Mongo Quick Reference

How to log in to your database

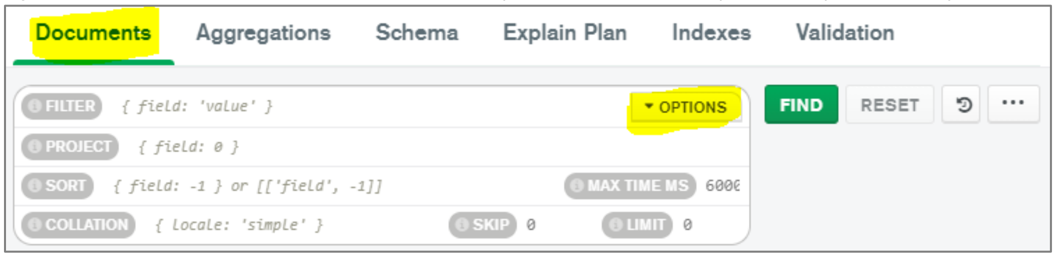
1. Login into your MongoDB Atlas account
2. Navigate to your project and then to Database Deployments.
3. Click **Connect**
4. Select the appropriate option to get your connection syntax or string. For the Compass connection string, you'll have to update it to include your database password and

Mongo Syntax

- Use this quick reference guide to figure out syntax

Mongo Compass

- To add filters, sorts, and projections (i.e. the equivalent of select specific columns but for fields in this case) click ***Options*** under the ***Documents*** tab in Compass. This will dropdown options for you to include these.



- To add aggregates, counts, and other functions to your Mongo query, click on Aggregations to select from a dropdown of options.

In-Class MongoDB Exercise

Just observe this as a demo by the Instructor

1. Login to mongo command line. *Note: This only works if mongo is installed on your client machine.* If interested in learning how to install MongoDB locally use this link.

2. Start mongo and list out all the databases using the following command: ***show dbs***

Mongo Command	Description
`show dbs`	Shows a list of all databases

3. Create a database called students

Mongo Command	Description
`use studentDB`	Creates the database called students

4. Insert a new document into students collection using a simple insert command

Mongo Command	Description
`db.students.insert({ fname:"Melissa", lname:"Wijono", email:"melissaw@mail.com", age: 21 })`	Creates the db and then Inserts a document for Tommy

5. Run ***show dbs*** command again to see that ***student db*** is now created and the ***use studentDB*** to access it

6. Pull all rows from the db using the

Mongo Command	Description
`db.students.find({})`	The mongo equivalent of the SQL statement: Select * from students.

7. Insert multiple records for more students using the insertMany command

Mongo Command	Description
`db.students.insertMany([ {fname:"Lisa", lname:"Desai", email:"lisad@ymail.com",age:20}, {fname:"Tito", lname:"Jackson", email:"tjackson@email.com"} ])`	Creates the db and then Inserts a document for Lisa and Tito. Tito has no specified age.

8. View all rows in command line and then again in MongoDB Compass which is the equivalent of SQLDeveloper (i.e. GUI-based Developer Tool) but used for json databases.
   a. Review the data json format.
   b. Use MongoDB Compass Table *View* to see data in table format. Draw context between json and relational table format.

9. Add a new student with a middle name and then look at the data in json and table formats
```
db.students.insert({fname:"John",midname:"Clinton", lname:"Tuttle", age: 41 })
```

10. Add a new students with just a *name*, *email*, and *currentAge*. Look at data in json and table formats
```
db.students.insert({name:"Clint Tuttle ", contact:"clint@gmail.com", currentage: 41 })
```

11. Next it's time to play around a little bit and do some basic querying with Mongo.

Once we're to this point turn the page over for the back portion.

Hands-on hacking / practice

1. Run the mongo command to show all your databases. `show dbs`
2. Run the command to use the ***sample_mflix*** database: `use sample_mflix`
3. Run the command to use show all the collections in this database: `show collections`
4. Run a command in Mongo syntax equivalent to the SQL command below and on the left.

Instructions	Mongo Command

SELECT *
 FROM movies;
	`db.movies.find()`

5. Run a command in Mongo syntax equivalent to the SQL command below and on the left.

Instructions	Mongo Command

SELECT count(*)
 FROM movies;
	db.movies.find().count()

6. Run a command in Mongo syntax equivalent to the SQL command below and on the left.

Instructions	Mongo Command

SELECT *
 FROM movies
 WHERE title = "Pulp Fiction";
	db.movies.find({title: "Pulp Fiction"})

7. Run a command in Mongo syntax equivalent to the SQL command below and on the left.

Instructions	Mongo Command

SELECT count(*)
 FROM movies
 WHERE year = 1994;
	db.movies.find({year: 1994}).count()

8. Run a command in Mongo syntax equivalent to the SQL command below and on the left. Tip: Use the $in operator

Instructions	Mongo Command

SELECT count(*)
 FROM movies
 WHERE year in (1994, 1995);
	db.movies.find({year: {$in: [1994,1995]} }).count()

9. Run a command in Mongo syntax equivalent to the SQL command below and on the left. Note, you'll need to add a **sort** to the query. Also to fetch only the first few rows you need to use the ***limit*** operator from MongoDB syntax. You

Instructions	Mongo Command

SELECT *
 FROM movies
 WHERE year = 1994
 ORDER BY awards.wins desc
 FETCH FIRST 3 ROWS ONLY;
	db.movies.find({year: 1994}).sort({"awards.wins": -1}).limit(3)

10. Run a command in Mongo syntax equivalent to the SQL command below and on the left. Tip. Since cast is actually an array that stores many actors, you'll need to google how to "***query an array***" in MongoDB using something call the **$all** operator

Instructions	Mongo Command

`SELECT count(*)`
`FROM movies`
`WHERE cast like '%Tom Hanks%';`
	db.movies.find({cast: {$all: ['Tom Hanks']}}).count()

`SELECT count(*)`
`FROM movies`
`WHERE cast like '%Tom Hanks%'`
`ORDER BY year asc`
`FETCH FIRST 1 ROW ONLY;`
	db.movies.find({cast: {$all: ['Tom Hanks']}}).sort({"year": 1}).limit(1)

11. Run a command in Mongo syntax equivalent to the SQL command below and on the left. Tip. To select specific fields in Mongo, we call this a ***projection***.

Instructions	Mongo Command

`SELECT title`
`FROM movies`
`WHERE awards.wins > 150;`
	db.movies.find({"awards.wins": {$gt: 150}},projection={title: 1})

12. Here's a freebee to see how you can also aggregate (i.e. group) data. Thoughts on whether SQL is easier? Doing complex queries like this present the value behind using tools like MongoDB Compass…a perfect segue to #13.

```
db.movies.aggregate([{"$group": {'_id':{'genres': "$genres"}, 'count':{'$sum':1}}}, {'$sort':{'count':-1}}])
```

13. Now that you tried command line, go log into your MongoDB cluster using the GUI-based Compass IDE and see if you can repeat the following queries using the UI interface.
    a. #6 – All movies with a title of "Pulp Fiction"
    b. #8 – All movies made in the year 1994 or 1995
    c. #9 – Top 3 movies in 1994 based on awards.wins
    d. #12 – Movies titles of all movies that have an awards.wins greater than 150

Reflection Question:

1. With mongoDB you have benefits of less structural constraints and the ability to store whatever you want and call it whatever you like. While flexibility can be a good thing, how could it be a bad thing too?
    a. Allows you to change your data model rapidly if your design changes a lot or structure of data isn't known.
    b. One issue is that you can store data for different instances of an entity differently e.g. (different fields and with different field names). This would make querying and managed data more complex and difficult
2. What syntax do you think makes more sense and is easier to read/use?
    a. Just discuss. Some people enjoy SQL more while some enjoy mongo's syntax more. This really becomes a personal preference that a developer has to consider while also weighing pros and cons of using MongoDB. Aside from comfort, the use of Mongo should be based on your need to scale and how known your database's schema (i.e. design structure) is.