MongoDB Commands with MongoDB Atlas

Connect to the Atlas cluster:

```
mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"
```

username: m001-student

password: m001-mongodb-basics

Find Command

```
show dbs


use sample_training


show collections


db.zips.find({"state": "NY"})
```

`it` iterates through the cursor.

```
db.zips.find({"state": "NY"}).count()


db.zips.find({"state": "NY", "city": "ALBANY"})


db.zips.find({"state": "NY", "city": "ALBANY"}).pretty()
```

Insert New Documents

Step two: navigate to the database that we need:

```
use sample_training
```

Step three, get a random document from the collection:

```
db.inspections.findOne();
```

Step four, copy this random document, and try to insert in into the collection:

```
db.inspections.insert({
      "_id" : ObjectId("56d61033a378eccde8a8354f"),
      "id" : "10021-2015-ENFO",
      "certificate_number" : 9278806,
      "business_name" : "ATLIXCO DELI GROCERY INC.",
      "date" : "Feb 20 2015",
      "result" : "No Violation Issued",
      "sector" : "Cigarette Retail Dealer - 127",
      "address" : {
              "city" : "RIDGEWOOD",
              "zip" : 11385,
              "street" : "MENAHAN ST",
              "number" : 1712
         }
  })
```

```
db.inspections.insert({
      "id" : "10021-2015-ENFO",
      "certificate_number" : 9278806,
      "business_name" : "ATLIXCO DELI GROCERY INC.",
      "date" : "Feb 20 2015",
      "result" : "No Violation Issued",
      "sector" : "Cigarette Retail Dealer - 127",
      "address" : {
              "city" : "RIDGEWOOD",
              "zip" : 11385,
              "street" : "MENAHAN ST",
              "number" : 1712
         }
  })


db.inspections.find({"id" : "10021-2015-ENFO",
"certificate_number" : 9278806}).pretty()
```

Insert three test documents:

```
db.inspections.insert([ { "test": 1 }, { "test": 2 },
{ "test": 3 } ])
```

Insert three test documents but specify the `_id` *values:*

```
db.inspections.insert([{ "_id": 1, "test": 1 },{ "_id": 1,
"test": 2 },

                       { "_id": 3, "test": 3 }])
```

Find the documents with `_id: 1`

```
db.inspections.find({ "_id": 1 })
```

Insert multiple documents specifying the `_id` *values, and using the* `"ordered": false` *option.*

```
db.inspections.insert([{ "_id": 1, "test": 1 },{ "_id": 1,
"test": 2 },

                       { "_id": 3, "test": 3 }],{ "ordered":
false })
```

Insert multiple documents with `_id: 1` *with the default* `"ordered": true` *setting*

```
db.inspection.insert([{ "_id": 1, "test": 1 },{ "_id": 3,
"test": 3 }])
```

View `collections` *in the active* `db`

```
show collections
```

Switch the active `db` *to* `training`

```
use training
```

View all available databases

```
show dbs
```

Updating Document

Use the `sample_training` *database as your database in the following commands.*

```
use sample_training
```

Find all documents in the `zips` *collection where the* `zip` *field is equal to* `"12434"`.

```
db.zips.find({ "zip": "12534" }).pretty()
```

Find all documents in the `zips` *collection where the* `city` *field is equal to* `"HUDSON"`.

```
db.zips.find({ "city": "HUDSON" }).pretty()
```

Find how many documents in the `zips` *collection have the* `city` *field equal to* `"HUDSON"`.

```
db.zips.find({ "city": "HUDSON" }).count()
```

Update all documents in the `zips` *collection where the* `city` *field is equal to* `"HUDSON"` *by adding* `10` *to the current value of the* `"pop"` *field.*

```
db.zips.updateMany({ "city": "HUDSON" }, { "$inc": { "pop":
10 } })
```

Update a single document in the `zips` *collection where the* `zip` *field is equal to* `"12534"` *by setting the value of the* `"pop"` *field to* `17630`.

```
db.zips.updateOne({ "zip": "12534" }, { "$set": { "pop":
17630 } })
```

Update a single document in the `zips` *collection where the* `zip` *field is equal to* `"12534"` *by setting the value of the* `"popupation"` *field to* `17630`.

```
db.zips.updateOne({ "zip": "12534" }, { "$set":
{ "population": 17630 } })
```

Find all documents in the `grades` *collection where the* `student_id` *field is* `151` *, and the* `class_id` *field is* `339`*.*

```
db.grades.find({ "student_id": 151, "class_id":
339 }).pretty()
```

Find all documents in the `grades` *collection where the* `student_id` *field is* `250` *, and the* `class_id` *field is* `339`*.*

```
db.grades.find({ "student_id": 250, "class_id":
339 }).pretty()
```

Update one document in the `grades` *collection where the* `student_id` *is ``250`` *, and the* `class_id` *field is* `339` *, by adding a document element to the* `"scores"` *array.*

```
db.grades.updateOne({ "student_id": 250, "class_id": 339 },

                    { "$push": { "scores": { "type": "extra
credit",

                                             "score": 100 }

                                 }

                     })
```

Deleting Documents and Collections

```
use sample_training
```

Look at all the docs that have `test` *field equal to* `1`.

```
db.inspections.find({ "test": 1 }).pretty()
```

Look at all the docs that have `test` *field equal to* `3`.

```
db.inspections.find({ "test": 3 }).pretty()
```

Delete all the documents that have `test` *field equal to* `1`.

```
db.inspections.deleteMany({ "test": 1 })
```

Delete one document that has `test` *field equal to* `3`.

```
db.inspections.deleteOne({ "test": 3 })
```

Inspect what is left of the `inspection` *collection.*

```
db.inspection.find().pretty()
```

View what collections are present in the `sample_training` *collection.*

```
show collections
```

Drop the `inspection` *collection.*

```
db.inspection.drop()
```

Query Operators - Comparison

Switch to this database:

```
use sample_training
```

Find all documents where the `tripduration` *was less than or equal to* `70` *seconds and the* `usertype` *was not* `Subscriber`:

```
db.trips.find({ "tripduration": { "$lte" : 70 },
                "usertype": { "$ne":
"Subscriber" } }).pretty()
```

Find all documents where the `tripduration` *was less than or equal to* `70` *seconds and the* `usertype` *was* `Customer` *using a redundant equality operator:*

```
db.trips.find({ "tripduration": { "$lte" : 70 },
                "usertype": { "$eq": "Customer" }}).pretty()
```

Find all documents where the `tripduration` *was less than or equal to* `70` *seconds and the* `usertype` *was* `Customer` *using the implicit equality operator:*

```
db.trips.find({ "tripduration": { "$lte" : 70 },
                "usertype": "Customer" }).pretty()
```

Query Operator

Find all documents where airplanes `CR2` *or* `A81` *left or landed in the* `KZN` *airport:*

```
db.routes.find({ "$and": [ { "$or" :[ { "dst_airport":
"KZN" },
                                     { "src_airport": "KZN" }
                                   ] },
                            { "$or" :[ { "airplane": "CR2" },
                                       { "airplane":
"A81" } ] }
                           ]}).pretty()
```