Notes

CRUD

💡 C ⇒ Create
R ⇒ Read
U ⇒ Update
D ⇒ Delete

💡 Just consider that we have created two collections as we have created in the live session itself.
1. users.
2. heroes.

Create

- Let us create a `database` first, before doing anything else (Ask them to do this along with you) ⇒ `use nxm`
- How to check in which `database` we are now ⇒ `db`
- Now let us create a `collection` in this `database`. ⇒ `db.createCollection("users")`
- Now let us insert `documents` ⇒ `db.users.insertOne({name:"Albert", org:"Masai"})` ⇒ This is to insert one document.
- To insert many you can use ⇒ `db.users.insertMany([{},{},{}])`

Read

- To see or read all the documents you can just simply do ⇒ `db.users.find()`
- This will show you `all` the documents available.
- Now, If I just want the documents whose org is Masai ⇒ `db.users.find({org:"Masai"})`

- If there are two documents with `{org:"Masai"}` but I just want any one of them then ⇒ `db.users.findOne({org:"Masai"})` ⇒ It will return the first one.

Update

- To update one ⇒ If I just want to add city as Bangalore whose org is Masai ⇒ `db.users.updateOne({org:"Masai"},{$set: {city: "Bangalore"}})`
- If you want to update many ⇒ `db.users.updateMany({org:"Masai"},{$set:{city: "Bangalore"}})`

> 💡 **Note:** If you want to see the data in a prettier way you can do ⇒ `db.users.find().pretty()`
>
> This will only work in terminal though, so that you can see the data in a proper manner.

Delete

- If you want to delete one then ⇒ `db.users.deleteOne({org:"Masai"})`
- If you want to delete many then ⇒ `db.users.deleteMany({city:"Bangalore"})`

> 💡 Now you can appreciate databases, to do this CRUD, it could have taken us so much time, we can just quickly run a query and get the work done.

Let us move on….

> 💡 Please open up the following link and copy the data from it.
>
> `pastebin.com/raw/2nuFWi3U`

- Just copy the data from the above link and create a new collection and insert this data, instead of creating a collection, we can directly insert many and that will do the job ⇒ `db.heroes.insertMany(copied_data)`
- The data will be inserted into a new collection.
- Now you can check all the available `collections` in the `database` ⇒ `show collections`

Now we are going to see filtering of data in advance manner…

Comparison Operators

- Talk about comparison operators `≤, <, ≥, ==, !=`
- How do we refer them in Mongo?

> 💡 **<= ⇒ lte**
>
> **< ⇒ lt**
>
> **>= ⇒ gte**
>
> **> ⇒ gt**
>
> **== ⇒ eq**
>
> **!= ⇒ ne**

- We can use these things for filtering out the data in a better way.
- Lets us get the heroes from the data whose health is **≤ 50** ⇒ `db.heroes.find({health: {$lte:50}}).pretty()`
- Similarly `gte, gt, lt` command can be used as well.

- For finding the equal to, there is no need to use `eq` , you can simply use ⇒ `db.heroes.find({health:86})`
- For `ne` ⇒ `db.heroes.find({health: {$ne:86}})` ⇒ this will give all the heroes except the one whose health is 86.

Logical Operators

- For understanding this we are going to use `users` collection
- Logical operators `and, or`
- We can use these things for filtering out the data in a better way.

> 💡 **Note:** If I want to add {country: “India”} in all the documents then ⇒ `db.users.updateMany({},{$set: {country:”India”}})`

- Find all the users whose org is Masai and country as India ⇒ `db.users.find({orq:”Masai”, country: “India”})`
- It is not necessary to use `and` over here as database will return the data only if both are present.
- What if you want to use `and` ⇒ `db.users.find({$and:[{org:”Masai”}, {country:”India”}]})`
- Similarly `or` can be used.
- You can also try this with the heroes collection data and show them.
- Want the data whose health is in between 40 and 60 ⇒ `db.heroes.find({$and: [{health:{$gt:40}}, {health:{$lt:60}}]}).pretty()`

> 💡 **Note:** Based on our requirement, we can use multiple things together as well.

Limit

- We can limit the result, criteria can be anything.
- `db.heroes.find().limit(2).pretty()` ⇒ First two documents will be returned.
- `db.heroes.find({org:"Masai"}).limit(2).pretty()` =⇒ Return the first two documents with org as Masai.

Skip

- We can use it to skip documents.
- `db.users.find({country:"India"}).skip(2).pretty()` ⇒ This will skip the first 2 documents.
- We can even combine `skip()` and `limit()` as well
- `db.users.find({country:"India"}).skip(1).limit(2).pretty()` ⇒ This will skip the first one and then limit the result to two documents.

💡 **Question**: Where exactly we can use this in real life?

Answer: This can be used in pagination.

Sort

- It will sort the `documents` in `ascending` or `descending` order.
- `db.heroes.find().sort({health:1}).pretty()` ⇒ This will sort in `ascending` order
- For `descending` order we can use `-1`

💡 **Question:** Can we sort Alphabets as well.

Answer: Yes, it can, it sorts on the basis of `ASCII` value, so you have to keep track of case as well.

Mongo Cheat Sheet

MongoDB Cheat Sheet | MongoDB

MongoDB Cheat Sheet by MongoDB for our awesome MongoDB Community <3.

https://www.mongodb.com/developer/products/mongodb/cheat-sheet/