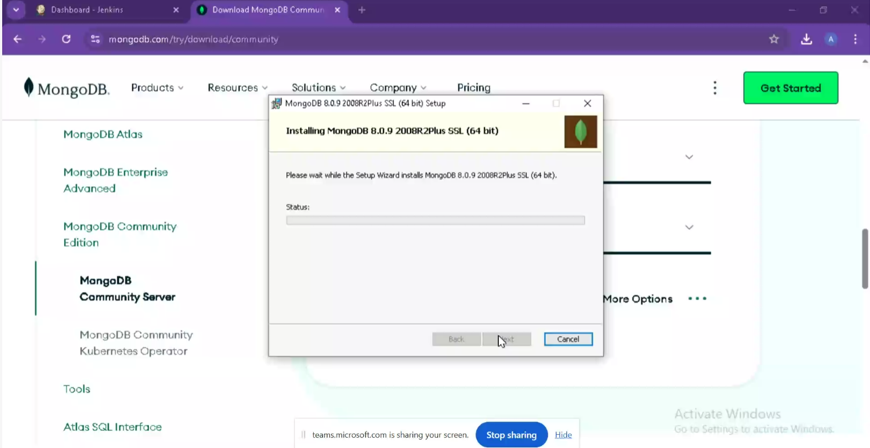
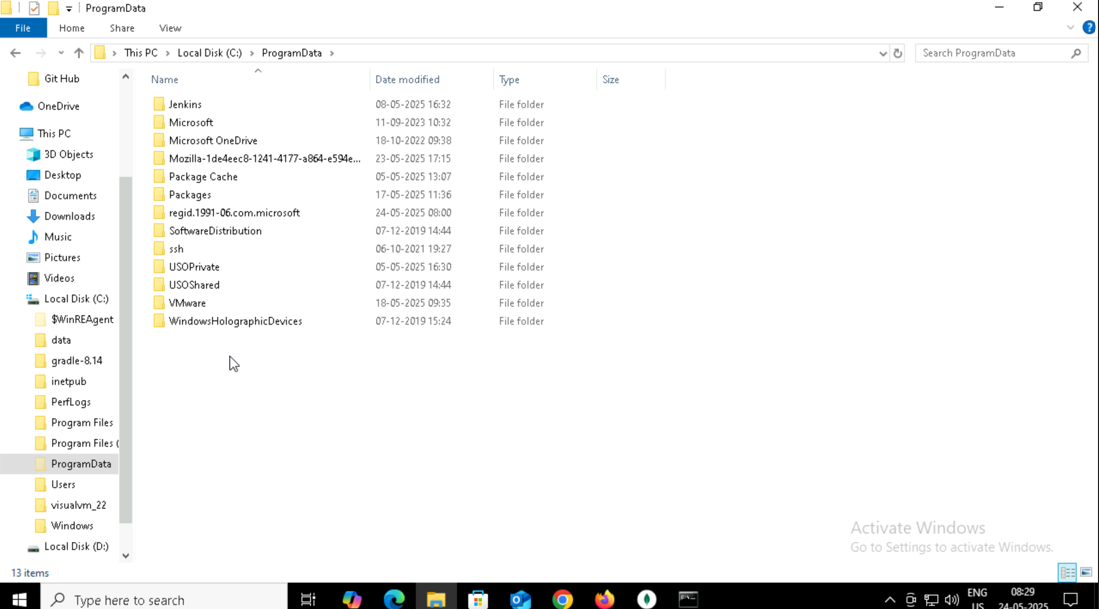
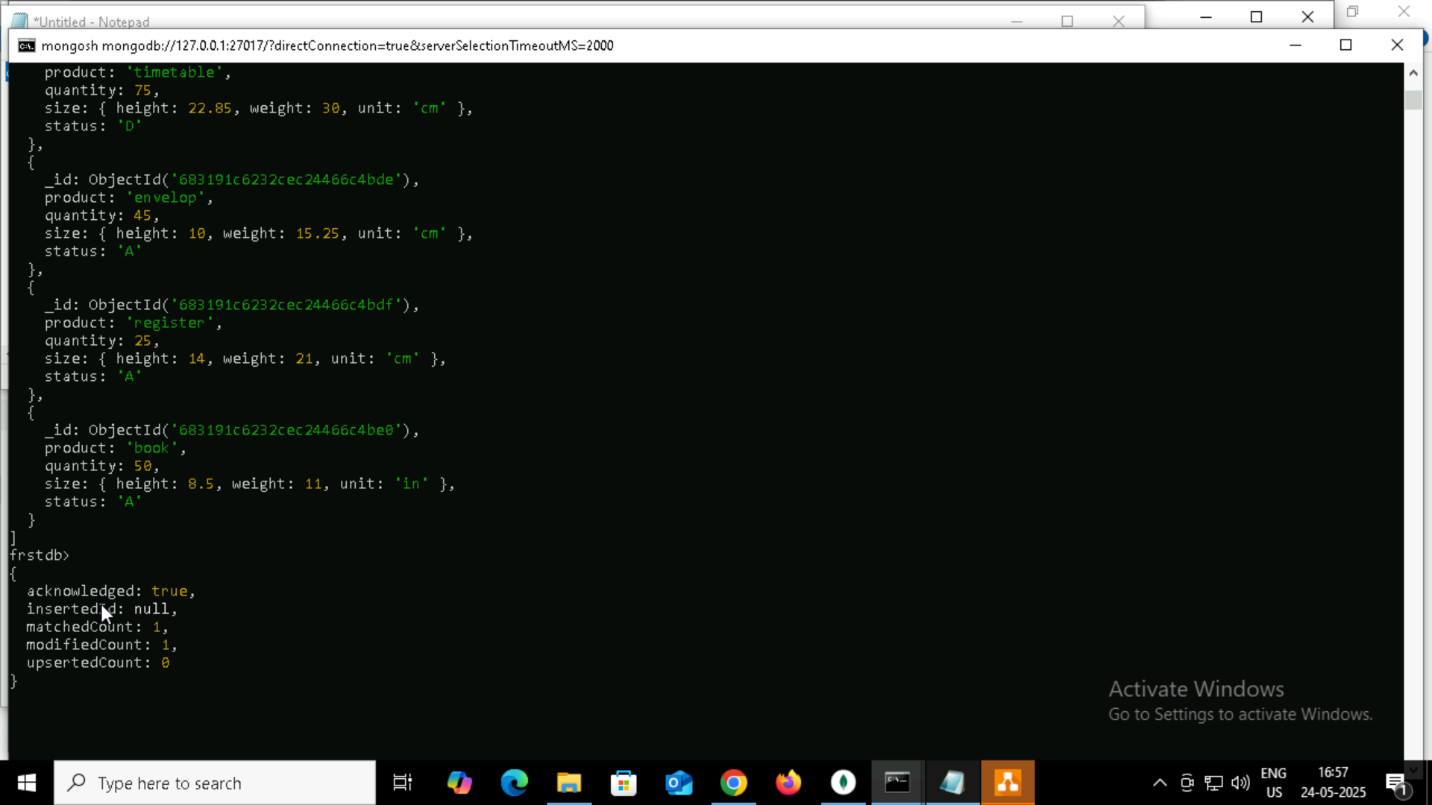
Day 4

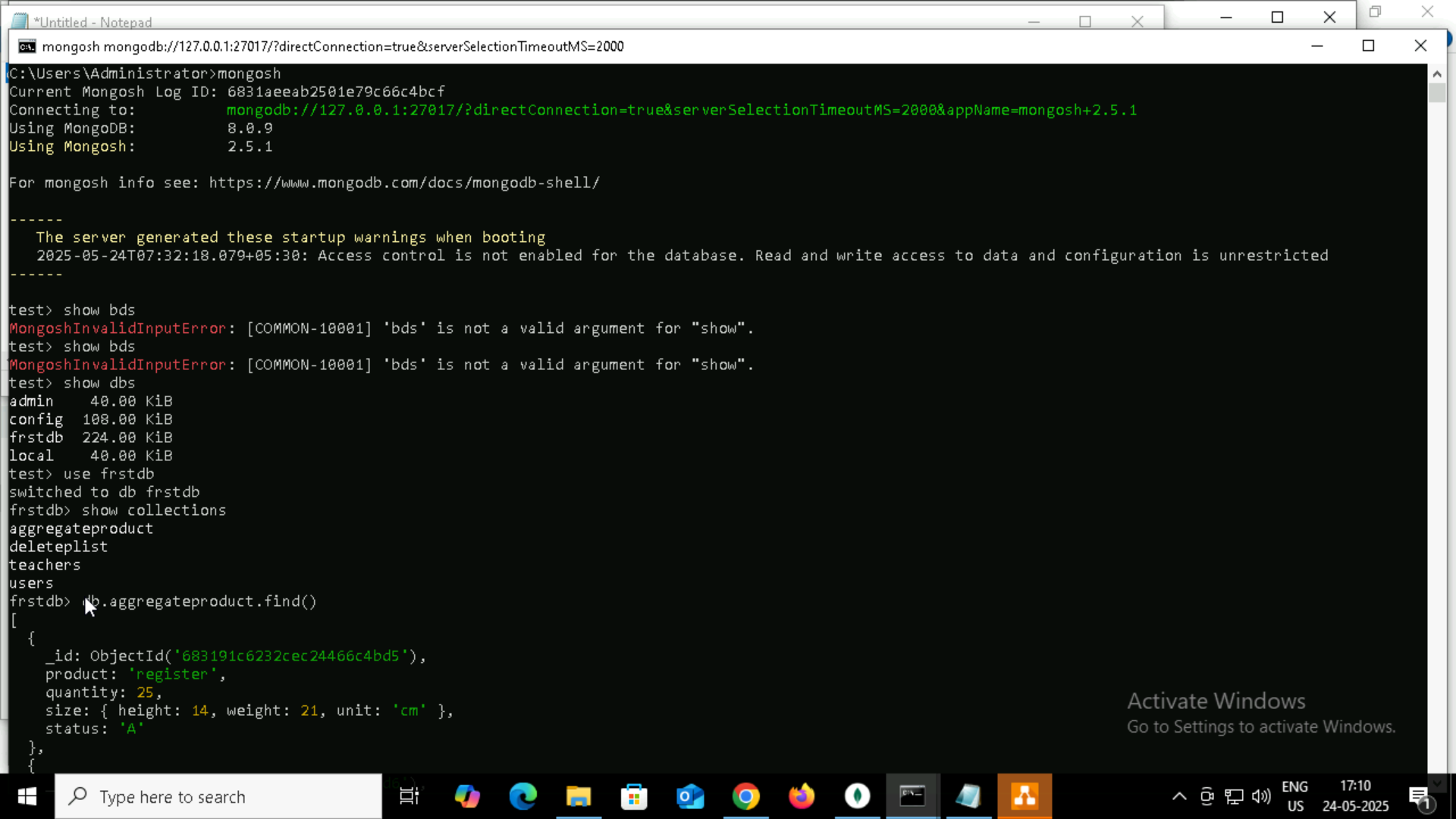
Installed MONGO DB.

Installed Mongo Shell.









Insert

use firstdb

# this will create db firstdb

#### create collection by 2 ways

db.students.insertOne({name: "kim", age: 12 , standard: "seventh"})

# This is going to create collection students with one document in it.

db.createCollection("teachers")

# This is going to create collection teachers specifically

db.teachers.insertMany([

{

name:"roger",

subject:"maths"

},

{

name:"anita",

subject:"chemistry"

},

{

name:"thomas",

subject:"physics"

},

{

name:"tina",

subject:"maths"

},

{

name:"jonty",

subject:"english"

}

]

)

db.teachers.find()

#this will show all the documents present in the teachers collection

db.teachers.find().forEach(printjson)

# This will print result in json format.

db.teachers.find({name : "roger"})

# this will find the document where name field is "roger"

db.cap2.insertMany(

[

{

\_id:14,

name:"sheena",

age:16,

standard:"junior"

},

{

\_id:15 ,

name:"kris",

age:18,

standard:"senior"

}

]

)

# This is going to insert 2 entries and since size is capped to 5 entries last entry from collection will get removed.

db.cap2.find()

# this will show only five entries and \_id is removed.

db.createCollection("convertcap1")

# create non capped collection

db.runCommand({"convertToCapped": "convertcap1",size : 10000})

#convert a collection to capped one

#Ordered bulk

var bulkOrdered = db.bulk1.initializeOrderedBulkOp()

bulkOrdered.insert({\_id:1, name:"Murugan"});

bulkOrdered.insert({\_id:2, name:"Niyas"});

bulkOrdered.insert({\_id:3, name:"Nitin"});

bulkOrdered.insert({\_id:1, name:"Murugan"});

bulkOrdered.insert({\_id:4, name:"Adarsh"});

bulkOrdered.execute();

Deleting:

db.deleteplist.insertMany( [

   { product: "register", quantity: 25, size: { height: 14, weight: 21, unit: "cm" }, status: "A" },

   { product: "book", quantity: 50, size: { height: 8.5, weight: 11, unit: "in" }, status: "A" },

   { product: "sheet", quantity: 100, size: { height: 8.5, weight: 11, unit: "in" }, status: "D" },

   { product: "timetable", quantity: 75, size: { height: 22.85, weight: 30, unit: "cm" }, status: "D" },

   { product: "envelop", quantity: 45, size: { height: 10, weight: 15.25, unit: "cm" }, status: "A" }

]);

db.deleteplist.deleteOne( { product: "sheet"} )

db.deleteplist.find( { product: "sheet"} )

db.deleteplist.deleteOne( { size: { height: 14, weight: 21, unit: "cm" } } )

db.deleteplist.find( { size: { height: 14, weight: 21, unit: "cm" } } )

db.deleteplist.deleteMany( { tags: ["red", "blank"] } )

db.deleteplist.deleteMany( { tags: "red" } )

Aggregarte:

db.aggregateproduct.insertMany( [

   { product: "register", quantity: 25, size: { height: 14, weight: 21, unit: "cm" }, status: "A" },

   { product: "book", quantity: 50, size: { height: 8.5, weight: 11, unit: "in" }, status: "A" },

   { product: "sheet", quantity: 100, size: { height: 8.5, weight: 11, unit: "in" }, status: "D" },

   { product: "timetable", quantity: 75, size: { height: 22.85, weight: 30, unit: "cm" }, status: "D" },

   { product: "envelop", quantity: 45, size: { height: 10, weight: 15.25, unit: "cm" }, status: "A" },

   { product: "register", quantity: 25, size: { height: 14, weight: 21, unit: "cm" }, status: "A" },

   { product: "book", quantity: 50, size: { height: 8.5, weight: 11, unit: "in" }, status: "A" },

   { product: "sheet", quantity: 100, size: { height: 8.5, weight: 11, unit: "in" }, status: "D" },

   { product: "timetable", quantity: 75, size: { height: 22.85, weight: 30, unit: "cm" }, status: "D" },

   { product: "envelop", quantity: 45, size: { height: 10, weight: 15.25, unit: "cm" }, status: "A" },

   { product: "register", quantity: 25, size: { height: 14, weight: 21, unit: "cm" }, status: "A" },

   { product: "book", quantity: 50, size: { height: 8.5, weight: 11, unit: "in" }, status: "A" }

]);

db.aggregateproduct.aggregate([

   { $match: { status: "A" } },

   { $group: { \_id: "$product", total: { $sum: "$quantity" } } }

])

db.aggregateproduct.aggregate([

   { $match: { status: "A" } },

   { $group: { \_id: "$product", total: { $sum: "$quantity" } } },

   {$sort : { total : 1 }

   }

])

db.aggregateproduct.aggregate([

   { $match: { status: "A" } },

   { $addFields: {

    maxWeight: { $max: "$size.weight" },

    minSizeHeight: { $min: "$size.height" }

} }

])

db.aggregateproduct.aggregate([

   { $match: { status: "A" } },

   { $addFields: {

    maxQ: { $max: "$quantity" },

    minSizeHeight: { $min: "$size.height" }

} }

])

db.aggregateproduct.aggregate([

   { $project : { product : 1,quantity : 1 } } , //optional line

   { $group: { \_id: "$product", total: { $sum: "$quantity" } } },

   {$sort : { total : 1 }

   }

])

firstdb> db.aggregateproduct.aggregate([

...    { $project : { product : 1,quantity : 1 } } ,

...    { $group: { \_id: "$product", total: { $sum: "$quantity" } } },

...    {$sort : { total : 1 }

...    }

... ])

[

  { \_id: 'register', total: 75 },

  { \_id: 'envelop', total: 90 },

  { \_id: 'book', total: 150 },

  { \_id: 'timetable', total: 150 },

  { \_id: 'sheet', total: 200 }

]

firstdb> db.aggregateproduct.aggregate([{ $group: { \_id: "$product", total: { $sum: "$quantity" } } }, { $sort: { total: 1 } }] )

[

  { \_id: 'register', total: 75 },

  { \_id: 'envelop', total: 90 },

  { \_id: 'book', total: 150 },

  { \_id: 'timetable', total: 150 },

  { \_id: 'sheet', total: 200 }

]

db.collection.mapReduce(

mapfunction() # this give document with key value pairs

reducefunction() # reduces earlier key and value pair to single key and single value pair.

{

query # this is basically same query as we throw it in the find function.

output # end result the way we want to print

})

lets see an example

db.aggregateproduct.mapReduce(

function() {

 emit(this.product,1);

 },

 function(key, values) {

 return Array.sum(values);

 },

{

query:{status:"A"},

out: { merge: "map\_reduce\_result" },

}

)

db.map\_reduce\_result.find();

INDEXING:

db.collection.createIndex( <key and index type specification>, <options> )

db.indexcollection.insertMany( [

  { product: "register", status: "A", size: { height: 14, weight: 21, unit: "cm" }, instock: [ { warehouse: "A", quantity: 5 } ] , creationts: ISODate("2014-11-01T12:30:15Z")},

  { product: "book", status: "A",  size: { height: 8.5, weight: 11, unit: "in" }, instock: [ { warehouse: "C", quantity: 5 } ], creationts: ISODate("2016-11-01T12:30:15Z") },

  { product: "sheet", status: "D", size: { height: 8.5, weight: 11, unit: "in" }, instock: [ { warehouse: "A", quantity: 60 } ] , creationts: ISODate("2017-11-01T12:30:15Z")},

  { product: "timetable", status: "D", size: { height: 22.85, weight: 30, unit: "cm" }, instock: [ { warehouse: "A", quantity: 40 } ] , creationts: ISODate("2018-11-01T12:30:15Z")},

  { product: "envelop", status: "A", size: { height: 10, weight: 15.25, unit: "cm" }, instock: [ { warehouse: "B", quantity: 15 }, { warehouse: "C", quantity: 35 } ], creationts: ISODate("2019-11-01T12:30:15Z") },

  { product: "register", status: "A", size: { height: 14, weight: 21, unit: "cm" }, instock: [ { warehouse: "A", quantity: 5 } ] , creationts: ISODate("2014-11-01T12:30:15Z")},

  { product: "register", status: "A",  size: { height: 8.5, weight: 11, unit: "in" }, instock: [ { warehouse: "C", quantity: 5 } ], creationts: ISODate("2016-11-01T12:30:15Z") },

  { product: "register", status: "D", size: { height: 8.5, weight: 11, unit: "in" }, instock: [ { warehouse: "A", quantity: 60 } ] , creationts: ISODate("2017-11-01T12:30:15Z")},

  { product: "book", status: "D", size: { height: 22.85, weight: 30, unit: "cm" }, instock: [ { warehouse: "A", quantity: 40 } ] , creationts: ISODate("2018-11-01T12:30:15Z")},

  { product: "book", status: "A", size: { height: 10, weight: 15.25, unit: "cm" }, instock: [ { warehouse: "B", quantity: 15 }, { warehouse: "C", quantity: 35 } ], creationts: ISODate("2019-11-01T12:30:15Z") }

]);

db.indexcollection.getIndexes()

db.indexcollection.dropIndexes()

db.indexcollection.createIndex( { product: 1 } )

db.indexcollection.getIndexes()

db.largecollectionforindex.createIndex( { product: 1 } )

db.indexcollection.createIndex( { "size.height": 1 } )

db.indexcollection.find( { "size.height" : 8.5 } )

db.indexcollection.createIndex( { size: 1 } )

db.indexcollection.find( {size: { height: 10, weight: 15.25, unit: "cm" } } )

db.collection.createIndex( { <field1>: <type>, <field2>: <type2>, ... } )

db.indexcollection.createIndex( { status : 1 , "size.height": 1 } )

db.indexcollection.getIndexes()

db.indexcollection.find( { product: 1 , "size.height": 1 } )

db.indexcollection.find( { product: -1 , "size.height": -1 } )

but below is not supported well

db.indexcollection.find( { product: 1 , "size.height": -1 } )

or

db.indexcollection.find( { product: -1 , "size.height": 1 } )

db.indexcollection.createIndex( { instock: 1 } )

db.indexcollection.find( { instock : [ { warehouse: "B", quantity: 15 }, { warehouse: "C", quantity: 35 } ] } )

db.indexcollection.createIndex( { product : 1, instock: 1 } )

db.indexcollection.createIndex( { \_id: "hashed" } )

db.indexcollection.find( { instock: [ { warehouse: "B", quantity: 15 } ] } )

db.indexcollection.createIndex( { product: "text" } )

db.indexcollection.createIndex( { product: "text" , status : "text" } )

db.indexcollection.createIndex( { "$\*\*": "text" } )

db.indexcollection.createIndex( {"size.weight" : 1,  "$\*\*": "text" } )

db.collection.createIndex( { <location field> : "2dsphere" } )

db.geoplaces.insertMany([

{

    name: "Muncipal Park",

   location: { type: "Point", coordinates: [ -73.97, 40.77 ] },

   category: "Parks"

} ,

{

   name: "City Park",

   location: { type: "Point", coordinates: [ -73.9928, 40.7193 ] },

   category: "Parks"

},

{

   name: "kannedy Grounds",

   location: { type: "Point", coordinates: [ -73.9375, 40.8303 ] },

   category: "Stadiums"

}

]

);

db.geoplaces.find(

   {

     location:

       { $near:

          {

            $geometry: { type: "Point",  coordinates: [ -73.9667, 40.78 ] },

            $minDistance: 1000,

            $maxDistance: 5000

          }

       }

   }

)

db.geoplaces.createIndex( { location: "2dsphere" } )

Regex

db.regex.insertMany([{

name: "Tony",

position: "Backend developer"

}

{

name: "Bruce",

position: "frontend developer"

}

{

name: "Nick",

position: "HR Manager"

}

])

db.geoplaces.find({

{ <field>: { $regex: /pattern/, $options: '<options>' } }

{ <field>: { $regex: 'pattern', $options: '<options>' } })

