Aggregation Commands :

{

\_id: ObjectId(7df78ad8902c)

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by\_user: 'tutorials point',

url: 'http://www.abc.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

},

{

\_id: ObjectId(7df78ad8902d)

title: 'NoSQL Overview',

description: 'No sql database is very fast',

by\_user: 'tutorials point',

url: 'http://www.abc.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 10

},

{

\_id: ObjectId(7df78ad8902e)

title: 'Neo4j Overview',

description: 'Neo4j is no sql database',

by\_user: 'Neo4j',

url: 'http://www.abc.com',

tags: ['neo4j', 'database', 'NoSQL'],

likes: 750

},

db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : 1}}}])

|  |  |  |
| --- | --- | --- |
| **Expression** | **Description** | **Example** |
| $sum | Sums up the defined value from all documents in the collection. | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : "$likes"}}}]) |
| $avg | Calculates the average of all given values from all documents in the collection. | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$avg : "$likes"}}}]) |
| $min | Gets the minimum of the corresponding values from all documents in the collection. | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$min : "$likes"}}}]) |
| $max | Gets the maximum of the corresponding values from all documents in the collection. | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$max : "$likes"}}}]) |
| $push | Inserts the value to an array in the resulting document. | db.mycol.aggregate([{$group : {\_id : "$by\_user", url : {$push: "$url"}}}]) |
| $addToSet | Inserts the value to an array in the resulting document but does not create duplicates. | db.mycol.aggregate([{$group : {\_id : "$by\_user", url : {$addToSet : "$url"}}}]) |
| $first | Gets the first document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage. | db.mycol.aggregate([{$group : {\_id : "$by\_user", first\_url : {$first : "$url"}}}]) |
| $last | Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage. | db.mycol.aggregate([{$group : {\_id : "$by\_user", last\_url : {$last : "$url"}}}]) |

group min max first laste sum avg

#index

exists

db.inventory.find( { qty: { $exists: **true**, $in: [ 5, 15 ] } } )

db.records.find( { a: { $exists: **true** } } )

in

{ "\_id" : 1, "location" : "24th Street",

"in\_stock" : [ "apples", "oranges", "bananas" ] }

{ "\_id" : 2, "location" : "36th Street",

"in\_stock" : [ "bananas", "pears", "grapes" ] }

{ "\_id" : 3, "location" : "82nd Street",

"in\_stock" : [ "cantaloupes", "watermelons", "apples" ] }

db.fruit.aggregate([

{

$project: {

"store location" : "$location",

"has bananas" : {

$in: [ "bananas", "$in\_stock" ]

}

}

}

])

{ $in: [ 2, [ 1, 2, 3 ] ] } true

{ $in: [ "abc", [ "xyz", "abc" ] ] } true

{ $in: [ "xy", [ "xyz", "abc" ] ] } false

{ $in: [ [ "a" ], [ "a" ] ] } false

{ $in: [ [ "a" ], [ [ "a" ] ] ] } true

{ $in: [ /^a/, [ "a" ] ] } false

{ $in: [ /^a/, [ /^a/ ] ] }

All

{

\_id: ObjectId("5234cc89687ea597eabee675"),

code: "xyz",

tags: [ "school", "book", "bag", "headphone", "appliance" ],

qty: [

{ size: "S", num: 10, color: "blue" },

{ size: "M", num: 45, color: "blue" },

{ size: "L", num: 100, color: "green" }

]

}

{

\_id: ObjectId("5234cc8a687ea597eabee676"),

code: "abc",

tags: [ "appliance", "school", "book" ],

qty: [

{ size: "6", num: 100, color: "green" },

{ size: "6", num: 50, color: "blue" },

{ size: "8", num: 100, color: "brown" }

]

}

{

\_id: ObjectId("5234ccb7687ea597eabee677"),

code: "efg",

tags: [ "school", "book" ],

qty: [

{ size: "S", num: 10, color: "blue" },

{ size: "M", num: 100, color: "blue" },

{ size: "L", num: 100, color: "green" }

]

}

{

\_id: ObjectId("52350353b2eff1353b349de9"),

code: "ijk",

tags: [ "electronics", "school" ],

qty: [

{ size: "M", num: 100, color: "green" }

]

}

db.inventory.find( { tags: { $all: [ "appliance", "school", "book" ] } } )

db.inventory.find( { "qty.num": { $all: [ 50 ] } } )

elemMatch

{ \_id: 1, results: [ 82, 85, 88 ] }

{ \_id: 2, results: [ 75, 88, 89 ] }

db.scores.find(

{ results: { $elemMatch: { $gte: 80, $lt: 85 } } }

)

{ \_id: 1, results: [ { product: "abc", score: 10 }, { product: "xyz", score: 5 } ] }

{ \_id: 2, results: [ { product: "abc", score: 8 }, { product: "xyz", score: 7 } ] }

{ \_id: 3, results: [ { product: "abc", score: 7 }, { product: "xyz", score: 8 } ] }

db.survey.find(

{ results: { $elemMatch: { product: "xyz", score: { $gte: 8 } } } }

)

db.survey.find(

{ results: { $elemMatch: { product: "xyz" } } }

)

db.survey.find(

{ "results.product": "xyz" }

)

Match

{ "\_id" : ObjectId("512bc95fe835e68f199c8686"), "author" : "dave", "score" : 80, "views" : 100 }

{ "\_id" : ObjectId("512bc962e835e68f199c8687"), "author" : "dave", "score" : 85, "views" : 521 }

{ "\_id" : ObjectId("55f5a192d4bede9ac365b257"), "author" : "ahn", "score" : 60, "views" : 1000 }

{ "\_id" : ObjectId("55f5a192d4bede9ac365b258"), "author" : "li", "score" : 55, "views" : 5000 }

{ "\_id" : ObjectId("55f5a1d3d4bede9ac365b259"), "author" : "annT", "score" : 60, "views" : 50 }

{ "\_id" : ObjectId("55f5a1d3d4bede9ac365b25a"), "author" : "li", "score" : 94, "views" : 999 }

{ "\_id" : ObjectId("55f5a1d3d4bede9ac365b25b"), "author" : "ty", "score" : 95, "views" : 1000 }

db.articles.aggregate(

[ { $match : { author : "dave" } } ]

);

db.articles.aggregate( [

{ $match: { $or: [ { score: { $gt: 70, $lt: 90 } }, { views: { $gte: 1000 } } ] } },

{ $group: { \_id: **null**, count: { $sum: 1 } } }

] );

--cond --may be