# Lab 9

# (MongoDB – Update and Replacement)

## Objective`

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

**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>)

## Getting Started

In this lab, you will use students.json dataset. Download students.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 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.

* cd C:\Program Files\MongoDB\Server\4.2\**bin**

**Execute the following command to create “student” collection under “university” database:**

> use university

> db.student.insert({"\_id" : 1, "name":"Paul Lee", "age" : 32, "courses" : ["java", "C", "DBS"]})

> db.student.insert({"\_id" : 2, "name":"Mark Zank", "age" : 22, " courses " : ["Maths", "C"]})

> db.student.insert({"\_id" : 3, "name":"Juke Peter", "age" : 19, " courses " : ["Web", "Java", "DBS"]})

> db.student.insert({"\_id" : 11, "name":"John Borg", "age" : 21, "courses" : ["java", "C", "DBS"]})

> db.student.insert({"\_id" : 22, "name":"Lee Mike", "age" : 32, " courses " : ["Maths", "Java"]})

> db.student.insert({"\_id" : 33, "name":"Paul Lin", "age" : 35, " courses " : ["Web", "Java", "DBS"]})

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.student.find().forEach(printjson)

or

* db.student.find().pretty()

## Submission

You submit this file with answers (in the provided space). Name the file L09\_ID#\_LASTNAME.docx”.

## Tasks

Use the above created student database to answer the following questions:

1. Write a statement to sort the documents in the *student* collection based on students ID ( *high to low values*) and student name (*from low to high values*).

|  |
| --- |
| db.student.find().sort({ "\_id":-1, "name":1}). forEach(printjson) |

1. Write a query to update the age of the students to 45, who are enrolled in the course Java.

|  |
| --- |
| db.student.updateMany({ "courses": { $all: ["Java"]}}, {$set: { "age": 45}}); |

1. Write a query to show the student IDs for last 2 documents in the given collection.

Hint: use - >db.collection.find().skip(2).

|  |
| --- |
| db.student.find({},{ \_id:1}).skip(4). forEach(printjson) |

1. Write a query to display total number the of students who have age not more than 25 and are have enrollment in exactly three courses.

|  |
| --- |
| db.student.find({ "$and":[{ "age": {"$lte":25}} ,{"courses" :{"$size":3 }} ]}). forEach(printjson) |

1. Write a query to add the following array to the student collection who have age greater than 30. The array – Phone [121313, 232412]

|  |
| --- |
| db.student.updateMany({ "age" : {"$gt":30}} , {"$push" : {"Phone" : {"$each" : [121313, 232412] }}}); |

1. Write a query to find the number of students who have enrolled in the C course. Display only first last two courses for them.

|  |
| --- |
| db.student.find({courses:"C"}, {"courses":{"$slice":-2}}). forEach(printjson) |