# Lab 10 – Week 12

# (MongoDB – UPDATE and Aggregate)

## Objective

In this lab, students learn how to update documents in a MongoDB database.

Use the aggregation framework to transform and combine documents in a collection.

Run simple aggregation commands such as count, distinct and group.

**update():** This method updates one document by default. If you want to update all documents that match the criteria using this method, you need the option {multi:true}.

update(<filter>,<update>,<option>)

The *filter* parameter specifies the criteria. For instance:

{“\_id”= 0}

{} for updating all documents

The *update* parameter specifies the changes that will be applied to a document.

**updateOne():** This method updates only the first document that matches the criteria.

updateOne(<filter>,<update>)

**updateMany():** This method updates all documents that match the criteria.

updateMany(<filter>,<update>)

Group work acknowledgment

We, ------------(mention your names), declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. No part of this assignment has been copied manually or electronically from any other source (including web sites) **or distributed to other students.**

Specify below what each member has done towards the completion of this work:

Name Task(s)

1. Hiu Fung Chan Q 1 - 4
2. Trung Kien Phan Q 5 - 7
3. Henry Lau Q 8 - 10

Importing data

1. Download MongoDB database tools zip file. <https://www.mongodb.com/try/download/database-tools?tck=docs_databasetools>
2. unzip the folder and place it under C:\Program Files\MongoDB. You can rename the folder as mongo-db-database-tools.
3. Create a folder named Mongo\_Import\_Export under C:\Program Files\MongoDB and place the students.json file under mongo\_Import\_Export folder.

The folder C:\Program Files\MongoDB should now contain:

Graphical user interface, text, application

Description automatically generated

1. In mongodb-database-tools/bin folder, double click on mongoimport, this will open up a command prompt. Or open a command prompt, type cd pathTo

pathTo is the path to where mongoimport is located.

1. It is time to import the data from the json file into a mongodb database. For this, type the following command:

mongoimport.exe "c:\program files\MongoDB\Mongo\_Import\_Export\students.json" /d college /c students --jsonArray

db.students.find()

This should show all documents in the students collection.

The database name is college.

When MongoDB starts successfully, open another Windows command prompt and go 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.

To import the *json* file, provide the full path to the students.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.students.find().forEach(printjson)

or

* db.students.find().pretty()

## Tasks

1. Write an update statement to add new fields *program* and *term* to all documents in the *students* collection and set them to values “*CPA*” and *1*.

|  |
| --- |
| db.students.updateMany(  {},  {  $set: {  program: “CPA”,  term: 1  }  }  ); |

1. Write an update statement to modify the value of the *program* field to “*BTM”* for all documents in the *students* collection.

|  |
| --- |
| db.students.updateMany(  {},  {  $set: {  Program: “BTM”  }  }  ); |

1. Write an update statement to modify the value of the program field to “*CPA”* for the student named *Jonie Raby*.

Before executing an update statement or a delete statement, you can use the *find()* method with the update or delete criteria, to see how many documents will be affected.

Write the update statement in the box below.

|  |
| --- |
| db.students.find({name: “Jonie Raby”});  db.students.updateOne(  { name: “Jonie Raby” },  {  $set: {  Program: “CPA”  }  }  ); |

How many documents are there with the value *Jonie Raby* for the *name* field? \_\_\_1\_\_\_

How many documents were updated? \_\_\_0\_\_\_

1. Write a query to show only the *program* field for the document that the value of the filed *name* is *Jonie Raby*.

|  |
| --- |
| db.students.find(  { name: "Jonie Raby" },  { \_id: 0, program: 1 }  ); |

1. Write an update statement to increase the value of the *term* field by 2 for documents with *\_id* 20, 22, and 24.

|  |
| --- |
| db.students.updateMany(  { \_id: { $in: [20, 22, 24] } },  { $inc: { term: 2 } }  ); |

1. Write an update statement to remove the *term* field from documents that the value of the *term* filed is 3.

|  |
| --- |
| db.students.updateMany(  { term: 3 },  { $unset: { term: "" } }  ); |

1. Show the number of students per term Hint: Use the aggregate() function and the $group operator.

|  |
| --- |
| db.students.aggregate(  [ { $group: { \_id: "$term", count: { $sum: 1 } } } ]  ); |

1. Unwind the students collection to turn each score into its own document.

|  |
| --- |
| db.students.aggregate(  [ { $unwind: "$scores" } ]); |

1. Unwind the students collection and show the average score, min score and max score per student and sort them by avgScore in descending order.

|  |
| --- |
| db.students.aggregate([  { $unwind: "$scores" },  { $group: { \_id: "$\_id",  name: { $first: "$name" },  avgScore: { $avg: "$scores.score" },  minScore: { $min: "$scores.score" },  maxScore: { $max: "$scores.score" }  }  },  {  $sort: { avgScore: -1 }  }  ]); |

1. Find the number of documents in the students collection.

|  |
| --- |
| db.students.countDocuments() |

SUBMISSION

Submit your lab10\_GroupX.doc file on BB. Replace X with your group number.

If a student does not contribute to the work, do not list his/her name(s) under the group section in the lab file and will get 0.