.

Monday, June 17, 2013

MongoDB course note 5

Simple Aggregation Example

Write the aggregation query that will find the number of products by category of a collection that has the form:

```
{
"_id" : ObjectId("50b1aa983b3d0043b51b2c52"),
"name" : "Nexus 7",
"category" : "Tablets",
"manufacturer" : "Google",
"price" : 199
}
```

Have the resulting key be called "num_products," as in the video lesson. Hint, you just need to change which key you are aggregating on relative to the examples shown in the lesson.
Please double quote all keys to make it easier to check your result.

```
db.products.aggregate([
 {$group:
   {_id: "$category",
    num_products: {$sum:1} }
  }
])
```

The Aggregation Pipeline

Which of the following are stages in the aggregation pipeline. Check all that apply.

☑ Match
□ Transpose
☑ Group
☑ Skip
☑ Limit
☑ Sort
☑ Project
☑ unwind

Simple Example Expanded

If you have the following collection of stuff:

```
> db.stuff.find()
{ "_id" : ObjectId("50b26f9d80a78af03b5163c8"), "a" : 1, "b" : 1, "c" : 1 }
{ "_id" : ObjectId("50b26fb480a78af03b5163c9"), "a" : 2, "b" : 2, "c" : 1 }
{ "_id" : ObjectId("50b26fbf80a78af03b5163ca"), "a" : 3, "b" : 3, "c" : 1 }
{ "_id" : ObjectId("50b26fcd80a78af03b5163cb"), "a" : 3, "b" : 3, "c" : 2 }
{ "_id" : ObjectId("50b26fd380a78af03b5163cc"), "a" : 3, "b" : 5, "c" : 3 }
```

and you perform the following aggregation:

Search This Blog

Search

Labels

- Java (48)
- Web Service (14)
- Ruby on Rails (13)
- Hadoop (11)
- SCM (10)
- MVNE / MVNO / Telecom (9)
- Mongodb (9)
- Git (5)
- Rest (5)
- Unit test (5)
- clean code (5)
- Software architecture (4)
- Ruby (3)
- Heroku (2)
- Lunix (2)
- Requirement engineering (2)
- C# (1)
- CSS (1)
- Python (1)

About Me

.

Copenhagen, Denmark

I'm a software developer, clean code practitioner and requirement analyst. I'm working on improving software quality from software engineering perspectives (Requirement engineering, software architecture, design, SCM, clean code, and testing). Currently, I work as a Telecom technical consultant in CPH.

View my complete profile

```
db.stuff.aggregate([{$group:{_id:'$c'}}])
```

How many documents will be in the result set from aggregate?
3

Compound Grouping

Given the following collection:

```
> db.stuff.find()
{ "_id" : ObjectId("50b26f9d80a78af03b5163c8"), "a" : 1, "b" : 1, "c" : 1 }
{ "_id" : ObjectId("50b26fb480a78af03b5163c9"), "a" : 2, "b" : 2, "c" : 1 }
{ "_id" : ObjectId("50b26fbf80a78af03b5163ca"), "a" : 3, "b" : 3, "c" : 1 }
{ "_id" : ObjectId("50b26fcd80a78af03b5163cb"), "a" : 3, "b" : 3, "c" : 2 }
{ "_id" : ObjectId("50b26fd380a78af03b5163cc"), "a" : 3, "b" : 5, "c" : 3 }
{ "_id" : ObjectId("50b27f7080a78af03b5163cd"), "a" : 3, "b" : 3, "c" : 2 }
```

And the following aggregation query:

```
db.stuff.aggregate([{$group:
     {_id:
      {'moe':'$a',
       'larry':'$b',
       'curly':'$c'
      }
     }
    }])
```

How many documents will be in the result set?
5

Aggregation Expressions

Which of the following aggregation expressions must be used in conjunction with a sort to make any sense?

☐ $addToSet
☑ $first
☑ $last
☐ $max
☐ $min
☐ $avg
☐ $push
☐ $sum

Using $sum

This problem, and some after it, use the zips collection frommedia.mongodb.org/zips.json. You don't need to download it, but you can if you want, allowing you to test your queries within MongoDB. You can import, once downloaded, using mongoimport
Suppose we have a collection of populations by postal code. The postal codes in are in the _id field, and are therefore unique. Documents look like this:

```
{
"city" : "CLANTON",
"loc" : [
 -86.642472,
 32.835532
],
"pop" : 13990,
"state" : "AL",
"_id" : "35045"
}
```

For students outside the United States, there are 50 non-overlapping states in the US with two letter abbreviations such as NY and CA. In addition, the capital of Washington is within an area designated the District of Columbia, and carries the abbreviation DC. For purposes of the mail, the postal service considers DC to be a "state." So in this dataset, there are 51 states. We call postal codes "zip codes." A city may overlap several zip codes.

Write an aggregation query to sum up the population (pop) by state and put the result in a field called population. Don't use a compound _id key (you don't need one and the quiz checker is not expecting one). The collection name is zips. so something along the lines of db.zips.aggregrate...

```
db.zips.aggregate([
{$group: {_id: "$state", population: {$sum: "$pop"}}}
])
```

Using $avg

This problem uses the same dataset as we described in using $sum quiz and you should review that quiz if you did not complete it.

Given population data by zip code (postal code) that looks like this:

```
{
"city" : "FISHERS ISLAND",
"loc" : [
 -72.017834,
 41.263934
],
"pop" : 329,
"state" : "NY",
"_id" : "06390"
}
```

Write an aggregation expression to calculate the average population of a zip code (postal code) by state. As before, the postal code is in the _id field and is unique. The collection is assumed to be called "zips" and you should name the key in the result set "average_pop".

```
db.zips.aggregate([
{$group: {_id:"$state", average_pop:{$avg: "$pop"} } }
])
```

Using $addToSet

This problem uses the same zip code data as the $using sum quiz. See that quiz for a longer explanation.

Suppose we population by zip code (postal code) data that looks like this (putting in a query for the zip codes in Palo Alto)

```
> db.zips.find({state:"CA",city:"PALO ALTO"})
{ "city" : "PALO ALTO", "loc" : [ -122.149685, 37.444324 ], "pop" : 15965, "state" : "CA", "_id" : "94301" }
{ "city" : "PALO ALTO", "loc" : [ -122.184234, 37.433424 ], "pop" : 1835, "state" : "CA", "_id" : "94304" }
{ "city" : "PALO ALTO", "loc" : [ -122.127375, 37.418009 ], "pop" : 24309, "state" : "CA", "_id" : "94306" }
```

Write an aggregation query that will return the postal codes that cover each city. The results should look like this:

```
 {
 "_id" : "CENTREVILLE",
 "postal_codes" : [
  "22020",
  "49032",
  "39631",
  "21617",
  "35042"
 ]
```

```
  },
```

Again the collection will be called zips. You can deduce what your result column names should be from the above output. (ignore the issue that a city may have the same name in two different states and is in fact two different cities in that case - for eg Springfield, MO and Springfield, MA)

```
db.zips.aggregate([
{$group: {_id: "$city", postal_codes:{$addToSet:"$_id"}} }
])
```

Using $push

Given the zipcode dataset (explained more fully in the using $sum quiz) that has documents that look like this:

```
> db.zips.findOne()
{
 "city" : "ACMAR",
 "loc" : [
  -86.51557,
  33.584132
 ],
 "pop" : 6055,
 "state" : "AL",
 "_id" : "35004"
}
```

would you expect the following two queries to produce the same result or different results?

```
db.zips.aggregate([{"$group":{"_id":"$city", "postal_codes":{"$push":"$_id"}}}])
```

```
db.zips.aggregate([{"$group":{"_id":"$city", "postal_codes":{"$addToSet":"$_id"}}}])
```

● Same result

○ Different Result

Using $max and $min

Again thinking about the zip code database, write an aggregation query that will return the population of the postal code in each state with the highest population. It should return output that looks like this:

```
{
   "_id" : "WI",
   "pop" : 57187
  },
  {
   "_id" : "WV",
   "pop" : 70185
  },
..and so on
```

Once again, the collection is named zips.

```
db.zips.aggregate([
{$group:{_id:"$state", pop: {$max:"$pop"}} }
])
```

Double $group stages

Given the following collection:

```
> db.fun.find()
{ "_id" : 0, "a" : 0, "b" : 0, "c" : 21 }
{ "_id" : 1, "a" : 0, "b" : 0, "c" : 54 }
{ "_id" : 2, "a" : 0, "b" : 1, "c" : 52 }
{ "_id" : 3, "a" : 0, "b" : 1, "c" : 17 }
{ "_id" : 4, "a" : 1, "b" : 0, "c" : 22 }
{ "_id" : 5, "a" : 1, "b" : 0, "c" : 5 }
{ "_id" : 6, "a" : 1, "b" : 1, "c" : 87 }
{ "_id" : 7, "a" : 1, "b" : 1, "c" : 97 }
```

And the following aggregation query

```
db.fun.aggregate([{$group:{_id:{a:"$a", b:"$b"}, c:{$max:"$c"}}}, {$group:{_id:"$_id.a", c:{$min:"$c"}}}])
```

What values are returned?

- ○ 17 and 54
- ○ 97 and 21
- ○ 54 and 5
- ● 52 and 22

$project

Write an aggregation query with a single projection stage that will transform the documents in the zips collection from this:

```
{
 "city" : "ACMAR",
 "loc" : [
  -86.51557,
  33.584132
 ],
 "pop" : 6055,
 "state" : "AL",
 "_id" : "35004"
}
```

to documents in the result set that look like this:

```
{
 "city" : "acmar",
 "pop" : 6055,
 "state" : "AL",
 "zip" : "35004"
}
```

So that the checker works properly, please specify what you want to do with the _id key as the first item. The other items should be ordered as above. As before, assume the collection is called zips. You are running only the projection part of the pipeline for this quiz.

A few facts not mentioned in the lesson that you will need to know to get this right: If you don't mention a key, it is not included, except for _id, which must be explicitly suppressed. If you want to include a key exactly as it is named in the source document, you just write key:1, where key is the name of the key. You will probably get more out of this quiz is you download the zips.json file and practice in the shell. zips.json link is in the using $sum quiz

```
db.zips.aggregate([
{$project: {_id:0,
'city':{$toLower:"$city"},'pop':1, 'state':1, 'zip':'$_id'}}
])
```

$match

Again, thinking about the zipcode collection, write an aggregation query with a single match phase that filters for zipcodes with greater than 100,000 people. You may need

to look up the use of the $gt operator in the MongoDB docs.
Assume the collection is called zips.

```
db.zips.aggregate([
{$match: {pop:{$gt:100000}}}
])
```

$sort

Again, considering the zipcode collection, which has documents that look like this,

```
{
 "city" : "ACMAR",
 "loc" : [
  -86.51557,
  33.584132
 ],
 "pop" : 6055,
 "state" : "AL",
 "_id" : "35004"
}
```

Write an aggregation query with just a sort stage to sort by (state, city), both ascending. Assume the collection is called zips.

```
db.zips.aggregate([
{$sort: {state:1, city:1} }
])
```

$limit and $skip

Suppose you change the order of skip and limit in the query shown in the lesson, to look like this:

```
db.zips.aggregate([
    {$match:
     {
  state:"NY"
     }
    },
    {$group:
     {
  _id: "$city",
  population: {$sum:"$pop"},
     }
    },
    {$project:
     {
  _id: 0,
  city: "$_id",
  population: 1,
     }
    },
    {$sort:
     {
  population:-1
     }
    },
    {$limit: 5},
    {$skip: 10}
])
```

How many documents do you think will be in the result set?

○ 10

○ 5

● 0

○ 100

Revisiting $first and $last

Given the following collection:

```
> db.fun.find()
{ "_id" : 0, "a" : 0, "b" : 0, "c" : 21 }
{ "_id" : 1, "a" : 0, "b" : 0, "c" : 54 }
{ "_id" : 2, "a" : 0, "b" : 1, "c" : 52 }
{ "_id" : 3, "a" : 0, "b" : 1, "c" : 17 }
{ "_id" : 4, "a" : 1, "b" : 0, "c" : 22 }
{ "_id" : 5, "a" : 1, "b" : 0, "c" : 5 }
{ "_id" : 6, "a" : 1, "b" : 1, "c" : 87 }
{ "_id" : 7, "a" : 1, "b" : 1, "c" : 97 }
```

What would be the value of c in the result from this aggregation query

```
db.fun.aggregate([
    {$match:{a:0}},
    {$sort:{c:-1}},
    {$group:{_id:"$a", c:{$first:"$c"}}}
])
```

○ 21

● 54

○ 97

○ 5

$unwind

Suppose you have the following collection:

```
db.people.find()
{ "_id" : "Barack Obama", "likes" : [ "social justice", "health care", "taxes" ] }
{ "_id" : "Mitt Romney", "likes" : [ "a balanced budget", "corporations", "binders full of women" ] }
```

And you unwind the "likes" array of each document. How many documents will you wind up with?

○ 2

○ 4

● 6

○ 9

$unwind example

Which grouping operator will enable to you to reverse the effects of an unwind?

○ $sum

○ $addToSet

● $push

○ $first

use the aggregation framework to calculate the author with the greatest number of comments.

db.posts.aggregate([{$project: {author:"$comments.author"}}, {$unwind: "$author"}, {$group: {_id:"$author", count:{$sum:1}}}, {$sort:{count:1}}])

Please calculate the average population of cities in California (abbreviation CA) and

New York (NY) (taken together) with populations over 25,000.

For this problem, assume that a city name that appears in more than one state represents two separate cities. (some cities span more than one zipcode)

```
db.zips.aggregate([
{$match: {state:{$in:["CA","NY"]}}},
{$group: {_id:{city:"$city",state:"$state"}, pop:{$sum:"$pop"}} },
{$match:{pop:{$gt:25000}}},
{$group:{_id:null, avg:{$avg:"$pop"}} }
])
```

There are documents for each student (student_id) across a variety of classes (class_id). Note that not all students in the same class have the same exact number of assessments. Some students have three homework assignments, etc.

Your task is to calculate the class with the best average student performance. This involves calculating an average for each student in each class of all non-quiz assessments and then averaging those numbers to get a class average. To be clear, each student's average includes only exams and homework grades. **Don't include their quiz scores in the calculation.**

What is the class_id which has the highest average student perfomance?

Hint/Strategy: You need to group twice to solve this problem. You must figure out the GPA that each student has achieved in a class and then average those numbers to get a class average. After that, you just need to sort. The hardest class is class_id=2. Those students achieved a class average of 37.6

Below, choose the class_id with the highest average student average.

○ 8

○ 9

● 1

○ 5

○ 7

○ 0

○ 6

```
db.grades.aggregate([
{$unwind:"$scores"},
{$match:{'scores.type':{$in:["homework", "exam"]}}},
{$group:{_id:{student_id:"$student_id",class_id:"$class_id"}, gpa:{$avg:"$scores.score"}}},
{$project:{_id:0, student_id:"$_id.student_id", class_id:"$_id.class_id", gpa:1}},
{$group:{_id:"$class_id", avg:{$avg:"$gpa"}}},
{$sort:{avg:1}}

])
```

Removing Rural Residents In this problem you will calculate the number of people who live in a zip code in the US where the city starts with a digit. We will take that to mean they don't really live in a city. Once again, you will be using the zip code collection you imported earlier.

The project operator can extract the first digit from any field. For example, to extract the first digit from the city field, you could write this query:

```
db.zips.aggregate([
    {$project:
     {
 first_char: {$substr : ["$city",0,1]},
     }
   }
])
```

Using the aggregation framework, calculate the sum total of people who are living in a zip code where the city starts with a digit. Choose the answer below.

Note that you will need to probably change your projection to send more info through than just that first character. Also, you will need a filtering step to get rid of all documents where the city does not start with a digital (0-9).

- ● 298015
- ○ 345232
- ○ 245987
- ○ 312893
- ○ 158249
- ○ 543282

```
db.zips.aggregate([
{$project: {_id:0, first_char: {$substr : ["$city",0,1]}, zip:"$_id", city:1, state:1, pop:1} },
{$match:{first_char:{$in:["0","1","2","3","4","5","6","7","8","9"]}}},
{$group: {_id:null, pops:{$sum:"$pop"}}}

])
```

Posted by . at 3:02 PM

g+1 Recommend this on Google

Labels: Mongodb

No comments:

Post a Comment

Enter your comment...

Comment as: Google Accoui ▼

Publish Preview

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Blog Archive

- ▼ 2013 (85)
  - ► November (1)
  - ► September (18)
  - ► August (6)
  - ► July (13)
  - ▼ June (11)
    - Better Rest API design 4
    - Better Rest API design 3
    - Better Rest API design 2
    - Better Rest API design 1
    - Liferay (Tomcat container) setup connections pool
    - MongoDB course note 7 (final)
    - ssh and scp
    - Aws, git, heroku env setup
    - MongoDB course note 6
    - MongoDB course note 5
    - MongoDB course note 4
  - ► May (14)
  - ► March (18)
  - ► February (4)
- ► 2012 (8)
- ► 2011 (37)
- ► 2010 (19)

@Hao. Simple template. Powered by Blogger.