**ALL MONGO DB COMMANDS FROM TADAS PRESENTATION:**

**Create Database:**

use database\_name

**See database:**

db

**See all databases:**

show dbs

**Create collection:**

db.createCollection(“collection\_name”)

**Delete collection:**

db.createCollection(“collection\_name”)

**Show collections:**

show collections

**Insert data:**

db.collection\_name.insert( {user\_id: “22” , age: 50, f\_name: “James”} )

**Delete from table where “status” = “D”:**

db.collection\_name.remove( {status: “D”} )

**Update collection “customers” set “status” to “pensioner” where age>65:**

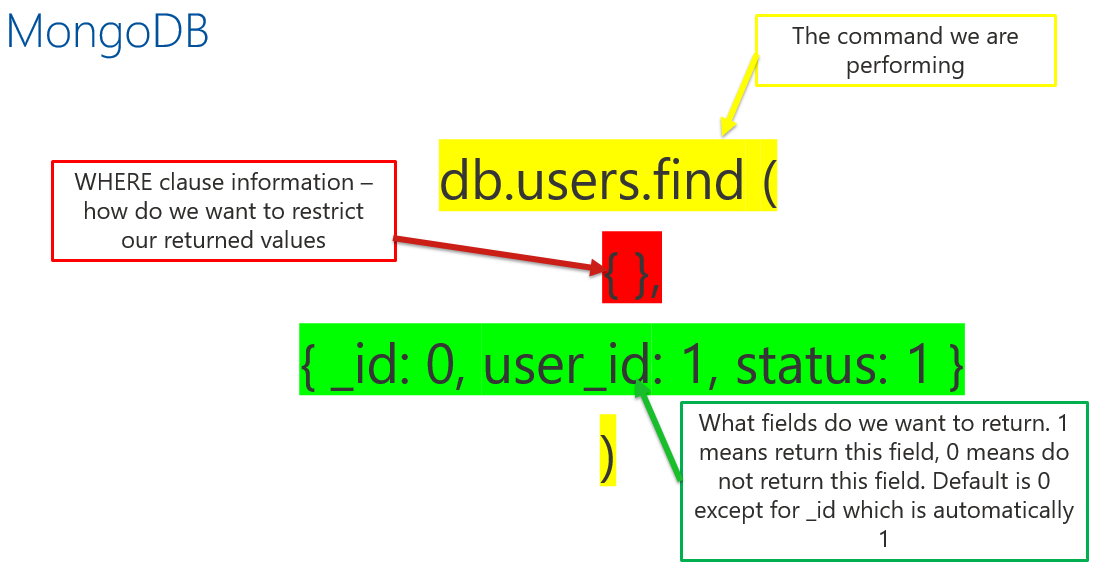
db.collection\_name.update( { age : { $gt : 65 } } , { $set : { status : “pensioner” } } , { multi : true } )

**Select \* From “users”:**

db.users.find()

**Select “user\_id”, “status” FROM “users”:**

db.users.find( { }, { \_id: 0, user\_id: 1, status: 1 } )

****

**SELECT \* FROM “users” WHERE status = “A”:**

db.users.find( { status : “A” } )

**SELECT \* FROM “users” WHERE status IS NOT = “A”:**

db.users.find( { status: { $ne : “A” } } )

**MONGODB OPERATORS:**

db.products.find ( { quantity : { $gt : 500 } ) 🡪 - $gt, greater than

db.products.find ( { quantity : { $gte : 500 } ) 🡪 - $gte, greater than or equal to

db.products.find ( { quantity : { $lt : 500 } ) 🡪 - $lt, less than

db.products.find ( { quantity : { $lte : 500 } ) 🡪 - $lte, less than or equal to

db.products.find ( { type : { $in : [ gnome, jacuzzi, ….. ] } } )

$in selects documents where the value of a field equals anything listed

$nin selects documents where the value of a field does not equal anything listed

db.products.find ( { $or : [ { quantity : { $lt 20 } }, { popularity : ‘high’ } ] } )

$or clause works to match on either condition

$and can be used in a similar way but must match both conditions

$nor used in a similar way but will return all documents that fail to match both clauses

$not returns documents that do NOT match the query expression

**Search database for a particular field:**

db.orders.find ( { complaints : { $exists : true } } )

**Output to a nicer looking format:**

db.customers.find().pretty()

**Limited find command returns:**

db.customers.findOne()

db.customers.find().limit(1)

**Order by command FIND “name” FROM “students” ORDER BY “score”:**

db.students.find({}, {“name” : 1,

“score” : 1}).sort({“score” : 1});

**Order by command FIND “name” FROM “students” ORDER BY “score” DESCENDING:**

db.students.find({}, {“name” : 1,

“score” : 1}).sort({“score” : -1});

**SELECT “name” + “score” FROM students WHERE score>30 AND or score<10:**

db.students.find({ “$or” : [{

“score” : { “$gt” : 30 } },

{“score” : { “$lt” : 10 } } ] },  
{“name” : 1, “score” : 1})

**SELECT “name” + “score” FROM students WHERE score>30 and “name” is ‘An%’:**

db.students.find({ “score” : { “$gt” : 30},

“name” : /^An/ },

{“name” : 1, “score” : 1 } )

**COUNT function (SELECT COUNT(\*) FROM users:**

db.users.count()

db.users.find().count()

**Aggregates (finding unique examples using a method):**

SELECT DISTINCT(type) FROM animal;

db.animal.distinct(“type”)

**Return information from an embedded document:**

db.customer.find({“address.city” : “Manchester”})

**Return a few pieces of information from embedded doc ($slice operator):**

db.customer.findOne({}, {“address”: { $slice : 4 }})