Interacting with MongoDB data from the command line

Follow on from last video where we created a database, inserted some documents and then retrieved them all again.

Start by looking at different ways to find documents and search the database, and then we'll look at updating and deleting documents.

Finding documents

Start by ensuring `mongod` is running - might be worth setting it to run on startup of your machine if you intend to use it a lot. Otherwise it's `mongod` in the command line.

Drop into the Mongo shell: `mongo`

Remind ourselves of the databases we have: `show dbs`

Use the `meantest` database: `use meantest`

Remind ourselves of the collections: `show collections`

find by id

The first type of query will return a single document based on the `_id`. So find an `ObjectId` value by a simple `find` on the collection: `db.tech.find()`. Copy an `_id` to the clipboard.

To find a document when you know its ID, you pass an object into the `find` query. The key of the pair is `_id` and the value is what you see in the Mongo shell. For example:

```
db.tech.find({"_id" : ObjectId("56b3d22468b2bfa5460d035b")})
```

And this will return back the single document:

```
{ "_id" : ObjectId("56b3d22468b2bfa5460d035b"), "name" : "MongoDB", "role" : "Database" }
```

find with query

This approach is not limited to finding IDs. It can work on with any property & value in a document For

example:

```
db.tech.find({ "name" : "Angular" })
```

Should return just the Angular document, like this:

```
{ "_id" : ObjectId("56b3d57168b2bfa5460d035d"), "name" : "Angular", "role" : "Front-end fra
```

Changing returned data

Sorting

You can change the order the results are returned by chaining a `sort` method to the `find` method, and passing it a parameters object.

This object contains the name of the property to sort by, and either `1` to sort ascending or `-1` to sort descending.

Ascending sort

```
db.tech.find().sort({"name": 1})
```

Returns:

```
{ "_id" : ObjectId("56b3d57168b2bfa5460d035d"), "name" : "Angular", "role" : "Front-end fra
{ "_id" : ObjectId("56b3d57168b2bfa5460d035c"), "name" : "Express", "role" : "Web applicati
{ "_id" : ObjectId("56b3d22468b2bfa5460d035b"), "name" : "MongoDB", "role" : "Database" }
{ "_id" : ObjectId("56b3d57168b2bfa5460d035e"), "name" : "Node.js", "role" : "Platform" }
```

Descending sort

```
db.tech.find().sort({"name": -1})
```

Returns

```
{ "_id" : ObjectId("56b3d57168b2bfa5460d035e"), "name" : "Node.js", "role" : "Platform" }
{ "_id" : ObjectId("56b3d22468b2bfa5460d035b"), "name" : "MongoDB", "role" : "Database" }
{ "_id" : ObjectId("56b3d57168b2bfa5460d035c"), "name" : "Express", "role" : "Web applicati
{ "_id" : ObjectId("56b3d57168b2bfa5460d035d"), "name" : "Angular", "role" : "Front-end fra
```

Returning specific fields

The act of returning just a specific set of fields from a query is known - in the world of MongoDB - as projection. The projection is added as a second parameter to the `find` method. So if you want to return all documents - which you normally leave blank in the `find` call - you first need to pass an empty object.

The projection parameter takes the name of field and a boolean value.

For example to find all documents, but only return the name:

```
db.tech.find({},{"name" : true})
```

Will return

```
{ "_id" : ObjectId("56b3d22468b2bfa5460d035b"), "name" : "MongoDB" }
{ "_id" : ObjectId("56b3d57168b2bfa5460d035c"), "name" : "Express" }
{ "_id" : ObjectId("56b3d57168b2bfa5460d035d"), "name" : "Angular" }
{ "_id" : ObjectId("56b3d57168b2bfa5460d035e"), "name" : "Node.js" }
```

Note: the `_id` field is automatically returned and has to be explicitly excluded if you don't want it.

```
db.tech.find({},{"name" : true, "_id" : 0})
```

Returns:

```
{ "name" : "MongoDB" }
{ "name" : "Express" }
{ "name" : "Angular" }
{ "name" : "Node.js" }
```

Note: excluding IDs is the only scenario in which you can combine include and exclude flags in the same parameter. Other than this, a projection must include either *only inclusions* or *only exclusions*.

Updating documents

To update documents you first have to find them, so that you can add or change existing data.

Updating a single document

To update an existing document you run an `update` method, in a similar way to the `find` method.

The `update` command starts off with two parameters. The first in the query to run to find a document, the second contains the data you want to add or update.

So to change the name of the Angular document from `Angular` to `AngularJS` you could run this query.

```
db.tech.update(
  { "name" : "Angular" },
  { $set : { "name" : "AngularJS" } }
)
```

And validate that it worked by running `db.tech.find()`.

This works for adding new fields to the document, like this:

```
db.tech.update(
  { "name" : "AngularJS" },
  { $set : { "language" : "JavaScript" } }
)
```

The update command will only work with a single document - if a query matches multiple documents, only the first will be updated.

Updating multiple docs

To update multiple documents we can add an options parameter to the `update` method and pass in a `multi : true` option.

This for example, will find all of the documents in the collection and add a field `language` with the value `JavaScript`.

```
db.tech.update(
  {},
  { $set : { "language" : "JavaScript" } },
  { multi : true}
)
```

If we run a `find()` we now get this:

```
{ "_id" : ObjectId("56b3d22468b2bfa5460d035b"), "name" : "MongoDB", "role" : "Database", "la
{ "_id" : ObjectId("56b3d57168b2bfa5460d035c"), "name" : "Express", "role" : "Web applicati
{ "_id" : ObjectId("56b3d57168b2bfa5460d035d"), "name" : "AngularJS", "role" : "Front-end fr
{ "_id" : ObjectId("56b3d57168b2bfa5460d035e"), "name" : "Node.js", "role" : "Platform", "la
```

Deleting data

Deleting documents

MongoDB comes with a `remove` method for deleting documents from a collection. You pass it a query to find matching documents to delete - just like with the `update` method.

So we can delete the Express document in our *tech* collection like this:

```
db.tech.remove( { name : "Express" })
```

Validate with `db.tech.find()`

if you pass an empty object you can delete all documents from a collection, like so:

```
db.tech.remove( {} )
```

Validate again.

Deleting collections

If you want to delete an entire collection from a database, you "drop" it.

```
db.tech.drop()
```

Validate with `show collections`.

Summary

In this lesson we've seen how to use the Mongo shell to find documents in a database, update them and even delete them. We've really only scratched the surface with these commands. There's a whole load of methods and capabilities which you can read about in the MongoDB documentation. I'm sure nobody wants to listen to me reading all of that out. So we're learning enough to be useful and keep moving forward.

If you've followed along right up to deleting the data, you might want to add some data back into this

database as in the next video we're going to look at exporting and importing data.
