

Lab 10 – MongoDB – Aggregation

Objective

In this lab, students learn how to transform and combine documents in a collection to do more complex analysis in a MongoDB database.

Students learn how to use the following operators:

\$match: is used to filter documents.

\$project: is used to select a field, rename a field, and apply some operations on the selected fields from documents.

\$group: This operator groups documents base on a certain field.

\$sort: This operator sorts documents based on a given field.

\$limit: This operator returns first n documents as a result of a query.

Skipping

Submission

For this lab, you should submit a file with the below exercises completed.

Your file should be called: **L10-*lastname-firstname*** (for example: L10-King-Les)

Getting Started

In this lab, you will use grades.json dataset. Download grades.json from Blackboard and store it in a folder named dataset.

Open your Windows command prompt and go the following directory where MongoDB is installed:

➤ `cd C:\Program Files\MongoDB\Server\4.2\bin`

To run MongoDB, execute ***mongod***

➤ `mongod`

When MongoDB starts successfully, open another Windows command prompt and go to the same *bin* directory:

➤ `cd C:\Program Files\MongoDB\Server\4.2\bin`

and execute *mongo*

➤ `mongo`

Or you execute a batch file to start up MongoDB.

You will import `grades.json` to the *college* database. To import data, go to the *bin* directory:

➤ `cd C:\Program Files\MongoDB\Server\4.2\bin`

Execute the following command:

➤ `mongoimport --db college --collection grades --file
..\dataset\grades.json`

To import the *json* file, provide the full path to the `grades.json`. After executing the command, the data is imported to the *college* database. To make sure data is imported successfully, go to the MongoDB shell and execute the following command to see the imported documents:

➤ `show dbs`

You should see the database *college* added to the list of your databases. To see the documents inside the database:

➤ `use college`
➤ `db.grades.find().forEach(printjson)`

or

➤ `db.grades.find().pretty()`

Tasks

1. Write an aggregate statement to sort the documents in the *grades* collection based on students ID and class ID. Display only the student ID and the class ID for each document. *Sort the result from high to low values for student ID and from low to high for class ID.*

```
db.grades.aggregate([
  {$project: {
    "_id": 0,
    "student_id": 1,
    "class_id": 1,
  }},
  {$sort: {"student_id": -1, "class_id": 1}}
]);
```

2. Revise the previous query to show the result for students with IDs between 10 and 12.

```
db.grades.aggregate([
  {$project: {
    "_id": 0,
    "student_id": 1,
    "class_id": 1,
  }},
  {$match: {
    "student_id": {$gte: 10, $lte: 12}
  }},
  {$sort: {"student_id": -1, "class_id": 1}}
]);
```

3. Show only existing class IDs in the grades collection. (Do not show duplicates.)

```
db.grades.aggregate([
  {$group: {_id: "$class_id"}},
  {$sort: {_id: 1}}
]);
```

4. Write a query to display student ID and class ID for students whose score are greater than 99.00. *Sort the result based on student ID from high to low and class ID from low to high.*

```
db.grades.aggregate([
  {$match: {
    scores: {$elemMatch: {score: { $gt: 99 }}}}
  },
  {$project: {
    "_id": 0,
    "student_id": 1,
    "class_id": 1,
  }},
  {$sort: {"student_id": -1, "class_id": 1}}
]);
```

5. Write a query to show the maximum and the minimum class ID for each student. *Sort the result based on student ID from low to high.* Show only the first **10** students.

See the following sample output:

```
{ "_id" : 0, "max_class_id" : 30, "min_class_id" : 2 }
{ "_id" : 1, "max_class_id" : 28, "min_class_id" : 13 }
{ "_id" : 2, "max_class_id" : 27, "min_class_id" : 24 }
{ "_id" : 3, "max_class_id" : 25, "min_class_id" : 3 }
{ "_id" : 4, "max_class_id" : 26, "min_class_id" : 0 }
{ "_id" : 5, "max_class_id" : 30, "min_class_id" : 0 }
{ "_id" : 6, "max_class_id" : 29, "min_class_id" : 8 }
{ "_id" : 7, "max_class_id" : 17, "min_class_id" : 17 }
{ "_id" : 8, "max_class_id" : 29, "min_class_id" : 0 }
{ "_id" : 9, "max_class_id" : 30, "min_class_id" : 0 }
```

```
db.grades.aggregate([
  {$group: {
    _id: "$student_id",
    max_class_id: {$max: "$class_id"},
    min_class_id: {$min: "$class_id"}
  }},
  {$sort: {"_id": 1}},
  {$limit: 10}
]);
```

6. Write a query to find the number of failed exams for student with ID 48.

```
db.grades.aggregate([
  {$match: {
    student_id: 48,
    scores: {
      $elemMatch: {
        score: {$lt: 59},
        type: "exam"
      }
    }
  }},
  {$count: "failed_exams"}
]);
```