Exercise 1 - Load sample document

Load sample data into the `training` database in `marks` collection.

```
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"})
```

Copied!

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}])
```

Copied!

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}}])
```

Copied!

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

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

Copied!

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's create a two stage pipeline that answers the question "What are the top 2 marks?"

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

Copied!

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"}
    }
}
])
```

Copied!

↓ similarly

The above query is equivalent to the below sql query.

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

Copied!

Exercise 6 - Putting it all together

Now let's 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
    }
])
```

Copied!

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

Summary

In this lab, you have gained an understanding of Aggregation pipelines in MongoDB.

Author(s)

Muhammad Yahya