**MongoDB**

https://www.geeksforgeeks.org/mongodb-tutorial/

use <database\_name>

show dbs

db.createCollection(“posts”)

db.student.find().count()

db.student.find().limit(2).pretty()

db.student.find({studentId:1}).size()

db.student.find().sort({studentId:-1}).pretty()

db.student.find().toArray()

db.student.find().next()

db.student.insertOne({Name: "Akshay", Marks: 500})

db.student.insertMany([{name:"Ajay",age:20},

{name:"Bina",age:24},

{name:"Ram",age:23}])

db.students.updateOne(

{ name: "Alice" },

{ $set: { age: 26 } }

)

### **Insert a New Document if No Match Exists (Upsert)**

db.students.updateOne(

{ name: "Charlie" },

{ $set: { age: 30 } },

{ upsert: true }

)

**Aggregate fundtions:**

count()

distinct()

$Group:

db.users.aggregate([

{ $group: { \_id: "$city", averageAge: { $avg: "$age" } } }

])

$project:

Merge

|  |
| --- |
| db.sales.aggregate( [ |
| { $project: { \_id: 0 } }, |
| { $merge : { into : "newCollection" } } |
| ] ) |

Upsert:

db.employee.findAndModify({

query:{name:"Ram"},

update:{$set:{department:"Development"}},

upsert:true

})

$match:

db.users.aggregate([

{ $match: { age: { $gt: 30 } } }

])

$sort:

db.users.aggregate([

{ $sort: { age: 1 } }

])

$limit:

db.users.aggregate([

{ $limit: 2 }

])

Pipeline

|  |
| --- |
| db.orders.insertMany( [ |
| { \_id: 0, name: "Pepperoni", size: "small", price: 19, |
| quantity: 10, date: ISODate( "2021-03-13T08:14:30Z" ) }, |
| { \_id: 1, name: "Pepperoni", size: "medium", price: 20, |
| quantity: 20, date : ISODate( "2021-03-13T09:13:24Z" ) }, |
| { \_id: 2, name: "Pepperoni", size: "large", price: 21, |
| quantity: 30, date : ISODate( "2021-03-17T09:22:12Z" ) }, |
| { \_id: 3, name: "Cheese", size: "small", price: 12, |
| quantity: 15, date : ISODate( "2021-03-13T11:21:39.736Z" ) }, |
| { \_id: 4, name: "Cheese", size: "medium", price: 13, |
| quantity:50, date : ISODate( "2022-01-12T21:23:13.331Z" ) }, |
| { \_id: 5, name: "Cheese", size: "large", price: 14, |
| quantity: 10, date : ISODate( "2022-01-12T05:08:13Z" ) }, |
| { \_id: 6, name: "Vegan", size: "small", price: 17, |
| quantity: 10, date : ISODate( "2021-01-13T05:08:13Z" ) }, |
| { \_id: 7, name: "Vegan", size: "medium", price: 18, |
| quantity: 10, date : ISODate( "2021-01-13T05:10:13Z" ) } |
| ] ) |

### **Calculate Total Order Quantity**

The following aggregation pipeline example contains two [stages](https://www.mongodb.com/docs/manual/reference/operator/aggregation-pipeline/#std-label-aggregation-pipeline-operator-reference) and returns the total order quantity of medium size pizzas grouped by pizza name:

|  |
| --- |
| db.orders.aggregate( [ |
|  |
| *// Stage 1: Filter pizza order documents by pizza size* |
| { |
| $match: { size: "medium" } |
| }, |
|  |
| *// Stage 2: Group remaining documents by pizza name and calculate total quantity* |
| { |
| $group: { \_id: "$name", totalQuantity: { $sum: "$quantity" } } |
| } |
|  |
| ] ) |

The [$match](https://www.mongodb.com/docs/manual/reference/operator/aggregation/match/#mongodb-pipeline-pipe.-match) stage:

* Filters the pizza order documents to pizzas with a size of medium.
* Passes the remaining documents to the [$group](https://www.mongodb.com/docs/manual/reference/operator/aggregation/group/#mongodb-pipeline-pipe.-group) stage.

The [$group](https://www.mongodb.com/docs/manual/reference/operator/aggregation/group/#mongodb-pipeline-pipe.-group) stage:

* Groups the remaining documents by pizza name.
* Uses [$sum](https://www.mongodb.com/docs/manual/reference/operator/aggregation/sum/#mongodb-group-grp.-sum) to calculate the total order quantity for each pizza name. The total is stored in the totalQuantity field returned by the aggregation pipeline.

Example output:

|  |
| --- |
| [ |
| { \_id: 'Cheese', totalQuantity: 50 }, |
| { \_id: 'Vegan', totalQuantity: 10 }, |
| { \_id: 'Pepperoni', totalQuantity: 20 } |
| ] |

### **Calculate Total Order Value and Average Order Quantity**

The following example calculates the total pizza order value and average order quantity between two dates:

|  |
| --- |
| db.orders.aggregate( [ |
|  |
| *// Stage 1: Filter pizza order documents by date range* |
| { |
| $match: |
| { |
| "date": { $gte: new ISODate( "2020-01-30" ), $lt: new ISODate( "2022-01-30" ) } |
| } |
| }, |
|  |
| *// Stage 2: Group remaining documents by date and calculate results* |
| { |
| $group: |
| { |
| \_id: { $dateToString: { format: "%Y-%m-%d", date: "$date" } }, |
| totalOrderValue: { $sum: { $multiply: [ "$price", "$quantity" ] } }, |
| averageOrderQuantity: { $avg: "$quantity" } |
| } |
| }, |
|  |
| *// Stage 3: Sort documents by totalOrderValue in descending order* |
| { |
| $sort: { totalOrderValue: -1 } |
| } |
|  |
| ] ) |

The [$match](https://www.mongodb.com/docs/manual/reference/operator/aggregation/match/#mongodb-pipeline-pipe.-match) stage:

* Filters the pizza order documents to those in a date range specified using [$gte](https://www.mongodb.com/docs/manual/reference/operator/aggregation/gte/#mongodb-expression-exp.-gte) and [$lt.](https://www.mongodb.com/docs/manual/reference/operator/aggregation/lt/#mongodb-expression-exp.-lt)
* Passes the remaining documents to the [$group](https://www.mongodb.com/docs/manual/reference/operator/aggregation/group/#mongodb-pipeline-pipe.-group) stage.

The [$group](https://www.mongodb.com/docs/manual/reference/operator/aggregation/group/#mongodb-pipeline-pipe.-group) stage:

* Groups the documents by date using [$dateToString.](https://www.mongodb.com/docs/manual/reference/operator/aggregation/dateToString/#mongodb-expression-exp.-dateToString)
* For each group, calculates:
  + Total order value using [$sum](https://www.mongodb.com/docs/manual/reference/operator/aggregation/sum/#mongodb-group-grp.-sum) and [$multiply.](https://www.mongodb.com/docs/manual/reference/operator/aggregation/multiply/#mongodb-expression-exp.-multiply)
  + Average order quantity using [$avg.](https://www.mongodb.com/docs/manual/reference/operator/aggregation/avg/#mongodb-group-grp.-avg)
* Passes the grouped documents to the [$sort](https://www.mongodb.com/docs/manual/reference/operator/aggregation/sort/#mongodb-pipeline-pipe.-sort) stage.

The [$sort](https://www.mongodb.com/docs/manual/reference/operator/aggregation/sort/#mongodb-pipeline-pipe.-sort) stage:

* Sorts the documents by the total order value for each group in descending order (-1).
* Returns the sorted documents.

Example output:

|  |
| --- |
| [ |
| { \_id: '2022-01-12', totalOrderValue: 790, averageOrderQuantity: 30 }, |
| { \_id: '2021-03-13', totalOrderValue: 770, averageOrderQuantity: 15 }, |
| { \_id: '2021-03-17', totalOrderValue: 630, averageOrderQuantity: 30 }, |
| { \_id: '2021-01-13', totalOrderValue: 350, averageOrderQuantity: 10 } |
| ] |