MongoDB CRUD

Exercise 1 - Getting the environment ready

```
theia@theiadocker-craigtrupp8:/home/project$ start_mongo
Starting your mongodb database....
This process can take up to a minute.

Mongodb started, waiting for all services to be ready....

Your mongodb server is now ready to use and available with username: root
password: MzEyOTktY3JhaWd0

You can access your mongodb database via:
 • The browser at:
https://craigtrupp8-8081.theiadocker-0-labs-prod-theiak8s-4-tor01.proxy.cog
nitiveclass.ai
 • CommandLine: mongo -u root -p MzEyOTktY3JhaWd0 --authenticationDatabase
admin local
theia@theiadocker-craigtrupp8:/home/project$ mongo -u root -p
MzEyOTktY3JhaWd0 --authenticationDatabase admin local
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/local
MongoDB server version: 3.6.3
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2023-10-13T21:35:27.439+0000 I STORAGE  [initandlisten]
2023-10-13T21:35:27.439+0000 I STORAGE  [initandlisten] ** WARNING: Using
the XFS filesystem is strongly recommended with the WiredTiger storage
engine
2023-10-13T21:35:27.439+0000 I STORAGE  [initandlisten] **          See
http://dochub.mongodb.org/core/prodnotes-filesystem
2023-10-13T21:35:28.417+0000 I CONTROL  [initandlisten]
2023-10-13T21:35:28.418+0000 I CONTROL  [initandlisten] ** WARNING: You are
running on a NUMA machine.
```

```
2023-10-13T21:35:28.418+0000 I CONTROL  [initandlisten] **          We
suggest launching mongod like this to avoid performance problems:
2023-10-13T21:35:28.418+0000 I CONTROL  [initandlisten] **
numactl --interleave=all mongod [other options]
2023-10-13T21:35:28.418+0000 I CONTROL  [initandlisten]
> SHOW DATABASES;
2023-10-13T17:36:49.811-0400 E QUERY    [thread1] SyntaxError: missing ;
before statement @(shell):1:5
> show dbs;
admin  0.000GB
local  0.000GB
> use training;
switched to db training
> db.createCollections('languages');
2023-10-13T17:37:39.021-0400 E QUERY    [thread1] TypeError:
db.createCollections is not a function :
@(shell):1:1
> db.createCollection('language');
{ "ok" : 1 }
> show collections
language
```

Exercise 2 - Insert documents

Let us insert five documents into the collection languages

```
> language_list = [{"name":"java", "type":"object oriented"},
{"name":"python", "type":"general purpose"}, {"name":"scala",
"type":"functional"}, {"name":"c", "type":"procedural"}, {"name":"c++",
"type":"object oriented"}]
[
        {
                "name" : "java",
                "type" : "object oriented"
        },
        {
                "name" : "python",
                "type" : "general purpose"
        },
        {
```

```
                "name" : "scala",
                "type" : "functional"
        },
        {
                "name" : "c",
                "type" : "procedural"
        },
        {
                "name" : "c++",
                "type" : "object oriented"
        }
]
> cd.insertMany(language_list)
2023-10-13T17:40:50.529-0400 E QUERY    [thread1] TypeError: cd.insertMany
is not a function :
@(shell):1:1
> db.insertMan(language_list)
2023-10-13T17:40:57.702-0400 E QUERY    [thread1] TypeError: db.insertMan
is not a function :
@(shell):1:1
> db.insertMany(language_list)
2023-10-13T17:41:08.761-0400 E QUERY    [thread1] TypeError: db.insertMany
is not a function :
@(shell):1:1
> db.language.insertMany(language_list)
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6529b988d98f7653794885ca"),
                ObjectId("6529b988d98f7653794885cb"),
                ObjectId("6529b988d98f7653794885cc"),
                ObjectId("6529b988d98f7653794885cd"),
                ObjectId("6529b988d98f7653794885ce")
        ]
}
```

- Had a fun few errors but finally created lol

Exercise 3 - Read document

- Querying Documents for random like select statements w/filters and using field values to pull certain records

```
> db.language.count()
5
> db.language.findOne()
{
        "_id" : ObjectId("6529b988d98f7653794885ca"),
        "name" : "java",
        "type" : "object oriented"
}
> db.language.find()
{ "_id" : ObjectId("6529b988d98f7653794885ca"), "name" : "java", "type" :
"object oriented" }
{ "_id" : ObjectId("6529b988d98f7653794885cb"), "name" : "python", "type" :
"general purpose" }
{ "_id" : ObjectId("6529b988d98f7653794885cc"), "name" : "scala", "type" :
"functional" }
{ "_id" : ObjectId("6529b988d98f7653794885cd"), "name" : "c", "type" :
"procedural" }
{ "_id" : ObjectId("6529b988d98f7653794885ce"), "name" : "c++", "type" :
"object oriented" }
> db.language.find().limit(3)
{ "_id" : ObjectId("6529b988d98f7653794885ca"), "name" : "java", "type" :
"object oriented" }
{ "_id" : ObjectId("6529b988d98f7653794885cb"), "name" : "python", "type" :
"general purpose" }
{ "_id" : ObjectId("6529b988d98f7653794885cc"), "name" : "scala", "type" :
"functional" }
> db.language.find({"name":"python"})
{ "_id" : ObjectId("6529b988d98f7653794885cb"), "name" : "python", "type" :
"general purpose" }
> db.language.find({"type":"object oriented"})
{ "_id" : ObjectId("6529b988d98f7653794885ca"), "name" : "java", "type" :
"object oriented" }
{ "_id" : ObjectId("6529b988d98f7653794885ce"), "name" : "c++", "type" :
"object oriented" }
```

List only specific fields.

Using a projection document you can specify what fields we wish to see or skip in the output.

- This command lists all the documents with only **name** field in the output.

```
> db.language.find({}, {"name":1})
{ "_id" : ObjectId("6529b988d98f7653794885ca"), "name" : "java" }
{ "_id" : ObjectId("6529b988d98f7653794885cb"), "name" : "python" }
{ "_id" : ObjectId("6529b988d98f7653794885cc"), "name" : "scala" }
{ "_id" : ObjectId("6529b988d98f7653794885cd"), "name" : "c" }
{ "_id" : ObjectId("6529b988d98f7653794885ce"), "name" : "c++" }
```

- This command lists all the documents without the name field in the output.
    - Recall the object we made only had two properties for each object we created in the collection

```
> db.language.find({}, {"name":0})
{ "_id" : ObjectId("6529b988d98f7653794885ca"), "type" : "object oriented"
}
{ "_id" : ObjectId("6529b988d98f7653794885cb"), "type" : "general purpose"
}
{ "_id" : ObjectId("6529b988d98f7653794885cc"), "type" : "functional" }
{ "_id" : ObjectId("6529b988d98f7653794885cd"), "type" : "procedural" }
{ "_id" : ObjectId("6529b988d98f7653794885ce"), "type" : "object oriented"
}
```

- This command lists all the “object oriented” languages with only “name” field in the output.
    - 0 and 1 like a boolean for the property visibility when using the find command

```
> db.language.find({"type":"object oriented"}, {"name":1})
{ "_id" : ObjectId("6529b988d98f7653794885ca"), "name" : "java" }
{ "_id" : ObjectId("6529b988d98f7653794885ce"), "name" : "c++" }
```

Exercise 4 - Update documents

Update documents based on a criteria.

Add a field to all the documents.

The ‘updateMany’ command is used to update documents in a mongodb collection, and it has the following generic syntax.

- Syntax

```
db.collection.updateMany({what documents to find},{$set:{what fields to set}})
```

- Here we are adding a field description with value programming language to all the documents.

```
> db.language.updateMany({}, {$set:{"description":"programming language"}})
{ "acknowledged" : true, "matchedCount" : 5, "modifiedCount" : 5 }
```

- Set the creator for python language.

```
> db.language.updateMany({"name":"python"}, {$set:{"creator":"Guido van Rossum"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

- Set a field named compiled with a value true for all the object oriented languages.

```
> db.language.updateMany({"type":"object oriented"},
{$set:{"compiled":true}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
```

- Review Updates

```
> db.language.find()
{ "_id" : ObjectId("6529b988d98f7653794885ca"), "name" : "java", "type" : "object oriented", "description" : "programming language", "compiled" : true }
{ "_id" : ObjectId("6529b988d98f7653794885cb"), "name" : "python", "type" : "general purpose", "description" : "programming language", "creator" : "Guido van Rossum" }
{ "_id" : ObjectId("6529b988d98f7653794885cc"), "name" : "scala", "type" : "functional", "description" : "programming language" }
```

```
{ "_id" : ObjectId("6529b988d98f7653794885cd"), "name" : "c", "type" :
"procedural", "description" : "programming language" }
{ "_id" : ObjectId("6529b988d98f7653794885ce"), "name" : "c++", "type" :
"object oriented", "description" : "programming language", "compiled" :
true }
```

Exercise 5 - Delete documents

- Delete documents based on a criteria.
- Delete the scala language document.

```
> db.language.remove({"name":"scala"})
WriteResult({ "nRemoved" : 1 })
```

- Delete the object oriented languages.

```
> db.language.remove({"type":"object oriented"})
WriteResult({ "nRemoved" : 2 })
```

- Let's validate the deletes worked

```
> db.language.find()
{ "_id" : ObjectId("6529b988d98f7653794885cb"), "name" : "python", "type" :
"general purpose", "description" : "programming language", "creator" :
"Guido van Rossum" }
{ "_id" : ObjectId("6529b988d98f7653794885cd"), "name" : "c", "type" :
"procedural", "description" : "programming language" }
```

- Delete all the documents in a collection.

```
> db.language.remove({})
WriteResult({ "nRemoved" : 2 })
> db.language.find()
```

Practice exercises

Run the below code on mongo console. It will insert 5 documents, which will serve as sample data for the next steps.

- Reset training collection and populate w/data

```
> use training
switched to db training
> db.languages.insert({"name":"java","type":"object oriented"})
WriteResult({ "nInserted" : 1 })
> db.languages.insert({"name":"python","type":"general purpose"})
WriteResult({ "nInserted" : 1 })
> db.languages.insert({"name":"scala","type":"functional"})
WriteResult({ "nInserted" : 1 })
> db.languages.insert({"name":"c","type":"procedural"})
WriteResult({ "nInserted" : 1 })
> db.languages.insert({"name":"c++","type":"object oriented"})
WriteResult({ "nInserted" : 1 })
> db.languages.find()
{ "_id" : ObjectId("6529bd2cd98f7653794885cf"), "name" : "java", "type" :
"object oriented" }
{ "_id" : ObjectId("6529bd2cd98f7653794885d0"), "name" : "python", "type" :
"general purpose" }
{ "_id" : ObjectId("6529bd2cd98f7653794885d1"), "name" : "scala", "type" :
"functional" }
{ "_id" : ObjectId("6529bd2cd98f7653794885d2"), "name" : "c", "type" :
"procedural" }
{ "_id" : ObjectId("6529bd2ed98f7653794885d3"), "name" : "c++", "type" :
"object oriented" }
>
```

- Insert an entry for ‘Haskell’ programming language which is of type ‘functional’ .

```
> db.languages.find()
{ "_id" : ObjectId("6529bd86d98f7653794885d4"), "name" : "java", "type" :
"object oriented" }
{ "_id" : ObjectId("6529bd86d98f7653794885d5"), "name" : "python", "type" :
"general purpose" }
{ "_id" : ObjectId("6529bd86d98f7653794885d6"), "name" : "scala", "type" :
"functional" }
{ "_id" : ObjectId("6529bd86d98f7653794885d7"), "name" : "c", "type" :
"procedural" }
```

```
{ "_id" : ObjectId("6529bd88d98f7653794885d8"), "name" : "c++", "type" :
"object oriented" }
> db.languages.insertOne({"name":"Haskell", "type":"functional"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6529be9ed98f7653794885d9")
}
```

- Query for all functional languages

```
> db.languages.find({"type":"functional"})
{ "_id" : ObjectId("6529bd86d98f7653794885d6"), "name" : "scala", "type" :
"functional" }
{ "_id" : ObjectId("6529be9ed98f7653794885d9"), "name" : "Haskell", "type"
: "functional" }
```

- We can also narrow the search to just include the name or type field with our query

```
> db.languages.find({"type":"functional"}, {"name":1})
{ "_id" : ObjectId("6529bd86d98f7653794885d6"), "name" : "scala" }
{ "_id" : ObjectId("6529be9ed98f7653794885d9"), "name" : "Haskell" }
> db.languages.find({"type":"functional"}, {"name":0})
{ "_id" : ObjectId("6529bd86d98f7653794885d6"), "type" : "functional" }
{ "_id" : ObjectId("6529be9ed98f7653794885d9"), "type" : "functional" }
```

    - Equivalent of select certain columns for a row or tuple of data

- Add ‘Bjarne Stroustrup’ as creator for c++.

```
> db.languages.updateMany({"name":"c++"}, {$set : {"creator": "Bjarne
Stroustrup"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.languages.find({"name":"c++"})
{ "_id" : ObjectId("6529bd88d98f7653794885d8"), "name" : "c++", "type" :
"object oriented", "creator" : "Bjarne Stroustrup" }
```

- Delete all functional programming languages.

```
> db.languages.remove({"type":"functional"})
WriteResult({ "nRemoved" : 2 })
> db.languages.find()
{ "_id" : ObjectId("6529bd86d98f7653794885d4"), "name" : "java", "type" :
"object oriented" }
{ "_id" : ObjectId("6529bd86d98f7653794885d5"), "name" : "python", "type" :
"general purpose" }
{ "_id" : ObjectId("6529bd86d98f7653794885d7"), "name" : "c", "type" :
"procedural" }
{ "_id" : ObjectId("6529bd88d98f7653794885d8"), "name" : "c++", "type" :
"object oriented", "creator" : "Bjarne Stroustrup" }
```

- Disconnect from the mongodb server.

```
> exit
bye
theia@theiadocker-craigtrupp8:/home/project$
```