from a query and view as JSON and also edit the JSON directly.

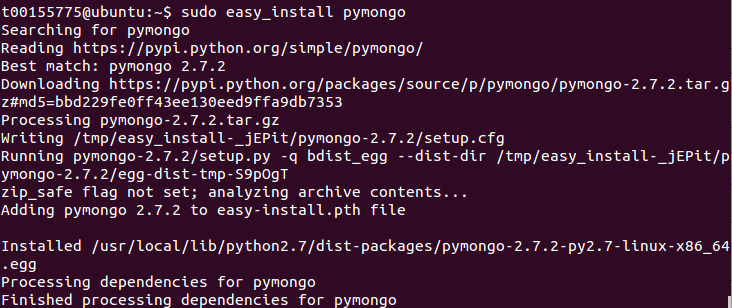


All other queries can be performed with the same ease using familiar SQL syntax. It is impossible to praise RazorSQL highly enough for ease of installation, use and incredible functionality on admittedly a relatively basic level. Easily create new collections, import existing data and more or less perform every operation available in sqldevloper, invaluable to any DBA struggling with a change to a new database system – HIGHLY recommended.

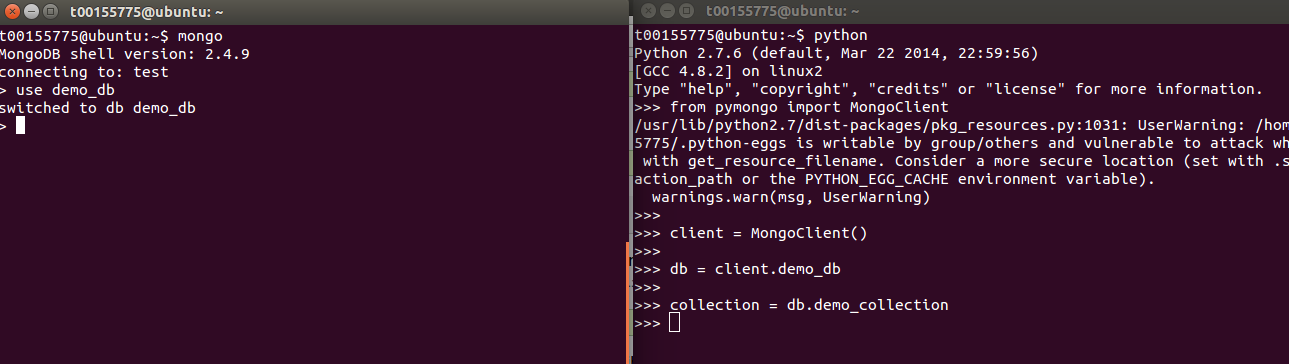
## Download a driver for a programming language of your choice and populate a collection through it.

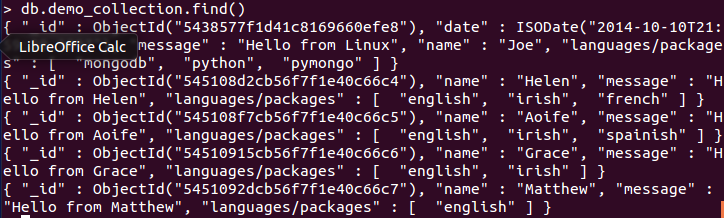
Similarly to part 1 of the portfolio Python running on a Linux virtual machine was chosen as the programming language. Researching drivers and plug-ins it was very noticeable that there appears to be much more support and tools/drivers etc. available for instances running on Linux systems. The fact that Python is built in on Ubuntu is also a big advantage. Some Java/C# drivers and plug-ins that were looked at appeared quite complex for the purposes of the exercise.

Initially a package was identified called pymongo which installed with no difficulty ( mongodb was also installed on the virtual machine)



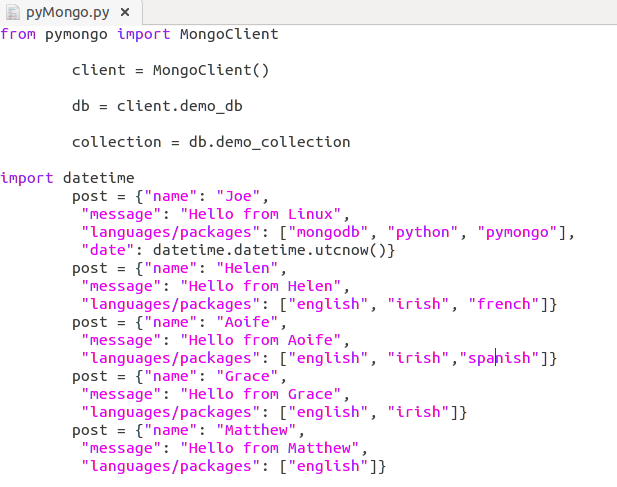
A MongoDB database and collection were created via Python and entries made to the collection.



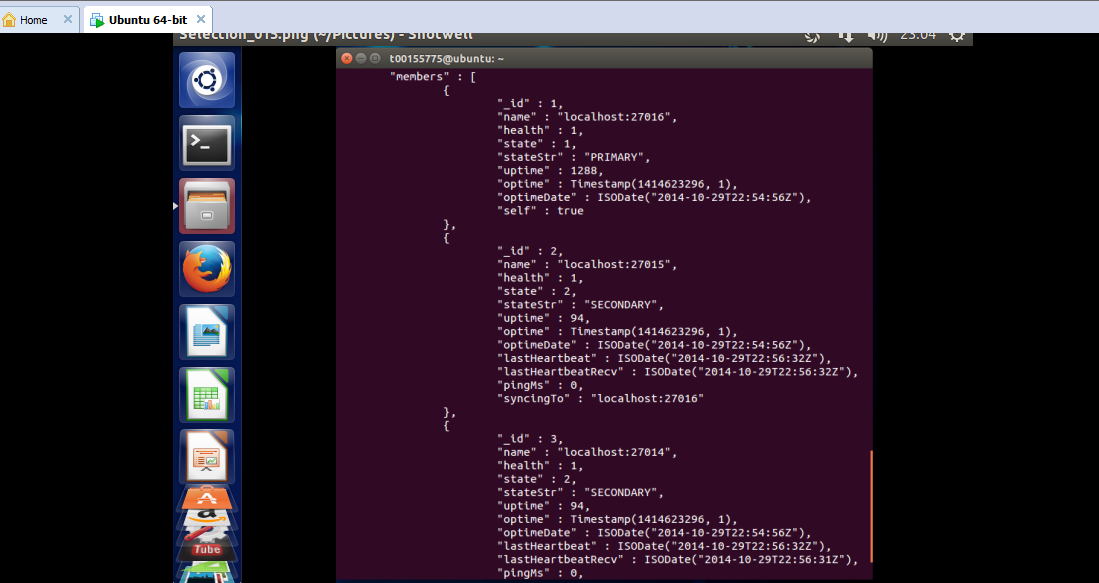


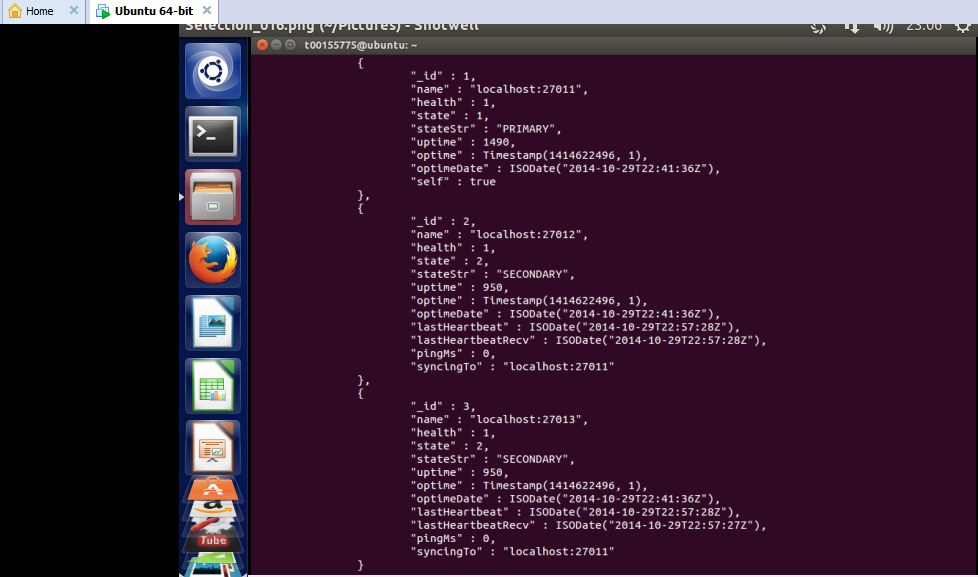
Confirmation of the post operations as completed via Python.

The Python script that was written to create the database, collection and populate the collection is shown below. One problem that occurred was that after the initial entry to the database the datetime.datetime.utcNow( ) failed to work. The simplest solution was to remove it from the script.

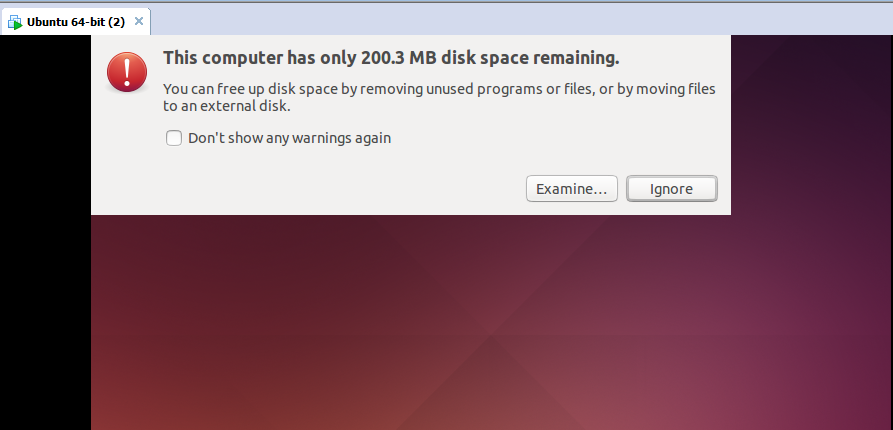


## (In Linux) run six servers: three servers in a replica set, and each replica set is one of two shards. Run a config server and mongos. Demonstrate the availability of your system under various scenarios. Run GridFS across them and show it working.



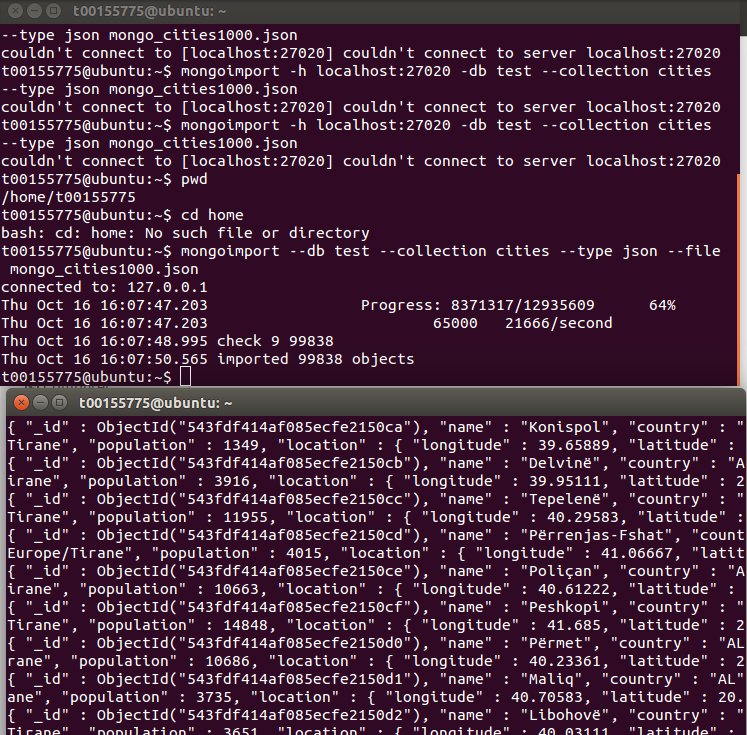


Six servers running in two replica sets, each replica set has a primary and 2 secondary servers. The remainder of this question proved extremely frustrating as the virtual machine in question repeatedly ran out of memory once the servers had been initiated. Repeated efforts were made on multiple virtual machines each with a larger hard drive capacity but the problem persisted.

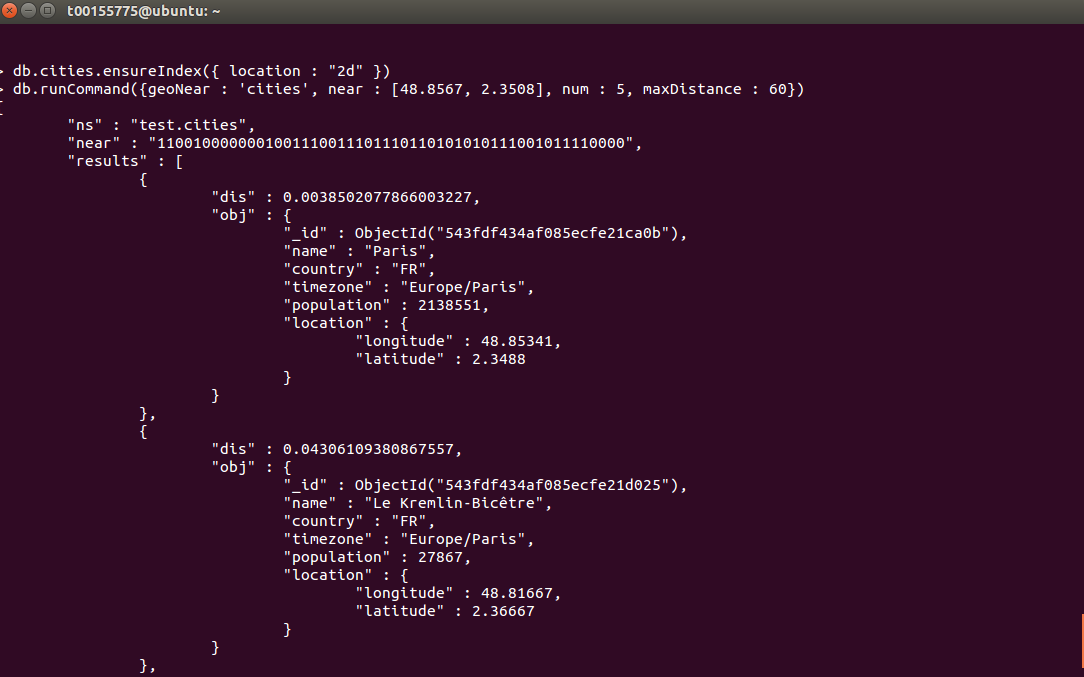


### Install the Cities database and show using Mongo’s bounding shapes, all the cities in a 60 mile box of Paris.

Importing the database proved very straightforward once it had been placed in the correct folder on Ubuntu prior to importing. The command used to import the file to ‘test’ db was **$ mongoimport –db test –collection cities –type json –file mongo\_cities1000.json** followed by **$ db.cities.find()** to confirm population of the database.



Once the file was imported the next portion of the question was relatively simple. Starting with indexing the collection which is required by the specific runCommand( ) in this case geoNear to operate successfully. A query very similar to this appears in 7 Databases in 7 Weeks and tailoring that command was all that completing this question required. Google was used to find the correct GPS co-ordinates for Paris which plugged into the near: parameter and maxDistance set to 60. The commands and results are shown in the following screenshots.





## Conclusion

MongoDB is similar in many ways to CouchDB and this definitely helped with this portfolio section. Most questions were completed quickly with most of the issues being simple typos and formatting errors in commands. Uploading an image initially caused problems associated with locating mongofiles and then realising that it needed to run from a regular CLI. The tool chosen for this database was easily sourced and extremely easy to understand and use, highly recommended. The only question that caused serious problems was running six servers on Linux, this was attempted on multiple occasions each having the same outcome – out of memory capacity on the virtual machine running Ubuntu. This was extremely frustrating and eventually a decision was taken to complete as much of the question as possible and leave it unfinished.