**5-1-A Install MongoDB compass & No SQL Booster ( windows)**

Google -> MongoDB Community Server Download

MongoDB Compass > Add new Collection > click on –localhost > open mongoDB shell >

Sometimes error occurred to connect / start this services : to resolve this , goto services > find MongoDB service > select & open > make it Automatice and state : start if it was stop > Ok .

# Use MongoDB Shell :

Mongosh:

test >show dbs

use practice > switched to db practice

db.getCollection(“test”).insertOne({name: “Next Level Web Dev”})

<{

acknowledged: true,

insertedId: ObjectId('6732fe24e375d51a06415fa3')

}

practice>

db.getCollection(“test”).find()

<{

\_id: ObjectId('6732fe24e375d51a06415fa3'),

name: 'Next Level Web Dev'

}

practice>

To work with in cmd , download mongoDB shell

Mongodb Shell download > package: msi (select) > download

Copy the path (example: C:\Program Files\MongoDB\Server\8.0\bin)

Windows>edit environment variables > path > new > paste > ok > ok

Cmd > mongod --version

:::

:::

Many people use > studio 3t

Here we’ll use :

[NoSQLBooster - The Smartest GUI Tool and IDE for MongoDB](https://nosqlbooster.com/)

After install > open > connect > new connection > test connection > close > ok > “localhost” > save & connect

localhost > practice >test

**5-1-B Install MongoDB compass & No SQL Booster ( Mac & Linux)**

**5-2 Insert,insertOne, find, findOne, field filtering, project**

Alt + Shift + F 🡪 to align

NoSQL Booster for MongoDB:

db.test.findOne({age:17})

db.test.find({ gender: "Male" }, { name: 1, age: 1, gender: 1, email: 1 })

db.test.find({ gender: "Male" }).project({ name: 1, age: 1, gender: 1, email: 1 })

**5-3 $eq, $neq, $gt, $lt, $gte, $lte**

<https://www.mongodb.com/docs/manual/reference/operator/query-comparison/>

Google -> mongodb operators -> Query and Projection Operators -> Comparison Query

db.test.find({ gender: { $eq: "Male" } })

db.test.find({ age: { $eq: 12 } })

db.test.find({ age: { $ne: 12 } })

db.test.find({ age: { $gt: 12 } })

db.test.find({ age: { $gte: 12 } })

db.test.find({ age: { $lt: 12 } }).sort({ age: 1 })

**5-4 $in, $nin, implicit and condition**

<https://www.mongodb.com/docs/manual/reference/operator/query/ne/#mongodb-query-op.-ne>

db.test.find({ age: { $lt: 30, $gt: 18 } }, { age: 1 }).sort({ age: 1 })

db.test.find({ gender: "Female", age: { $lt: 30, $gt: 18 } }, { age: 1, gender: 1 }).sort({ age: 1 })

db.test.find({ gender: "Female", age: { $lt: 30, $gt: 18 } }, { age: 1, gender: 1, name:1 }).sort({ age: 1 })

// implicit and

db.test.find(

{

gender: "Female",

age: { $in: [18, 20, 22, 24, 26, 28, 30] },

interests: "Cooking"

},

{ age: 1, gender: 1, name: 1 }).

sort({ age: 1 }

)

// implicit and

db.test.find(

{

gender: "Female",

age: { $in: [18, 20, 22, 24, 26, 28, 30] },

interests: { $in: ["Cooking", "Gaming"] }

},

{ age: 1, gender: 1, name: 1, interests: 1 }).

sort({ age: 1 }

)

**5-5 $and, $or, implicit vs explicit**

db.test.find({ age: { $ne: 15, $lt: 30 } })

// ----------------Explicit $and ----------------

// ------------ ascending

// db.test.find({ age: { $ne: 15, $lt: 30 } })

db.test.find({

$and: [

{ age: { $ne: 15 } },

{ age: { $lt: 30 } }

]

}).project({

age: 1

}).sort({ age: 1 })

// --------- descending

db.test.find({

$and: [

{ age: { $ne: 15 } },

{ age: { $lt: 30 } }

]

}).project({

age: 1

}).sort({ age: -1 })

// ----------- Explicit $or -------------

db.test.find({

$or: [

{ interests: "Travelling" },

{ interests: "Cooking" }

]

}).project({

interests: 1

}).sort({ age: 1 })

db.test.find({

$or: [

{ "skills.name": "JAVASCRIPT" },

{ "skills.name": "PYTHON" }

]

}).project({

skills: 1

}).sort({ age: 1 })

// ---- same work using : $in

db.test.find({ "skills.name": { $in: ["JAVASCRIPT", "PYTHON"] } }).project({

skills: 1

}).sort({ age: 1 })

**5-6 $exists, $type,$size**

db.test.find({ age: { $exists: false } })

db.test.find({ phone: { $exists: false } }, { phone: -1,name: 1 })

db.test.find({age: {$type: "string"}})

db.test.find({age: {$type: "number"}})

db.test.find({friends: {$type: "array"}})

db.test.find({friends: {$size: 0}}).project({friends:1})

db.test.find({friends: {$size: 4}}).project({friends:1})

db.test.find({friends: {$size: 5}}).project({friends:1})

**5-7 $all , $elemMatch**

db.test.find({ interests: "Cooking" }).project({ interests: 1 })

db.test.find({ interests: ["Gardening", "Gaming", "Cooking"] }).project({ interests: 1 }) // query -> same to same value, organize, order

db.test.find({ interests: { $all: ["Gardening", "Gaming", "Cooking"] } }).project({ interests: 1 }) // to find all value just matched , no order

db.test.find({

skills: {

name: "JAVASCRIPT",

level: "Intermidiate",

isLearning: false

}

}).project({ skills: 1 })

// ----- elemMatch -----

db.test.find({

skills: { $elemMatch: {

name: "JAVASCRIPT",

level: "Intermidiate"

}}

}).project({ skills: 1 })

**5-8 $set, $addToSet, $push**

<https://www.mongodb.com/docs/manual/reference/operator/update/set/>

db.test.updateOne(

{ \_id: ObjectId("6406ad63fc13ae5a40000065") },

{

$set: {

age: 23,

// interests: [ "Cooking", "Writing", "Reading" ],

interests: [ "Gaming", "Writing", "Reading" ],

}

}

)

db.test.updateOne(

{ \_id: ObjectId("6406ad63fc13ae5a40000065") },

{

$addToSet: {

interests: "Writing"

}

}

)

db.test.updateOne(

{ \_id: ObjectId("6406ad63fc13ae5a40000065") },

{

$addToSet: {

interests: { $each: ["Reading", "Driving"] }

}

}

)

db.test.updateOne(

{ \_id: ObjectId("6406ad63fc13ae5a40000065") },

{

$push: {

interests: { $each: ["Reading", "Driving"] }

}

}

)