Big Data Lab – Week 4: Working with MongoDB Database

In this lab you will practice the basics of creating and working with a MongoDB database, importing data into MongoDB, and also retrieving data from MongoDB. You can either using the VM or MongoDB server installed on your computer to do all the tasks.

To run the VM on a lab PC on campus, click **This PC** on the desktop and then navigate to the **VMwar**e folder on the **C** drive.

Click on the folder **SCEBE-LinuxMint_17.xxxx**

Inside this folder you are looking for the executable file in the file set in front of you.

Using the following username and password to log into the VM:

Username: ***bdtech***
Password: ***bdtech***

The VM has MongoDB version 3.4.5 installed, which is fine for our labs.

You can install *MongoDB Community Server* on your own machine. The current version is 5.0.3 (Released in Sep 2021), which is ok for this lab. However, if you install the current version (or any version from version 4.4), you need to download and install *MongoDB Database Tools* separately. Because from version 4.4, MongoDB releases its Database Tools (e.g., **mongoimport**, which will be used in task 2 of this lab) separately from the Server.

You can download the database tools from:
https://www.mongodb.com/try/download/database-tools

Documentation of the database tools can be found at:
https://docs.mongodb.com/database-tools/

To avoid the separate installation of the database tools, you can install an older MongoDB Community Server, e.g., version 4.2.

Task 1: Basics of Working with MongoDB

In this task you will learn to run a single ***MongoDB server*** instance and use the ***Mongo shell*** to create a database and insert documents in it.

1. Start the MongoDB server

   Open a terminal window. From your home directory start a MongoDB server instance with the **mongod** command:

```
mongod --dbpath data/db
```

Note that mongod option names are preceded by a double dash (--).

You should see a set of server startup messages which should include one indicating that the server is alive and waiting for connections on port 27017.

Note the above command assumes that the ***bin*** subdirectory of the MongoDB installation directory on your computer is on the current path, and that there is a subdirectory ***data/db*** in your home directory, which is a valid directory where the *mongod* instance stores its data. Otherwise, you may have to include the full path to the ***bin*** directory, and you may have to create the data directory.

If you are running the mongod command on a Windows machine, make sure you use backword slash in the data directory, for example:

```
mongod --dbpath c:\data\db
```

2. Open **another** terminal window. Run the **mongo** command to start a Mongo shell client:

```
mongo
```

Again, you may have to include the full path to the ***bin*** directory.

You should see a welcome message and a shell command prompt '**>**', indicating that you have connected to the server.

3. Enter the **db** command in the shell, which will show the name of the current database.

```
> db
```

4. Enter the following **use** command in the shell, which will create a new database called *myDatabase* if it does not already exist, and will switch to using that database.

```
> use myDatabase
switched to db myDatabase
```

5. Enter the following (**insert** or **insertOne**) commands, which will store two new JSON documents in the **collection** ***basket***. The *basket* collection will be created if it does not already exist. The ***db*** is a reference to the current database.

```
> db.basket.insert({"customer": "Customer03", "orderDate": "2021-09-12",
"product":["Milk", "Baked Beans", "Cornflaks"]})
```

```
> db.basket.insert({"customer": "Customer05", "orderDate": "2021-10-08",
"product":["Apple", "Baked Beans"]})
```

6. Enter the following **show** command, which lists the collections in the current database:

```
> show collections
```

You should see *basket* listed.

7. Run the ***find*** function to check the documents have been stored. You should see response similar to the one listed in the screenshot below:

```
> db.basket.find()
{ "_id" : ObjectId("5da45a3f8cddaa957adce4be"), "customer" : "Custome
r03", "orderDate" : "2015-08-09", "product" : [ "Milk", "Baked Beans"
, "Cornflakes" ] }
{ "_id" : ObjectId("5da45aaa8cddaa957adce4bf"), "customer" : "Custome
r05", "orderDate" : "2015-08-10", "product" : [ "Apple", "Baked Beans
" ] }
```

8. Enter the following ***find*** commands which run simple queries on the *basket* collection:

```
> db.basket.find({product : "Baked Beans"})
```

```
> db.basket.find({product : "Baked Beans"}, {"customer" : 1})
```

What is the difference between the results of these two queries? Why?

9. Run the following ***update*** commands. Use ***find*** function to check that the updates have been applied (you can refer to the output from exercise 7):

```
> db.basket.update({customer: "Customer03"}, {$set:{"orderDate":
"2021-09-15"}})
```

```
> db.basket.update({"customer": "Customer05"},{$push:{ "product":
"Orange"}})
```

```
> db.basket.find()
```

What is the effect of each of these updates?

10. Enter the following **createIndex** command. It creates a single key index, sorted in ascending order, on the ***customer*** field:

```
> db.basket.createIndex({"customer": 1})
```

Check the response, and identify how many indexes did the collection have before running this command, and how many does it have now?

11. Enter the ***getIndexes*** command to list the indexes:

```
> db.basket.getIndexes()
```

Check the response, and find out what the names of the indexes are, and which fields they index.

12. Enter the following **remove** command to delete a document. Using the ***find*** command to check if that the document has been deleted.

```
> db.basket.remove({"customer": "Customer03"})
```

13. Enter the following **drop** command to delete the whole ***basket*** collection and then the **dropDatabase** command to delete the ***myDatabase*** database:

```
> db.basket.drop()
```

```
> db.dropDatabase()
```

14. Exit the shell and return to the system prompt using the **exit** command:

```
> exit
```

Task2: Importing Data into MongoDB

In this task you will use the ***mongoimport*** utility to import JSON text data into MongoDB. As mentioned at the beginning of this document, if you have installed MongoDB Community Server version 4.4 or newer, you should also download and install MongoDB Database Tools to use *mongoimport*.

You can refer to the MongoDB document (for version 3.4) at:

https://docs.mongodb.com/v3.4/reference/program/mongoimport/

If you are using a recent MongoDB (version after 4.0), the document for *mongoimport* can be find at:

https://docs.mongodb.com/database-tools/mongoimport/

1. In your web browser, find the MongoDB "***restaurants***" sample database at the following URL, and save it to a file named ***primer-dataset.json*** in your home directory.

   https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json

2. Enter the following command **at the system prompt** to run the ***mongoimport*** utility from the system command line and import this documents into the ***restaurants*** collection in the database ***test*** (again you may need to include the full path to the bin directory):

```
mongoimport --db test --collection restaurants --drop
--file primer-dataset.json
```

3. Check the response and find out how many documents were imported.

   What do you think the purpose of the "***--drop***" option is?

4. Start the mongo shell again using the **mongo** command, then make sure you are using the ***test*** database. Enter a command to show the documents in the ***restaurants*** collection. Note that the shell shows 20 documents at a time.

5. Enter a command to find the details of the restaurant with the *name* "*Taste The Tropics Ice Cream*".
   (Hint: you should specify the "query filter" in find function.)

   What borough is it in? Which grades has it?
   (Hint: specify the "projection document" in find function to include these fields in the result.)

Task 3: Querying MongoDB

In this task you will devise and run some queries to extract specific data from the ***restaurants*** and ***movieDetails*** data sets. You can refer to the following two snapshots for the data models for these two data sets.


```
{
        "_id" : ObjectId("59e565aa8b8edcdd5b7a43bc"),
        "address" : {
                "building" : "469",
                "coord" : [
                        -73.961704,
                        40.662942
                ],
                "street" : "Flatbush Avenue",
                "zipcode" : "11225"
        },
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "grades" : [
                {
                        "date" : ISODate("2014-12-30T00:00:00Z"),
                        "grade" : "A",
                        "score" : 8
                },
                {
                        "date" : ISODate("2014-07-01T00:00:00Z"),
                        "grade" : "B",
                        "score" : 23
                },
                {
                        "date" : ISODate("2013-04-30T00:00:00Z"),
                        "grade" : "A",
                        "score" : 12
                },
                {
                        "date" : ISODate("2012-05-08T00:00:00Z"),
                        "grade" : "A",
                        "score" : 12
                }
        ],
        "name" : "Wendy'S",
        "restaurant_id" : "30112340"
}
```


Figure 1: snapshot of a document in the *restaurants* collection

```
{
        "_id" : ObjectId("59ee697aaa1d2db3dc4fbfa5"),
        "title" : "Star Wars: Episode II - Attack of the Clones",
        "year" : 2002,
        "rated" : "PG",
        "runtime" : 142,
        "countries" : [
                "USA"
        ],
        "genres" : [
                "Action",
                "Adventure",
                "Fantasy"
        ],
        "director" : "George Lucas",
        "writers" : [
                "George Lucas",
                "Jonathan Hales",
                "George Lucas"
        ],
        "actors" : [
                "Ewan McGregor",
                "Natalie Portman",
                "Hayden Christensen",
                "Christopher Lee"
        ],
        "plot" : "Ten years after initially meeting, Anakin Skywalker shares a f
orbidden romance with Padmé, while Obi-Wan investigates an assassination attempt
 on the Senator and discovers a secret clone army crafted for the Jedi.",
        "poster" : "http://ia.media-imdb.com/images/M/MV5BMTY5MjI5NTIwN15BM15Ban
BnXkFtZTYwMTM1Njg2._V1_SX300.jpg",
        "imdb" : {
                "id" : "tt0121765",
                "rating" : 6.7,
                "votes" : 425728
        },
        "tomato" : {
                "meter" : 66,
                "image" : "fresh",
                "rating" : 6.7,
                "reviews" : 242,
                "fresh" : 159,
                "consensus" : "Star Wars Episode II: Attack of the Clones benefi
ts from an increased emphasis on thrilling action, although they're once again u
ndercut by ponderous plot points and underdeveloped characters.",
                "userMeter" : 58,
                "userRating" : 3.3,
                "userReviews" : 844634
        },
        "metacritic" : 54,
        "awards" : {
                "wins" : 13,
                "nominations" : 47,
                "text" : "Nominated for 1 Oscar. Another 13 wins & 47 nomination
s."
        },
        "type" : "movie"
}
```


Figure 2: snapshot of a document in the *movieDetails* collection

1. Download the ***movieDetailsData.js*** file from GCULearn and import it into the ***test*** database as ***movieDetails*** collection by execute the .js file using **load** in mongo shell:

```
> load("movieDetailsData.js")
```

The ***restaurants*** collection has been created in task 2.

You should create and test queries to retrieve the data specified in each of the following exercises. You can refer to your lecture notes and the MongoDB manual:

https://docs.mongodb.com/v3.4/tutorial/query-documents/
https://docs.mongodb.com/v3.4/reference/operator/query/

For each exercise the expected output is shown, and unless stated otherwise it is the complete output you should get for your query (it's quite long in some cases!)

You can create a document and copy each of your completed queries in it as a record of your work.

2. Find how many different types of cuisine are represented in the restaurants data. (Hint: using the **distinct** method)

```
85
```

3. List the address of all the **Hawaiian** restaurants (there are 3)
(Hint: specify the query filter and projection document in the **find** method)

```
{
	"address" : {
		"building" : "2245",
		"coord" : [
			-73.93600099999999,
			40.795675
		],
		"street" : "1 Avenue",
		"zipcode" : "10029"
	},
	"cuisine" : "Hawaiian"
}
{
	"address" : {
		"building" : "360",
		"coord" : [
			-73.9854351,
			40.7419304
		],
		"street" : "Park Avenue South",
		"zipcode" : "10010"
	},
	"cuisine" : "Hawaiian"
}
{
	"address" : {
		"building" : "84",
		"coord" : [
			-73.9562856,
			40.7139315
		],
		"street" : "Havemeyer Street",
		"zipcode" : "11211"
	},
	"cuisine" : "Hawaiian"
}
```

4. List the title and awards of movies directed by “**Sergio Leone**” (there are 2)
(Hint: specify the query filter and projection document in the **find** method)


```
{
	"title" : "Once Upon a Time in the West",
	"director" : "Sergio Leone",
	"awards" : {
		"wins" : 4,
		"nominations" : 5,
		"text" : "4 wins & 5 nominations."
	}
}
{
	"title" : "Once Upon a Time in America",
	"director" : "Sergio Leone",
	"awards" : {
		"wins" : 11,
		"nominations" : 7,
		"text" : "Nominated for 2 Golden Globes. Another 11 wins & 7 nominations."
	}
}
```


5. Find the title and countries of movies released before the year of **1897**. (there are 2)
(Hint: using the **$lt** operator on the year field)


```
{
	"title" : "L.L. M.M. le Tsar et la Tsarine entrant dans l'église de l'As
somption [Moscou]",
	"year" : 1896,
	"countries" : [
		"France"
	]
}
{
	"title" : "Monza, L.L. M.M. le Roi et la Reine d'Italie",
	"year" : 1896,
	"countries" : [
		"France"
	]
}
```


6. Find the details of the latest movie directed by “**George Lucas**”, omit the poster information in the result
(Hint: using **sort()** to sort the results and **limit()** to specify how many docuemnts to return)


```
{
	"_id" : ObjectId("59ee697aaa1d2db3dc4fbfa3"),
	"title" : "Star Wars: Episode III - Revenge of the Sith",
	"year" : 2005,
	"rated" : "PG-13",
	"runtime" : 140,
	"countries" : [
		"USA"
	],
```

```
        "genres" : [
                "Action",
                "Adventure",
                "Fantasy"
        ],
        "director" : "George Lucas",
        "writers" : [
                "George Lucas"
        ],
        "actors" : [
                "Ewan McGregor",
                "Natalie Portman",
                "Hayden Christensen",
                "Ian McDiarmid"
        ],
        "plot" : "Three years after the onset of the Clone Wars; the noble Jedi
Knights are spread out across the galaxy leading a massive clone army in the war
 against the Separatists. After Chancellor ...",
        "imdb" : {
                "id" : "tt0121766",
                "rating" : 7.6,
                "votes" : 483613
        },
        "tomato" : {
                "meter" : 79,
                "image" : "certified",
                "rating" : 7.3,
                "reviews" : 282,
                "fresh" : 222,
                "consensus" : "With Episode III: Revenge of the Sith, George Luc
as brings his second Star Wars trilogy to a suitably thrilling and often poignan
t -- if still a bit uneven -- conclusion.",
                "userMeter" : 65,
                "userRating" : 3.1,
                "userReviews" : 33674396
        },
        "metacritic" : 68,
        "awards" : {
                "wins" : 21,
                "nominations" : 49,
                "text" : "Nominated for 1 Oscar. Another 21 wins & 49 nomination
s."
        },
        "type" : "movie"
}
```


7. Find all Scandinavian restaurants in Manhattan, showing only the street and name of each restaurant.
   (Hint: specify the query filter and projection document in the **find** method)


```
{ "address" : { "street" : "Stone Street" }, "name" : "Smorgas Chef/ Crepes Du Nord" }
{ "address" : { "street" : "East 55 Street" }, "name" : "Aquavit" }
{ "address" : { "street" : "Park Avenue" }, "name" : "Smorgas Chef At Scandinavi
a House" }
{ "address" : { "street" : "Pearl Street" }, "name" : "Fika" }
{ "address" : { "street" : "Washington Street" }, "name" : "Fika" }
{ "address" : { "street" : "Laight Street" }, "name" : "Aamanns-Copenhgen" }
```

8. Find the details of the movies which have actors "**Nicolas Cage**" and "**Dennis Hopper**". (there is 1)
(Hint: using the **$all** operator on the actors field)


```
{
	"_id" : ObjectId("59ee697aaa1d2db3dc4fbf9b"),
	"title" : "Red Rock West",
	"year" : 1993,
	"rated" : "R",
	"runtime" : 98,
	"countries" : [
		"USA"
	],
	"genres" : [
		"Crime",
		"Drama",
		"Thriller"
	],
	"director" : "John Dahl",
	"writers" : [
		"John Dahl",
		"Rick Dahl"
	],
	"actors" : [
		"Nicolas Cage",
		"Dennis Hopper",
		"Lara Flynn Boyle",
		"J.T. Walsh"
	],
	"plot" : "When a promised job for Texan Michael fails to materialise in
Wyoming, Mike is mistaken by Wayne to be the hitman he hired to kill his unfaith
ful wife, Suzanne. Mike takes full advantage of...",
	"poster" : "http://ia.media-imdb.com/images/M/MV5BMTk0Mjg4OTc3OF5BMl5Ban
BnXkFtZTYwNzgzOTg5._V1_SX300.jpg",
	"imdb" : {
		"id" : "tt0105226",
		"rating" : 7,
		"votes" : 15007
	},
	"awards" : {
		"wins" : 0,
		"nominations" : 3,
		"text" : "3 nominations."
	},
	"type" : "movie"
}
```


9. Find out how many movies include "**Jim Thomas**" or "**Flint Dille**" as one of the writers
(Hint: using the **$in** operator on the writers field)

```
3
```

10. List the names of all restaurants with a grade dated **2011-03-03**.
    [Hint: use ISODate("2011-03-03T00:00:00Z") to specify the date]

```
{ "name" : "The Assembly Bar" }
{ "name" : "Benny'S Famous Pizza" }
{ "name" : "Ray'S Pizza Restuarant" }
{ "name" : "La Piazza Pizzeria & Restaurant" }
{ "name" : "Kennedy Fried Chicken" }
{ "name" : "Barbara Blum Residence / Good Shepherd Services" }
{ "name" : "Le Pain Quotidien" }
{ "name" : "Blimpie" }
```

    Challenge: use the $*elemMatch* operator in the projection document to include the grade with that date, and no other grades. Only the first document in the desired result is shown here:

```
{
        "grades" : [
                {
                        "date" : ISODate("2011-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                }
        ],
        "name" : "The Assembly Bar"
}
```

11. Find all Irish restaurants that have at least one for each of the grades A, B, C and Z.
    (Hint: using the **$all** operator on the grade field)

```
{
        "grades" : [
                {
                        "grade" : "Z"
                },
                {
                        "grade" : "A"
                },
                {
                        "grade" : "A"
                },
                {
                        "grade" : "C"
                },
                {
                        "grade" : "B"
                }
        ],
        "name" : "Niall'S On 52Nd"
}
```