Week2_MongoDB_commands

General

`start_mongo`

`mongo -u root -p MTY3NTMtam9hb3Bh --authenticationDatabase admin local`

`db.version()`

`show dbs`

`use training`

`db.createCollection("mycollection")`

`show collections`

`db.mycollection.insert({"color":"white","example":"milk"})`

`db.mycollection.insert({"color":"blue","example":"sky"})`

`db.mycollection.count()`

`db.mycollection.find()`

`exit`

CRUD

Create and Read

```
use training
db.languages.insert({"name":"java","type":"object oriented"})
db.languages.insert({"name":"python","type":"general purpose"})
db.languages.insert({"name":"scala","type":"functional"})
db.languages.insert({"name":"c","type":"procedural"})
db.languages.insert({"name":"c++","type":"object oriented"})
```

`db.languages.find().limit(3)`

This command lists all the documents with only name field in the output.
`db.languages.find({},{"name":1})`
This command lists all the documents without the name field in the output.

`db.languages.find({},{"name":0})`
This command lists all the "object oriented" languages with only "name" field in the output.
`db.languages.find({"type":"object oriented"},{"name":1})`

Update
The 'updateMany' command is used to update documents in a mongodb collection, and it has the following generic 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.languages.updateMany({},{$set:{"description":"programming language"}})
```

Set the creater for python language.

```
db.languages.updateMany({"name":"python"},{$set:{"creator":"Guido van
Rossum"}})
```

set a field named `compiled` with a value `true` for all the `object oriented` languages

```
db.languages.updateMany({"type":"object oriented"},{$set:{"compiled":true}})
```

Delete

Delete documents based on a criteria.

Delete the scala language document.
`db.languages.remove({"name":"scala"})`

Delete the object oriented languages.
`db.languages.remove({"type":"object oriented"})`

Delete all the documents in a collection
`db.languages.remove({})`

Indexing

`db.createCollection("bigdata")`

Let us insert a lot of documents into the newly created collection.
This should take around 3 minutes, so please be patient.
The code given below will insert 200000 documents into the 'bigdata' collection.
Each document would have a field named account_no which is a simple auto increment number.

And a field named balance which is a randomly generated number, to simulate the bank balance for the account.

```
use training
for (i=1;i<=200000;i++)
{print(i);db.bigdata.insert({"account_no":i,"balance":Math.round(Math.random
()*1000000)})}
```

`db.bigdata.count()`

Let us run a query and find out how much time it takes to complete.

Let us query for the details of account number 58982.

We will make use of the explain function to find the time taken to run the query in milliseconds.

`db.bigdata.find({"account_no":58982}).explain("executionStats").executionStats.executionTimeMillis`

Before you create an index, choose the field you wish to create an index on. It is usually the field that you query most.

Run the below command to create an index on the field account_no.

```
db.bigdata.createIndex({"account_no":1})
```

`db.bigdata.getIndexes()`

```
db.bigdata.find({"account_no":
69271}).explain("executionStats").executionStats.executionTimeMillis
```

`db.bigdata.dropIndex({"account_no":1})`

Aggregation

```

use training
db.marks.insert({"name":"Ramesh","subject":"maths","marks":87})
db.marks.insert({"name":"Ramesh","subject":"english","marks":59})
db.marks.insert({"name":"Ramesh","subject":"science","marks":77})
db.marks.insert({"name":"Rav","subject":"maths","marks":62})
db.marks.insert({"name":"Rav","subject":"english","marks":83})
db.marks.insert({"name":"Rav","subject":"science","marks":71})
db.marks.insert({"name":"Alison","subject":"maths","marks":84})
db.marks.insert({"name":"Alison","subject":"english","marks":82})
db.marks.insert({"name":"Alison","subject":"science","marks":86})
db.marks.insert({"name":"Steve","subject":"maths","marks":81})
db.marks.insert({"name":"Steve","subject":"english","marks":89})
db.marks.insert({"name":"Steve","subject":"science","marks":77})
db.marks.insert({"name":"Jan","subject":"english","marks":0,"reason":"absent
"})
```

Using the `$limit` operator we can limit the number of documents printed in the output.
This command will print only 2 documents from the marks collection.

```
use training
db.marks.aggregate([{"$limit":2}])
```

We can use the $sort operator to sort the output.

This command sorts the documents based on field marks in ascending order.

`db.marks.aggregate([{"$sort":{"marks":1}}])`

This command sort the documents based on field marks in descending order.

`db.marks.aggregate([{"$sort":{"marks":-1}}])`

Aggregation usually involves using more than one operator.
A pipeline consists of one or more operators declared inside an array.
The operators are comma separated.
Mongodb executes the first operator in the pipeline and sends its output to the next operator.

Let us create a two stage pipeline that answers the question “What are the top 2 marks?”.

```
db.marks.aggregate([
{"$sort":{"marks":-1}},
{"$limit":2}
])


# Exercise 5 - Group by

#The operator $group by, along #with operators like $sum, #$avg, $min, $max,
allows us to #perform grouping operations.
#This aggregation pipeline #prints the average marks #across all subjects.


db.marks.aggregate([
{
    "$group":{
        "_id":"$subject",
        "average":{"$avg":"$marks"}
        }
}
])
```

The above query is equivalent to the below sql query.

```
SELECT subject, average(marks)
FROM marks
GROUP BY subject
```

Now let us put together all the operators we have learnt to answer the question. “Who are the top 2 students by average marks?”
This involves:

finding the average marks per student.
sorting the output based on average marks in descending order.
limiting the output to two documents.

```
db.marks.aggregate([
{
    "$group":{
        "_id":"$name",
        "average":{"$avg":"$marks"}
        }
},
{
    "$sort":{"average":-1}
},
{
    "$limit":2
}
])
```

Mongo DB and python

`python3 -m pip install pymongo`

mongo_connect.py

```

from pymongo import MongoClient
user = 'root'
```

```
password = 'MjQwOTgtcnNhbm5h' # CHANGE THIS TO THE PASSWORD YOU NOTED IN THE
EARLIER EXCERCISE - 2
host='localhost'
#create the connection url
connecturl = "mongodb://{}:{}@{}:27017/?
authSource=admin".format(user,password,host)
# connect to mongodb server
print("Connecting to mongodb server")
connection = MongoClient(connecturl)
# get database list
print("Getting list of databases")
dbs = connection.list_database_names()
# print the database names
for db in dbs:
    print(db)
print("Closing the connection to the mongodb server")
# close the server connecton
connection.close()
```

`python3 mongo_connect.py`

- connect to the mongodb server.
- select a database named training.
- select a collection named python.
- insert a sample document.
- query all the documents in the training database and python collection.
- close the connection to the server.

```
from pymongo import MongoClient
user = 'root'
password = 'MjQwOTgtcnNhbm5h' # CHANGE THIS TO THE PASSWORD YOU NOTED IN THE
EARLIER EXCERCISE - 2
host='localhost'
#create the connection url
connecturl = "mongodb://{}:{}@{}:27017/?
authSource=admin".format(user,password,host)

# connect to mongodb server
print("Connecting to mongodb server")
connection = MongoClient(connecturl)

# select the 'training' database
```

```
db = connection.training

# select the 'python' collection

collection = db.python

# create a sample document

doc = {"lab":"Accessing mongodb using python", "Subject":"No SQL Databases"}

# insert a sample document

print("Inserting a document into collection.")
db.collection.insert_one(doc)

# query for all documents in 'training' database and 'python' collection

docs = db.collection.find()

print("Printing the documents in the collection.")

for document in docs:
    print(document)

# close the server connecton
print("Closing the connection.")
connection.close()
```

- connect to the mongodb server.
- select a database named training.
- select a collection named mongodb_glossary.
- insert the following documents into the collection mongodb_glossary.
- {“database”:”a database contains collections”}
- {“collection”:”a collection stores the documents”}
- {“document”:”a document contains the data in the form of key value pairs.”}
- query and print all the documents in the training database and mongodb_glossary collection.
- close the connection to the server.

```
from pymongo import MongoClient
user = 'root'
password = 'MjQwOTgtcnNhbm5h' # CHANGE THIS TO THE PASSWORD YOU NOTED IN THE EARLIER EXCERCISE - 2
```

```
host='localhost'
#create the connection url
connecturl = "mongodb://{}:{}@{}:27017/?
authSource=admin".format(user,password,host)
# connect to mongodb server
print("Connecting to mongodb server")
connection = MongoClient(connecturl)
# select the 'training' database
db = connection.training
# select the 'python' collection
collection = db.mongodb_glossary
# create documents
doc1 = {"database":"a database contains collections"}
doc2 = {"collection":"a collection stores the documents"}
doc3 = {"document":"a document contains the data in the form or key value
pairs."}
# insert documents
print("Inserting documents into collection.")
db.collection.insert_one(doc1)
db.collection.insert_one(doc2)
db.collection.insert_one(doc3)
# query for all documents in 'training' database and 'python' collection
docs = db.collection.find()
print("Printing the documents in the collection.")
for document in docs:
    print(document)
# close the server connecton
print("Closing the connection.")
connection.close()
```