1. **To create a database in mongoDB - use <database name>**

use niit

>use niit

switched to db niit

**2. To check your currently selected database, use the command db**

db

>db

niit

**3. To check list of databases, use the command show dbs**

make sure you have atleast inserted one row or a document in the collection or a table

>db.employee.insert ({"name":"Krishna"})

Now run

>show dbs

local 0.000GB

niit 0.000GB

In MongoDB default database is test. If you didn't create any database, then collections will be stored in test database.

**4. If you want to delete database <local>, then dropDatabase() command would be as follows −**

>use local

switched to db local

>db.dropDatabase()

>{ "dropped" : "local", "ok" : 1 }

> show dbs

niit 0.000GB

**5.If you want to drop a table or a collection  <employee>, then  command would be as follows −**

>use niit

switched to db niit

show collections

>db.employee.drop()

true

**6. To insert data into MongoDB collection, you need to use MongoDB's insert() or save() method.**

the basic command is >db.COLLECTION\_NAME.insert(document)

> db.employee.insert({

\_id : 'Emp001',

fname: 'Sanjay',

lname: 'Garg',

dob : new Date ("1972-06-20"),

doj : new Date ("2016-04-01"),

address : [

{

Block : '2/20',

Street : '4th Avenue',

City : 'Bangalore',

State : 'karnataka',

pin : '560037'

}

]

})

db.employee.insert({

\_id : 'Emp002',

fname: 'Rajiv',

lname: 'Kumar',

dob : new Date ("1972-06-20"),

doj : new Date ("2016-04-01"),

address : [

{

Block : '4/801',

Street : '3rd Road',

City : 'Mumbai',

State : 'Maharashtra',

pin : '400005'

}

]

})

**7. To display the contents of the collection.**

>db.COLLECTION\_NAME.find() or db.COLLECTION\_NAME.find().pretty()

db.employee.find().pretty()

>db. employee.find() - display in a unstuctured way

OR

>db.employee.find().pretty() - display in a structured way

OR with where clause

>db.employee.find({'fname':'Rajiv'},{}).pretty()

db.employee.find({'customers.rating':'100'}).pretty()

8. **To update the values in the document**

>db.employee.update({'lname':'Kumar'}, {$set:{'lname':'Shah'}})

9. **To delete the document**

>db.employee.remove({'lname':'Shah'}, 1)

**10. To truncate the table**

>db.employee.remove()

**11. To show the limit number of records and skip the first record**

>db.employee.find().limit(1).skip(1)

**12. To sort the data**

>db.employee.find({},{"fname":1,\_id:0}).sort({"fname":-1})

**13. To index the data**

>db.employee.ensureIndex({"fname":1,"lname":-1})

**AGGREGATE FUNCTIONS FOR TRANSACTIONAL TABLE**

|  |  |  |
| --- | --- | --- |
| $sum | Sums up the defined value from all documents in the collection. | db.employee.aggregate([{$group : {\_id : "$lname", num\_employee : {$sum : "$likes"}}}]) |
| $avg | Calculates the average of all given values from all documents in the collection. | db.employee.aggregate([{$group : {\_id : "$lname", num\_employee : {$avg : "$likes"}}}]) |
| $min | Gets the minimum of the corresponding values from all documents in the collection. | db.employee.aggregate([{$group : {\_id : "$lname", num\_employee : {$min : "$likes"}}}]) |
| $max | Gets the maximum of the corresponding values from all documents in the collection. | db.employee.aggregate([{$group : {\_id : "$lname", num\_employee : {$max : "$likes"}}}]) |
| $push | Inserts the value to an array in the resulting document. | db.employee.aggregate([{$group : {\_id : "$lname", url : {$push: "$url"}}}]) |
| $addToSet | Inserts the value to an array in the resulting document but does not create duplicates. | db.employee.aggregate([{$group : {\_id : "$lname", url : {$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.employee.aggregate([{$group : {\_id : "$lname", 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.employee.aggregate([{$group : {\_id : "$lname", last\_url : {$last : "$url"}}}]) |

> db.employee.aggregate([{$group : {\_id : "$lname", num\_employee : {$sum : 1}}}])

{

"result" : [

{

"\_id" : "Kumar",

"num\_employee" : 1

},

{

"\_id" : "Garg",

"num\_employee" : 1

}

],

"ok" : 1

}

>

**select lname, count(\*) from employee group by lname**

SELECT \* FROM employee INTO OUTFILE '/tmp/employee.csv'

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

**Importing data into mongodb table from csv format**

$ cat > locations.csv

Name,Address,City,State,ZIP

Jane Doe,123 Main St,Whereverville,CA,90210

John Doe,555 Broadway Ave,New York,NY,10010

ctrl-d

$ mongoimport -d mydb -c things --type csv --file locations.csv --headerline

connected to: 127.0.0.1

imported 3 objects

$ mongo

MongoDB shell version: 1.7.3

connecting to: test

> use mydb

switched to db mydb

> db.things.find()

{ "\_id" : ObjectId("4d32a36ed63d057130c08fca"), "Name" : "Jane Doe", "Address" : "123 Main St", "City" : "Whereverville", "State" : "CA", "ZIP" : 90210 }

{ "\_id" : ObjectId("4d32a36ed63d057130c08fcb"), "Name" : "John Doe", "Address" : "555 Broadway Ave", "City" : "New York", "State" : "NY", "ZIP" : 10010

.

var lts = db.Lines.find({$and: [{Stops: 73565}]}); //Lines through stop

for (i=0;i<lts.length();i++){

var direction = db.Stops.find({stopID: lts[i].Stops[lts[i].Stops.length-1]},{name:1})[0].name;

print({'Line': lts[i].Line, 'Vehicle': lts[i].Vehicle, 'Direction': direction});

}

{

"\_id":ObjectId("52ffc33cd85242f436000001"),

"contact": "987654321",

"dob": "01-01-1991",

"name": "Tom Benzamin",

"address": [

{

"building": "22 A, Indiana Apt",

"pincode": 123456,

"city": "Los Angeles",

"state": "California"

},

{

"building": "170 A, Acropolis Apt",

"pincode": 456789,

"city": "Chicago",

"state": "Illinois"

}

]

}

>db.users.findOne({"name":"Tom Benzamin"},{"address":1})

{

"\_id":1,

"product\_name": "Samsung S3",

"category": "mobiles",

"product\_total": 5,

"product\_available": 3,

"product\_bought\_by": [

{

"customer": "john",

"date": "7-Jan-2014"

},

{

"customer": "mark",

"date": "8-Jan-2014"

}

]

}

>db.products.findAndModify({

query:{\_id:2,product\_available:{$gt:0}},

update:{

$inc:{product\_available:-1},

$push:{product\_bought\_by:{customer:"rob",date:"9-Jan-2014"}}

}

})

Creating a text search

{

"post\_text": "enjoy the mongodb articles on weblink.com",

"tags": [

"mongodb",

"weblink.com"

]

}

index name is text

>db.posts.ensureIndex({post\_text:"text"})

>db.posts.find({$text:{$search:"weblink.com"}})

>db.posts.getIndexes()

>db.posts.dropIndex("post\_text\_text")

1. List all the columns for the salespeople table

> db.salespeople.find().pretty();

2. List all the customers with the rating of 100

db.salespeople.find

>db.employee.find({'fname':'Rajiv'}).pretty()