30/03/2018

[ATND-IBO2]

Project Report

NEU Robin – REBILLARD Maxime – LABARRE Julien

IBO2

Table des matières

[Introduction 2](#_Toc510184373)

[Dataset Description 3](#_Toc510184374)

[Import JSON file 4](#_Toc510184375)

[Organisation in MongoDB 5](#_Toc510184376)

[Web application 7](#_Toc510184377)

[User interface 7](#_Toc510184378)

[Admin interface 8](#_Toc510184379)

[Queries Integrated 9](#_Toc510184380)

[ Easy queries 9](#_Toc510184381)

[ Medium queries 12](#_Toc510184382)

[ Complex query 13](#_Toc510184383)

# Introduction

The web application was made using MongoDB and NodeJS. We wanted our application to be fairly easy to use and to install on a computer. NodeJS is great for this type of project since It has a great number of drivers and allows you to easily connect on different localhost ports.

We can also give the results in an organized way, thanks to HTML and CSS.

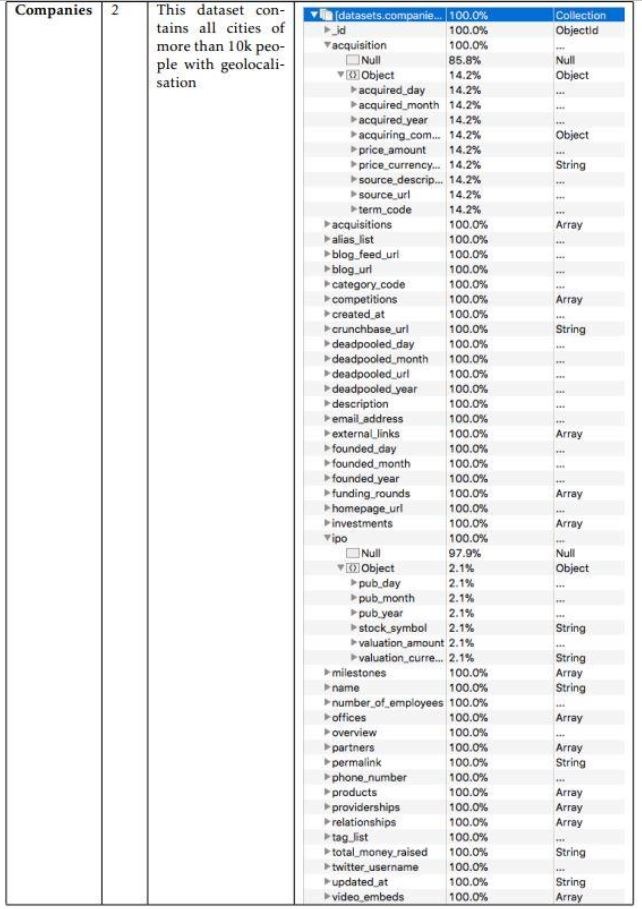
In order to make this app work, you can download it, go in the repository and then run:

npm install .

node mongoDBNode.js

From here, you can just go on your favorite web browser to enter the address: “localhost/port” where “port” is given in the console after typing the 2nd command

# Dataset Description



# Import JSON file

The method is much simpler since a lot of typos and non-valid JSON are taken care of by the mongoimport command.

In the system shell or command prompt, use mongoimport to insert the documents into the companies collection in the test database. If the collection already exists in the test database, the operation will **drop** the companies collection first.

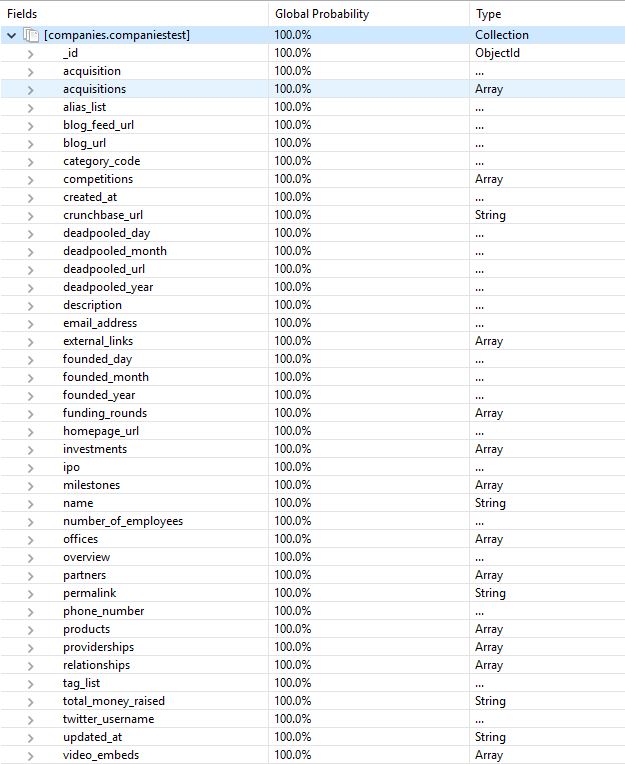
mongoimport --db test --collection companies --drop --file ~/downloads/companies2.json

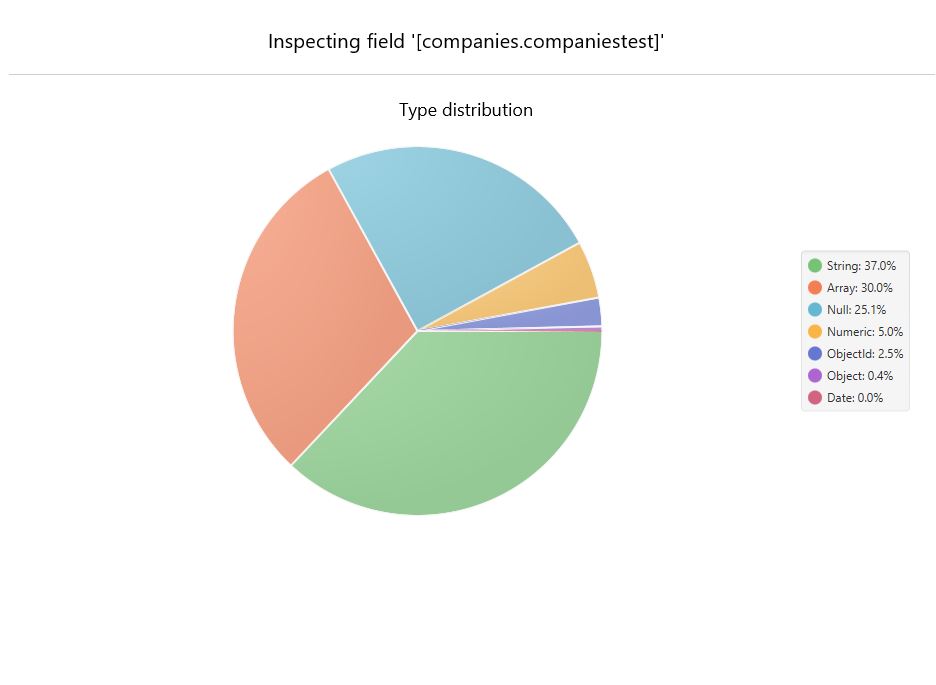
The mongoimport connects to a mongod instance running on localhost on port number 27017. The --file option provides the path to the data to import, in this case, ~/downloads/primer-dataset.json.

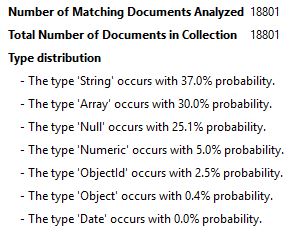
To import data into a mongod instance running on a different host or port, specify the hostname or port by including the --host and the --port options in your mongoimport command.

# Organisation in MongoDB

As you can see just below, there are a lot of fields. By expending all of them, we find that there are even more fields. And that lead us to a question: What is the data type proportion? A more precise result will lead us doing better requests.







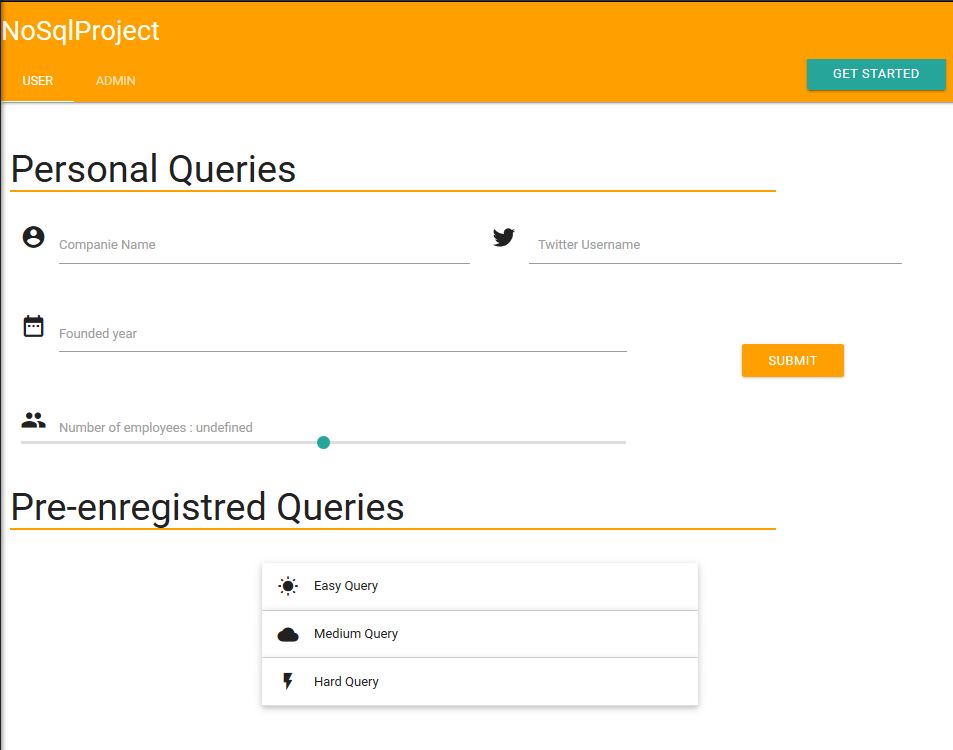
In our dataset we have 18801 documents. The advantage is that we do not need to adjust our JSON document since all those JSON object are saved as documents. So, every JSON Line is well saved without the need to adapt the document beforehand.

Just by looking the distribution of data types, we can see that 25% are “Null”, it explains well that we came across some difficulties for our Cassandra report. The most common data type is the “String” one. Now on to the queries to get some interesting results.

# Web application

## User interface

First, you’ll need to launch mongodb on you computer and import the dataset in mongodb, just be careful to name your db and collections the same way in mongo and in the program: otherwise, your queries won’t work. In our program, the name of the database and collections were “Compagnies” and “companies2” respectively.



Here you have an overview on how our web application will look like.

You have the 5 registered queries that will be displayed once you click on the buttons.

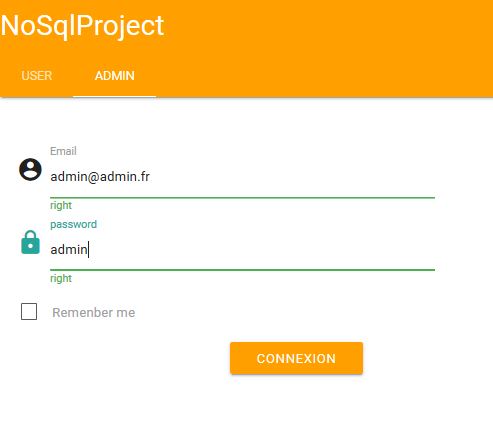
We also have a user interface and a login protected admin page.

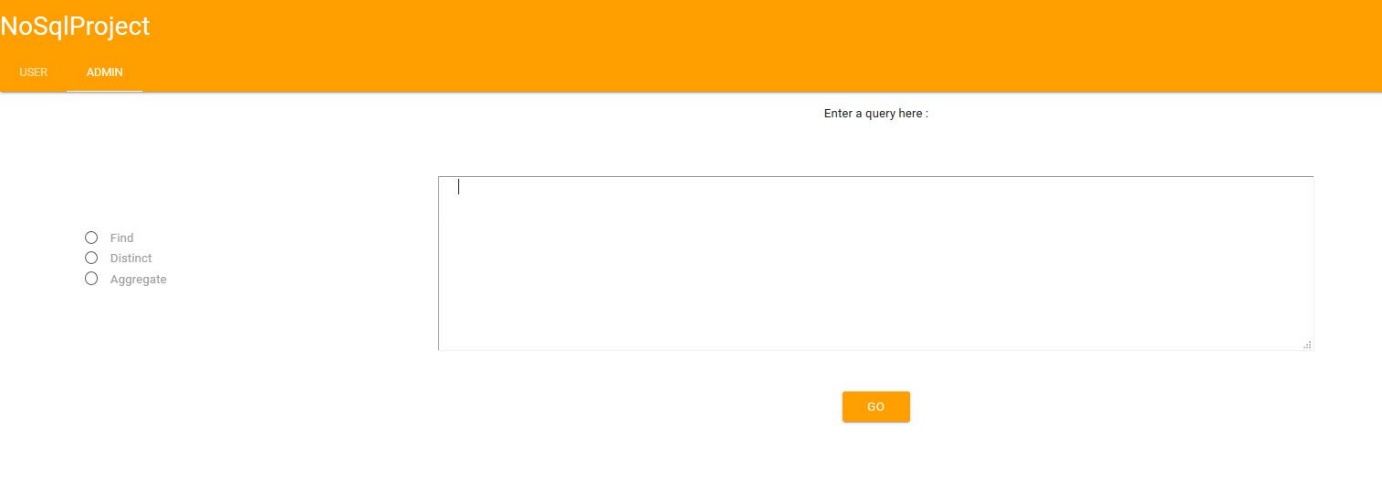
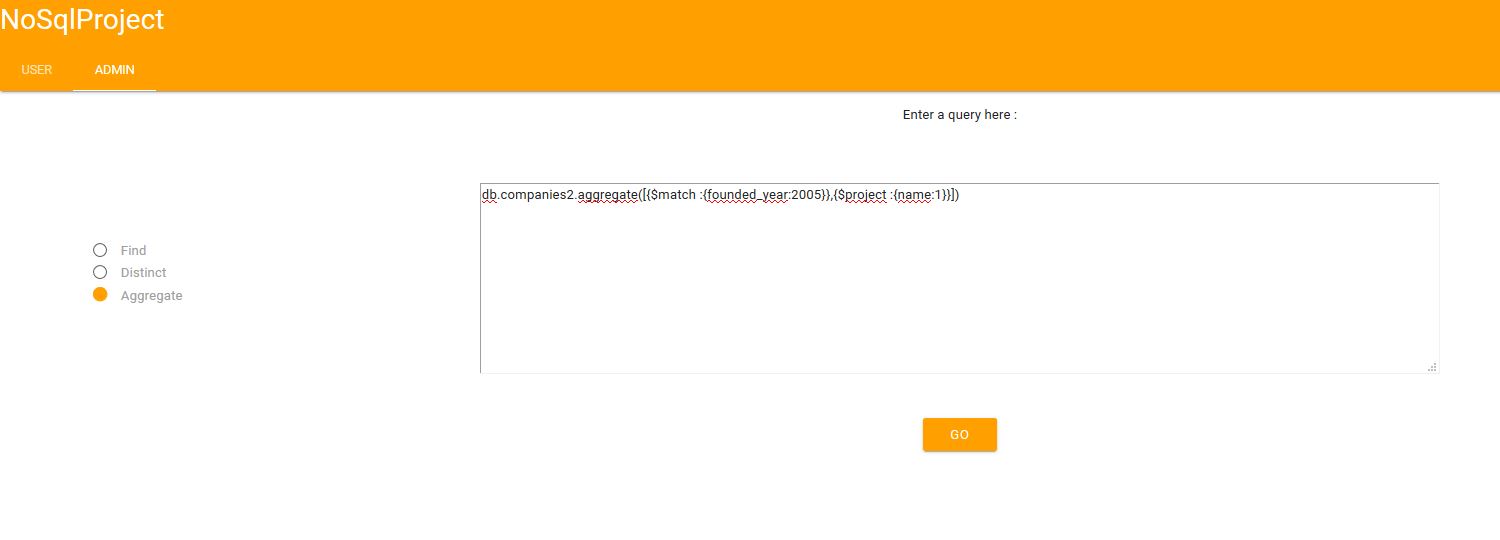
The user page allows you to execute pre-enregistred queries, and the personal queries allows you to change the parameters of the request. You can search for any existing companies or a company that has your search criteria.

To have help, click on the get started button.

Password and email are given in the next screeshot. You can’t create a new admin.

## Admin interface

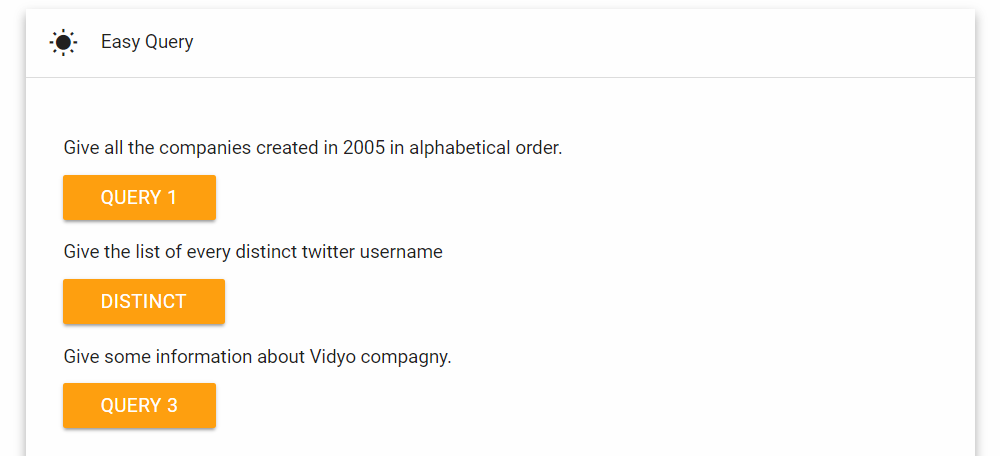
The admin interface is password protected:

The admin interface allows you to enter whenever query you like, with some examples:

PS : Aggregate queries not working well.. due to a problem of parsing the queries you write.

# Queries Integrated

## Easy queries



*Give all the companies created in 2005 in alphabetical order.*

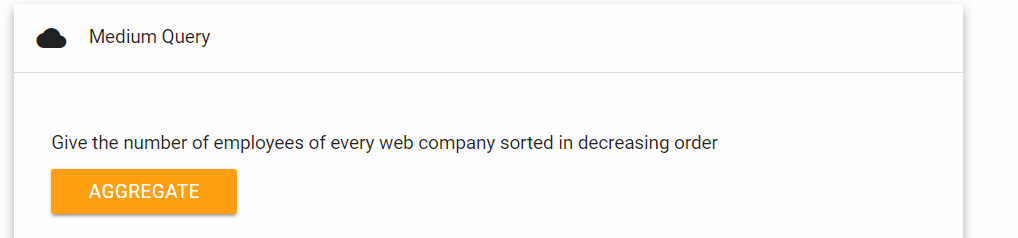
db.companies2.find({"founded\_year":2005}).sort({"name":1});

*Give the list of every distinct twitter username*

db.companies2.distinct("twitter\_username");

*Find informations on the Vidyo company*

db.companies2.find({"name": “vidyo”});



db.aggregate([

{$match: {"category\_code" : "web"}},

{$group: {

\_id: "$name",

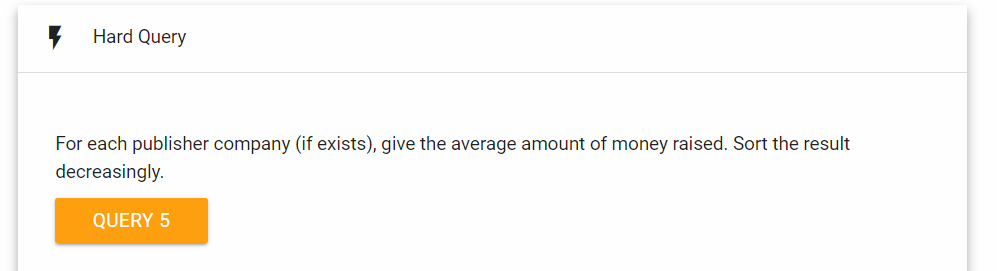
nbemployees: { $sum: "$number\_of\_employees" }

}

},

{$sort: {nbemployees: -1}}

]);



db.aggregate([

{$match: {"name": {$ne:null}}},

{$group: {"\_id":

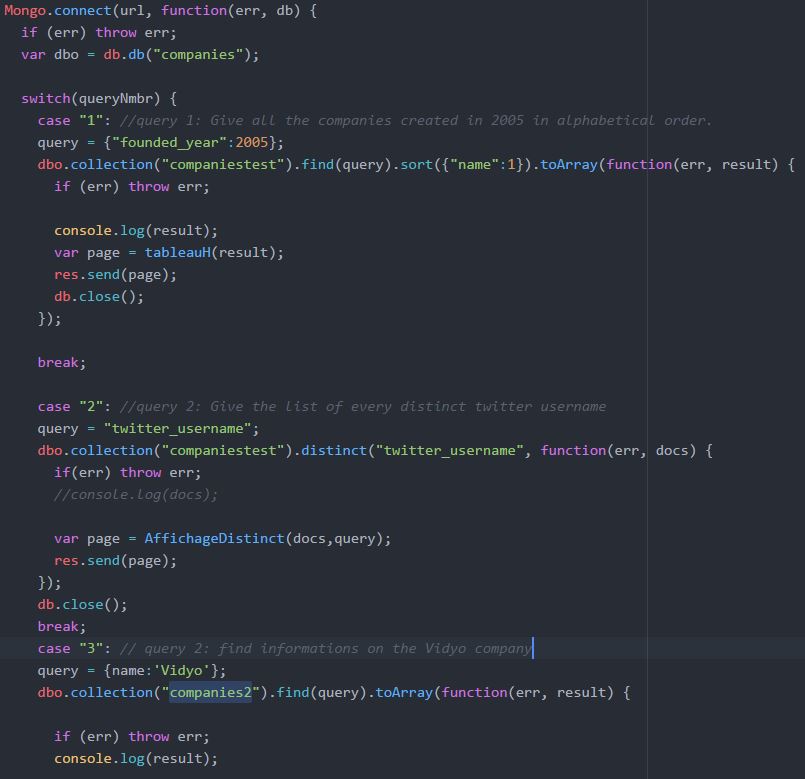
{"name": "$name",total\_money\_raised":"$total\_money\_raised"},

"count": {$sum:1}}},

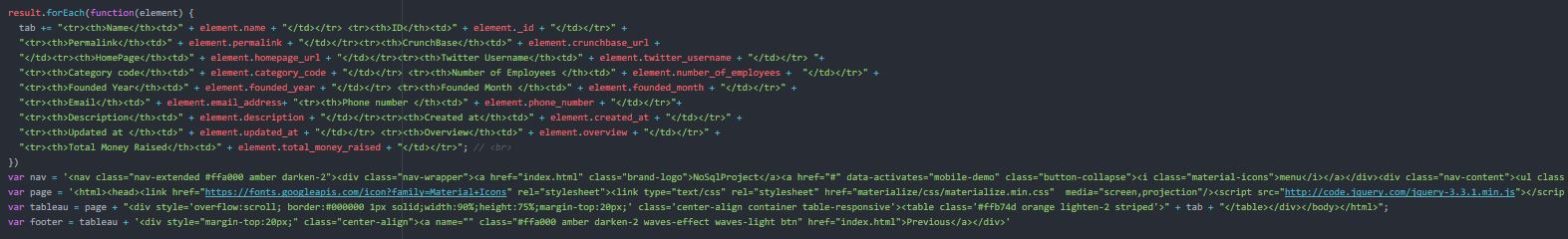
{$group: {"\_id":"$\_id", "avg":{$avg: "count"}}},

{$sort: {"avg":-1}}

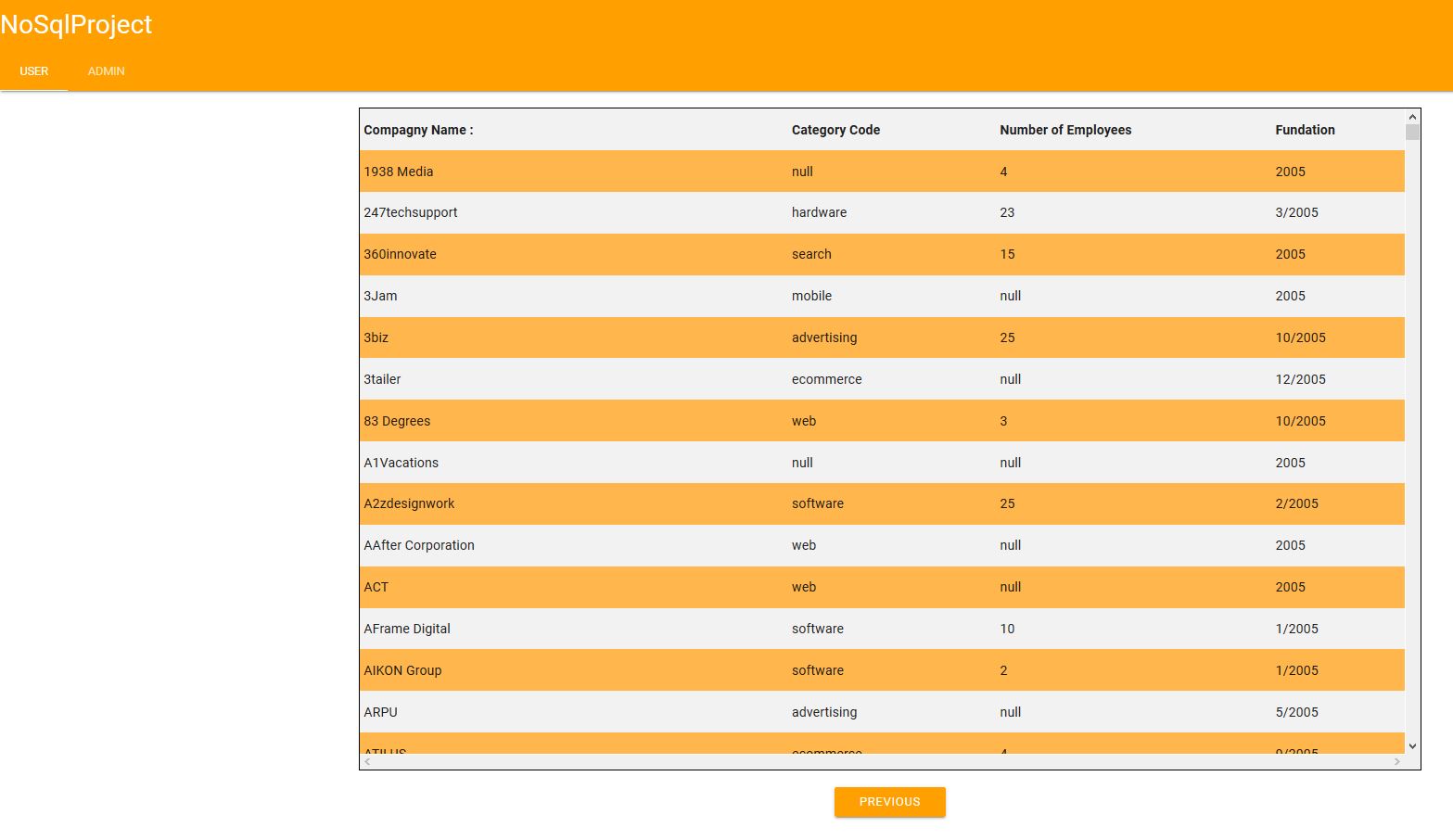
]);



On the image just above, you can see how the different queries are processed in mongodb from the user interface.

And this is how they are displayed:

The web app result page should look like this, when you execute a query:



## Complex query

*For each publisher company (if exists), give the average amount of money raised. Sort the result decreasingly.*

db.companiestest.aggregate(

[

{$match: {"name": {$ne:null}}},

{$group: {"\_id":{"name":"$name","total\_money\_raised":"$total\_money\_raised"}, "count": {$sum:1}}},

{$group: {"\_id":"$\_id", "avg":{$avg: "count"}}},

{$sort: {"avg":-1}}

]

);