Mongo Quick Reference

# How to log in to your database

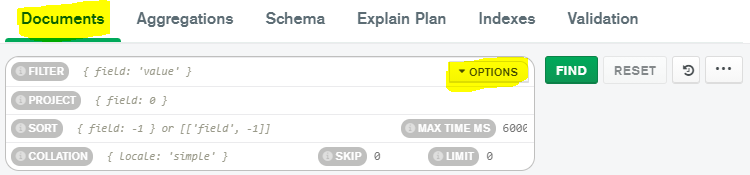
1. Login into your MongoDB Atlas account
2. Navigate to your project and then to Database Deployments.
3. Click **Connect**
4. Select the appropriate option to get your connection syntax or string. For the Compass connection string, you’ll have to update it to include your database password and

# Mongo Syntax

* [Use this quick reference guide to figure out syntax](https://docs.mongodb.com/manual/reference/mongo-shell/#command-helpers)

# 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](https://docs.mongodb.com/mongodb-shell/install/).
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 |

1. Create a database called students

|  |  |
| --- | --- |
| **Mongo Command** | **Description** |
| **use studentDB** | Creates the database called students |

1. 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 |

1. Run ***show dbs*** command again to see that ***student*** ***db*** is now created and the ***use studentDB*** to access it
2. Pull all rows from the db using the

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

1. 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. |

1. 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.
   1. Review the data json format.
   2. Use MongoDB Compass Table *View* to see data in table format. Draw context between json and relational table format.
2. 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 })**

1. 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 })**

1. 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:
2. Run the command to use the ***sample\_mflix*** database:
3. Run the command to use show all the collections in this database:
4. Run a command in Mongo syntax equivalent to the SQL command below and on the left.

|  |  |
| --- | --- |
| **Instructions** | **Mongo Command** |
| **SELECT \***  **FROM movies;** |  |

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

|  |  |
| --- | --- |
| **Instructions** | **Mongo Command** |
| **SELECT count(\*)**  **FROM movies;** |  |

1. 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”;** |  |

1. 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;** |  |

1. 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);** |  |

1. 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;** |  |

1. 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%’;** |  |
| **SELECT count(\*)**  **FROM movies**  **WHERE cast like ‘%Tom Hanks%’**  **ORDER BY year asc**  **FETCH FIRST 1 ROW ONLY;** |  |

1. 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;** |  |

1. 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}}])

1. 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.
   1. #6 – All movies with a title of “Pulp Fiction”
   2. #8 – All movies made in the year 1994 or 1995
   3. #9 – Top 3 movies in 1994 based on awards.wins
   4. #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?
   * +
   * -
2. What syntax do you think makes more sense and is easier to read/use?