MongoDB basic

**1.** Mongodb তে একটা ডাটা insert করানোর জন্য insertOne( ) ফাংশান ব্যবহার করা হয়। যেমনঃ

db.test.insertOne({name : 'Jonathan'})

**2.** Mongodb তে একের অধিক বা অনেকগুলো ডাটা insert করার জন্য insertMany( ) ফাংশান ব্যবহার করা হয়। যেমনঃ

db.test.insertMany([

{name : 'abir rahman'},

{name : 'shafiqul islam'}

])

3. specific কিছু ডাটা পাওয়ার জন্য আমরা field filtering ব্যবহার করতে পারি। উদাহরনঃ

db.test.find({age : 17}, {name : 1, address : 1})

এখানে আমরা কোন ডাটাবেজের age : 17 এর সাথে মিল পাওয়া সকল ডাটা পাবো। কিন্ত পরের ব্রাকেটে {name : 1, address : 1}) দেওয়ার কারণে age : 17 এর সাথে মিল পাওয়া ডাটাগুলোর শুধুমাত্র name এবং address ডাটাগুলো পাবো।

উপরের কোডটি আমরা project মেথড ব্যবহার করেও করতে পারি। তবে project মেথড শুধু find এর ক্ষেত্রে কাজ করে। findOne এর ক্ষেত্রে কাজ করেনা।

db.test.find({age : 17}).project({name : 1, address : 1})

3. mongodb operator দিয়েও আমরা ফিল্ড ফিল্টারিং করতে পারি। যেমনঃ

$eq :

db.test.find({age : {$eq : 18}}) //-- give the all data that matches age=18

$gt :

db.test.find({age : {$gt : 18}}) //-- give the all data that matches age>18

$gte :

db.test.find({age : {$gte : 18}}) //-- give the all data that matches age>=18

$lt :

db.test.find({age : {$lt : 18}}) //-- give the all data that matches age<18

$lte :

db.test.find({age : {$lte : 18}}) //-- give the all data that matches age<=18

sort :

db.test.find({age : {$eq : 18}}).sort({birthday : 1}) //-- give the all data that matches age=18 and will be sorted by Birthday in ascending order

4. $in এর ব্যবহারঃ

**db.test.find({age : {$gt : 18, $lt : 40}}, {age : 1}).sort({age:1})**

//-- give the all data that matches

age : {$gt : 18, $lt : 40} = //-- age 18 এর উপরে এবং 40 এর নিচের ডাটাগুলো দেখাবে ।

{age : 1} = //-- শুধু মাত্র age ডাটা দেখাবে।

.sort({age:1})= //-- ডাটাগুলো age অনুসারে ascending order এ sorting হয়ে দেখাবে।

-------------------------

//-- same code with one more field

**db.test.find({gender :'female' ,age : {$gt : 18, $lt : 40}}, {age : 1}).sort({age:1})**

{gender :'female' ,age : {$gt : 18, $lt : 40}} //-- age 18 এর উপরে এবং 40 এর, এবং শুধুমাত্র female ডাটাগুলো দেখাবে ।

-------------------------

**db.test.find(**

**{age : {$in: [18,19,20,22,32]}},**

**{age : 1})**

**.sort({age:1}**

**)**

{age : {$in: [18,19,20,22,32]}} //-- যেসব ডাটার age এর ভ্যালু 18, 19, 20, 22, 32 সেগুলো শো করবে।

-------------------------

**db.test.find(**

**{age : {$nin: [18,19,20,22,32]}},**

**{age : 1})**

**.sort({age:1}**

**)**

{age : {$nin: [18,19,20,22,32]}} //-- যেসব ডাটার age এর ভ্যালু 18, 19, 20, 22, 32 সেগুলো বাদে আর সবগুলো ডাটা শো করবে।

-------------------------

db.test.find(

{

age: { $nin: [18, 19, 20, 22, 32] },

interests: 'Writing'

},

{ age: 1 })

.sort({ age: 1 }

)

interests: 'Writing' //-- যেসব ডাটার age এর ভ্যালু 18, 19, 20, 22, 32 সেগুলো , সেইসাথে সেই ডাটাগুলোর মধ্যে যেগুলাতে interests ফিল্ডে ‘Writing’ আছে সেগুলো শো করবে।

-------------------------

db.test.find(

{

age: { $nin: [18, 19, 20, 22, 32] },

interests: { $in: ['Writing', 'Cooking'] }

},

{

age: 1,

interests : 1

})

.sort({ age: 1 }

)

interests: 'Writing' //-- যেসব ডাটার age এর ভ্যালু 18, 19, 20, 22, 32 সেগুলো , সেইসাথে সেই ডাটাগুলোর মধ্যে যেগুলাতে interests ফিল্ডে ‘Writing’ অথবা ‘Cooking’ যে কোন একটি অথবা ২টিই আছে সেগুলো শো করবে।

-------------------------

5. $and, $or, implicit vs explicit :

db.test.find({

$and: [

{ age: { $gt: 3, $lt: 30 } },

{ gender: 'Female' }

]

}).project({

age: 1,

gender: 1

}).sort(

{ age: 1 }

)

$and: [

{ age: { $gt: 3, $lt: 30 } },

{ gender: 'Female' }

]

//-- $and অপারেটর দ্বারা বুঝাচ্ছে এখানে [ ] ব্রাকেটের ভেতর থাকা সবগুলো শর্ত পূরণ করা ডাটাগুলো দেখাবে।

//-- তদ্রুপভাবে এখানে $and এর জায়গায় $or থাকলে বোঝাতো [ ] ব্রাকেটের মধ্যে থাকা যেকোন একটা শর্ত পূরণ করা ডাটাগুলো দেখাবে। অবশ্য $or এর কাজ আমরা $in দিয়েও করতে পারি যেটা উপরে করেছি। তবে $in শুধুমাত্র একই ফিল্ডের ক্ষেত্রে কাজ করে।

-------------------------

6. array উপর mongodb অপারেশানঃ

একটি array-র যেকোন একটা ডাটা অনুযায়ী সার্চের নিয়মঃ

db.test.find({ interests: 'Cooking' })

.project({

interests: 1

})

এখানে "interests" : [ "Cooking", "Writing", "Reading" ] এর মত ডাটার ভেতর cooking থাকলেই সেগুলো শো করবে।

আর যদি এরকম লিখিঃ

db.test.find({ interests: [ "Cooking", "Writing", "Reading" ] })

.project({

interests: 1

})

তবে interests এরের মধ্যে ঠিক **[ "Cooking", "Writing", "Reading" ]** ঠিক এইভাবে সিরিয়াল অনুযায়ী সাজানো আছে যে ডাটাগুলো , শুধু সেগুলো দেখাবে। এখন আমরা যদি চাই জাস্ট এই ৩টা ডাটা থাকলেই হবে , সিরিয়াল যেমন ইচ্ছা তেমন থাকুক তবে সেটা $all দিয়ে করতে হবে ,

db.test.find({ interests: { $all : [ "Cooking", "Writing", "Reading" ]} })

.project({

interests: 1

})

-------------------------

7. array of objects [{ }] এর উপর mongodb অপারেশানঃ

db.test.find(

{

"skills.name" : 'JAVASCRIPT'

})

.project({

skills : 1

})

উপরের কোডটি skills নামক array-র ভেতর যেসব অবজেক্ট আছে সেসব অবজেক্টের মধ্যে name প্রপার্টিতে JAVASCRIPT পেলেই সেগুলো শো করবে।

-------------------------

db.test.find(

{

skills: {

"name": 'JAVA',

"level": 'Expert',

"isLearning": false

}

})

.project({

skills: 1

})

উপরের কোডটি এরের মত এক্স্যাক্ট সিরিয়াল অনুযায়ী ডাটা দিবে । অর্থাৎ যে ডাটাগুলোতে

skills: {

"name": 'JAVA',

"level": 'Expert',

"isLearning": false

}

সিরিয়াল মত থাকবে, শুধু সেই ডাটাগুলোই দিবে। কিন্ত আমরা যদি চাই এখানে array এর $all মেথডের মত করে ডাটা শো করতে অর্থাৎ query তে দেওয়া ডাটাগুলো থাকলেই হলো, সিরিয়াল মেইন্টেইন ম্যান্ডাটরি না , তবে আমরা $elemMatch অপারেটর ব্যবহার করতে পারি,

db.test.find(

{

skills: {

$elemMatch: {

"level": 'Expert',

"name": 'JAVA'

}

}

})

.project({

skills: 1

})

-------------------------

8. Update or edit data on mongodb :

কোন ডাটাকে অন্য কোন ডাটা দিয়ে replace করার জন্য $and ব্যবহার করা হয় ।

db.test.updateOne(

{\_id : ObjectId("6406ad65fc13ae5a400000c7")},

{$set : {

friends : ['abir rahman', 'shafiqul islam', 'jonathan wick']

}}

)

Note :

1. এটি সম্পূর্ন ডাটাকে রিপ্লেস করবে, ডাটা টাইপ সহ।
2. একই ডাটা আবার দিলে সেটা রিপ্লেস হবেনা। আগেরটাই থাকবে।

যেহেতু $set সম্পূর্ণ ডাটা রিপ্লেস করে, কিন্ত আমরা চাই সম্পূর্ণ রিপ্লেস না হয়ে আগের ডাটার সাথে নতুন ডাটা এড হবে সেক্ষেত্রে $addToSet মেথড ব্যবহার করতে পারি ।

db.test.updateOne(

{\_id : ObjectId("6406ad65fc13ae5a400000c7")},

{$addToSet : {

friends : 'jonathan' // this will add data to db like 'push' method in array

}}

)

এই মেথডও কিন্ত $set মত করে ডুপ্লিকেট ডাটা এড করবেনা । তবে এই মেথড দিয়ে আবার একের অধিক ডাটা এড করা যাবেনা। একের অধিক ডাটা এড করার জন্য যদি [‘rasel’, ‘jonathan’, ‘cash’] ব্যবহার করি তবে সেটা এরে হিসেবে আগের ডাটার সাথে এড হয়ে যাবে। এখন আমরা একের অধিক ডাটা এড করতে চাইলে, $addToSet এর সাথে $each মেথড ব্যবহার করতে পারি।

db.test.updateOne(

{\_id : ObjectId("6406ad65fc13ae5a400000c7")},

{$addToSet : {

friends : {$each : ['rasel', 'jonathan', 'cash']}

}}

)

তবে এটিতেও ডুপ্লিকেট ডাটা এড হবেনা। সেক্ষেত্রে $push মেথড ব্যবহার করতে হবে।

db.test.updateOne(

{\_id : ObjectId("6406ad65fc13ae5a400000c7")},

{$push : {

friends : {$each : ['rasel', 'jonathan', 'cash']}

-------------------------

8. Update or edit data for object , array of object:

**For object :**

db.test.updateOne(

{\_id : ObjectId("6406ad65fc13ae5a400000c7")},

{$set : {

"address.street" : 'North Shialkathi',

"address.city" : 'Bhandaria'

}}

)

**For Array of Object :**

db.test.updateOne(

{\_id : ObjectId("6406ad65fc13ae5a400000c7"), "education.major" : "Philosophy"},

{$set : {

"education.$.major" : 'Zoology'

}}

)

-------------------------

10. delete :

Delete with $unset :

db.test.updateOne(

{\_id : ObjectId("6406ad65fc13ae5a400000c7")},

{$unset : {phone : 1}} //-- {phone : ‘’} will work too

)

Delete last item from an array with $pop:

db.test.updateOne(

{\_id : ObjectId("6406ad65fc13ae5a400000c7")},

{$pop : {interests : 1}}

)

Delete first item from an array with $pop :

db.test.updateOne(

{\_id : ObjectId("6406ad65fc13ae5a400000c7")},

{$pop : {interests : -1}}

)

Delete specific item by name from an array with $pull :

db.test.updateOne(

{\_id : ObjectId("6406ad65fc13ae5a400000c7")},

{$pull : {friends : "cash"}}

)

Delete a bunch of specific items by name from an array with $pullAll :

db.test.updateOne(

{\_id : ObjectId("6406ad65fc13ae5a400000c7")},

{$pullAll : {friends : ['rasel', 'jonathan']}}

)

-------------------------

10. aggregation :

db.test.aggregate([

// stage - 1

{$match: {gender : 'Male', age : {$lt : 30}}},

// stage - 2

{$project : {gender : 1, name : 1}}

])

In the aggregation stage $match will work like .find( ) method in mongodb.

{$match: {gender : 'Male', age : {$lt : 30}}} will return the data that matches ‘Male’ gender and age<30 in database.

{$project : {gender : 1, name : 1}} will show the data of gender and name only .

-------------------------

11. aggregation :

db.test.aggregate([

// stage - 1

{$match: {gender : 'Male', age : {$lt : 30}}},

// stage - 2

{$addFields: {games : 'Mobile Legends', city: 'barisal'}},

// stage - 3

// {$project : {games : 1, city : 1}},

// stage - 4

{$out : "gamer-group"}

])

Explanation for this aggregation :

stage-1 এ আমরা মংগোডিবি ডাটাবেজ এর test নামক কালেকশান থেকে gender : Male এবং age ৩০ এর নিচে যে ডাটাগুলো আছে সেগুলোকে বের করে আনবো।

stage-2 তে আমরা বের করে আনা ডাটাগুলোর সাথে games এবং city নামক নতুন ফিল্ড add করবো। ।

stage-4 : তবে নতুন এড হওয়া ডাটাগুলো test নামক কালেকশানের সাথে এড হবেনা। নতুন কালেকশান বানিয়ে সেখানে এড করতে হবে। সেই নতুন কালেকশান বানিয়ে সেখানে এড করার কাজটাই আমরা stage-4 এ করেছি।

এখন আমরা যদি নতুন করে এড হওয়া ডাটাগুলো test নামক কালেকশানে এড করতে চাই সেক্ষেত্রে merge অপারেটর ব্যবহার করতে হবে।

db.test.aggregate([

// stage - 1

{$match: {gender : 'Male', age : {$lt : 30}}},

// stage - 2

{$addFields: {games : 'Mobile Legends', city: 'barisal'}},

// stage - 3

// {$project : {games : 1, city : 1}},

// stage - 4

{$merge : "test"}

])

-------------------------

12. group :

db.test.aggregate([

// stage - 1

{$group : {\_id : '$gender', count : {$sum: 1}}}

])

Explanation :

উপরের কোডটি test নামক কালেকশান থেকে gender নামে যতগুলো প্রপার্টি আছে সবগুলোকে নিয়ে group করে ডাটা শো করবে। যেমন test নামক কালেকশানে gender প্রোপার্টিতে male, female , polygender ইত্যাদি নামক gender আছে। এখন $group প্রোপার্টি যেটা করবে সেটা হলো, male গুলোকে নিয়ে একটি গ্রুপ করবে , female গুলোকে নিয়ে একটি গ্রুপ করবে, polygender নিয়ে একটি গ্রুপ করবে , এভাবে প্রত্যেকটা আলাদা ভ্যালু নিয়ে গ্রুপ করবে। এবং count : {$sum: 1} একেকটি গ্রুপে ঐ ডাটাগুলো কয়টি করে আছে সেগুলো কাউন্ট করে শো করবে।

যেমন

{

"\_id" : "Female",

"count" : 52

}

{

"\_id" : "Polygender",

"count" : 1

}

{

"\_id" : "Male",

"count" : 38

}

-------------------------

12. group, $push, $$root :

$group : {

\_id : '$address.country',

count : {$sum: 1},

showMedetails : {

$push : '$email'

}

}

এখানে কান্ট্রি নামক প্রোপার্টির উপর ভিত্তি করে গ্রুপ করা হয়েছে । showmeDetails নামক একটি প্রোপার্টি নতুন করা হয়েছে যেখানে ওই গ্রুপভিত্তিক ইমেইলগুলা এরে আকারে থাকবে। যেমন

{

"\_id" : "Russia",

"count" : 6,

"showMedetails" : [

"cwhitechurch6@facebook.com",

"jlye9@hhs.gov",

"klovittr@oaic.gov.au",

"sjaffa14@soup.io",

"dcolwell29@tuttocitta.it",

"dbeidebeke2b@reference.com"

]

},

{

"\_id" : "Sweden",

"count" : 5,

"showMedetails" : [ "ojesson13@springer.com", "vdunstone1e@de.vu", "kbuckthorpe1j@house.gov", "ohuyton1p@mashable.com", "dmanson2n@godaddy.com" ]

},

{

"\_id" : "Philippines",

"count" : 4,

"showMedetails" : [ "cpeaker5@nationalgeographic.com", "lfullicks1a@jimdo.com", "censtone20@foxnews.com", "javes24@disqus.com" ]

},

এখন এখানে যেহেতু showMedetails : {$push : '$email'} এ email দিয়েছি তাই showMedetails প্রোপার্টির এরে তে ওইগ্রুপে থাকা ইউজারদের ইমেইলগুলো দেখাচ্ছে কিন্তু আমরা যদি চাই ইউজারের পুরো ডিটেইলস মানে পুরো প্রোপার্টিই দেখাবে, তাহলে $$root ইউজ করতে হবে।

$group : {

\_id : '$address.country',

count : {$sum: 1},

showMedetails : {

$push : '$$ROOT'

}

}

-------------------------

13. group:

db.test.aggregate([

// stage - 1

{

$group : {

\_id : null

}}

])

$group : { \_id:null } দ্বারা আমরা টোটাল কালেকশানকে একটি আইডির আন্ডারে নিয়ে এসেছি। এতে করে আমরা খুব সহজে কিছু ম্যাথম্যাটিকাল ক্যালকুলেশান করতে পারি। যেমন এই কালেকশানে থাকা সকল ইউজারের টোটাল স্যালারির যোগফল বের করতে পারিঃ

$group : {

\_id : null ,

totalSalary : {$sum : "$salary"}

}

ঠিক এইভাবে তাদের ম্যাক্স স্যালারি, মিনিমাম স্যালারি, অথবা এভারেজ স্যালারিও বের করতে পারিঃ

$group : {

\_id : null ,

totalSalary : { $sum : "$salary" },

maxSalary : { $max : "$salary"},

minSalary : { $min : "$salary"},

avgSalary : { $avg : "$salary"}

}

$project দিয়েও আমরা কিছু ক্যালকুলেশান করতে পারি, অথবা প্রোপার্টির নাম চেঞ্জ করতে পারিঃ

db.test.aggregate([

// stage - 1

{

$group : {

\_id : null ,

totalSalary : { $sum : "$salary"},

maxSalary : { $max : "$salary"},

minSalary : { $min : "$salary"},

avgSalary : { $avg : "$salary"}

}},

// stage - 2

{

$project :{

totalSalary : 1,

maxSalary : 1,

minSalary : 1,

averageSalary : "$avgSalary",

rangebetweenMaxMin : {$subtract: ["$maxSalary","$minSalary"]}

}

}

])

averageSalary : "$avgSalary" দিয়ে avgSalary প্রোপার্টির নাম rename করেছি ।

rangebetweenMaxMin : {$subtract: ["$maxSalary","$minSalary"]} দিয়ে মিনিমাম স্যালারি এবং ম্যাক্সিমাম স্যালারির পার্থক্য বের করেছি।

-------------------------

14. group for Array:

Array এর উপর গ্রুপ করতে গেলে আমাদেরকে $unwind ব্যবহার করতে হবে। $unwind ব্যবহার করলে ডাটার যেকোন প্রোপার্টির এরের ভেতর থাকা প্রত্যেকটা আলাদা ইলেমেন্টকে নিয়ে আলাদা আলাদা গ্রুপ হবে।

db.test.aggregate([

// stage - 1

{

$unwind: "$friends",

},

// stage - 2

{

$group : {\_id : "$friends", count : {$sum : 1} }

}

])

-------------------------

14. $bucket:

db.test.aggregate([

// stage - 1

{

$bucket: {

groupBy: "$age",

boundaries: [20, 40, 60],

default: "More than 60",

output: {

count: { $sum: 1 },

whoAreInList: { $push: "$age" }

}

}

},

// stage - 2

{

$sort : {count : 1}

},

// stage - 3

{

$limit : 1

},

{

$project : {

count : 1

}

}

])

-------------------------

14. $facet:

কয়েকটা পাইপলাইনকে একটা কোডের ভেতর লিখতে হলে সেটা $facet দিয়ে লিখতে হয়ঃ

db.test.aggregate([

{

$facet: {

//--- pipeline - 1

"friendsCount": [

// stage-1

{ $unwind: "$friends" },

// stage-2

{ $group: { \_id: "$friends", count: { $sum: 1 } } }

],

//--- pipeline - 2

"educationCount": [

// stage - 1

{$unwind : "$education"},

// stage-2

{$group : {\_id : "$education", count: { $sum:1 }}}

],

//--- pipeline - 3

"skillsCount" : [

// stage-1

{$unwind: "$skills"},

// stage-2

{$group :{\_id : "$skills", count:{$sum:1}}}

]

}

}

])