IMY 220 ● Lecture 20

What is MongoDB?

"MongoDB is a source-available cross-platform document-oriented database program.

*Classified as a **NoSQL** database program, MongoDB uses JSON-like documents with optional schemas."*

https://en.wikipedia.org/wiki/MongoDB *(29/09/2021)*

(In other words, there are no tables, rows, or columns)

Terminology

Database: one/more collections

Collection: one/more documents

Document: a single JSON object containing any number of key-value pairs to store some data

```
{
    "userid": "000001",
    "name": "Curtis"
}
```
(Document)

```
{
    "userid": "000002",
    "name": "Nathan"
}
```
(Document)

Users
(Collection)

```
{
    "productid": "000123",
    "name": "Running shoes"
}
```
(Document)

```
{
    "productid": "009567",
    "name": "Apple pie"
}
```
(Document)

Products
(Collection)

DBExample
(Database)

MongoDB vs MySQL

We can liken a single **document** to a single **row**

…a **collection** to a **table**…

…and a database is still a database

JSON vs BSON

"…there are several issues that make JSON less than ideal for usage inside of a database.

1. *JSON is a text-based format, and text parsing is very slow*
2. *JSON's readable format is far from space-efficient, another database concern*
3. *JSON only supports a limited number of basic data types"*

Solution: Binary JSON (BSON)

https://www.mongodb.com/json-and-bson

JSON vs BSON

"BSON simply stands for "Binary JSON", and that's exactly what it was invented to be. BSON's binary structure encodes type and length information, which allows it to be parsed much more quickly."

"MongoDB stores data in BSON format both internally, and over the network, but that doesn't mean you can't think of MongoDB as a JSON database. Anything you can represent in JSON can be natively stored in MongoDB, and retrieved just as easily in JSON."

https://www.mongodb.com/json-and-bson

Atlas

Cloud hosting platform for MongoDB

Includes free tier: 512MB storage + no expiration date

Atlas setup

Create a MongoDB account at:
https://account.mongodb.com/account/register

Once logged in, create a new organisation (name it whatever you want)
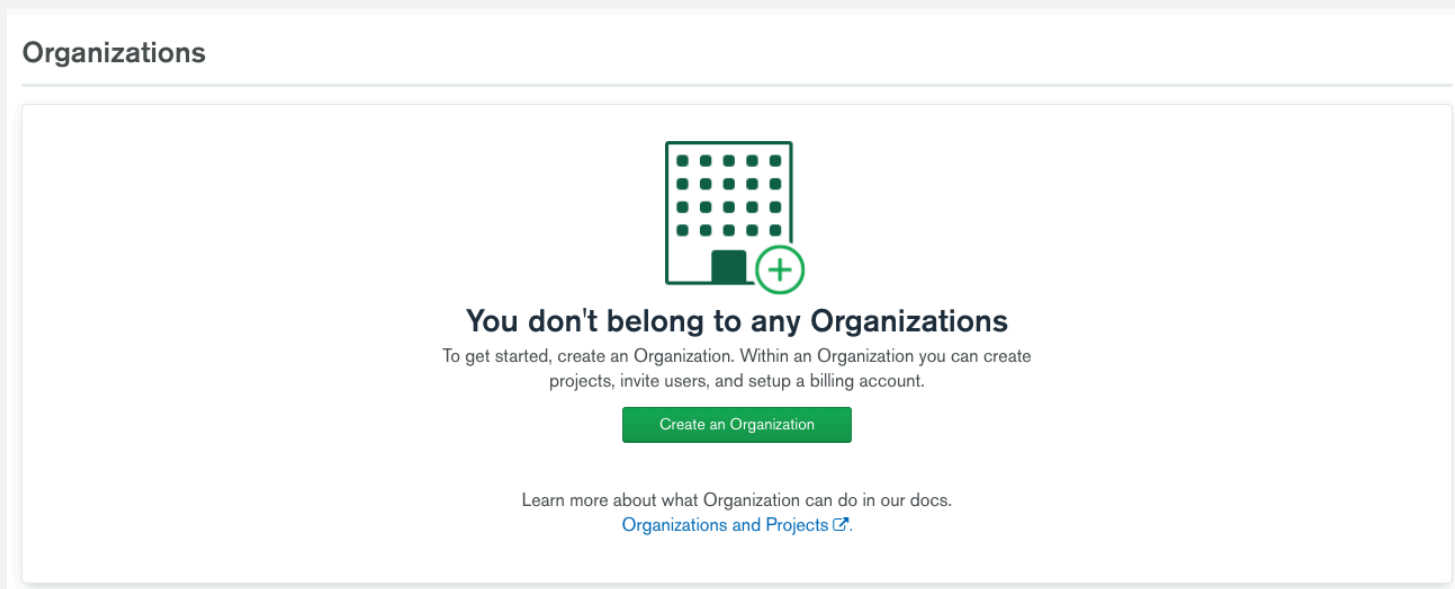
Atlas setup

Under "Select Cloud Service" select MongoDB Atlas and click Create Organization

Click on New Project (top right) and name the project IMY220

Click on Create Project

Atlas setup

Click on Build a Database

Atlas setup

Select the free option and click Create

Atlas setup

This will provide a list of hosting options and locations.

(The free tier limits you to a much smaller list, so I'd recommend going with whatever is selected automatically)

Atlas setup

Scroll down to the Cluster Name and change this to "demo"

Click Create Cluster

Atlas setup

Success! You should see something like this, which takes a few minutes

As an aside, you can read up on what a cluster is here: https://www.mongodb.com/basics/clusters

Atlas setup

When the cluster is finished setting up, your page should look like this
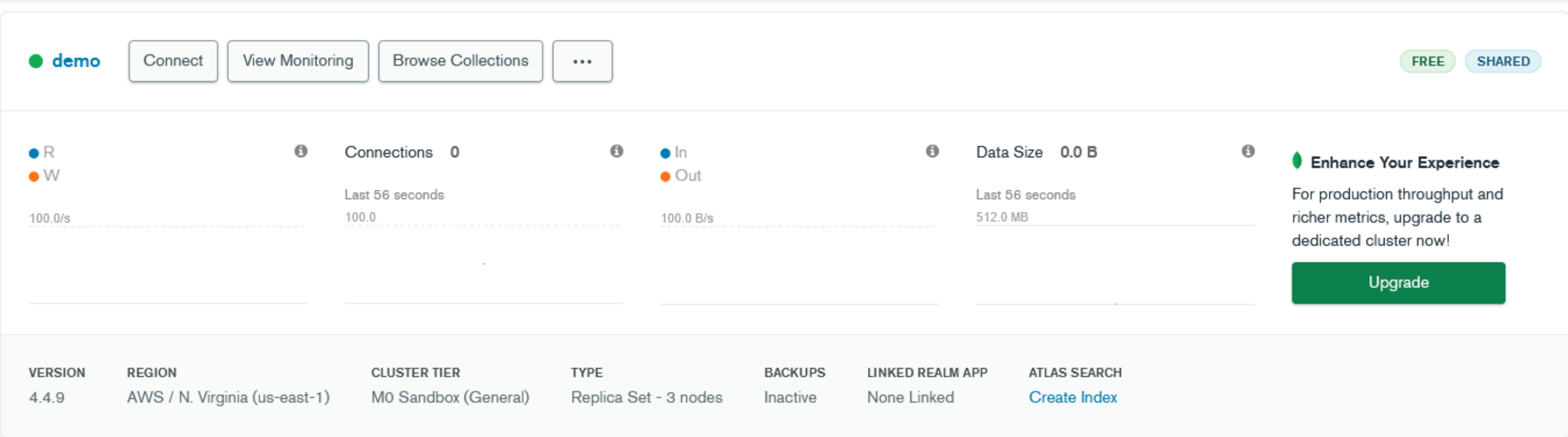
Atlas setup

What this means is that we have a hosted space (a cluster) that contains nothing so far

We can start adding data by clicking Browse Collections
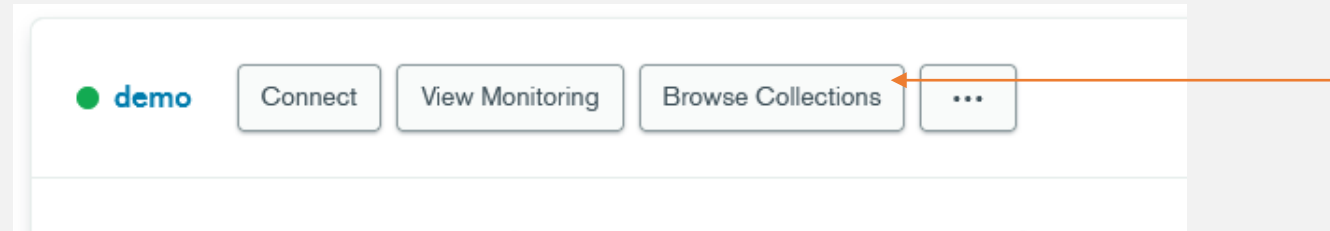
This is where we can browse through our data

Atlas setup

You should see two buttons to
- Load a Sample Dataset
- Add My Own Data
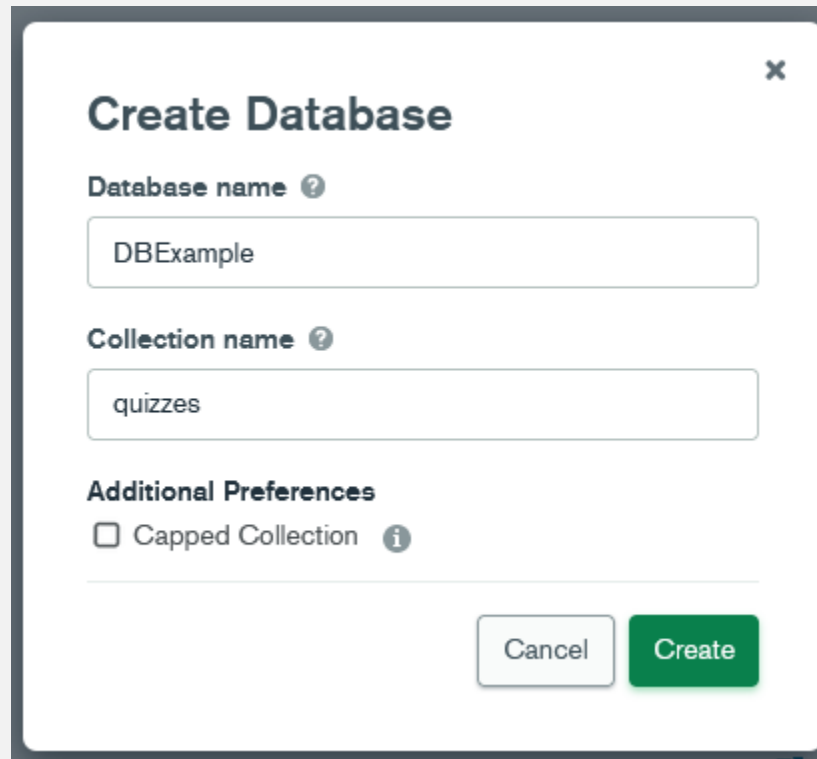
As a way to get started, Atlas provides large datasets with example data

However, for this module we will work with our own data

Atlas setup

Click on Add My Own Data, enter the following details, and click Create

Atlas setup

Now we have a database called DBExample that contains one collection called quizzes. The next step is to start adding Documents. Click on INSERT DOCUMENT
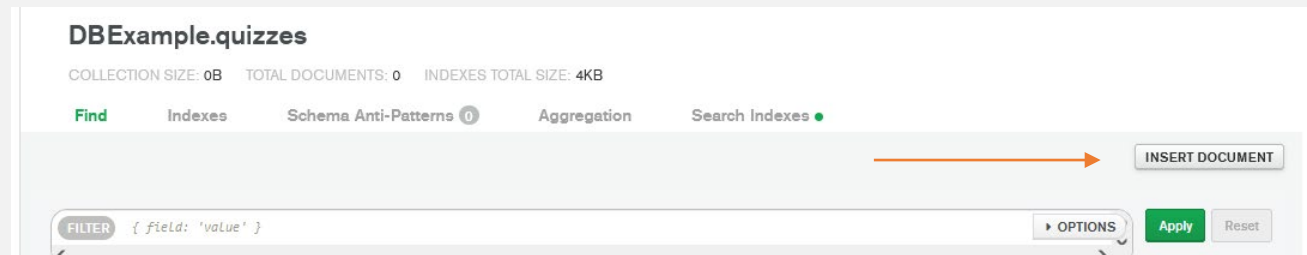
The interface allows us to create documents using standard JSON format as well as using a simplified format
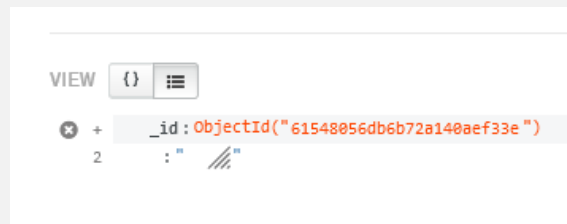
Atlas setup

We'll start by adding a single document with some basic data

```
{
        "question": "What is the name of Thor's hammer?",
        "answers":
        [
                {
                        "answer": "Mjolnir",
                        "correct": true
                },
                {
                        "answer": "Frostmourne",
                        "correct": false
                }
        ]
}
```
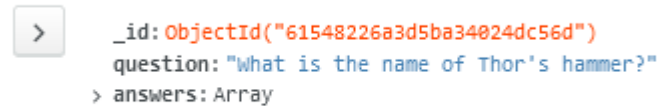
Atlas setup

Select the JSON interface and paste the data, then click Insert

Atlas setup

The document now contains the string and array we entered, but also an _id key with an auto-generated value


```
_id: ObjectId("61548226a3d5ba34024dc56d")
question: "What is the name of Thor's hammer?"
answers: Array
```

Whenever new documents are added, MongoDB automatically generates _id which contains the primary key

Atlas setup

For the remainder of this class we will need more complex test data

Download users.json and classes.json from ClickUP

Click on the **+** next to DBExample to create two new collections titled users and classes
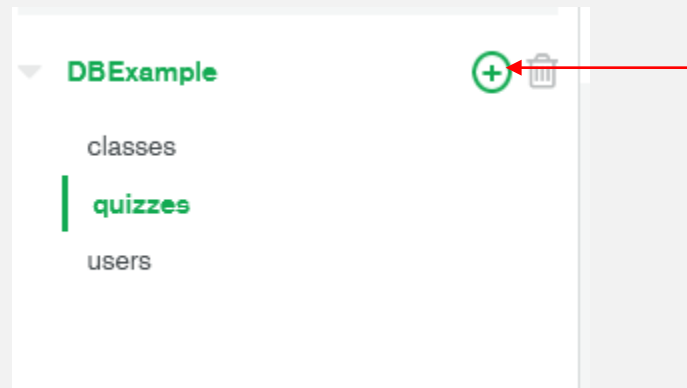
Atlas setup

Open each collection, click INSERT DOCUMENT and copy the contents of the corresponding JSON file into the JSON interface

If you view one of the new collections, you should see a list of documents (one for each user/class respectively)

In other words, when you add an array of JSON objects, each gets added as a unique document (with an auto-generated _id)

Atlas setup

DBExample.users

COLLECTION SIZE: **1.89KB**   TOTAL DOCUMENTS: **13**   INDEXES TOTAL SIZE: **36KB**

Find   Indexes   Schema Anti-Patterns ⓪   Aggregation   Search Indexes ●

FILTER   `{ field: 'value' }`

<

QUERY RESULTS **1-13 OF** 13

```
_id: ObjectId("6162c83b13dadb8fa421f48e")
id: 23985677
name: "Jeff"
surname: "Winger"
age: 47
position: "student"
> enrolled: Array
> passed: Array
```

```
_id: ObjectId("6162c83b13dadb8fa421f498")
id: 56472819
name: "Marshall"
surname: "Kane"
age: 52
position: "teacher"
```

```
_id: ObjectId("6162c83b13dadb8fa421f492")
```

Atlas setup

Open each collection, click INSERT DOCUMENT and copy the contents of the corresponding JSON file into the JSON interface

If you view one of the new collections, you should see a list of documents (one for each user/class respectively)

In other words, when you add an array of JSON objects, each gets added as a unique document (with an auto-generated _id)

Atlas setup

With our test data we can now start exploring basic CRUD operations (Create, Read, Update, Delete)

While we can do some of this in the web interface, we will focus on using MongoDB statements to do so for the remainder of this class

To be able to do this we need to connect to our database somehow (e.g. through a command line interface)

Atlas setup

Download the Mongo Shell from
https://www.mongodb.com/try/download/shell

The installation should set up the correct PATH variable

If it does not, you need to manually add it to the PATH

https://sysadmindata.com/set-mongodb-path-windows/

Atlas setup

We also need to tell our Mongo server to accept incoming connections

Click on Connect ——→   ● demo ——→ | Connect | | View Monitoring | | Browse Collections | | ... |
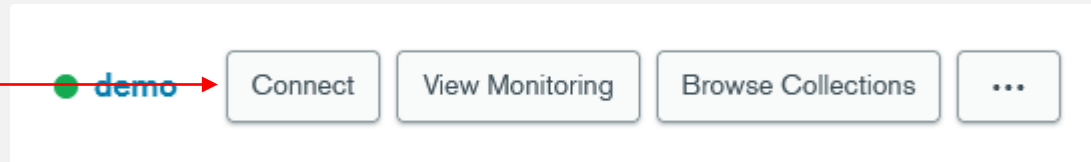
Under Add a connection IP address click on Allow Access from Anywhere
And then Click Add IP Address

1 Add a connection IP address

| Add Your Current IP Address | Add a Different IP Address | Allow Access from Anywhere | ←——————
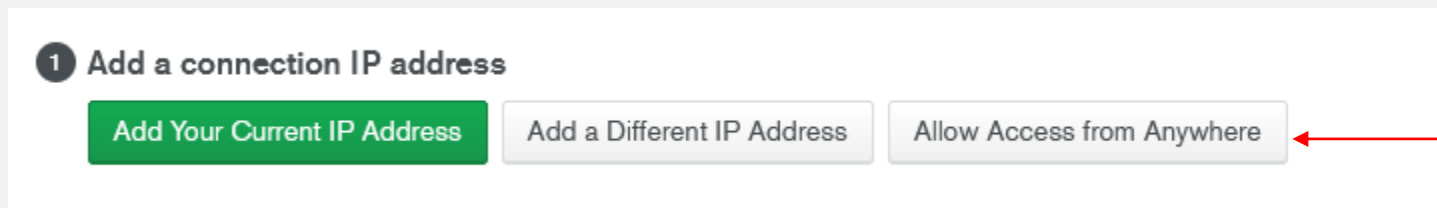
Atlas setup

NB! Enabling access from anywhere presents a huge security risk and is NOT recommended for production servers

However, since we are only using this server as a testbed to learn MongoDB, we are making things easier for ourselves by allowing this option

Atlas setup

Under Create a Database User, create a new user by entering a username and password

Note that you will need to enter this password every time you connect to MongoDB

After creating the Database User, click on Choose a connection method

Atlas setup

Since we will use the MongoDB Shell, click on Connect with the MongoDB Shell

Since we've already installed the MongoDB Shell (mongosh) we can skip to Step 3: Run your connection string in your command line

Atlas setup

Copy the connection string which should look something like this:

```
mongosh "mongodb+srv://demo.xxxxx.mongodb.net/myFirstDatabase" --username yourUsername
```

xxxxx will instead be a unique alphanumeric string and **yourUsername** will be your chosen username

(For security reasons, it is also not a good idea to share these)

Atlas setup

Paste the connection string into your command line and run it. It will prompt for your password as well

You should see something like this

```
Current Mongosh Log ID:
Connecting to:          mongodb+srv://demo.          .mongodb.net/myFirstDatabase
Using MongoDB:          4.4.9
Using Mongosh:          1.1.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
  You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
```

MongoDB statements

We are now ready to start executing statements to interact with our database using MQL (MongoDB query language)

Run the *show dbs* command to show the list of databases on your server

```
Atlas atlas-kjhvps-shard-0 [primary] myFirstDatabase> show dbs
DBExample    156 kB
admin        373 kB
local       7.75 GB
```

(We will not be working with the **admin** and **local** databases at all)

Select database

First we need to specify which database we want to use. The statement for this is *use <dbname>*

```
Atlas atlas-kjhvps-shard-0 [primary] myFirstDatabase> use DBExample
switched to db DBExample
```

We can then type *show collections* to view collections in this database

```
Atlas atlas-kjhvps-shard-0 [primary] DBExample> show collections
classes
quizzes
users
```

Retrieve data

To retrieve data we use the *find* method, which takes the following form

db.<collection>.find(query, projection)

    <collection> refers to the name of the collection we are querying

    query JSON object that specifies the query we want the data to match

    projection JSON object that specifies the fields we want to return

This is roughly equivalent (~) to the following MySQL statement

SELECT projection FROM <collection> WHERE query

Retrieve data

To retrieve the entire collection we can leave the query and projection parameters blank, e.g.

```
db.classes.find()
```

(~MySQL: *SELECT * FROM classes*)

Retrieve data - queries

The simplest query is to retrieve all entries where a key matches a value

For example, if we want to retrieve all users who are students, we can use

```
db.users.find({"position":"student"})
```

(~MySQL: *SELECT * FROM users WHERE position="student"*)

Retrieve data - queries

To retrieve based on something other than equivalence and to build more complex queries we need to use query operators

The general syntax is *db.<collection>.find({"key":{"$operator":"value"}})*

The operators can be found here:
https://docs.mongodb.com/manual/reference/operator/query/

Retrieve data - queries

For example, if we want to retrieve all users above the age of 50 we can use the *$gt* (greater than) operator

```
db.users.find({"age":{"$gt":50}})
```

(~MySQL: *SELECT * FROM users WHERE age > 50*)

Retrieve data - queries

We have actually already used the equivalence operator (*$eq*), albeit implicitly, since

```
db.users.find({"position":"student"})
```

…is the same as

```
db.users.find({"position":{"$eq":"student"}})
```

(But there are scenarios where we would need to use *$eq* explicitly)

Retrieve data - queries

We can combine query statements with logic operators, which have the following syntax: *{"$operator":[{query1},{query2},…]}*

e.g. to find all users who are above 50 and teachers

```
db.users.find({"$and":[
    {"age":{"$gt":50}},
    {"position":"teacher"}
]})
```

(~MySQL: *SELECT * FROM users WHERE age > 50 AND position = "teacher"*)

Retrieve data - queries

MongoDB also has query operators specifically for dealing with arrays, such as $all which checks if an array field contains at least all given values (in any order)

The syntax for *$all* is *{"$all":["elem1","elem2",…]}*

E.g. to retrieve all students who passed Spanish 101 and History 101

```
db.users.find({"passed":{"$all":["ESP101","HST101"]}})
```

Retrieve data - projection

Next, let's investigate the syntax for retrieving only certain fields using what is known as projection (in MongoDB terminology)

Remember that the projection is the (optional) second parameter in the find method, so we always need to specify it after a query

db.<collection>.find(query, projection)

Retrieve data - projection

The general syntax for specifying a projection is *{"key1":1|0,"key2":1|0,...}*

The 1|0 (i.e. 1 or 0) is a TRUE|FALSE flag which specifies whether we want to include/exclude a key from the results

Note that _id is included by default

Retrieve data - projection

E.g. if we only want to retrieve the names of all students

```
db.users.find({"position":"student"},{"name":1})
```

(~MySQL: *SELECT name FROM users WHERE position = "student"*)

```
Atlas atlas-kjhvps-shard-0 [primary] DBExample> db.users.find({"position":"student"},{"name":1})
[
  { _id: ObjectId("6162c83b13dadb8fa421f48e"), name: 'Jeff' },
  { _id: ObjectId("6162c83b13dadb8fa421f48f"), name: 'Britta' },
  { _id: ObjectId("6162c83b13dadb8fa421f490"), name: 'Abed' },
  { _id: ObjectId("6162c83b13dadb8fa421f491"), name: 'Troy' },
  { _id: ObjectId("6162c83b13dadb8fa421f492"), name: 'Pierce' },
  { _id: ObjectId("6162c83b13dadb8fa421f493"), name: 'Shirley' },
  { _id: ObjectId("6162c83b13dadb8fa421f494"), name: 'Annie' }
]
```

Retrieve data - projection

Or if we want to retrieve everything except the name and surname

```
db.users.find({"position":"student"},{"name":0,"surname":0})
```

Note that you cannot mix the use of 1s and 0s here

E.g. the following will throw a syntax error

```
db.users.find({"position":"student"},{"name":0,"surname":1})
```

Retrieve data - projection

The only exception to this is for _id (since it is included by default)

```
db.users.find({"position":"student"},{"name":1,"_id":0})
```

```
Atlas atlas-kjhvps-shard-0 [primary] DBExample> db.users.find({"position":"student"},{"name":1,"_id":0})
[
  { name: 'Jeff' },
  { name: 'Britta' },
  { name: 'Abed' },
  { name: 'Troy' },
  { name: 'Pierce' },
  { name: 'Shirley' },
  { name: 'Annie' }
]
```

The End

"The UI is quite simple, client will easily figure it out"

Client:

References

https://university.mongodb.com/

https://docs.atlas.mongodb.com/

https://docs.mongodb.com/manual/reference/method/db.collection.find

https://docs.mongodb.com/manual/reference/operator/query/