MongoDB Cheat Sheet

By Web Dev Simplified https://courses.webdevsimplified.com

Terminology

Database	A container for collections. This is the same as a database in SQL and usually each project will have its own database full of different collections.
Collection	A grouping of documents inside of a database. This is the same as a table in SQL and usually each type of data (users, posts, products) will have its own collection.
Document	A record inside of a collection. This is the same as a row in SQL and usually there will be one document per object in the collection. A document is also essentially just a JSON object.
Field	A key value pair within a document. This is the same as a column in SQL. Each document will have some number of fields that contain information such as name, address, hobbies, etc. An important difference between SQL and MongoDB is that a field can contain values such as JSON objects, and arrays instead of just strings, number, booleans, etc.

Basic Commands

`mongosh`	Open a connection to your local MongoDB instance. All other commands will be run within this mongosh connection.
`show dbs`	Show all databases in the current MongoDB instance
`use <dbname>` `use myDatabase`	Switch to the database provided by dbname Switch to myDatabase
`db`	Show current database name
`cls`	Clear the terminal screen
`show collections`	Show all collections in the current database
`db.dropDatabase()`	Delete the current database
`exit`	Exit the mongosh session

Complex Update Object

Any combination of the below can be use inside an update object to make complex updates

$set

```
db.users.updateOne({ age: 12 }, { $set: { name: “Hi” } })
```

Update only the fields passed to $set. This will not affect any fields not passed to $set.

Update the name of the first user with the age of 12 to the value Hi

---

$inc

```
db.users.updateOne({ age: 12 }, { $inc: { age: 2 } })
```

Increment the value of the field by the amount given

Add 2 to the age of the first user with the age of 12

$rename

```
db.users.updateMany({}, { $rename: { age: “years” } })
```

Rename a field

Rename the field age to years for all users

$unset

```
db.users.updateOne({ age: 12 }, { $unset: { age: “” } })
```

Remove a field

Remove the age field from the first user with an age of 12

$push

```
db.users.updateMany({}, { $push: { friends: “John” } })
```

Add a value to an array field

Add John to the friends array for all users

$pull

```
db.users.updateMany({}, { $pull: { friends: “Mike” } })
```

Remove a value from an array field

Remove Mike from the friends array for all users

Read Modifiers

Any combination of the below can be added to the end of any read operation

sort

```
db.users.find().sort({ name: 1, age: -1 })
```

Sort the results of a find by the given fields

Get all users sorted by name in alphabetical order and then if any names are the same sort by age in reverse order

---

limit

```
db.users.find().limit(2)
```

Only return a set number of documents

Only return the first 2 users

---

skip

```
db.users.find().skip(4)
```

Skip a set number of documents from the beginning

Skip the first 4 users when returning results. This is great for pagination when combined with limit.