Advanced Database Programming Portfolio 1 - CouchDB

Niall Stack T00174406

2016

# Q1 - Implementing My Own CouchDB Database

## Introduction:

I was tasked with creating and documenting the creation of a CouchDB database for my Advanced Database Programming class and as I had previously never used a NoSQL database it was a new experience. The following is the documentation of me creating my database for a PC parts shop.

## Vision:

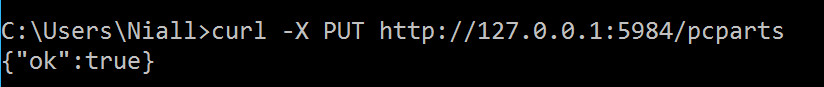
I wanted to create a database for a PC parts shop that I envisioned requiring a database that would be available both on desktop and mobile so that staff can check the database while moving around and so customers and see what’s in stock online. CouchDB is a good fit for this database as it has the ability to replicate between mobile and desktop as opposed to MongoDB, another NoSQL database. It is also more suited to use a NoSQL database since the products can vary significantly and so might need different properties which can’t really be done with an SQL database also new properties can be added with relative ease.

## Creating the Database:

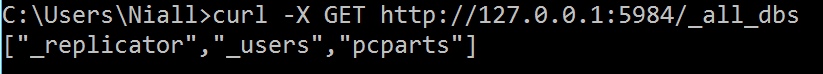
For part one I decided to create a database of various computer components for the PC parts shop. I would have 6 fields:

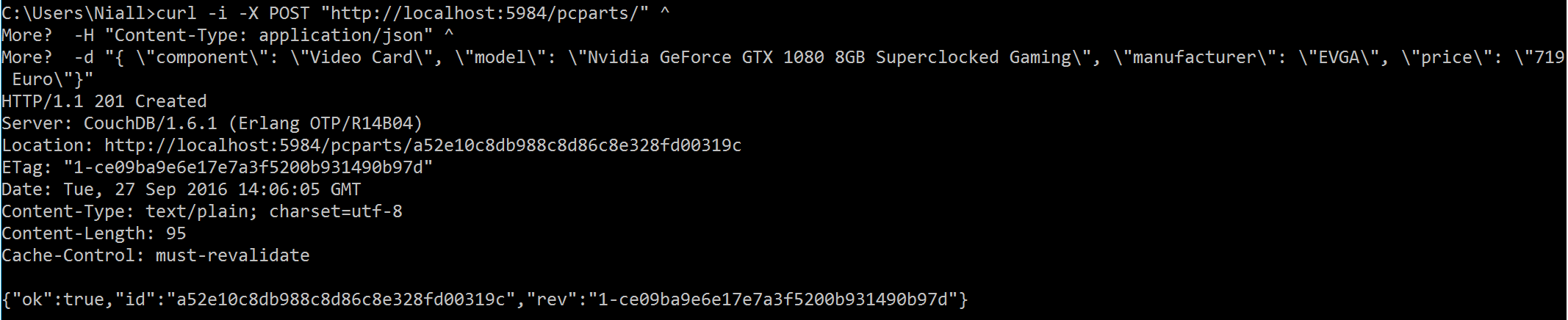
* \_id
* \_rev
* component
* model
* manufacturer
* price
* picture

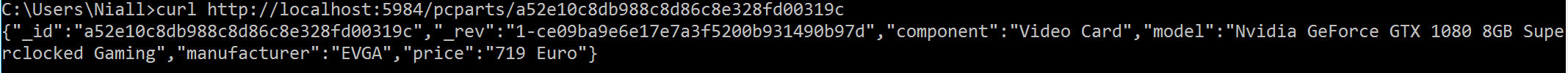
I created the database with the following command.

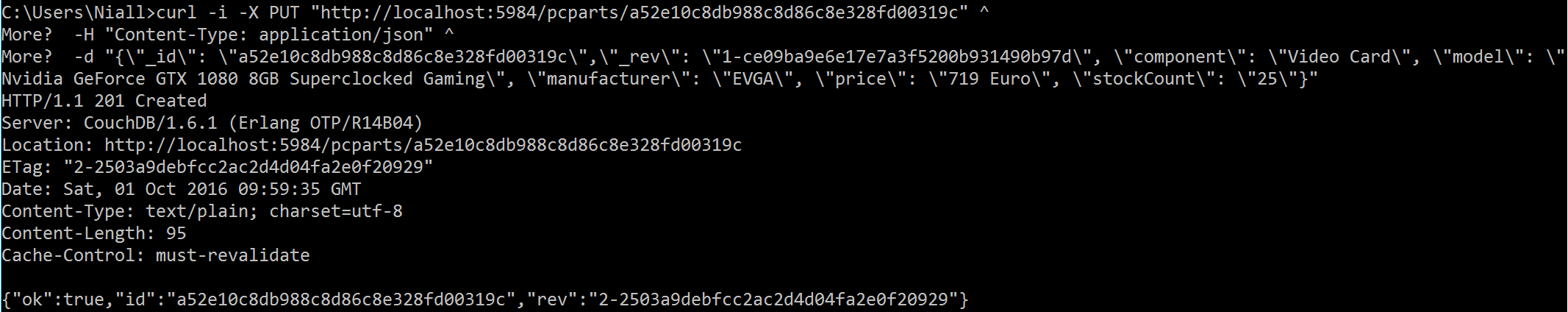


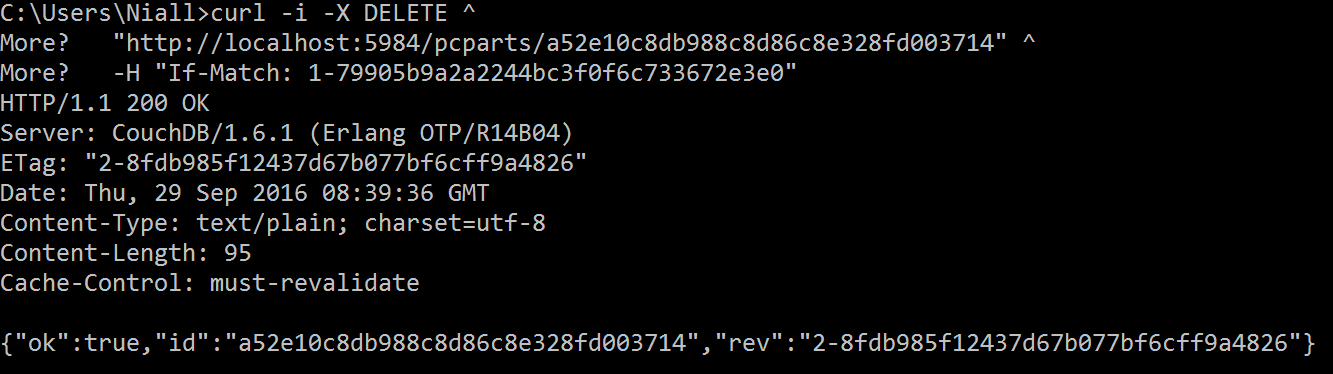
Then I viewed all the databases with a GET request.



Once I had created the database I could now add documents to it.

I could then see this document in the database.

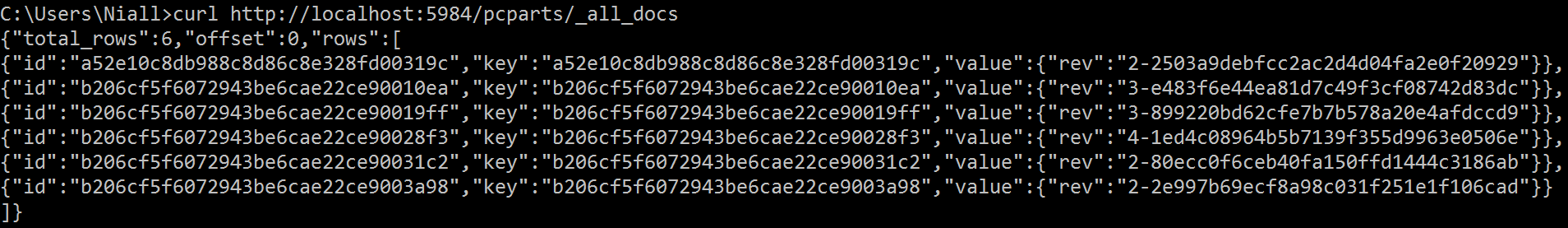
**I then decided to add both a stock count field to my document so I added it with a PUT request.

I later created a document that I wanted to delete.

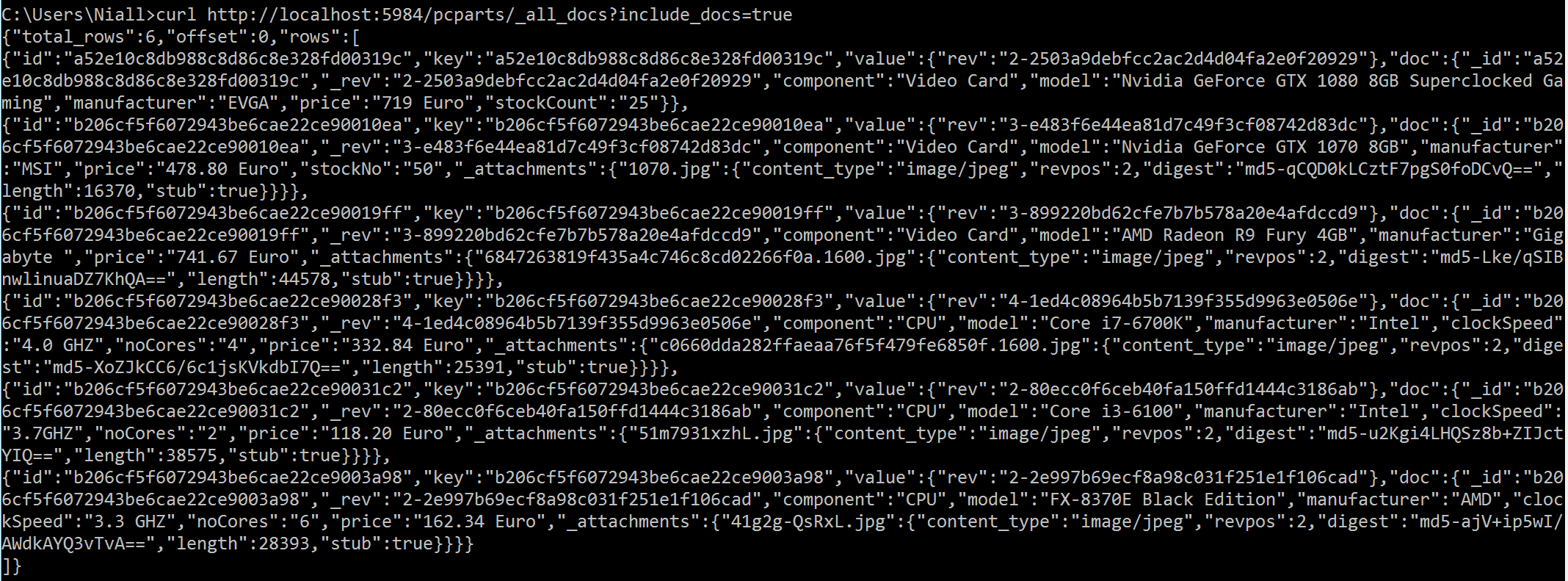
I also added images to the database but was unable to get this to work through the command line so I had to use Futon. So one problem with CouchDB is the difficulty of uploading images without using a REST client.

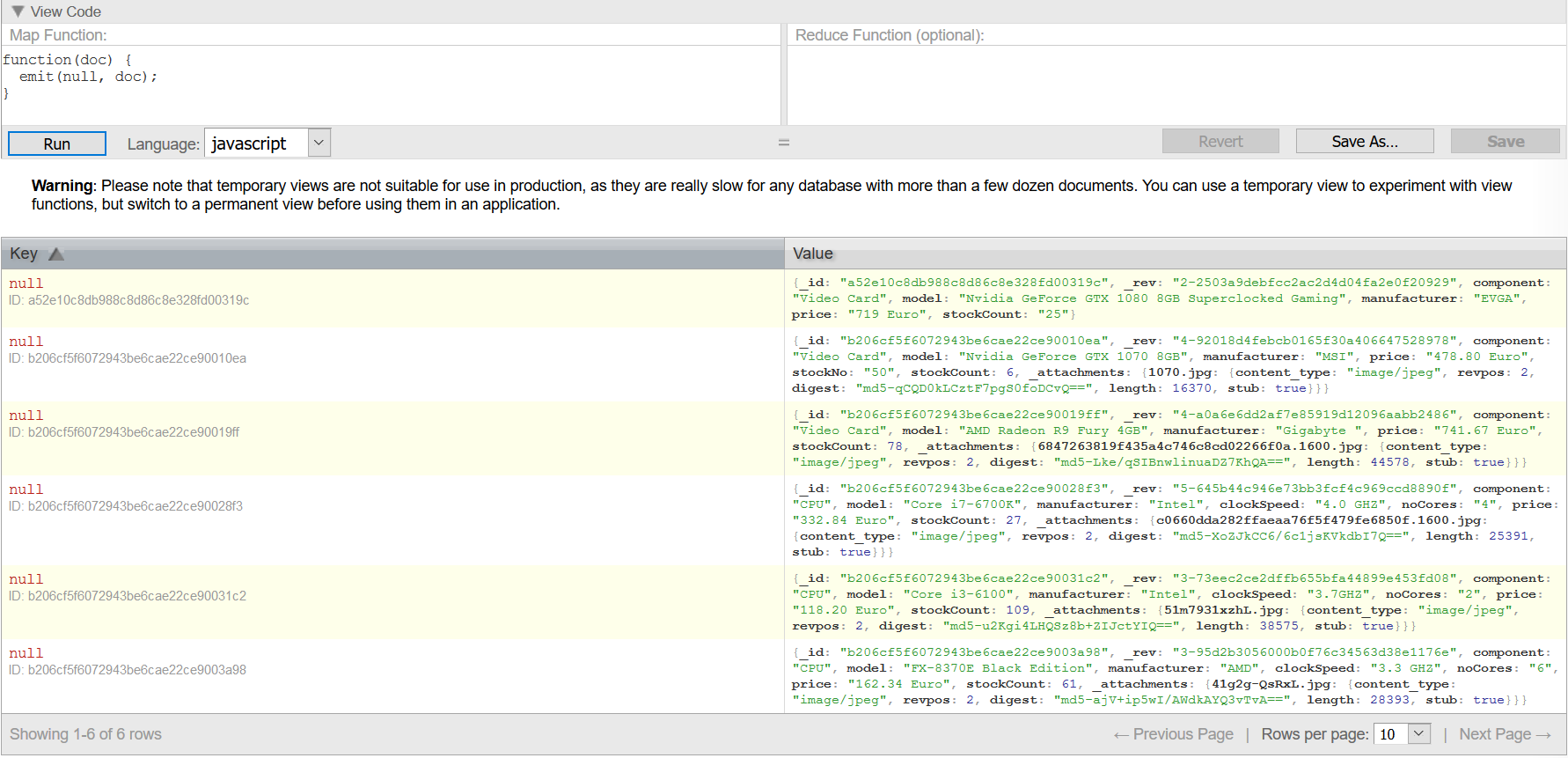
## Views:

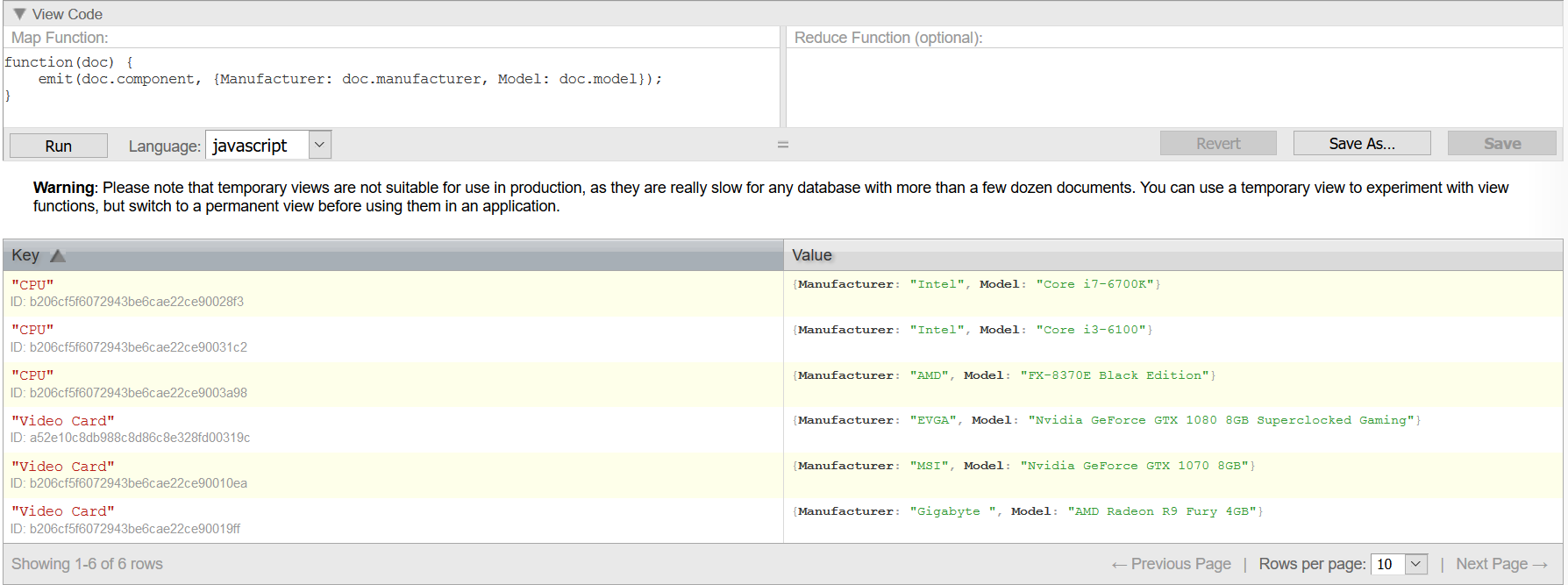
I later added more documents with various different categories, a useful feature of CouchDB which is a more practical for me than if I used an SQL database and had to use all the same fields. I used a basic view to see all documents in my PC parts database.



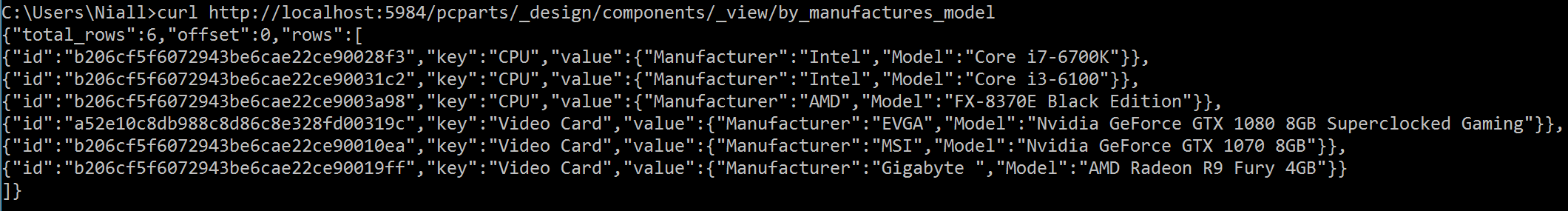
### 

To view all the headers I changed the command.

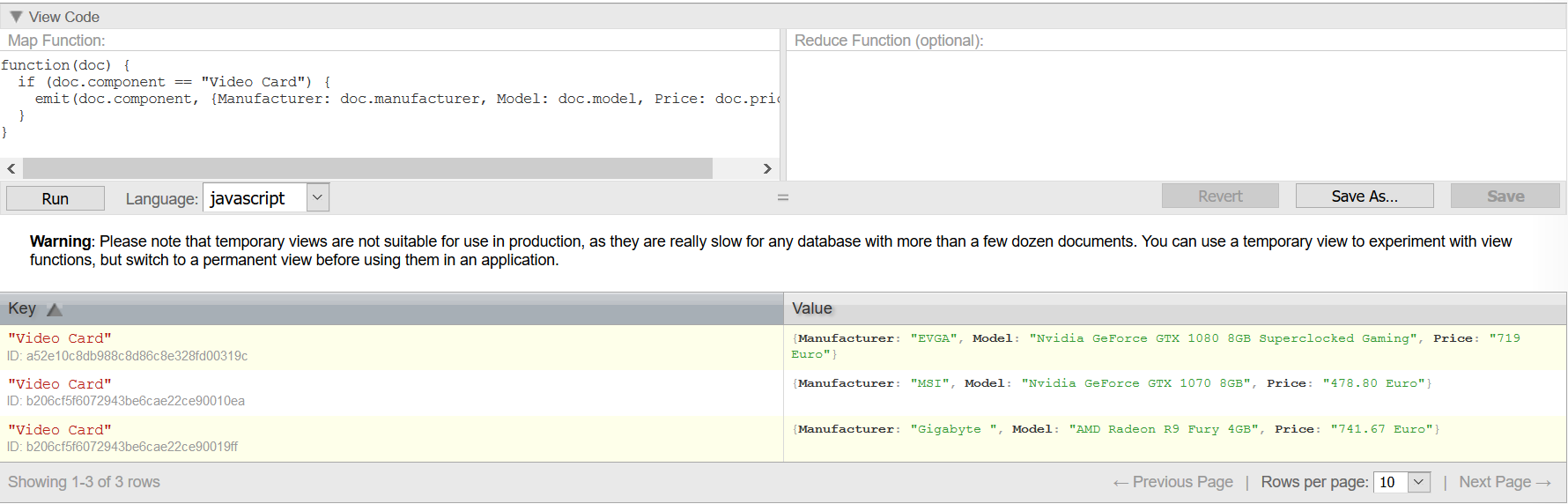
CouchDB allows you to create views with JavaScript functions, here is the default one that emits a key of null and a value of the whole document.

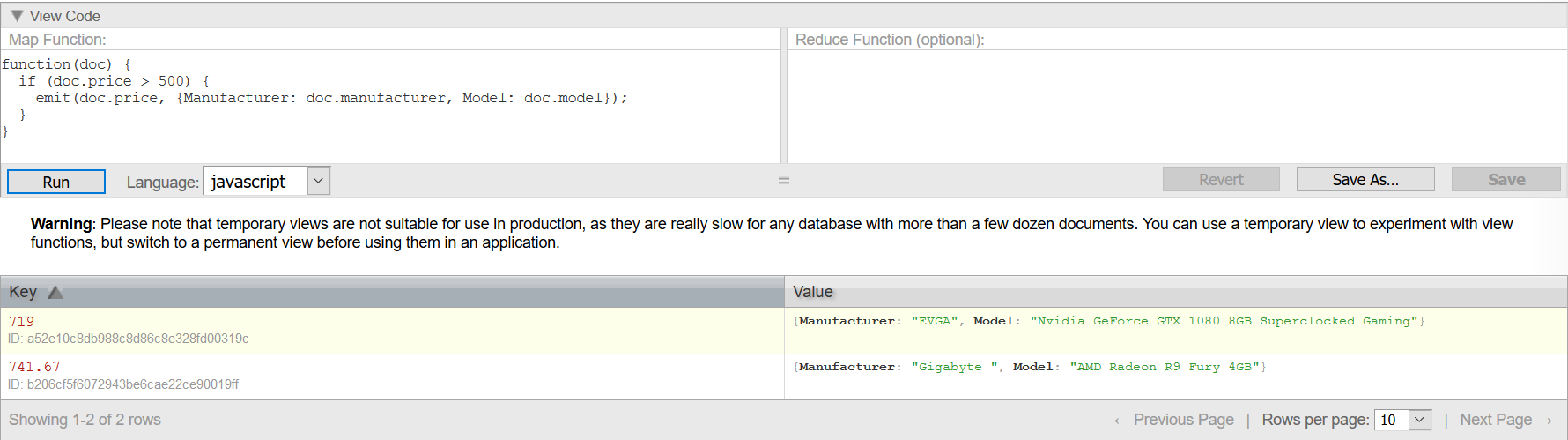
I created my own view so I could sort by component and just be able to see the manufacturer and model instead of all the information.

The simple map function above simply emits the component as the key and the manufacturer and model as the value, creating a simple view that is easy to read. I can also see this view through the command line.

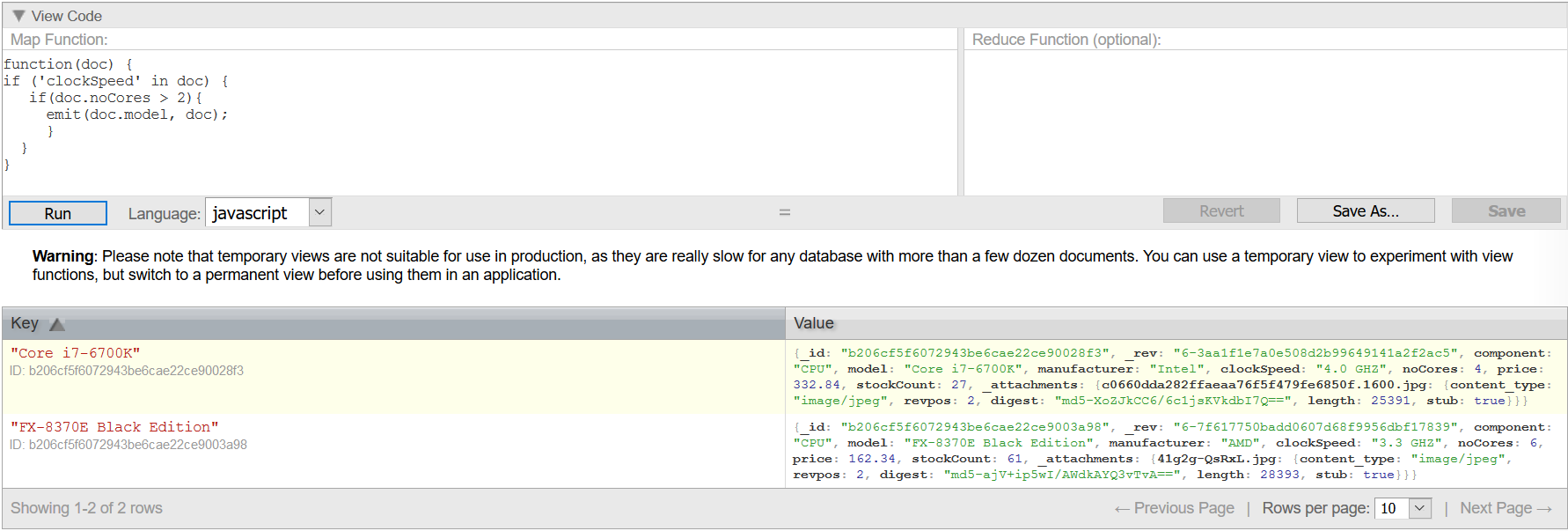


I then decided that instead of seeing all components I wanted to just see video cards and their price which I could do with an if statement that would only emit the component if the component name was “Video Card”.

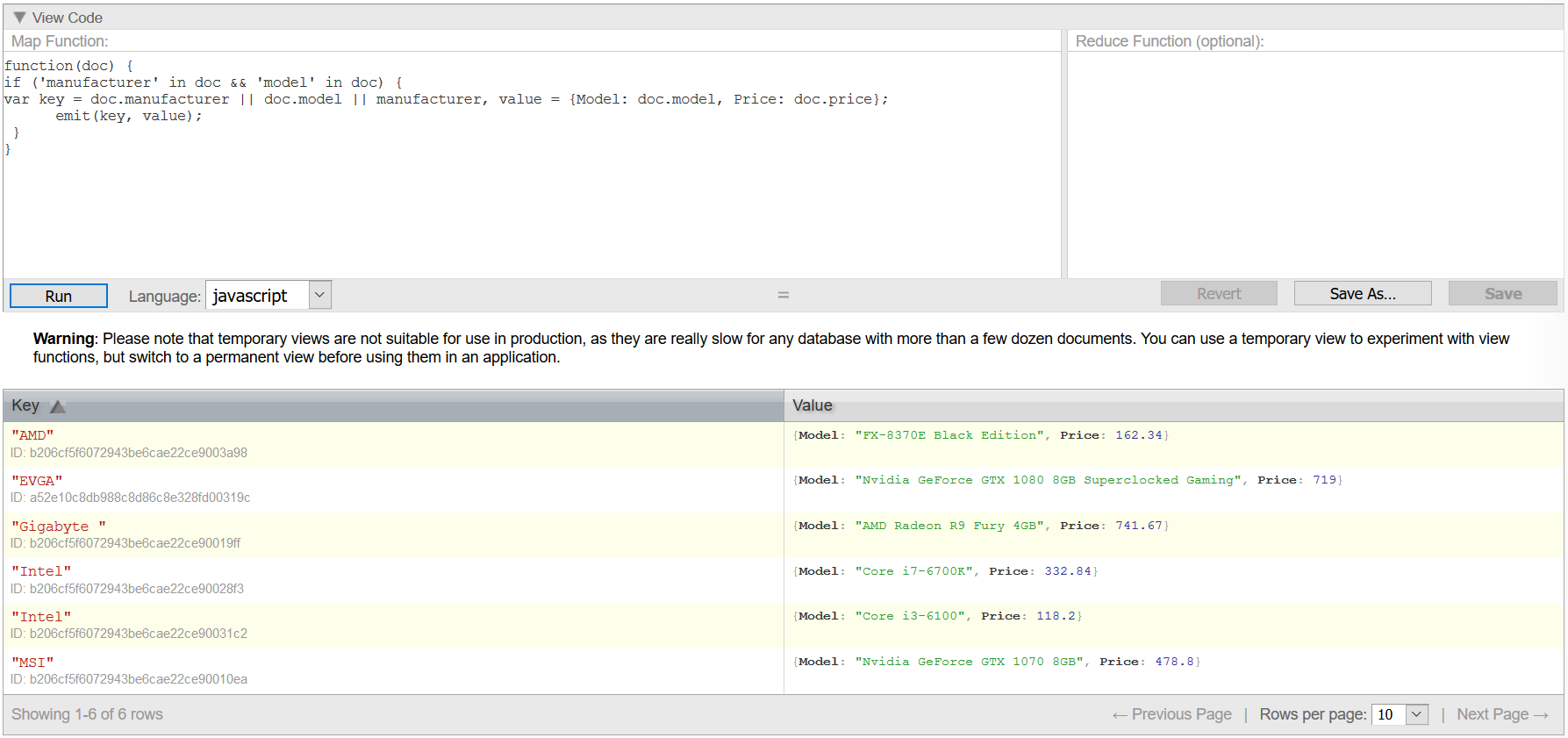


I was also able to use an if statement to search for components over a certain price and emit that price as the key with the manufacturer and model as the values.

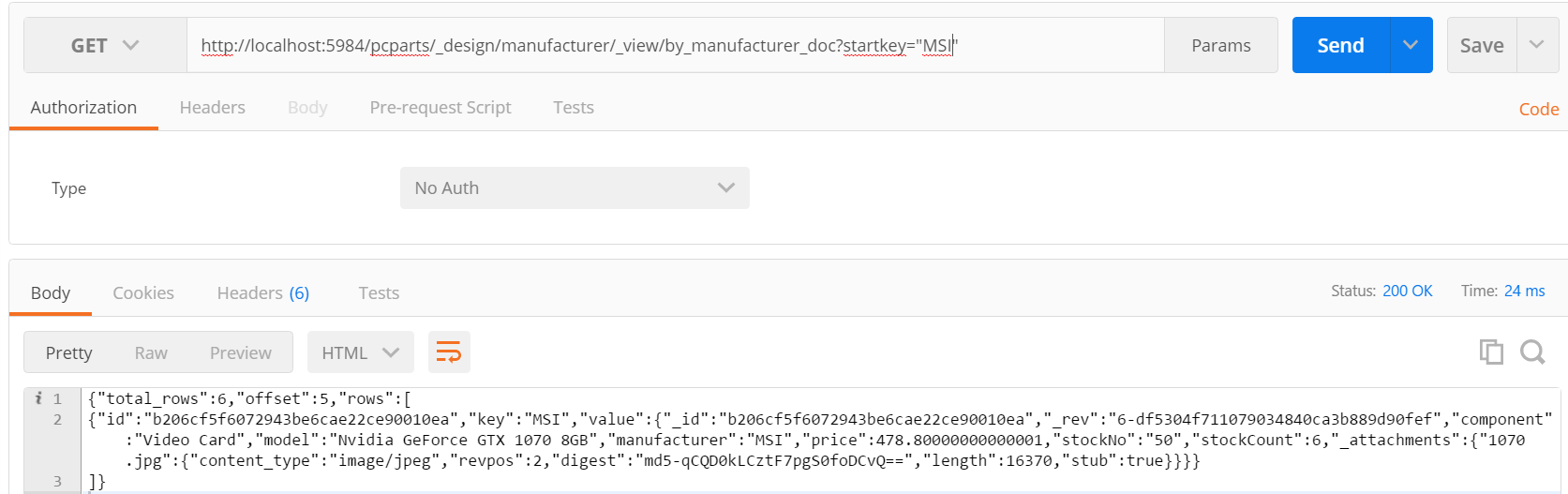
I created a more complex view with two if statements that searched for documents that said “clockSpeed” and if the number of cores was greater than 2.

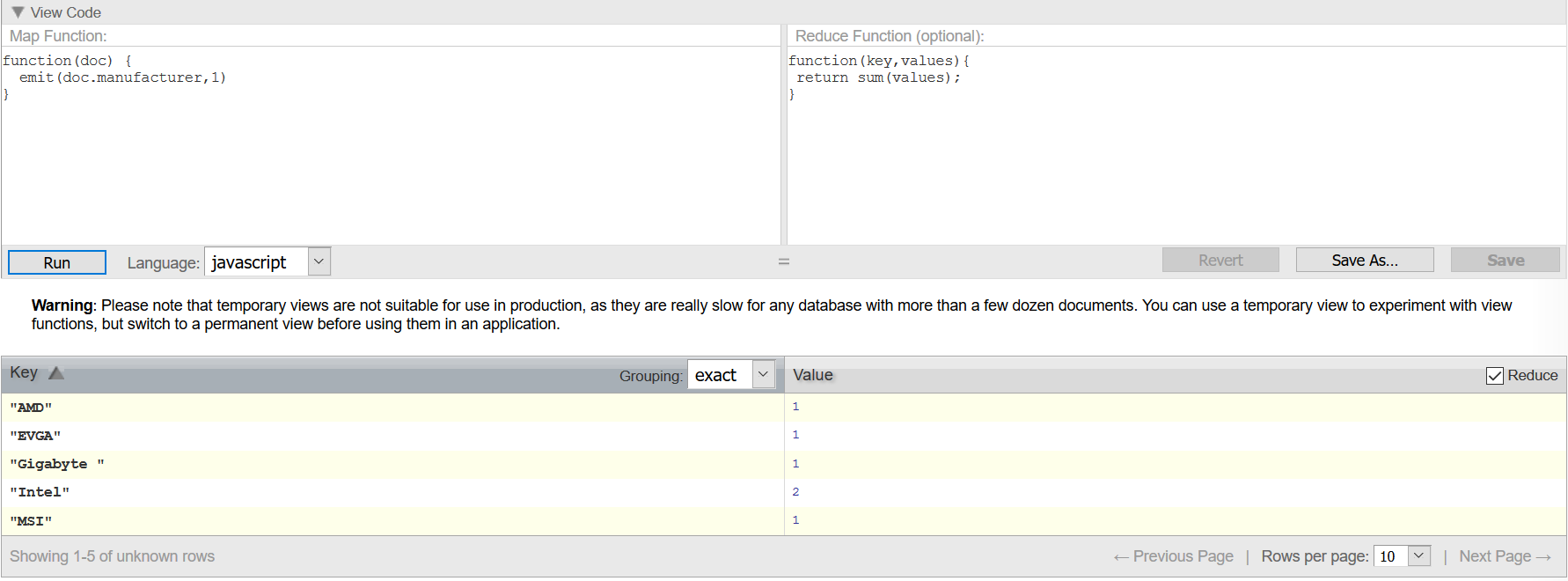


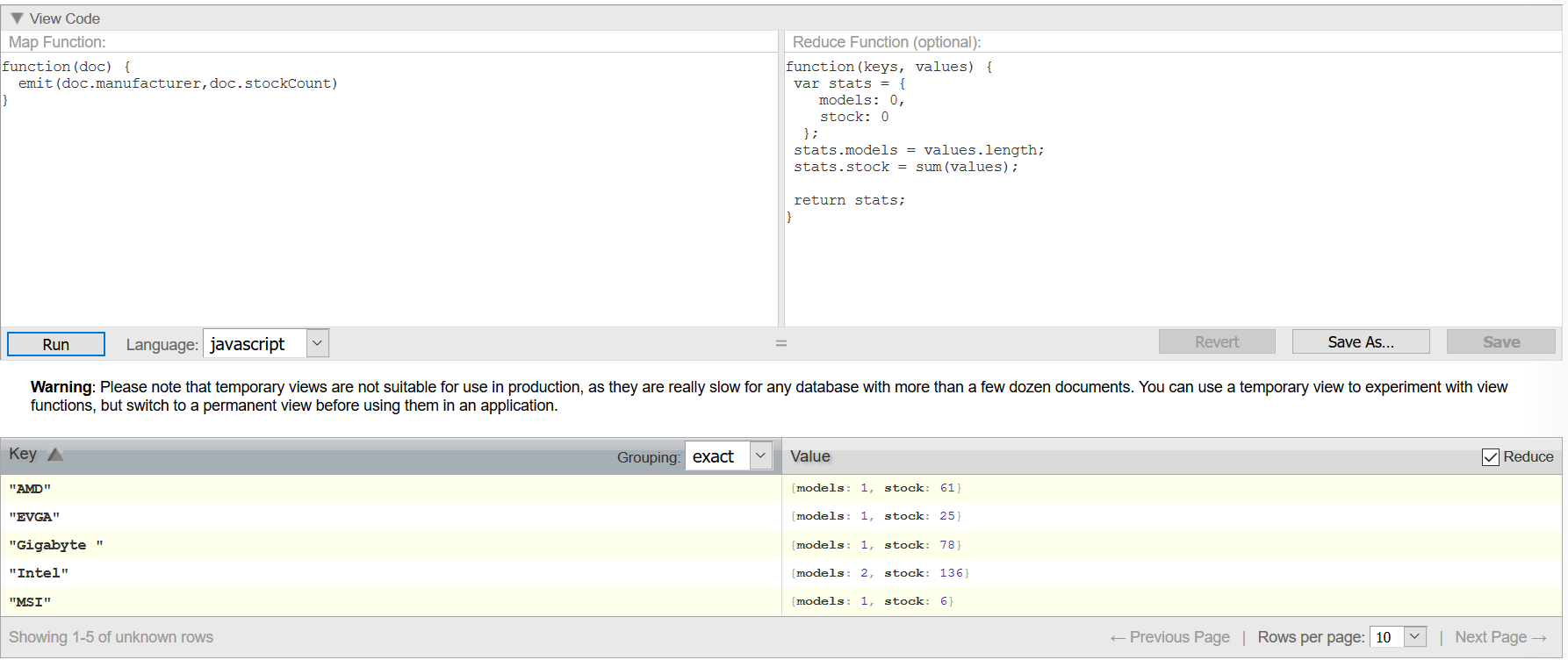
It’s possible to create much more complex views in CouchDB by using things such as variables which I used in the next example, that uses an if statement to check that manufacturer and model were in the database.



Using a REST client like Postman you can use an existing view and search for a keyword, like looking for all parts manufactured my MSI in the view by manufacture view.



In CouchDB you can create more advanced views with MapReduce’s that can aggregate and index documents. I created a basic MapReduce that summed up how many products were created by each manufacturer.

This uses a simple map function but includes the reduce value which sums the manufacturers. After this I created a more complex view to count how many models by each manufacturer and the amount of stock left.

## Conclusion:

I eventually created the database I had envisioned at the beginning and it was relatively simple to add, update, view and delete documents. I knew that be able to create map and reduce functions to create custom views was going to be of benefit later when I had to make a GUI for my database. Overall my experience with CouchDB has been positive and I would definitely consider using it for future projects

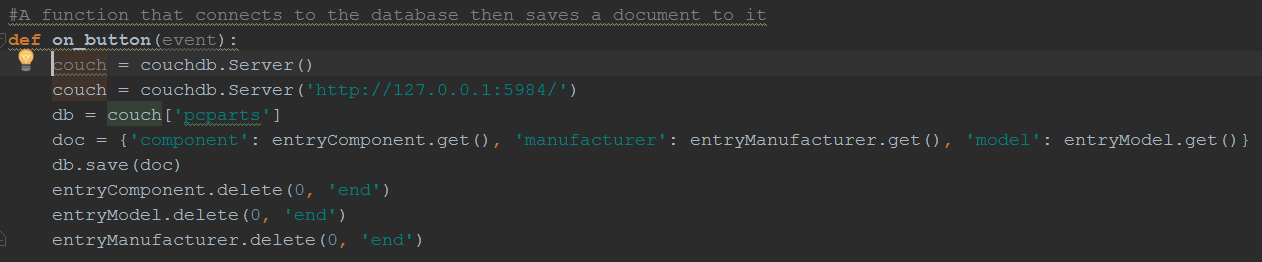
# Q2 – Implementing the Database into a GUI

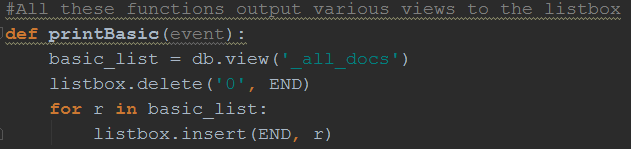
## Introduction:

I was tasked with creating a GUI that would interact with the database that I created in part one and so had to decide which Language to use. I first looked at Java and although there are a few library’s for CouchDB I found the support and information on how to use those libraries to be lacking, so then I looked at C# which had very few libraries to choose from. I found a lot of support and information on using JavaScript but I have no experience with it so it would take me too long to learn how to use it first. Eventually I looked at Python which had a single package that connected to CouchDB with very few commands and there was plenty of support and information on it, so I decided to create my GUI in Python.

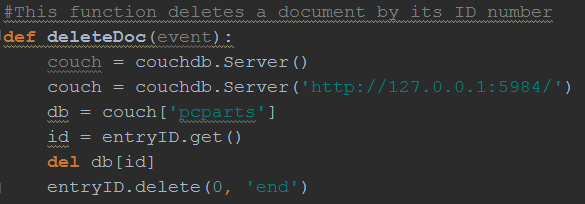
## Code Snippets Showing CRUD:

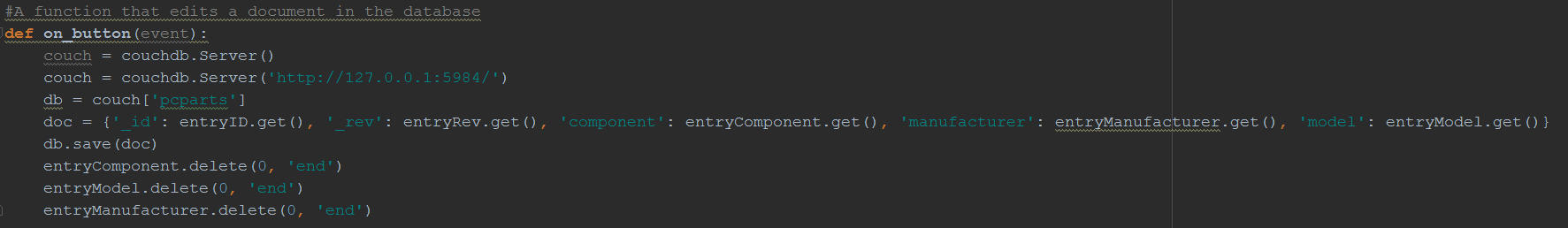
As mentioned in my introduction connecting to CouchDB is not very difficult with only three lines of code to connect. The below code creates a document by getting variables entered in text boxes and then saving them to the database, the code after that simply wipes the text boxes.



Very little code was needed to add a document to the database and even less is needed to view a document. All that was needed was to save a view to a list, make sure the list box was empty and then insert the list into the list box.

I wrote the code so that all that is needed is an ID number to delete a document. Once the code to connect to the database had been written all that was needed was to get an ID from the text field and then use the  command to delete the document by its ID.

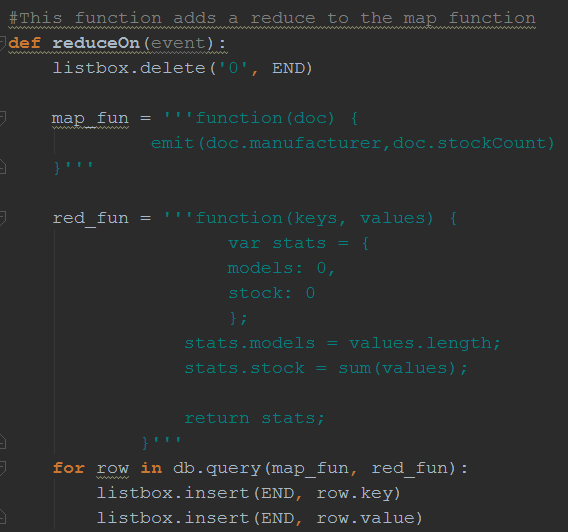


For the view I used essentially the same code as my create as once the id and revision number are entered CouchDB will update the document with the changed fields.

Those were the CRUD operations that I used in my code and as it can be seen, very little code is needed to do them.

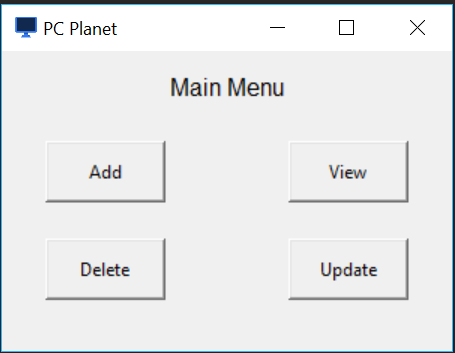
## Code Snippet Showing MapReduce:

When it came to doing a MapReduce from within the code more code was needed but I was able to take the JavaScript and put it directly into the code which made it very simple to execute the MapReduce. So to create it I first made a basic map function that emitted the manufacturer and a stock count, then I made a reduce function that displayed the amount of manufacturers and the total amount of stock. Then all I had to do was insert the information into a list box.

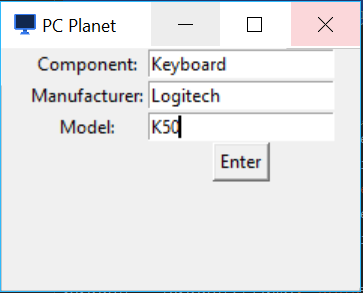


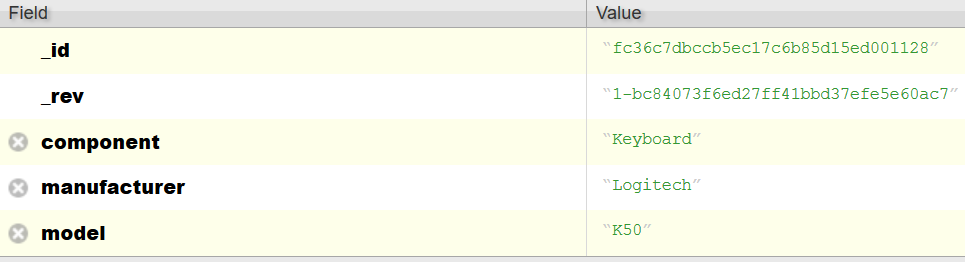
## User Interface:

To begin with I made a basic user interface.



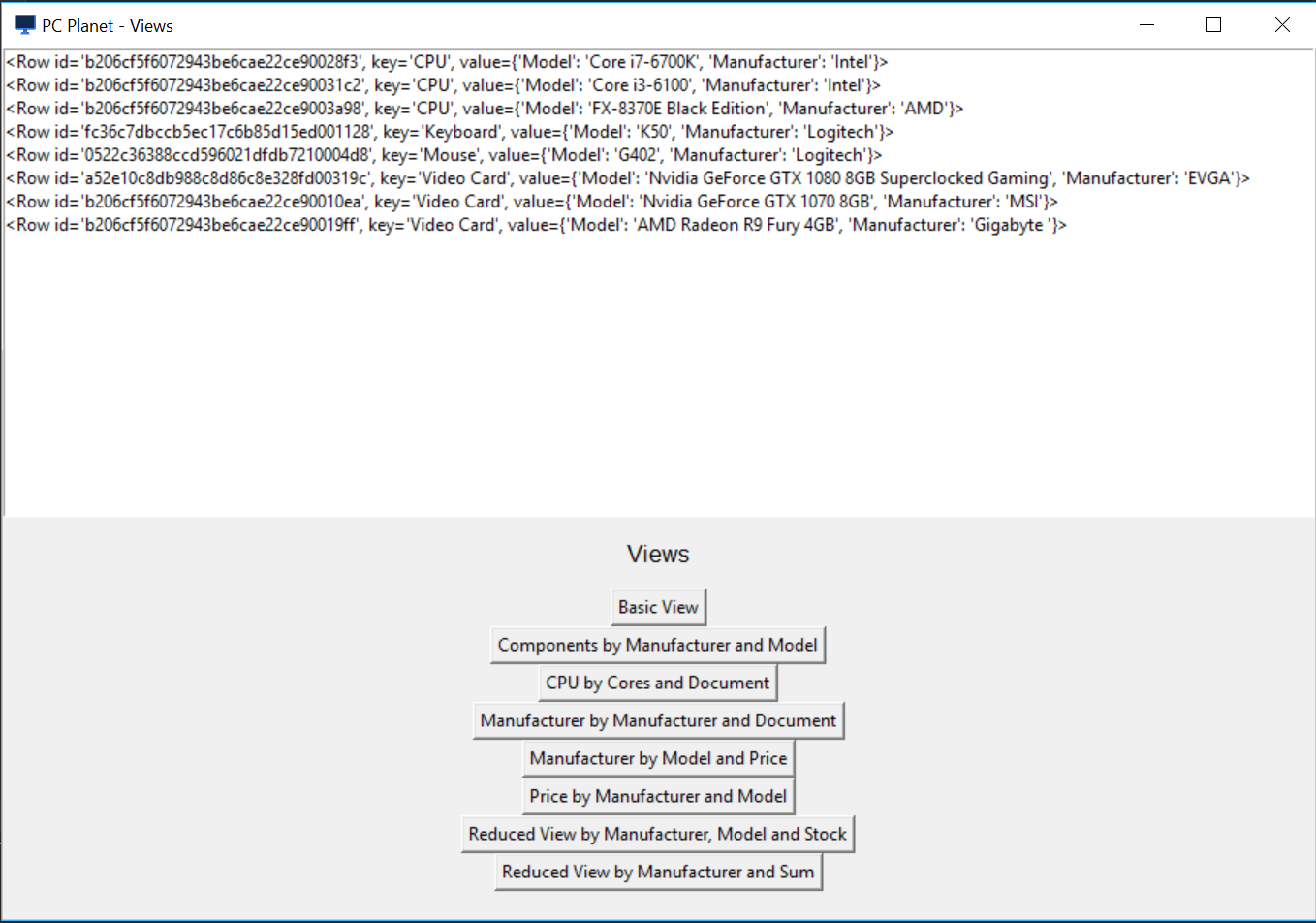
Once I had set up the main menu I made the window to add documents to the database. I made a basic layout that allows the user to enter three fields and once enter is pressed the code to save the new document is executed and the fields wipe.



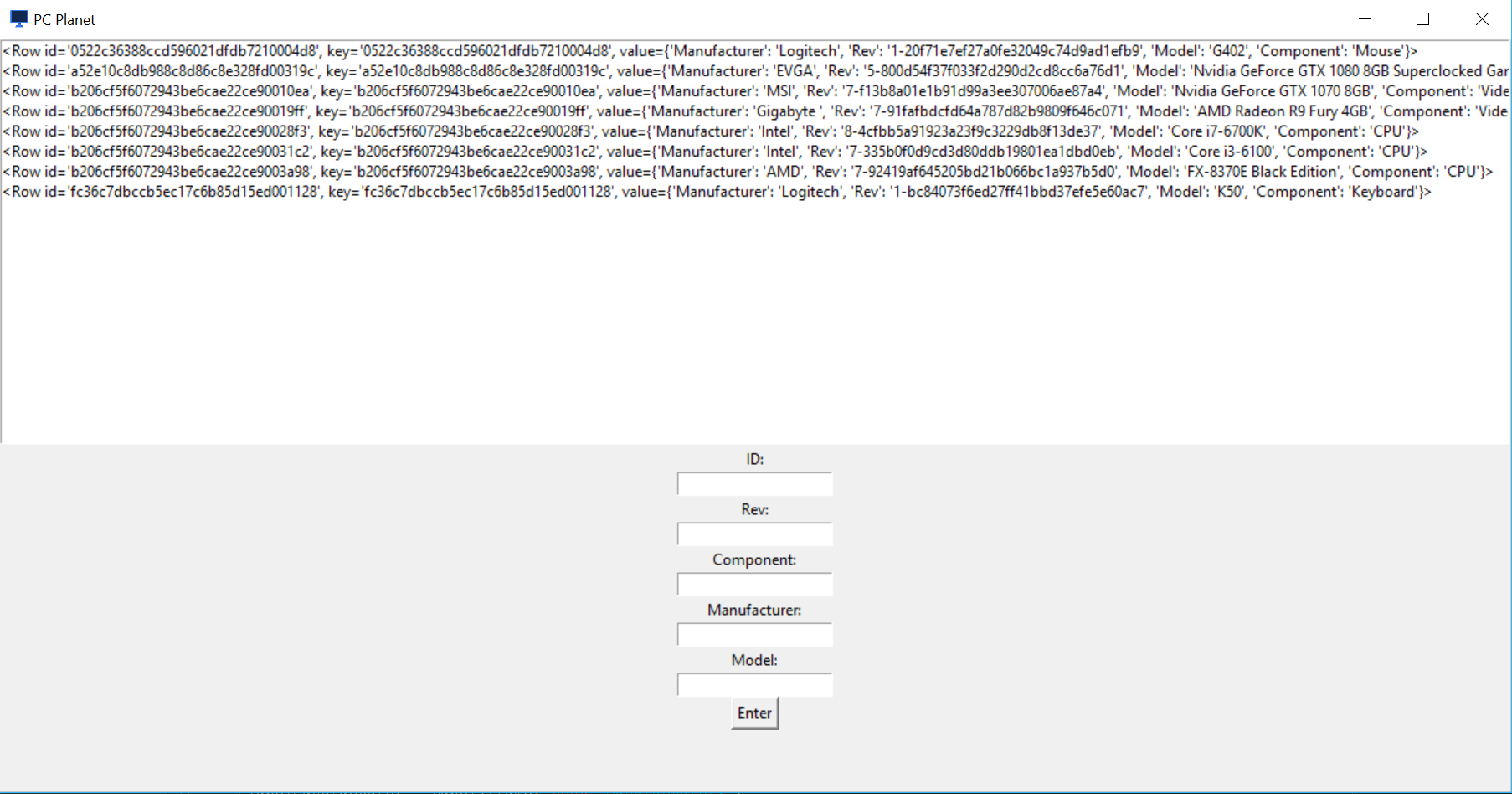


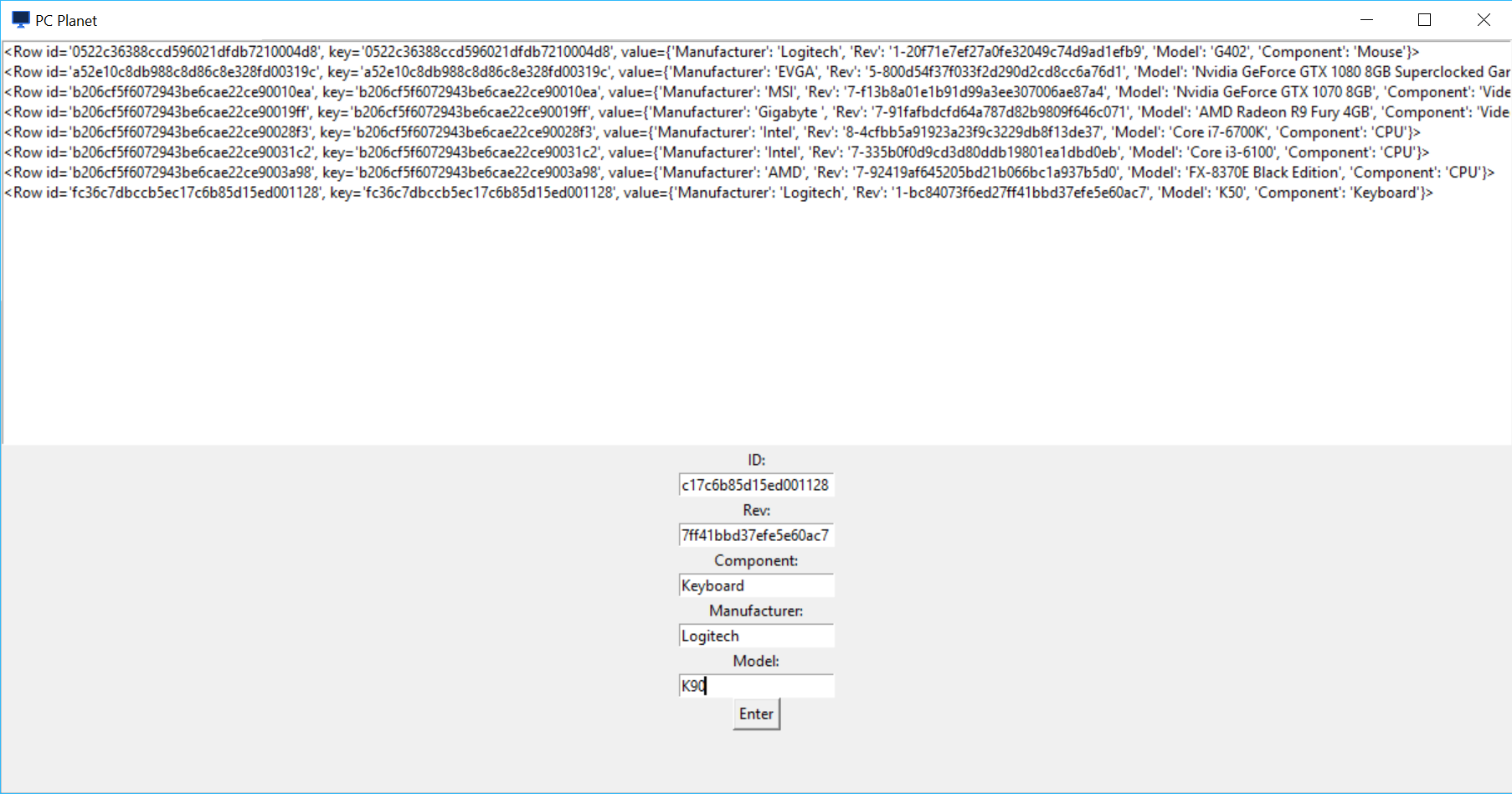
Then there was the view document for which I added the option to display several different views within a list box.

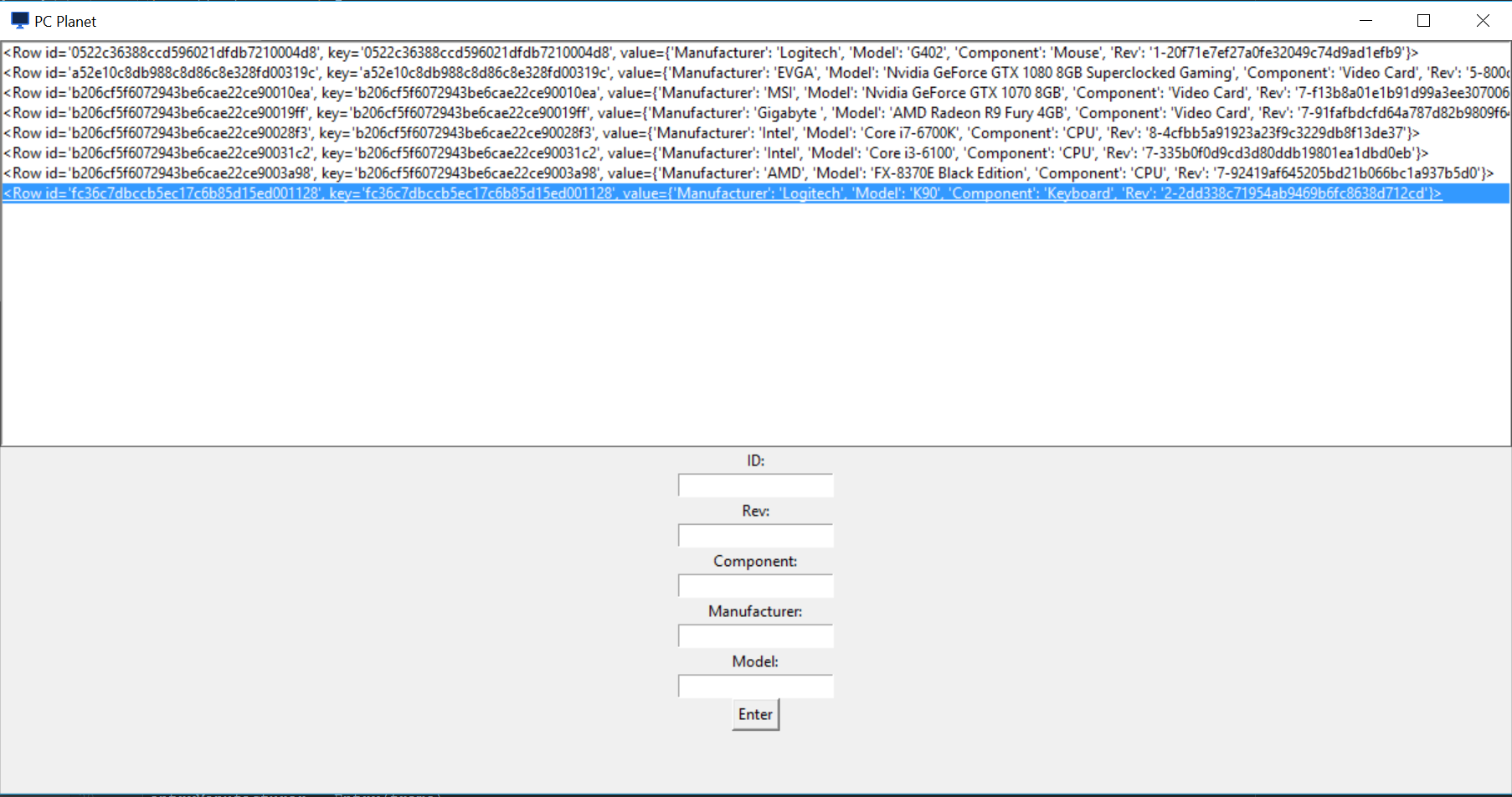
(Basic View) 

(Components by Manufacturer and Model View)

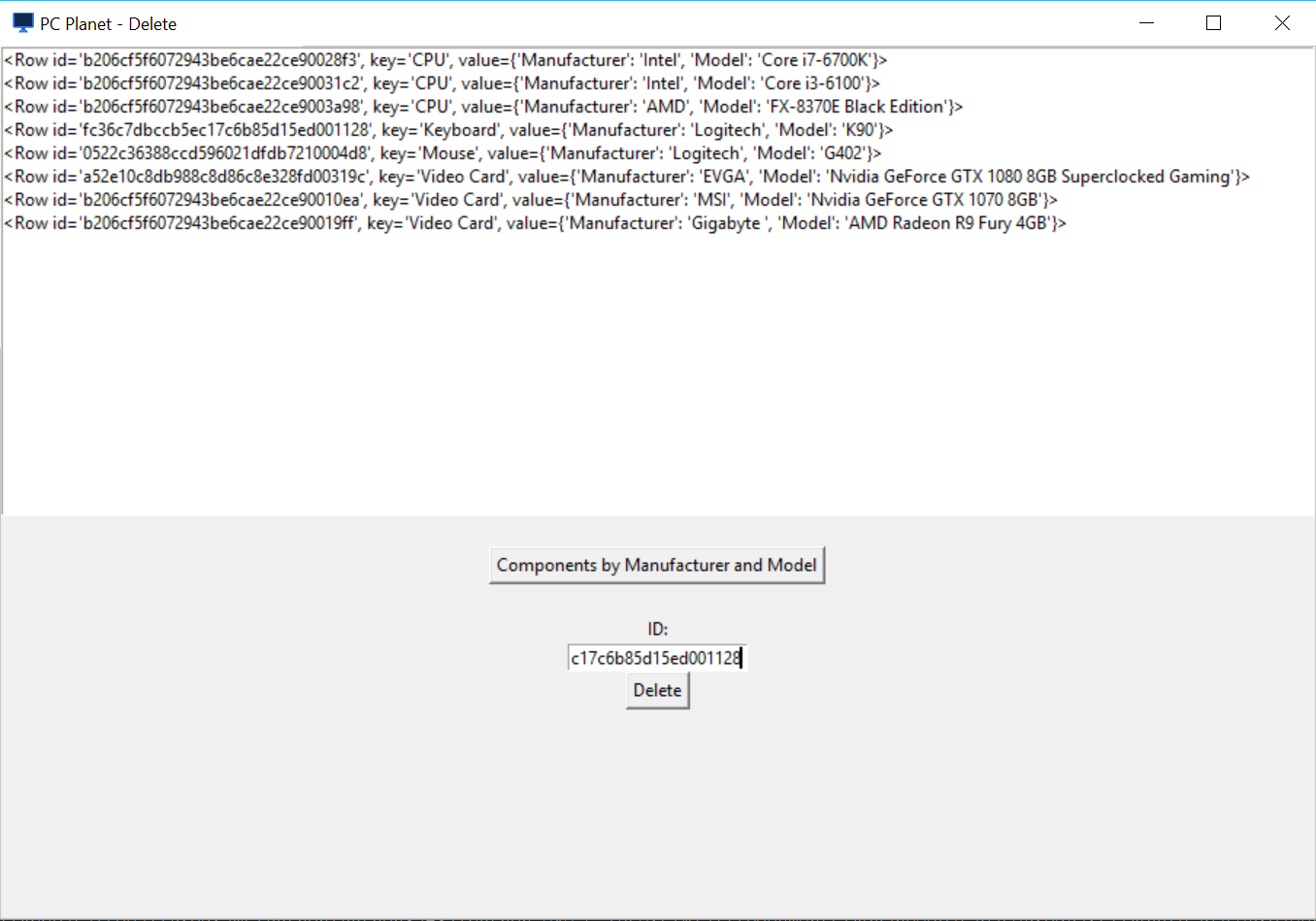
Once I created the view I could reuse the code to display current documents in my delete and update windows. For the update I displayed all current documents and added some fields that could be edited.

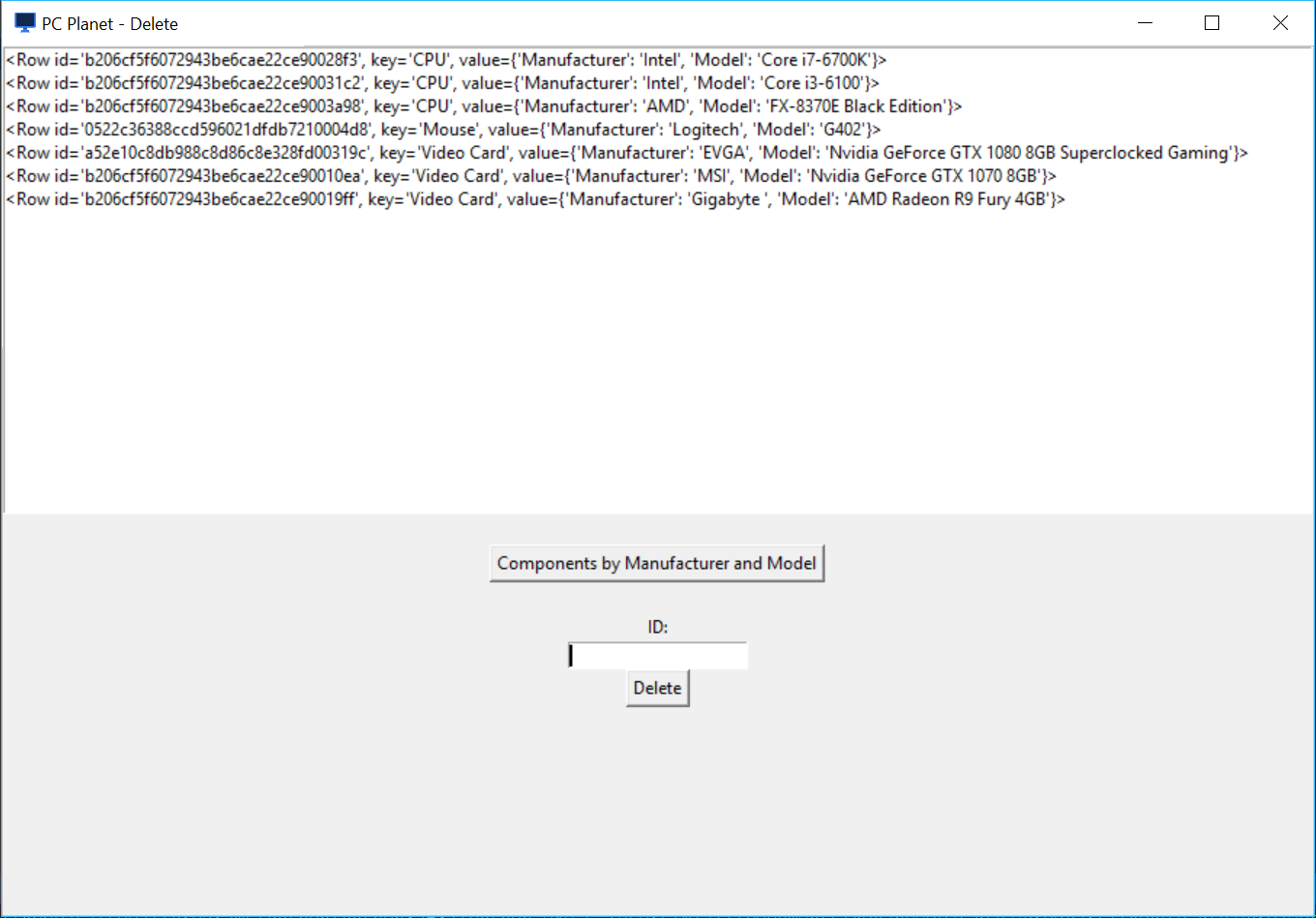


 (Fields Entered)

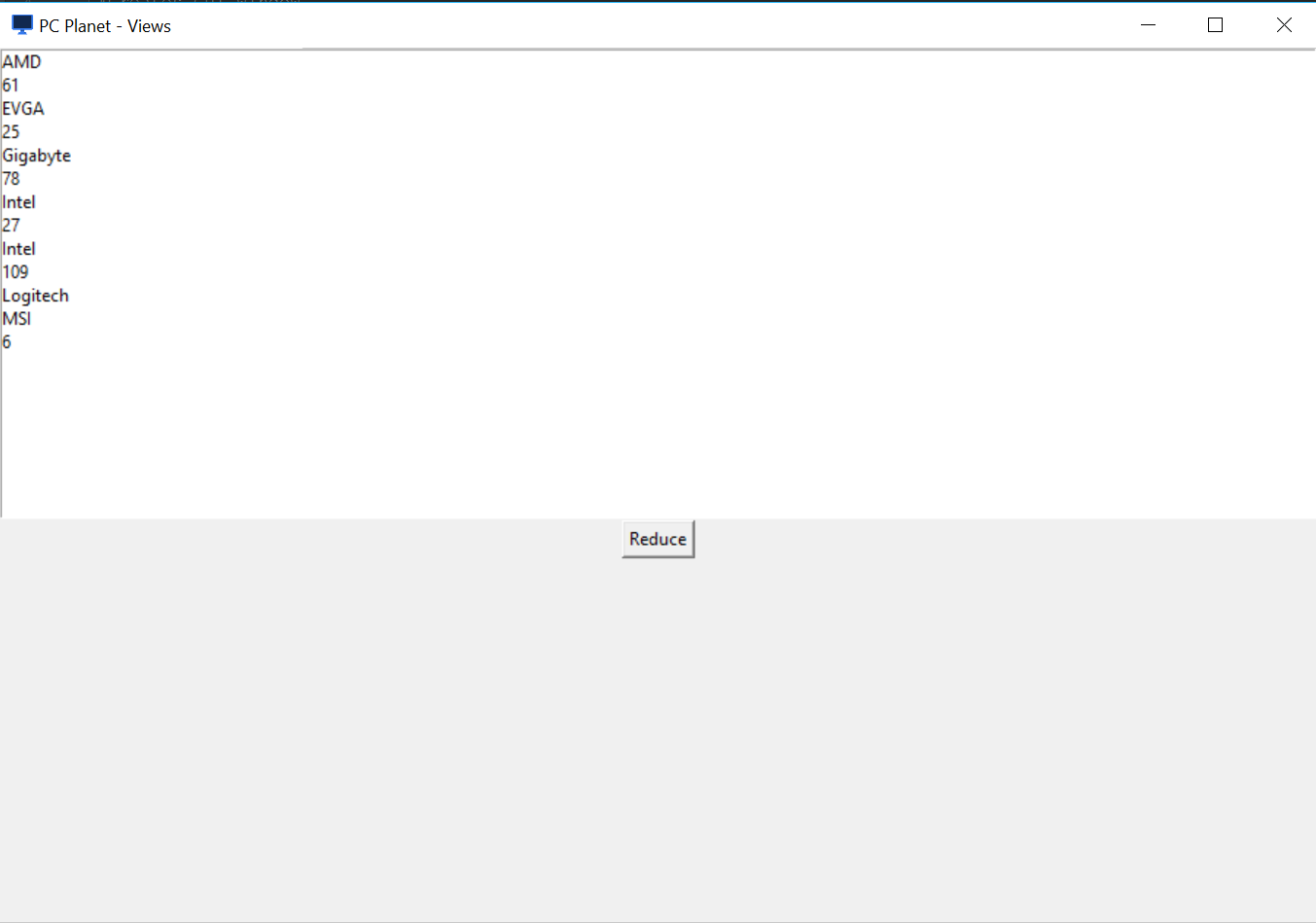
 (After Update)

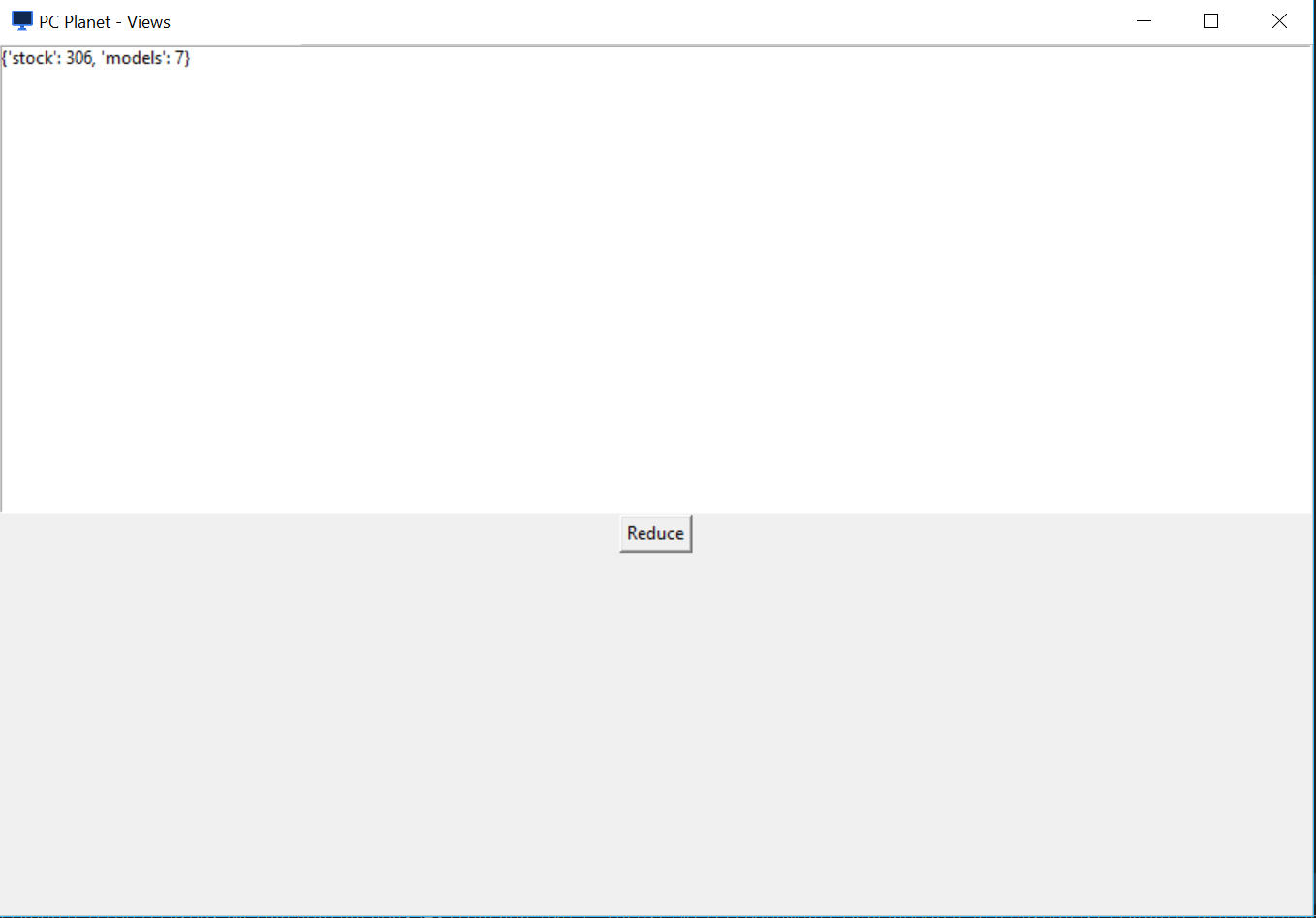
After the update window was complete I needed to do the delete window which again used the view list box I made earlier.



 (After Deletion)

Once all the CRUD windows had been completed I created a window to display a Map function but then when a button is clicked it add a MapReduce function to it and displays it in a list box.

 (Before MapReduce)

 (After Reduce)

## Advantages and Disadvantages of Using Python:

The ease of connecting to the database was the big advantage for me and I found plenty of help online for when I was unsure how to do things, especially GUI building in Python which I have never done before. Learning in Python is also quick enough as it is a relatively easy to read language.

The lack of GUI building tools was the only real disadvantage I found with using Python, I spent considerable time manually placing widgets and frames where as a UI designer like the ones in NetBeans and Visual Studio would have been a lot faster and would have worked a lot better. Although I got all my windows working separately, they didn’t seem to work when put together in a single program so I couldn’t create a simple executable file.

## Conclusion:

Working with CouchDB has been very different from using SQL databases which are all I’ve ever used and I found it much easier to work with as changing a document does not affect relationships with other documents and it’s also possible to have different fields for every document. I found it difficult to manage different fields in the GUI as I could only allow the user to enter certain ones as opposed to letting them add any fields they want, which would have been significantly more difficult to code. I could have let the user enter the exact JSON code they want for the document but this would not have been very user friendly, defeating the purpose of having a UI.