# insertOne() Method

db.collection.insertOne(

<document>,

{

writeConcern: <document>,

bypassDocumentValidation: <boolean>

}

)

# insertOne() Method

db.student.insertOne({name: "David",age: 25, marks: 90})

2

# Syntax of insertMany() method

db.collection.insertMany(

<documents>,

{

writeConcern: <document>,

ordered: <boolean>

}

)

# Usage of insertMany() method

db.student.insertMany([{name: "Billy", age: 21, marks: 85},

{name: "Alaric", age: 24, marks: 75},

{name: "Henna", age: 22, marks: 80}])

3

db.student.insertOne({

\_id : 234,

info : [{"name": "Alice"},{"age": 29}],

courses : ["Python", "MongoDB"]

});

4)

# Syntax of distinct()

db.collection.distinct(field, query)

# Retrieve Distinct Values of Field

db.student.distinct("name")

5)

db.student.distinct("name").length

**Retrieve Distinct Array Values of Field**

db. student.distinct("course")

db. student.distinct("course").length

**4. Retrieve Distinct Values of Fields From Condition**

# Retrieve Distinct Values of Fields From Condition

db.student.distinct("name", {"age": {$gt: 23}})

**5. Retrieve Distinct Values using runCommand()**

6

DELETE & Remove

# etrieve Distinct Values of Fields Using Key

db.runCommand ( { distinct: "student", key: "gender" } )

db.student.deleteOne({ marks:70 })

db.student.deleteMany({ age:20 })

db.student.remove({ "\_id": 6 })

7

# Usage of OR operator in MongoDB

db.student.find({$or: [{name : "Mickel"}, {email : "elena@gmail.com"}]})

db.student.find( { "name" : "Elena" , $or : [{"age" : 23},{"age" : 24},{"age":20}] } )

db.student.find({$or: [{ age: {$lt: 23} },{ age: {$gt: 24} }]})

db.student.find({$or: [{course: {$in: ["C++", "MongoDB"]}}]})

db.student.find({$or: [{"personal.cell": 10918129},{"personal.city": "Austin"}]})

8

**Using the $match operator**

The $match stage is similar to the find() method for querying documents but is typically used within an aggregation pipeline.

db.student.aggregate( [

{ $match : { name : "Zack" } } ]);

# Usage of $in operator in the $match stage

db.student.aggregate([

{ $match: { course: { $in: ["MongoDB", "SQL"] } } }])

# Usage of $match operator with $gt and $lt operator

db.student.aggregate( [

{ $match : { age : { $gt : 21, $lt : 24 } } }

] );

# Usage of $match operator with logical operator

db.student.aggregate( [

{ $match: { $or: [ { marks: { $gt: 80, $lt: 90 } },

{ age: { $gte: 24 } } ] } }

] );

# Usage of $match operator with the $group operator

db.student.aggregate([

{ $match:{ gender:'M'}},

{ $group:{ \_id:'$course', totalStudent: { $sum:1 } }

}])

**Using the `db.stats()` Method**

use yourDatabaseName

var stats = db.stats()

print("Database Size: " + (stats.dataSize / (1024 \* 1024 \* 1024)).toFixed(2) + " GB");

print("Storage Size: " + (stats.storageSize / (1024 \* 1024 \* 1024)).toFixed(2) + " GB");

The dataSize field represents the total size of all the documents in the database, while the storageSize field shows the total amount of space allocated to collections for document storage.

**Using the `show dbs` Command**

show dbs

**How to create an admin user and enable authentication – CREATE USER**

use admin

db.createUser({ user: "myAdminUser", pwd: "myAdminPassword", roles: [{ role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase"] });

Replace "myAdminUser" and "myAdminPassword" with your desired admin username and secure password respectively.

**Advanced User Management – CREATE ROLE**

**db.createRole({ role: "myCustomRole", privileges: [ { resource: { db: "myDatabase", collection: "" }, actions: [ "find", "update", "insert", "remove" ] } ], roles: [] });**

Once the role is created, you can assign it to a user with the following command:

db.grantRolesToUser("existingUser", ["myCustomRole"]);

Ensure that you replace "existingUser" with the actual username of the user to whom you wish to grant the newly created role.

**Revoking Privileges**

Sometimes you need to revoke privileges for security reasons. This can be done using the revokeRolesFromUser command:

use admin

db.revokeRolesFromUser('myReadOnlyUser', ['readWrite'])

**Change the Root Password**:

In the MongoDB shell, use the following commands to change the root password:

use admin

db.changeUserPassword("root", "newpassword")

Replace "newpassword" with your desired password.

use admin

db.updateUser("root", {pwd: "newRootPassword"});

Restart MongoDB with Access Control

After setting the new password, you need to enable access control again. First, stop the MongoDB server process that’s running without access control:

use admin

db.shutdownServer()

AGGREGATE FUNCTIONS;

db.customer.aggregate([

{ $count: "age" }

]);

db.customer.aggregate([

{

$group: {

\_id: null,

minPrice: { $min: "$age" },

maxPrice: { $max: "$age" }

}

}

]);

db.customer.aggregate([

{

$group: {

\_id: null,

totalRevenue: { $sum: { $multiply: ["$age", "$salary"] } },

averagePrice: { $avg: "$age" }

}

}

]);

db.employees.aggregate([

{

$project:{

"employeeId":1,

"firstName":1,

"lastName":1,

}

},

{

$limit : 5

}

]);

$lookup:

db.books.insertMany([{"bid":1,"bname":"java"},{"bid":2,"bname":"cpp"},{"bid":3,"bname":"python"}])

db.author.insertMany([{"aid":1,"aname":"balaji","bo\_id":ObjectId('66a8e5e7d8ee471782712a05')},{"aid":2,"aname":"mohan"},{"aid":3,"aname":"jeno"}])

db.author.aggregate({$lookup:{from:"books",

localField:"bo\_id",

foreignField:"\_id",

as: "bookDetails"}})

db.employees.updateMany({},{$set:{salary:200}})

db.employees.aggregate([

{

$group: {

\_id: "$departmentId",

totalSales: { $sum: "$salary" }

}

},

{

$out: "totalSales"

}

])

db.employees.aggregate([

{

$project: {

"employeeId": 1,

"firstName": 1,

"lastName": 1

}

},

{

$limit: 5

}

])