MongoDB Aggregation

Skills Network

Estimated time needed: **45** minutes

Objectives

After completing this lab you will be able to:

- Describe simple aggregation operators that process and compute data such as $sort, $limit, $group, $sum, $min, $max, and $avg
- Combine operators to create multi-stage aggregation pipelines
- Build aggregation pipelines that draw insights about the data by returning aggregated values

Prerequisites

Before starting this lab, it'll be helpful to have knowledge about accessing and performing basic operations with MongoDB. If you're unfamiliar with MongoDB, feel free to take a look at the Getting Started with MongoDB and MongoDB CRUD labs!

About Skills Network Cloud IDE

Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands on labs for course and project related labs. Theia is an open source IDE (Integrated Development Environment), that can be run on desktop or on the cloud. to complete this lab, we will be using the Cloud IDE based on Theia and MongoDB provided by Skills Network.

Important Notice about this lab environment

Please be aware that sessions for this lab environment are not persisted. Every time you connect to this lab, a new environment is created for you. Any data you may have saved in the earlier session would get lost. Plan to complete these labs in a single session, to avoid losing your data.

Exercise 1 - Getting the environment ready

Open the MongoDB database page by clicking the button below:

Open MongoDB Page in IDE

On that page, click the `Create` button to create a MongoDB database.

After creation has finished, click the `MongoDB CLI` button (below the `Create` button from the previous step) to connect to the database using the mongosh CLI.

Load sample data into the **training** database.

```
use training
db.marks.insert({"name":"Ramesh","subject":"maths","marks":87})
db.marks.insert({"name":"Ramesh","subject":"english","marks":59})
db.marks.insert({"name":"Ramesh","subject":"science","marks":77})
db.marks.insert({"name":"Rav","subject":"maths","marks":62})
db.marks.insert({"name":"Rav","subject":"english","marks":83})
db.marks.insert({"name":"Rav","subject":"science","marks":71})
db.marks.insert({"name":"Alison","subject":"maths","marks":84})
db.marks.insert({"name":"Alison","subject":"english","marks":82})
db.marks.insert({"name":"Alison","subject":"science","marks":86})
db.marks.insert({"name":"Steve","subject":"maths","marks":81})
db.marks.insert({"name":"Steve","subject":"english","marks":89})
db.marks.insert({"name":"Steve","subject":"science","marks":77})
db.marks.insert({"name":"Jan","subject":"english","marks":0,"reason":"absent"})
```

Exercise 2 - Limiting the rows in the output

Using the $limit operator we can limit the number of documents printed in the output.
This command will print only 2 documents from the **marks** collection.

```
use training
db.marks.aggregate([{"$limit":2}])
```

Exercise 3 - Sorting based on a column

We can use the $sort operator to sort the output.

This command sorts the documents based on field marks in ascending order.

```
db.marks.aggregate([{"$sort":{"marks":1}}])
```

This command sort the documents based on field marks in descending order.

```
db.marks.aggregate([{"$sort":{"marks":-1}}])
```

Exercise 4 - Sorting and limiting

Aggregation usually involves using more than one operator.
A pipeline consists of one or more operators declared inside an array.
The operators are comma separated.
Mongodb executes the first operator in the pipeline and sends its output to the next operator.

Let us create a two stage pipeline that answers the question “What are the top 2 marks?”.

~~~
db.marks.aggregate([
{"$sort":{"marks":-1}},
{"$limit":2}
])
```
# Exercise 5 - Group by
The operator $group by, along with operators like $sum, $avg, $min, $max, allows us to perform grouping operations.
This aggregation pipeline prints the average marks across all subjects.
```
db.marks.aggregate([
{
    "$group":{
        "_id":"$subject",
        "average":{"$avg":"$marks"}
        }
}
])
~~~

The above query is equivalent to the below sql query.

SELECT subject, average(marks)
FROM marks
GROUP BY subject

Exercise 6 - Putting it all together

Now let us put together all the operators we have learnt to answer the question. “Who are the top 2 students by average marks?”
This involves:

- finding the average marks per student.

- sorting the output based on average marks in descending order.
- limiting the output to two documents.

```
db.marks.aggregate([
{
    "$group":{
        "_id":"$name",
        "average":{"$avg":"$marks"}
        }
},
{
    "$sort":{"average":-1}
},
{
    "$limit":2
}
])
```

Practice exercises

1. Problem:

   Find the total marks for each student across all subjects.

▶ Click here for Hint
▶ Click here for Solution

2. Problem:

   Find the maximum marks scored in each subject.

▶ Click here for Hint
▶ Click here for Solution

3. Problem:

   Find the minimum marks scored by each student.

▶ Click here for Hint
▶ Click here for Solution

4. Problem:

   Find the top two subjects based on average marks.

▶ Click here for Hint
▶ Click here for Solution

5. Problem:

   Disconnect from the mongodb server.

▶ Click here for Hint
▶ Click here for Solution

Authors

Ramesh Sannareddy

Other Contributors

Rav Ahuja