**PSG COLLEGE OF TECHNOLOGY, COIMBATORE - 641 004**

**DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES**

**Practice sheet –3 MongoDB**

**Practice Sheet-3- MongoDB Aggregation**

|  |  |
| --- | --- |
| **Session** | **Description** |
| $sum | Sums up the defined value from all documents in the collection. |
| $avg | Calculates the average of all given values from all documents in the collection. |
| $min | Gets the minimum of the corresponding values from all documents in the collection. |
| $max | Gets the maximum of the corresponding values from all documents in the collection. |
| $push | Inserts the value to an array in the resulting document. |
| $addToSet | Inserts the value to an array in the resulting document but does not create duplicates. |
| $first | Gets the first document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage. |
| $last | Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage. |

**Example: Create 'universities'  collection**

{

 country : 'Spain',

 city : 'Salamanca',

 name : 'USAL',

 location : {

   type : 'Point',

   coordinates : [ -5.6722512,17, 40.9607792 ]

 },

 students : [

   { year : 2014, number : 24774 },

   { year : 2015, number : 23166 },

   { year : 2016, number : 21913 },

   { year : 2017, number : 21715 }

 ]

}

{

 country : 'Spain',

 city : 'Salamanca',

 name : 'UPSA',

 location : {

   type : 'Point',

   coordinates : [ -5.6691191,17, 40.9631732 ]

 },

 students : [

   { year : 2014, number : 4788 },

   { year : 2015, number : 4821 },

   { year : 2016, number : 6550 },

   { year : 2017, number : 6125 }

 ]

}

The second and last collection is called 'courses' and looks like this:

{

 university : 'USAL',

 name : 'Computer Science',

 level : 'Excellent'

}

{

 university : 'USAL',

 name : 'Electronics',

 level : 'Intermediate'

}

{

 university : 'USAL',

 name : 'Communication',

 level : 'Excellent'

}

**Aggregation stages**

**MongoDB $match**

The $match stage allows us to choose just those documents from a collection that we want to work with.

db.universities.aggregate([

 { $match : { country : 'Spain', city : 'Salamanca' } }

]).pretty()

### MongoDB $group

With the $group() stage, we can perform all the aggregation or summary queries that we need, such as finding counts, totals, averages or maximums.

db.universities.aggregate([

{ $group : { \_id : '$name', totaldocs : { $sum : 1 } } }

]).pretty()

### MongoDB $out

This is an unusual type of stage because it allows you to carry the results of your aggregation over into a new collection, or into an existing one after dropping it, or even adding them to the existing documents (new in 4.1.2 version).

The $out() operator must be the last stage in the pipeline.

For the first time, we are using an aggregation with more than one stage. We now have two, a $group() and an $out():

db.universities.aggregate([

{ $group : { \_id : '$name', totaldocs : { $sum : 1 } } },

{ $out : 'aggResults' }

])

### MongoDB $unwind

The $unwind() stage in MongoDB is commonly found in a pipeline because it is a means to an end.

You cannot work directly on the elements of an array within a document with stages such as $group(). The $unwind() stage enables us to work with the values of the fields within an array.

Where there is an array field within the input documents, you will sometimes need to output the document several times, once for every element of that array.Each copy of the document has the array field replaced with the successive element.

This is the document:

{

country : 'Spain',

city : 'Salamanca',

name : 'USAL',

location : {

type : 'Point',

coordinates : [ -5.6722512,17, 40.9607792 ]

},

students : [

{ year : 2014, number : 24774 },

{ year : 2015, number : 23166 },

{ year : 2016, number : 21913 },

{ year : 2017, number : 21715 }

]

}

### MongoDB $sort

You need the $sort() stage to sort your results by the value of a specific field.

For example, let’s sort the documents obtained as a result of the $unwind stage by the number of students in descending order.

db.universities.aggregate([

{ $match : { name : 'USAL' } },

{ $unwind : '$students' },

{ $project : { \_id : 0, 'students.year' : 1, 'students.number' : 1 } },

{ $sort : { 'students.number' : -1 } }

]).pretty()

### MongoDB $limit

What if you are only interested in the first two results of your query? It is as simple as:

db.universities.aggregate([

{ $match : { name : 'USAL' } },

{ $unwind : '$students' },

{ $project : { \_id : 0, 'students.year' : 1, 'students.number' : 1 } },

{ $sort : { 'students.number' : -1 } },

{ $limit : 2 }

]).pretty()

{ "students" : { "year" : 2014, "number" : 24774 } }

{ "students" : { "year" : 2015, "number" : 23166 } }

### MongoDB $count

The $count() stage provides an easy way to check the number of documents obtained in the output of the previous stages of the pipeline.

db.universities.aggregate([

{ $unwind : '$students' },

{ $count : 'total\_documents' }

]).pretty()

### MongoDB $lookup

Because MongoDB is document-based, we can shape our documents the way we need. However, there is often a requirement to use information from more than one collection.

Using the $lookup(), here is an aggregate query that merges fields from two collections.

db.universities.aggregate([

{ $match : { name : 'USAL' } },

{ $project : { \_id : 0, name : 1 } },

{ $lookup : {

from : 'courses',

localField : 'name',

foreignField : 'university',

as : 'courses'

} }

]).pretty()

### MongoDB $sortByCount

This stage is a shortcut for grouping, counting and then sorting in descending order the number of different values in a field.

Suppose you want to know the number of courses per level, sorted in descending order. The following is the query you would need to build:

db.courses.aggregate([

{ $sortByCount : '$level' }

]).pretty()

### Exercise

Now, try to resolve the next exercise by yourself.

How do we get the total number of students that have ever belonged to each one of the universities?

db.universities.aggregate([

{ $unwind : '$students' },

{ $group : { \_id : '$name', totalalumni : { $sum : '$students.number' } } }

]).pretty()

The output:

{ "\_id" : "UPSA", "totalalumni" : 22284 }

{ "\_id" : "USAL", "totalalumni" : 91568 }

Yes, I have combined two stages. But, how do we build a query that sorts the output by the totalalumni field in a descending order?

db.universities.aggregate([

{ $unwind : '$students' },

{ $group : { \_id : '$name', totalalumni : { $sum : '$students.number' } } },

{ $sort : { totalalumni : -1 } }

]).pretty()

Right, we need to apply the $sort() stage at the output of the $group().

**Learning Link**

<https://docs.mongodb.com/manual/reference/method/db.collection.aggregate/>

<https://docs.mongodb.com/manual/core/aggregation-pipeline/>

<https://docs.mongodb.com/manual/reference/operator/aggregation/match/>

https://docs.mongodb.com/manual/reference/method/db.collection.aggregate/#db.collection.aggregate

https://docs.mongodb.com/manual/tutorial/insert-documents/