Practical G01
Creating and Manipulating data in NoSQL

What you will learn / do in this lab

1. How to store and manipulate data stored in NoSQL databases such as MongoDB
2. https://docs.mongodb.com/manual/crud/

Table of Contents

1. Overview

A. What you will do for this lab

In this lab, you will learn how to set up your own NoSQL database server (MongoDB) on your laptop and manipulate the data stored in the database using:

1) MongoDB command-line tool (mongo.exe)

B. Intro to MongoDB

MongoDB is an open-source NoSQL database that provides high performance, high availability, and automatic scaling.

A record in MongoDB is a document, which is a data structure composed of field and value pairs. For example, the following shows a record which stores user information.

MongoDB stores its documents in collections; for example, the document above can be stored in a collection called “users”. If you have documents that contain product information, you may store them in a collection called “products”. You can think of collections as the equivalent to tables in relational databases. Finally, collections are stored in databases. So the users and products collections could be stored inside a database called mydatabase.

2. Install MongoDB on your laptop

A. Windows Installation

No	Task
a)	Download the latest production release of **MongoDB Community Server** from the MongoDB Download Center (180 MB file) https://www.mongodb.com/download-center/community
b)	• Locate the downloaded MongoDB .msi file, which typically is located in the default Downloads folder and double-click the .msi file • A set of screens will appear to guide you through the installation process • Choose "Full Installation" which will install MongoDB to C:\Program Files\MongoDB\Server\verx.x\ by default • During the installation process you will be given the option to install MongoDB Compass in addition to MongoDB Server, select Yes

3. Start MongoDB Server

A. Set up the MongoDB environment

No	Task

a)	MongoDB requires a data directory to store all data.
	
	MongoDB's default data directory path is the absolute path \data\db on the drive from which you start MongoDB
	
	Create this folder by running the following command in a Command Prompt
	
	`mkdir C:\data`
	`mkdir C:\data\db`

B. Start MongoDB with mongod.exe

No	Task
a)	To start MongoDB, run ***mongod.exe***. For example, from the Command Prompt: `"C:\Program Files\MongoDB\Server\3.6\bin\mongod.exe"` This starts the main MongoDB database process. The waiting for connections message in the console output indicates that the mongod.exe process is running successfully. Depending on the security level of your system, Windows may pop up a Security Alert dialog box about blocking "some features" of C:\Program Files\MongoDB\Server\3.6\bin\mongod.exe from communicating on networks. All users should select Private Networks, such as my home or work network and click Allow access.

C. Verify that MongoDB started successfully

No	Task
a)	Verify that MongoDB has started successfully by checking the process output for the following line:

```
[initandlisten] waiting for connections on port 27017
```

The output should be visible in the terminal or shell window.

You may see non-critical warnings in the process output. As long as you see the log line shown above, you can safely ignore these warnings during your initial evaluation of MongoDB.

```
2017-12-16T00:24:26.995-0700 I CONTROL  [initandlisten] See http://dochub.mongodb.org/core/wt-
windows-system-file-cache
2017-12-16T00:24:26.996-0700 I CONTROL  [initandlisten]
2017-12-16T15:24:27.453+0800 I FTDC     [initandlisten] Initializing full-time diagnostic data
 capture with directory 'C:/data/db/diagnostic.data'
2017-12-16T15:24:27.456+0800 I NETWORK  [initandlisten] waiting for connections on port 27017
```

4. Use MongoDB with Mongo Shell

A. Start mongo.exe

No	Task
a)	We can connect to our Mongo Server using a command-line tool called ***mongo.exe***. https://docs.mongodb.com/manual/reference/method/#collection https://docs.mongodb.com/manual/reference/method/#cursor
b)	Open a new Command prompt window and type the following command `"C:\Program Files\MongoDB\Server\3.6\bin\mongo.exe"`
c)	You should see output similar to that below with the following starting lines as shown in screen 1 and ending lines in screen 2

```
C:\Users\Dora Chua>"C:\Program Files\MongoDB\Server\3.6\bin\mongo.exe"
MongoDB shell version v3.6.0
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.0
```

```
2017-12-16T00:24:26.995-0700 I CONTROL  [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2017-12-16T00:24:26.996-0700 I CONTROL  [initandlisten]
>
```

B. View the databases in Mongo

No	Task
a)	We can know what databases are there already in Mongo Server by typing this command at the > symbol `>show dbs`

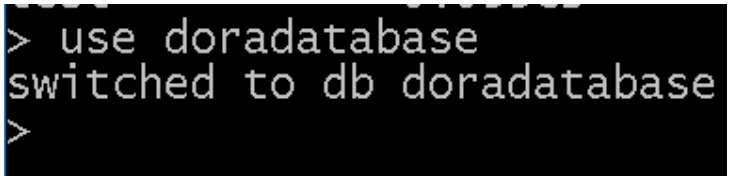
No	Task

b) For example, this is what I see in my MongoDB server.

You will not see the same output on your screen, as you probably have less databases than me since yours is a fresh installation while I already have been creating many other databases since I first installed.

```
> show dbs
admin          0.000GB
botdb          0.013GB
botdb2         0.000GB
botdb3         0.000GB
chiabot        0.000GB
db             0.000GB
local          0.000GB
myNewDatabase  0.000GB
mycompanydata  0.000GB
test           0.033GB
```

C. Switch to a specific database

No	Task
a)	We can switch to a particular database by typing the use command e.g. `>use doradatabase`
b)	This is the output you will see 

D. View collections

No	Task
a)	MongoDB uses the concept of “collections” to represent tables in a database.
b)	We can view the collections in a Mongo database using the ***show collection*** command `>show collections`
c)	For example, I already have a database called ***mycompanydata*** and as you can see from the screen below, once I switch over to this database using the use command, I can see there are 2 collections in it, named inventory and users respectively. 

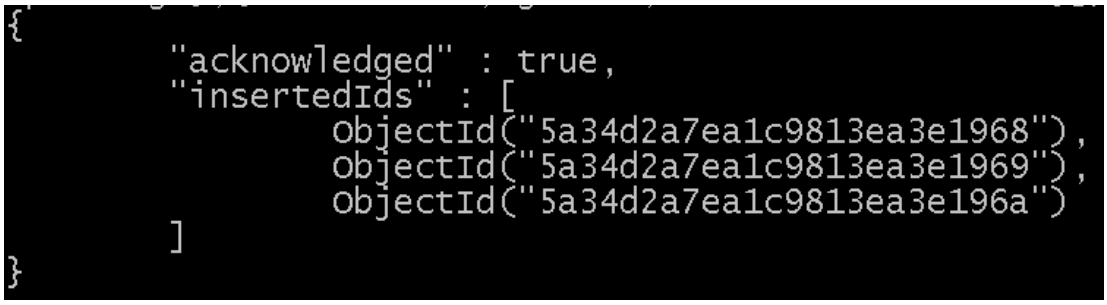
E. Insert new documents into a collection

Add one document

No	Task
a)	In this section, let's practise how to create a new database, a new collection and a new document all at once.
b)	First, let's start a new database `>use st1501_database`
c)	let's insert one new document into this database with the following command `> db.users.insertOne({name: "dora",age:44, status: "pending"})`
d)	You should see the following output if the insertion was successful 

Add multiple documents

No	Task
a)	Now that you know how to insert a single document, let's practise how to insert multiple documents.
b)	Ensure you are in the correct database `>use st1501_database`

No	Task
c)	Next, let's insert a few new documents into this database with the following command (Note: The paramerts is a list [] of {}.) `> db.users.insertMany([{name: "sue",age: 26,status: "confirmed"},{name: "zan",age: 18,status: "pending"},{name: "don",age: 20,status: "confirmed"}])`
d)	You should see the following output if the insertion was successful 

F. View documents in collection

View all documents

No	Task
a)	In this section, let's practise how to view the newly created documents you added in the previous section.
b)	First, make sure you are in the correct database `>use st1501_database`
c)	Type the following command to view ALL documents `> db.users.find({})`

No	Task

d)	You should see the following output if the query was successful
	`> db.users.find({})`
	`{ "_id" : ObjectId("5a34d273ea1c9813ea3e1967"), "name" : "dora", "age" : 44, "status" : "pending" }`
	`{ "_id" : ObjectId("5a34d2a7ea1c9813ea3e1968"), "name" : "sue", "age" : 26, "status" : "confirmed" }`
	`{ "_id" : ObjectId("5a34d2a7ea1c9813ea3e1969"), "name" : "zan", "age" : 18, "status" : "pending" }`
	`{ "_id" : ObjectId("5a34d2a7ea1c9813ea3e196a"), "name" : "don", "age" : 20, "status" : "confirmed" }`

View document with criteria, projection and limit

No	Task
a)	In this section, let's practise how to view documents using specified criteria, projection and limit. We can control the documents that are returned in a query by specifying equality conditions. E.g. db.users.find({ status: "Pending" }) or db.users.find({ age: {$gt:40}}) We can also control the fields that are returned in a query by using projection by setting the <field> to 1. E.g. db.users.find({ status: "Pending" }, {name:1}) We can also control the number of documents that are returned by using the limit command. E.g. db.users.find({age: {$lt:45}}, {name:1,age:2}).limit(3)
b)	First, make sure you are in the correct database `>use st1501_database`
c)	Type the following command to view the names of the first 3 users who have age less than 45 `> db.users.find({age: {$lt:45}}, {name:1,age:2}).limit(3)`
d)	You should see the following output if the query was successful

No	Task

```
> db.users.find({age: {$lt:45}}, {name:1,age:2}).limit(3)
{ "_id" : ObjectId("5a34d273ea1c9813ea3e1967"), "name" : "dora", "age" : 44 }
{ "_id" : ObjectId("5a34d2a7ea1c9813ea3e1968"), "name" : "sue", "age" : 26 }
{ "_id" : ObjectId("5a34d2a7ea1c9813ea3e1969"), "name" : "zan", "age" : 18 }
```

Sort documents

No	Task
a)	In this section, let's practise how we can modify the previous query with a sort instruction
b)	First, make sure you are in the correct database `>use st1501_database`
c)	Type the following command to view the names of the first 3 users who have age less than 45, sorted in ascending order (youngest first) `> db.users.find({age: {$lt:45}}, {name:1,age:2}).limit(3) .sort( { age: 1 } )`
d)	You should see the following output if the query was successful

```
> db.users.find({age: {$lt:45}}, {name:1,age:2}).limit(3) .sort( { age: 1 } )
{ "_id" : ObjectId("5a34d2a7ea1c9813ea3e1969"), "name" : "zan", "age" : 18 }
{ "_id" : ObjectId("5a34d2a7ea1c9813ea3e196a"), "name" : "don", "age" : 20 }
{ "_id" : ObjectId("5a34d2a7ea1c9813ea3e1968"), "name" : "sue", "age" : 26 }
```

G. Update documents in collection

Update all documents

No	Task
a)	In this section, let's practise how to update **all** documents to have a status of pending
b)	First, make sure you are in the correct database `>use st1501_database`
c)	Type the following command to update ALL documents `> db.users.updateMany({}, {$set: {status: 'pending'}})`

d)	You should see the following output if the query was successful
	`> db.users.updateMany({}, {$set: {status: 'pending'}})`
	`{ "acknowledged" : true, "matchedCount" : 4, "modifiedCount" : 2 }`

Update specific documents with specific fields

No	Task
a)	In this section, let's practise how to update **specific** documents to have a status of pending as well as
b)	First, make sure you are in the correct database `>use st1501_database`

No	Task

c)	Next, let's insert some documents in a new collection so that we can update them later.
	
	`db.inventory.insertMany( [`
	`  { item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" },`
	`  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },`
	`  { item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" }, status: "A" },`
	`  { item: "mousepad", qty: 25, size: { h: 19, w: 22.85, uom: "cm" }, status: "P" },`
	`  { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },`
	`  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },`
	`  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },`
	`  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },`
	`  { item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },`
	`  { item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }`
	`] );`

d)	Type the following command to update only those documents that have quantity less than 50. The 'size.uom' and 'status' should be updted, and a property 'lastModified' added.
	`db.inventory.updateMany(`
	`  { "qty": { $lt: 50 } },`
	`  {`
	`  $set: { "size.uom": "in", status: "P" },`
	`  $currentDate: { lastModified: true }`
	`  }`
	`)`

e)	You should see the following output if the query was successful
	`> db.inventory.updateMany(`
	`...  { "qty": { $lt: 50 } },`
	`...  {`
	`...  $set: { "size.uom": "in", status: "P" },`
	`...  $currentDate: { lastModified: true }`
	`...  }`
	`... )`
	`{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }`

f)	Let's view the updated documents to verify `db.inventory.find({})`

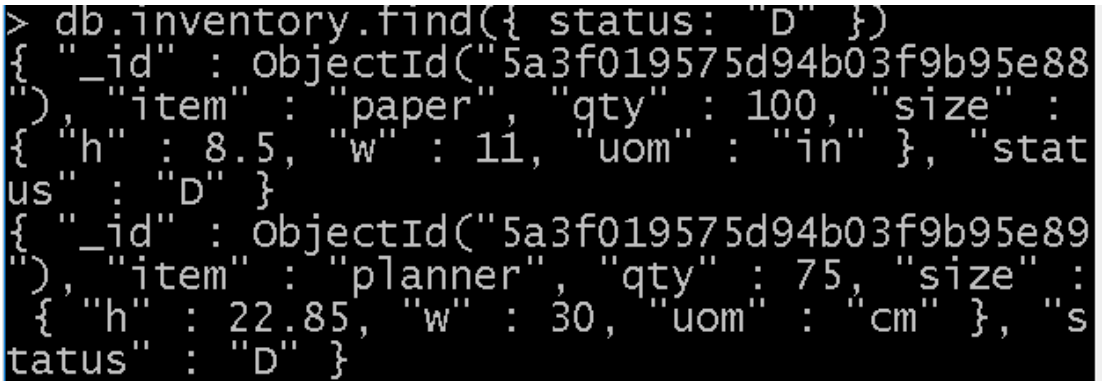
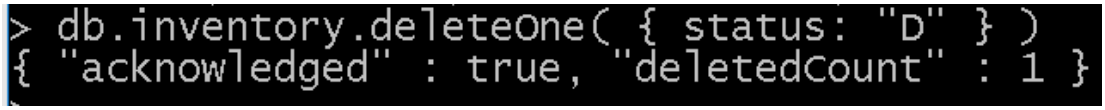
No	Task

g) You should see output similar to that below

```
> db.inventory.find({})
{ "_id" : ObjectId("5a3f019575d94b03f9b95e83"), "item" : "canvas", "qty" : 100, "size" : { "h"
 : 28, "w" : 35.5, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5a3f019575d94b03f9b95e84"), "item" : "journal", "qty" : 25, "size" : { "h"
 : 14, "w" : 21, "uom" : "in" }, "status" : "P", "lastModified" : ISODate("2017-12-24T01:24:05
.846Z") }
{ "_id" : ObjectId("5a3f019575d94b03f9b95e85"), "item" : "mat", "qty" : 85, "size" : { "h" : 2
7.9, "w" : 35.5, "uom" : "cm" }, "status" : "A" }
{ "_id" : ObjectId("5a3f019575d94b03f9b95e86"), "item" : "mousepad", "qty" : 25, "size" : { "h
" : 19, "w" : 22.85, "uom" : "in" }, "status" : "P", "lastModified" : ISODate("2017-12-24T01:2
4:05.847Z") }
```

H. Delete documents in collection

Delete specific documents with specific criteria

No	Task
a)	In this section, let's practise how to delete **specific** documents that fit a given criteria
b)	First, make sure you are in the correct database `>use st1501_database`
c)	Let's view the current inventory collection to see the existing documents inside it. `db.inventory.find({ status: "D" })`
d)	Note that there are 2 documents with status D inside this collection 
e)	Type the following command to delete the first documentt that has status equal to D. `db.inventory.deleteOne( { status: "D" } )`
f)	You should see the following output if the operation was successful 
g)	Let's view the updated collection to verify `db.inventory.find({ status: "D" })`

No	Task

h)	Note that there is now only one document left with status D inside this collection since the other one has been deleted.
	`> db.inventory.find({ status: "D" })`
	`{ "_id" : ObjectId("5a3f019575d94b03f9b95e89"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "D" }`

Delete all documents

No	Task
a)	In this section, let's practise how to delete **all** documents in the inventory collection.
b)	First, make sure you are in the correct database `>use st1501_database`
c)	First, let's check how many documents are there in the entire inventory collection. `db.inventory.count({})`

d)	You should see similar output as that below.
	`> db.inventory.count({})`
	`9`

e)	Type the following command to delete ALL documents in the inventory collection `> db.inventory.deleteMany({})`
f)	You should see the following output if the query was successful

No	Task

```
> db.inventory.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 9 }
```

-- End of Lab --