HOW TO USE MONGODB:

Open cmd as administrator

cd.., cd.., until you see C:\>

cd Users

cd MongoDB

cd bin

mongod

(wait for it to run/generate loads of text)

Open a new cmd terminal as an administrator, keep other one open (or, if using a GUI, run this now)

do the same steps as above - navigate to bin folder

then type mongo.exe

should be working now.

SPACES and COMMAS matter!

Examples: adding collections and documents (one or many at a time), and deleting them:

use employee

switched to db employee

> db.createCollection("EmployeePay")

{ "ok" : 1 }

> show collections

EmployeePay

> db.EmployeePay.drop()

true

> show collections

> db.createCollection("users")

{ "ok" : 1 }

> db.users.insertOne(

... {

... name: "sue",

... age: 19,

... status: "P"

... }

... )

Here is how you delete the FIRST document that matches a given characteristic:

db.users.deleteOne( { status: "D" } )

Another way to do the same thing:

db.users.remove( { status: "D" }, 1)

delete ALL documents that match a given condition:

db.users.deleteMany({status:"A"})

another way to delete all that match a condition:

db.users.remove({status: "P"})

See all the entries (documents) in the table (collection):

db.users.find()

Or, you can do the same thing with:

db.users.find( {} )

If you want to query all of the documents with a given characteristic:

db.users.find( {status: "A" } )

Some other types of queries, for example 'all the documents whose status is P or D':

db.users.find( {status: {$in:["P", "D"]}})

Syntax for this is 'first is 'field', then 'operator', then 'value'.

Finding "AND" conditions: here we want to find ALL of the documents for people with status = A AND age is less than 30.

db.users.find( {status: "A", age: {$lt: 30 } } )

comma means 'and'. $lt means 'less than'.

Finding "OR" conditions: all the docs in this collection where status = A OR age is less than 30:

db.users.find({$or: [ {status: "A"}, {age: {$lt:30} } ] } )

you put the 'or' at the start. and the comma means 'or' here.

To combine AND and OR queries... for example, 'find all whose status is A, and (some set of other conditions).' Here, we want all whose status = A, AND (comma) who are either less than 30 or whose type =1.

db.users.find(

... {

... status: "A",

... $or: [ {age: {$lt:30}}, {type:1}]

... }

... )

Common operations:

: equality

$lt: less than

$lte: less than or equal to

$gt: greater than

$gte: greater than or equal to

$ne: not equals

Regular expressions: example query those whose names do not start with 'b':

db.users.find({name: {$not: /^b.\*/}})

Where you have to specify the field you're examining the text of (here, it's 'name'). then /^ = 'starts with', and .\* = the 'rest of the string'. (any character, any number of times.)

Count number of documents in a collection:

db.users.find().count()

limit the number of results displayed:

db.users.find().limit(NUMBER)

or:

db.users.find({}).limit(2).pretty()

where 'pretty' means you want it to be formatted nicely.

SKIP = skip a certain number of documents.

db.users.find().limit(NUMBER).skip(NUMBER)

If you say 'skip(2)', the first result you'll see is the third one in the collection.

SORT ascending (1) or descending (-1). = sort()

db.COLLECTION\_NAME.find().sort({KEY:1})

for example display 'users' in alphabetical order of name:

db.users.find().sort({"name":1}).pretty()

Sort by more than one key: sort by ascending age AND descending status...sort by age and THEN by status, in that order.

db.users.find().sort({"age":1, "status":-1}).pretty()

The second parameter of the db.collection.find(QUERYFILTER, PROJECTION) is the projection, which specifies which fields from the matching documents to return. It limits the amount of data that mongodb returns. so you don't always want the result of every column.

1 means yes

0 means no

so for example:

db.users.find({status:"A"},{name:1, status:1})

this says 'find all whose status =A, but only show me the name and status columns of the result.' (and the \_id field, by default)

UPDATE A DOCUMENT (change value):

db.users.update({"name": "xi"}, {$set: {"age":30})

Check the result to see if it updated properly by using:

db.users.find() (or query this person's name)

Update THE FIRST document that matches filter:

db.users.updateONE({"status":"P"},{$set:{"name":"andy","age":31}})

If you want to insert a lot of data into mongodb at once: use mongoImport tool:

https://docs.mongodb.com/manual/reference/program/mongoimport/#bin.mongoimport

There are 3 ways to do aggregation:

aggregation pipeline,

map-reduce

single-purpose methods

if the collection name is 'orders', the pipeline method is:

db.orders.aggregate({

$match:{status:"A"}},

{$group:{\_id: "$cust\_id",total:{$sum:"$amount"}}}

] )

this basically says the first procedure is 'find all records w/status=A". then, group records by customer id, and then calculate the sum of the amounts."

MAP-REDUCE IS AN OPTIONAL METHOD.