**Portfolio Part 1: CouchDB**

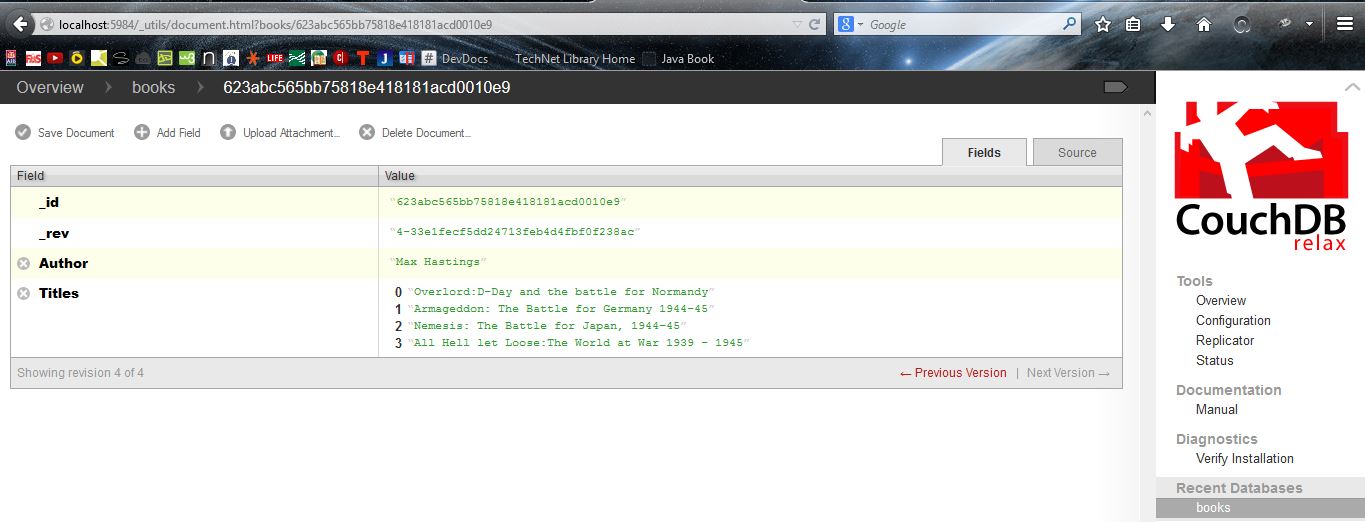
**T00155775**

**Joe O Flaherty**

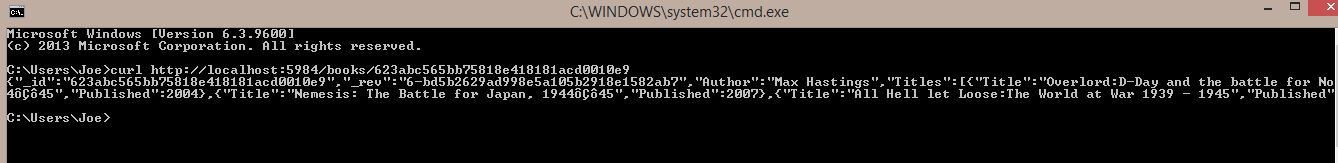
**BSc (Hons) Computing with Multimedia**

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

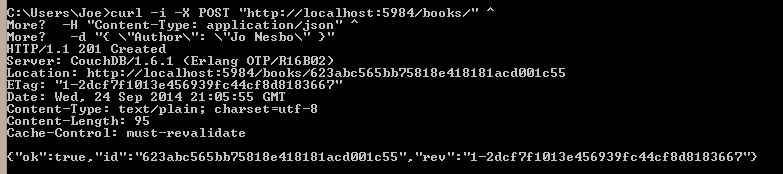
**Create:** Database of books and authors created with 1 author and a number of titles added.



**Read:** Read operation performed against the database from the command line via **curl**



**Post:** New author to the database

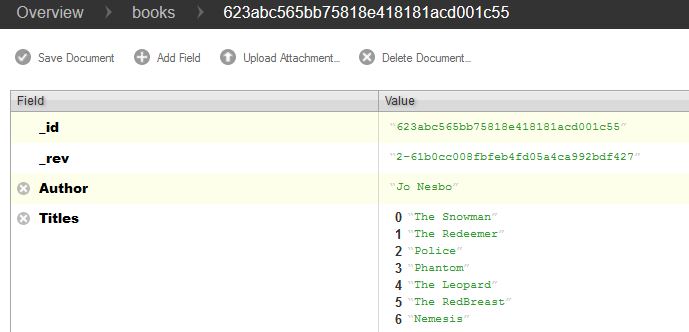


2 Futon views following POST request

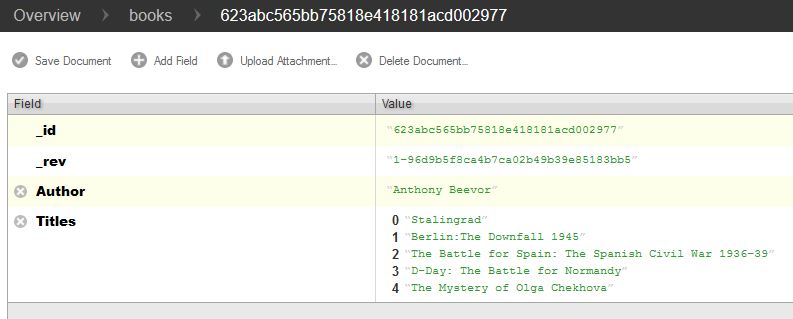




New titles added to Author “Jo Nesbo”



3rd Author added to database and titles populated





**D**eleting Anthony Beevor from the database using the Futon interface.

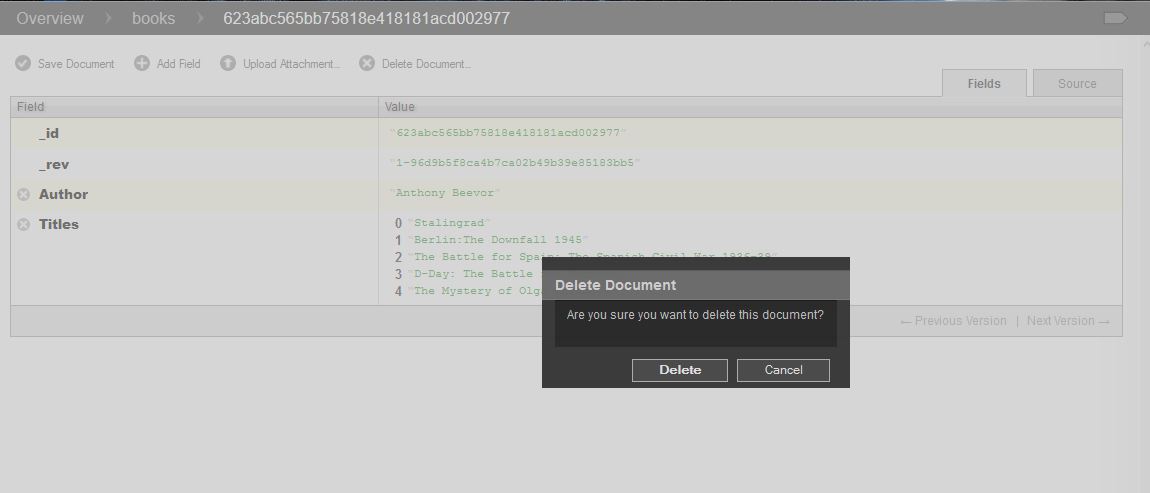
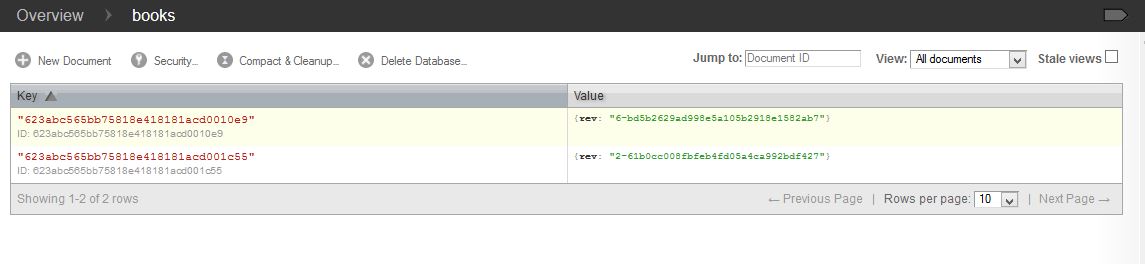
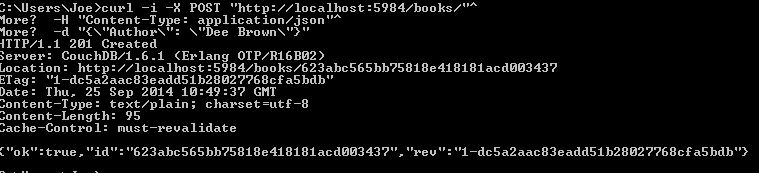


Image of the database following the deletion.



Added a further author to the database and populated titles via the command line interface using **PUT** the equivalent of Update









## Add attached documents (images) to the database

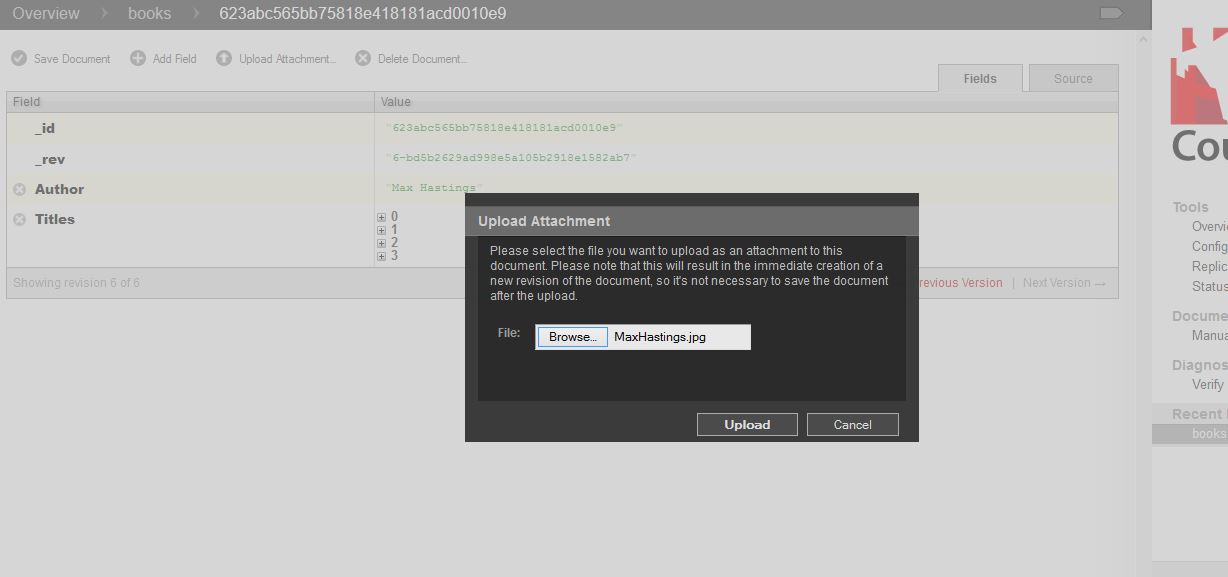
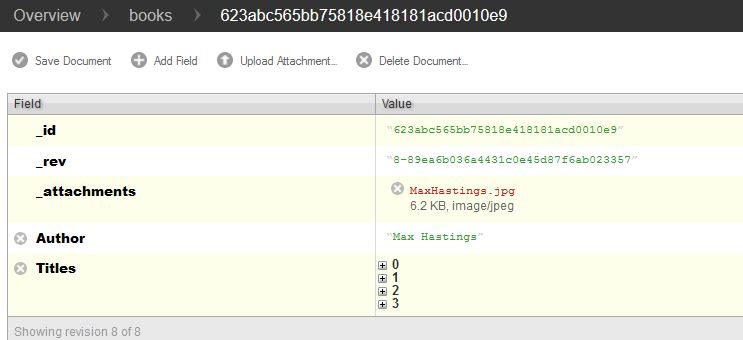
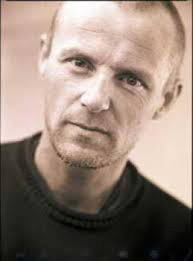
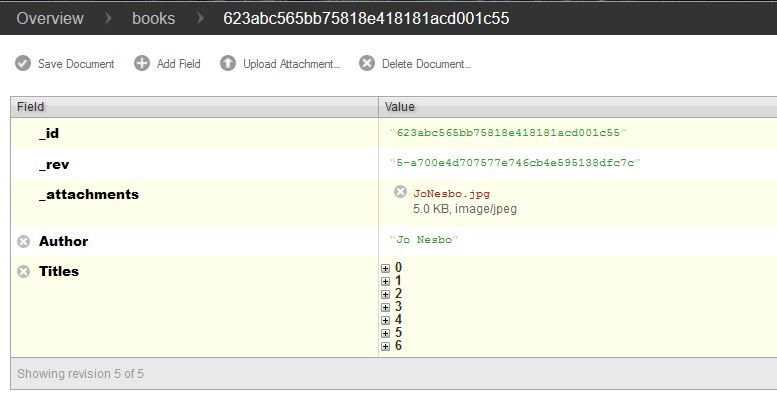
 

Image is attached to the correct Author in the database.



And a second image attached to “Jo Nesbo”

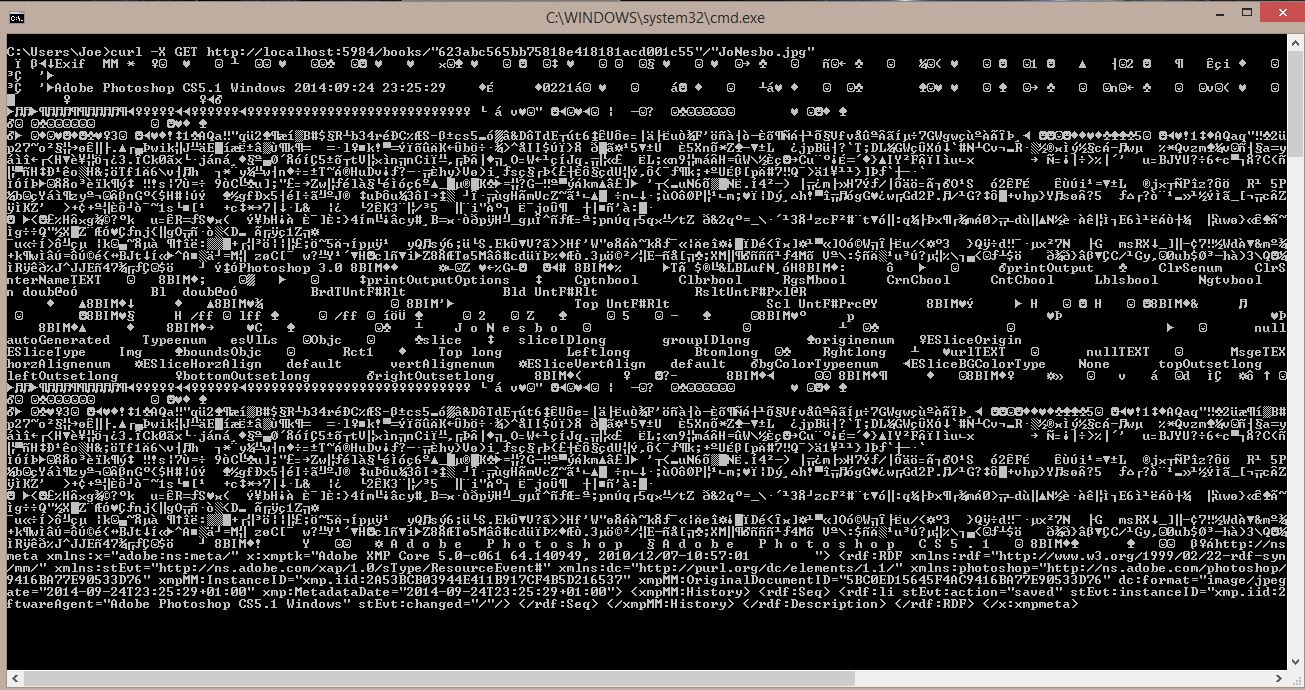
## Craft and execute a cURL request that will return just an image

This question proved a little problematic – **not** constructing the curl command which was very straight-forward.

The command used to return an image from the database was:

**Curl –X GET http://localhost:5984/books/”623abc565bb75818e418181acd001c55”/”JoNesbo.jpg”**

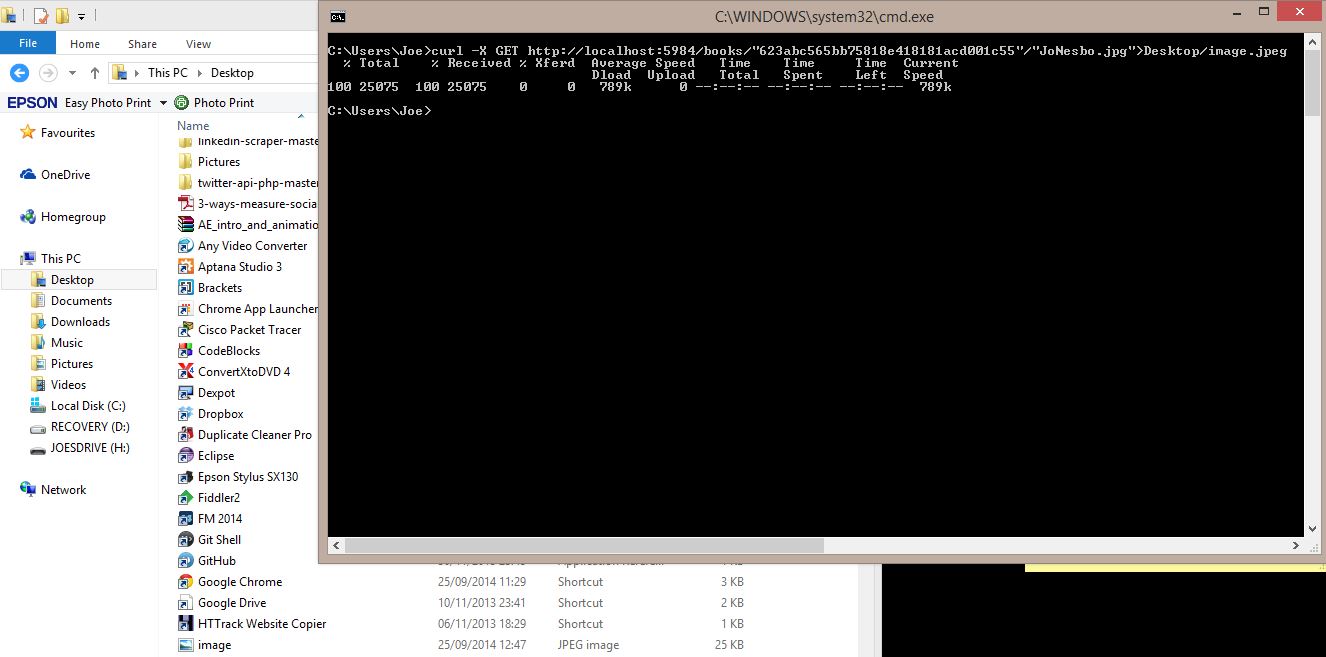
When this command is run from the command line it returns completely useless binary code as seen below.



The data returned to the command line does contain some information about the image file but remains totally useless.

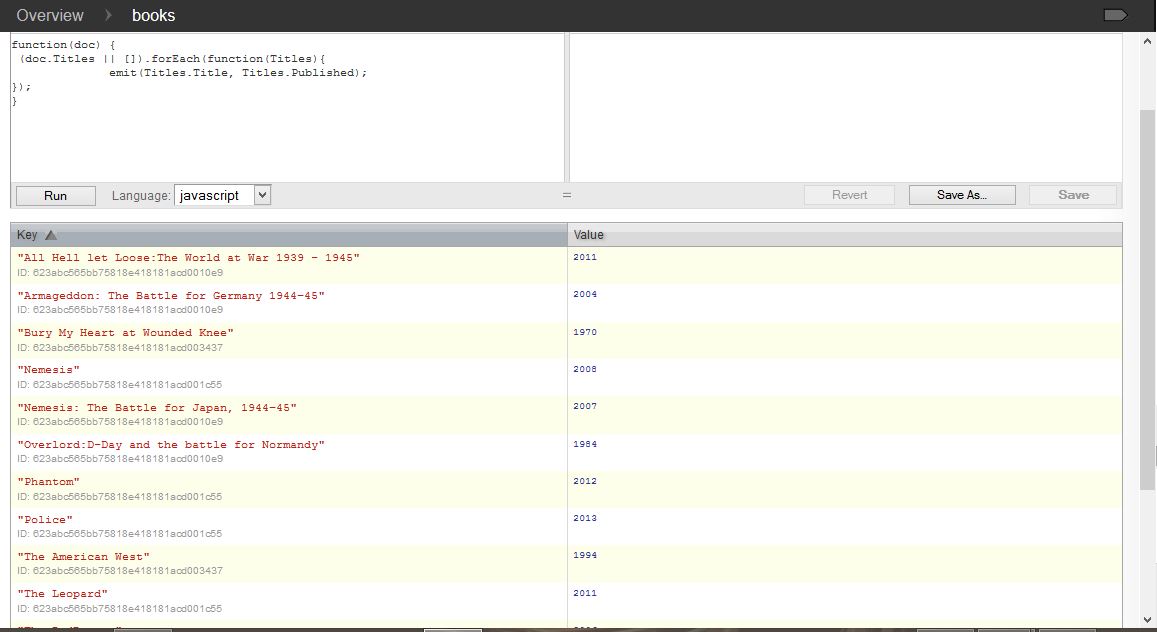
I decide to overcome this problem for demonstration purposes in a relatively simple way by sending the returned image file to a location on the desktop of my personal laptop.

This is accomplished by adding **>Desktop/image.jpg** to the end of the original curl request; now the image can be viewed as a normal image file.



## Develop your own map reduce function on the database and explain how it works

Prior to implementing a map-reduce function simple views similar to the example below which show a book title and date of publication were used.

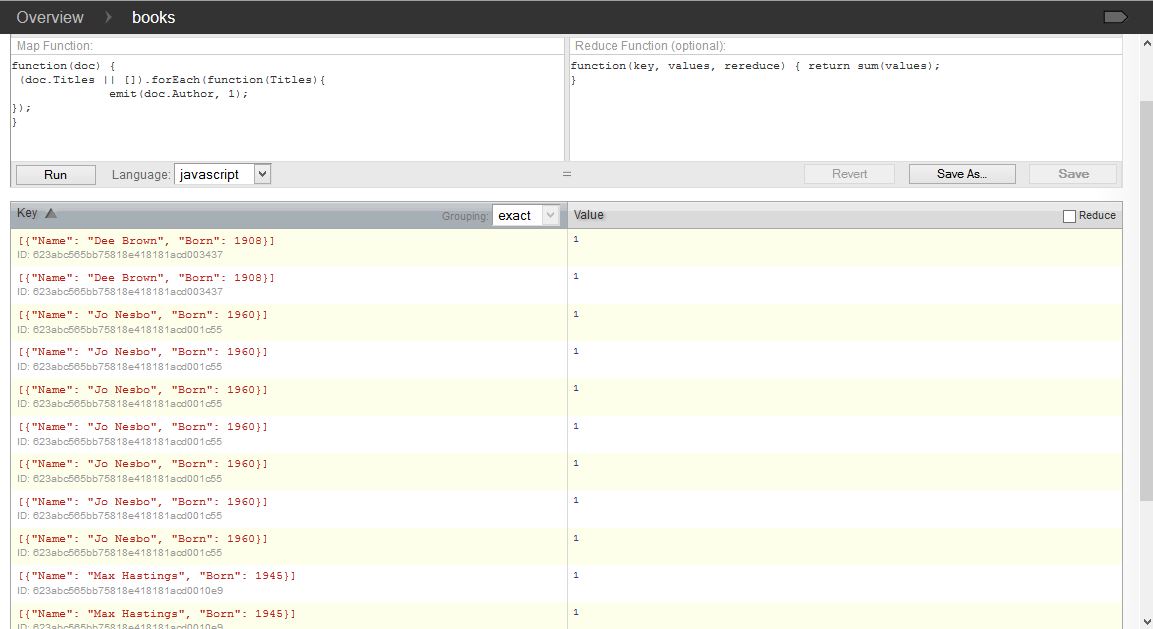


As a result of the database implemented being quite small it proved difficult to implement a useful map-reduce function. It was decided to simply find out how many books each author has in their Titles document.

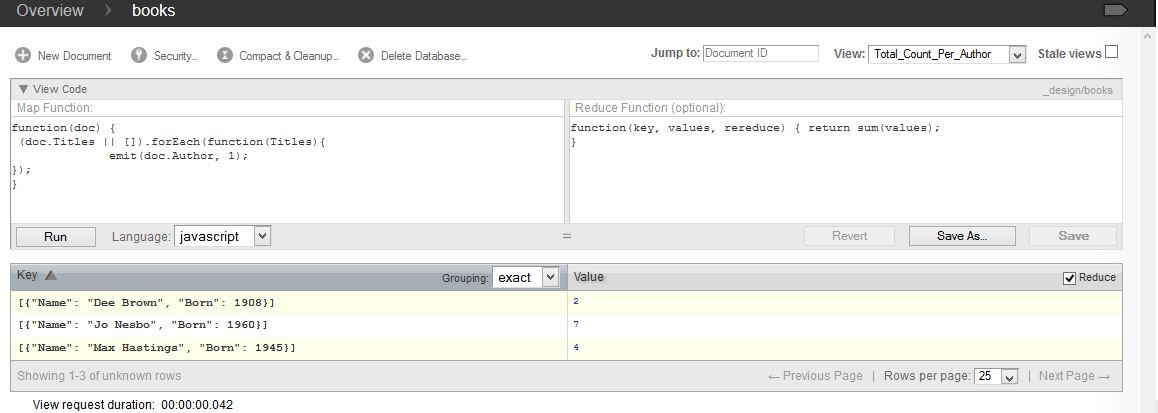
The initial view shown below with reduce de-selected simply examines each document in Titles and emits the **key** (author) of each title and a **value** (1) per individual title returned.

The reduce function is then selected which takes the key value from the map function which in this case is author and the value from the map function in this case 1 and returns the sum of titles per individual author. The operation as can be seen at the bottom of the second image took 42ms to perform on this very small database.

**Map Function – Reduced not enabled**



**Map Function with Reduce enabled**

****

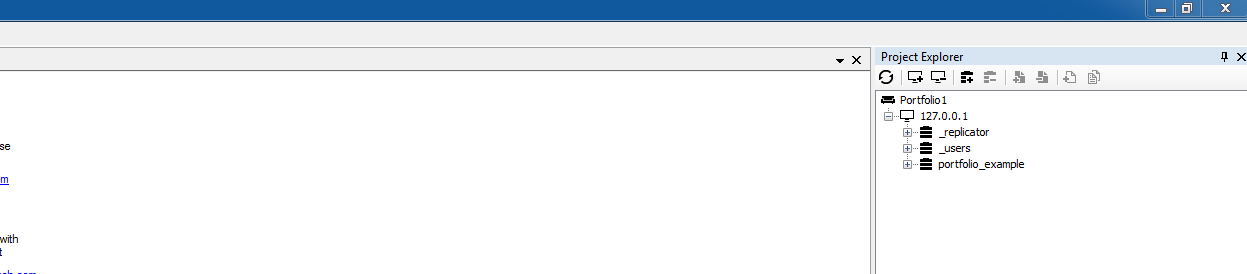
## Download, install and evaluate a tool to use with CouchDB

This was surprisingly difficult as there are a lot of tools/plug-ins available but very few that either (a) come with adequate documentation (b) run on windows based instances of CouchDb or (c) do not require installation and/or knowledge of specific programming languages.

Resulting from these problems it was decided to install and trial an IDE for use with CouchDB called Kanapes IDE <http://kanapeside.com/>

This is described as the most complete development environment for CouchDB but this turns out to be an easy claim to make as there is little or no competition.

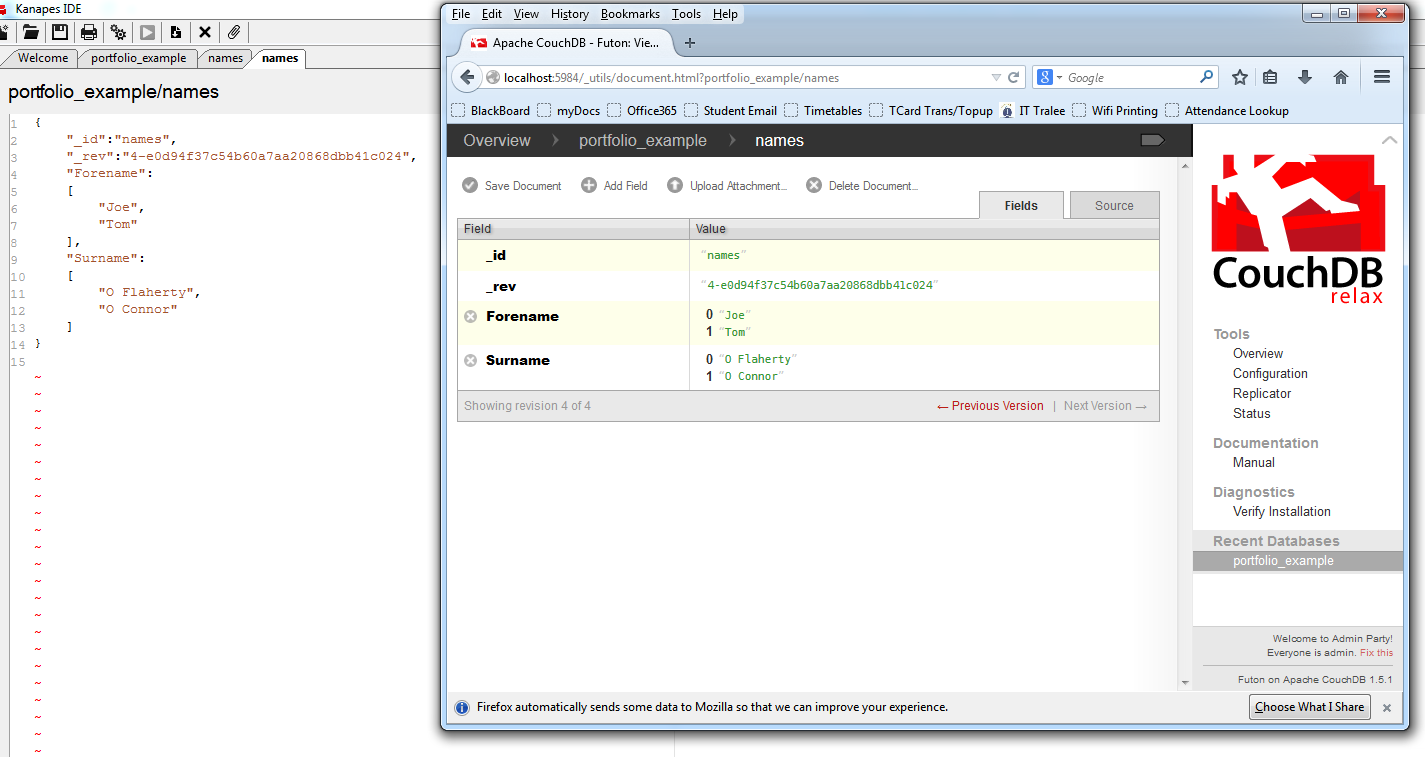
Initially the biggest difficulty and the main criticism of Kanapes is the almost total lack of documentation, instructions and tutorials available, this makes initial usage of the tool quite difficult and frustrating. Following a little experimentation the opinion of the IDE has changed, once a new project is created the other basic operations one may wish to perform are quite straightforward and intuitive. The simple project explorer panel used by Kanapes greatly helps in creating this intuitive feeling; however a little bit of documentation should be provided to give users a ‘kick-start’ so to speak.



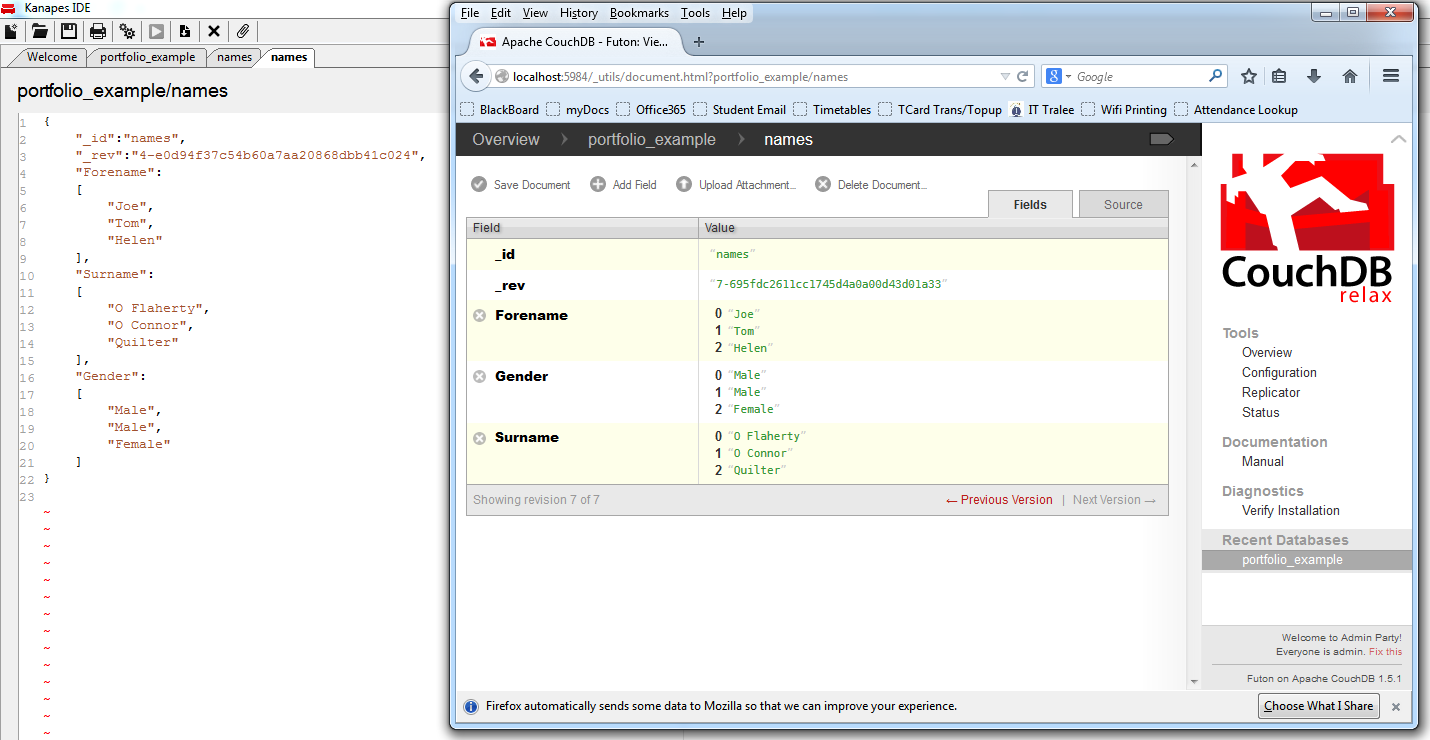
**A database (portfolio\_example) created in Kanapes IDE**

Once a user has discovered how to add a server to their project everything else is simply a matter of hovering over the icons in the explorer window and reading the tool tip to discover what each option does.

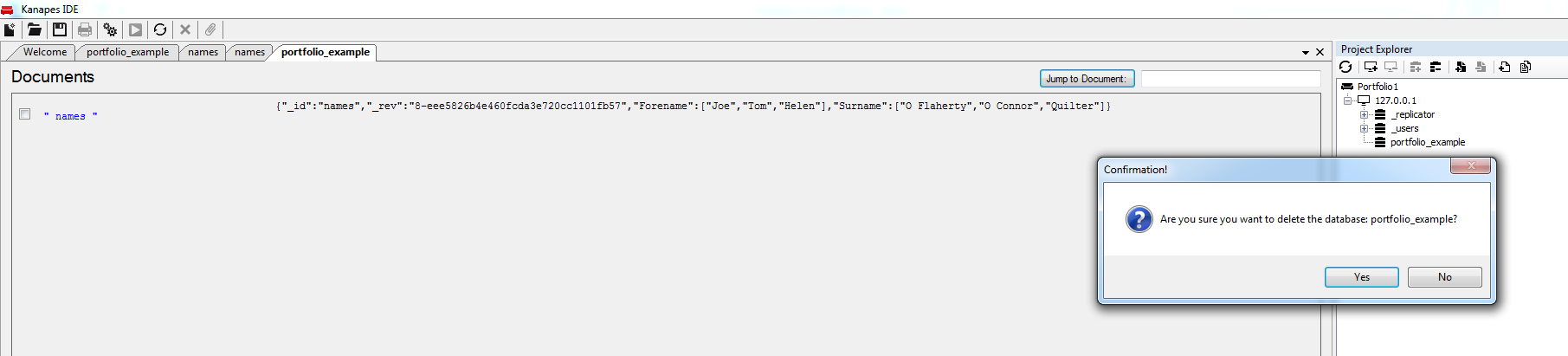
Performing the basic CRUD operations from the IDE proved very straightforward once the various options had been tested and a degree of familiarity achieved: **See following screenshots for examples**.



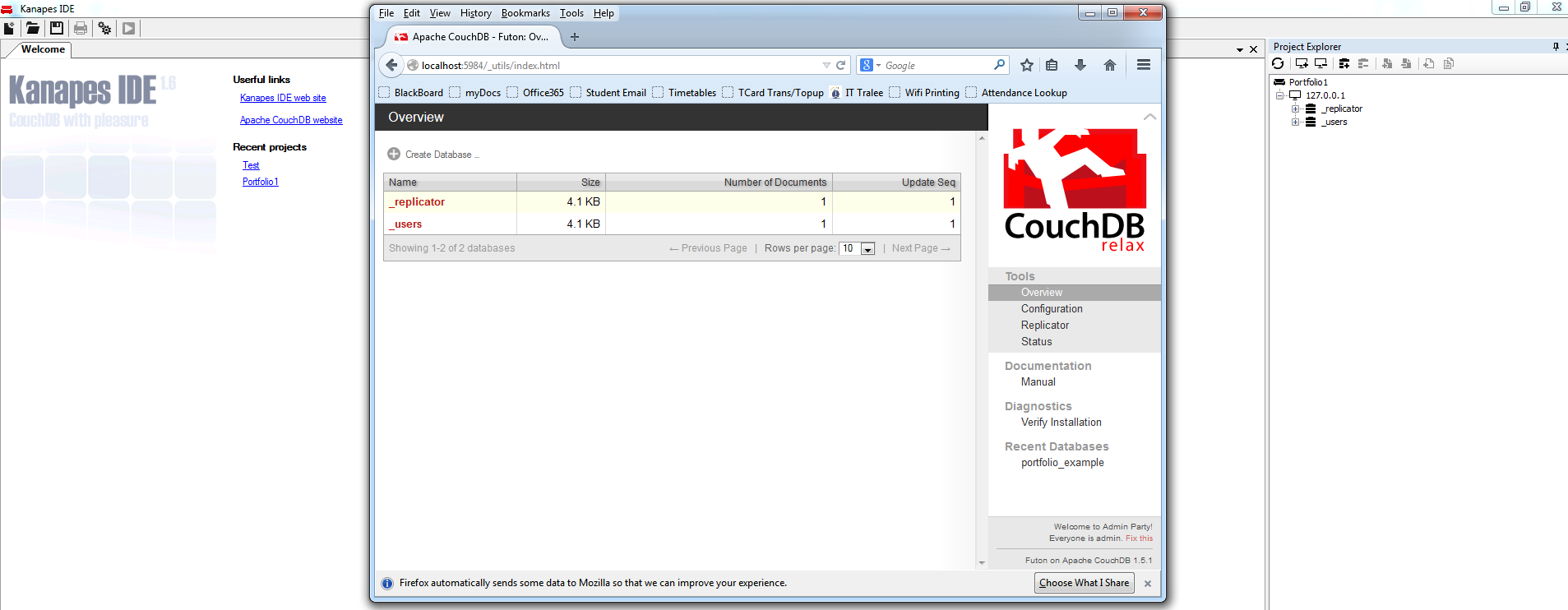
**Creating a new document from the IDE confirmed by accessing Futon.**

****

**Update performed in IDE – adding a Gender value and populating it.**

****

**Delete the database from the IDE – confirmed via Futon – shown below**

****

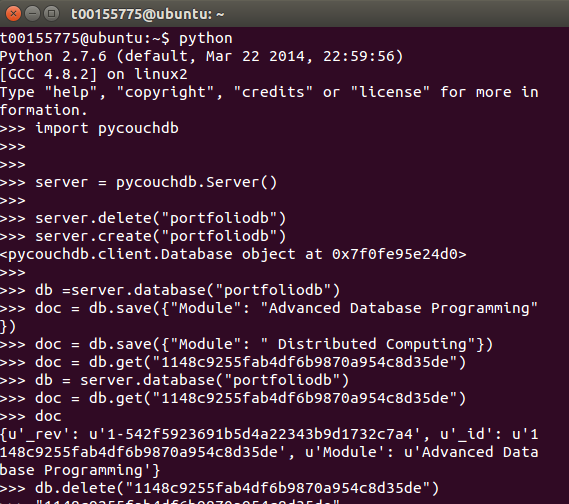
## Download a driver for a programming language of your choice (Android would be nice!) and run the CRUD operations and a map reduce operation from the programming language

For this question initially plug-ins and drivers that would work on a windows based system in a familiar language were considered. Following some experimentation and research the decision was made to change this approach.

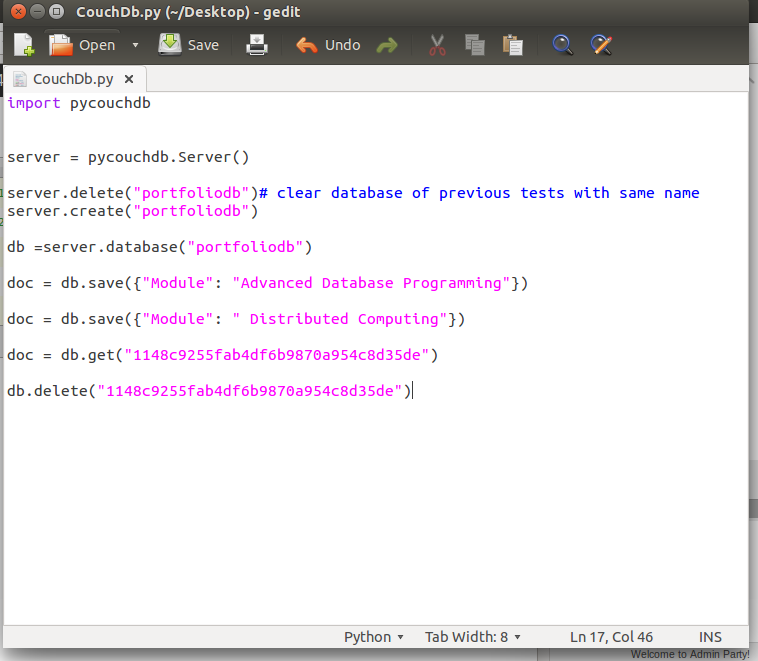
Python was chosen as the language and work was undertaken on a virtual machine spun up on a personal laptop running Ubuntu (Linux). The reason for choosing a Linux distro is that Python comes pre-installed with Ubuntu thus reducing a lot of the complication installing on Windows can cause. Having decided on a language and operating system a Python module called **pycouchdb** was selected**.** This module mustbe installed via Linux terminal before it becomes visible from any Python scripts it is called from. One problem with this as at first was denial permission to access the destination required for the installation. The first command used was **pip install pycouchdb** which failed, the issue was solved by placing **sudo** in front of the command which is the Linux equivalent of run as administrator. Once the package was installed it was a matter of following a tutorial and writing a simple Python script to run operations against CouchDb.

**Note:** pycouchdb connects by default to http://localhost:5984/ so this is not specified in the script, this can be changed if needed and authentication provided via a constructor.

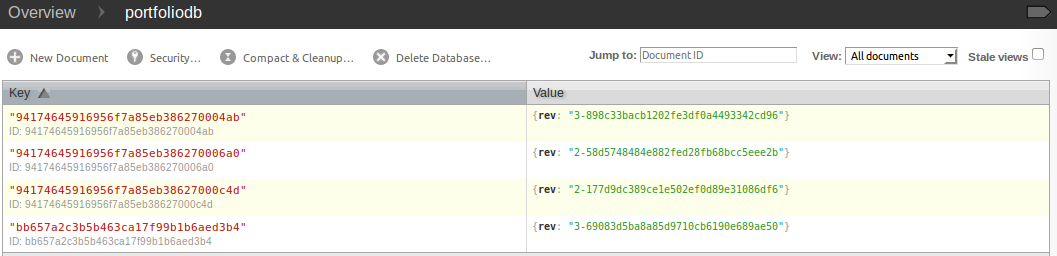
Most of the functions were relatively straight-forward however a problem emerged i.e. specific update functionality is not provided for.



**Commands running from Python interpreter on Ubuntu: Create, Read, Update (not supported) and Delete.**

****

**The same commands shown as a Python script CouchDb.py**

****

**Futon view of database on Linux VM**

## Map-Reduce:

Unfortunately pycouchdb proved a very poor choice as a driver due to the lack of functionality provided for multiple operations. A second driver was used for the more complex operations i.e. Map-Reduce simply used as an import statement from the command shell and then connected to the existing ‘portfoliodb’ database.

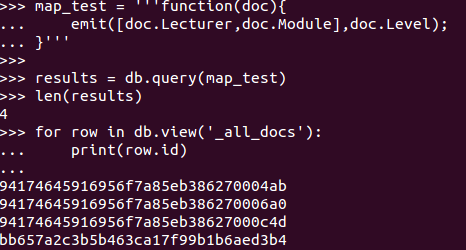
import couchdb

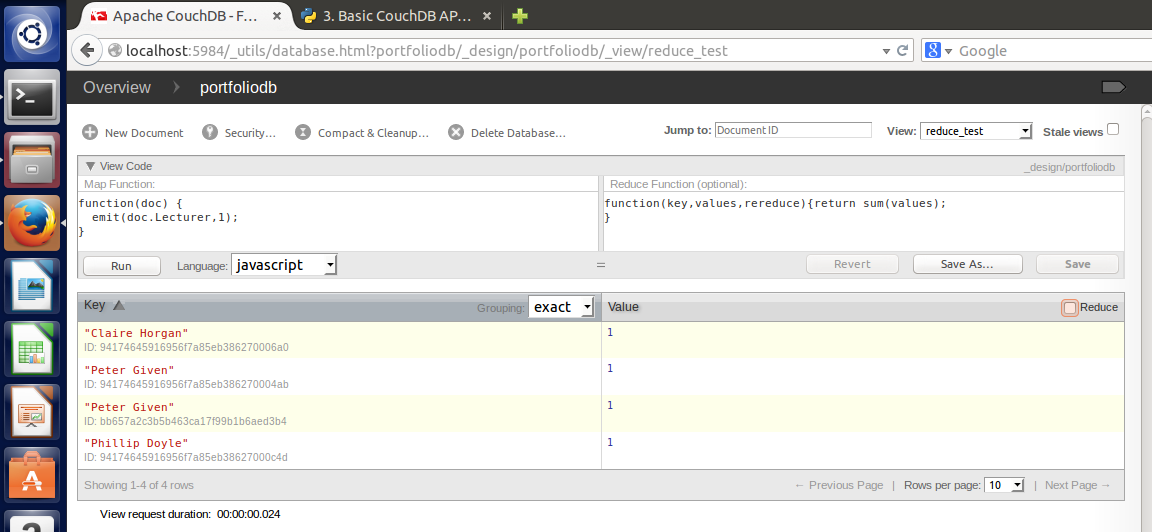
>>> couch = couchdb.Server()

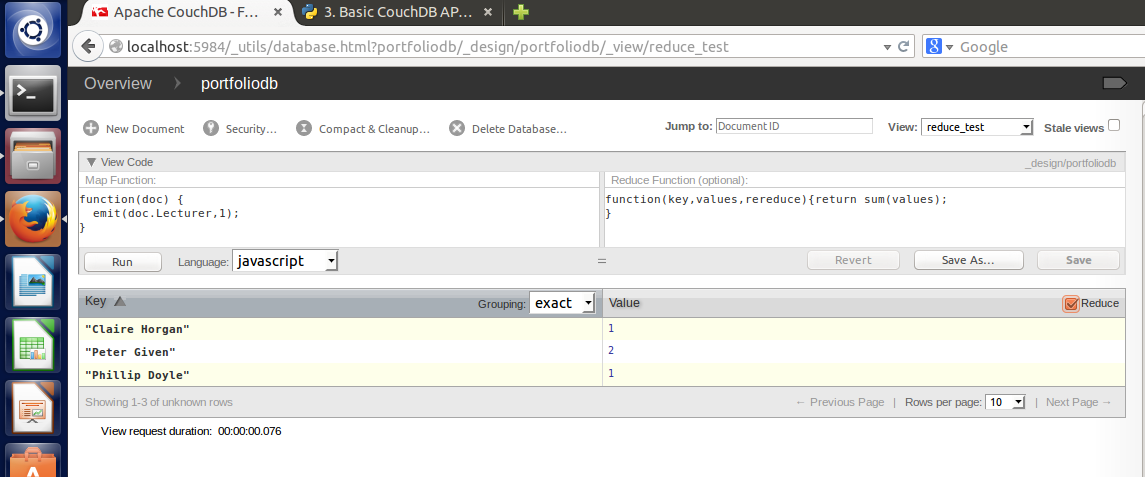
>>> db = couch['portfoliodb']

>>> db.name

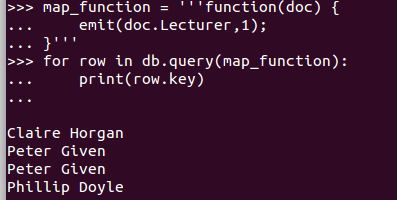
Some simple functions and queries were used to experiment with the API as shown below.





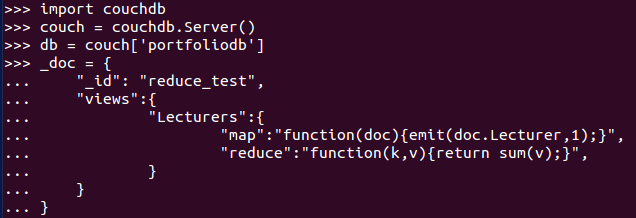


The map and reduce functions that will be re-created using Python.



The map portion returns just lecturer names. Unfortunately the driver chosen does not provide the required reduce functionality, a fact that was not clear until quite a lot of work had been done on the database.

Following a lot of searching and trials the following function was hoped to work, unfortunately it too failed.



## Conclusion:

This portfolio ultimately proved the easiest to complete except for the final question. CRUD operations were easily implemented and posed no significant problems, the Futon interface being an excellent environment. Map reduce was a little trickier and took a lot of time to run a successful reduce operation. Sourcing a suitable tool to evaluate also proved more difficult than anticipated with not many tools available that were accessible quickly. Choosing Python as a programming language was very successful for the purpose of performing CRUD operations, unfortunately the module chosen did not implement reduce making completion of the final question impossible.