MONGO DB

MONGO IS CASE SENSITIVE DATABASE QUERY LANGUAGE.

1.CREATE A DATABASE BY USING “USE”

USE \_ \_ \_DATABASE NAME\_ \_ \_ \_ \_

:- THIS IS SOMETHING WHEN THE DATABASE YOU’VE GIVEN IS EXIST THEN CONTROL WILL COME IN THAT PARTICULAR DATABASE, IF NOT AVAILABLE THEN, IT WILL CREATE A NEW DATABASE.

2. COLLECTION . THESE ARE LIKE TABLES IN MONGO DB.

AND THIS WILL CONTAIN THE KEY, VALUE PAIR.

KEY IS THE COLUMN NAME AND PAIR IS ITS VALUE. FOR EXAMPLE(‘EID: 1001). EID = KEY, 1001-= PAIR.

3.- LETS CREATE A DATABASE

####################################################################################################################

use lb23

\* - db.stu.insert({“rno”: 1, “name”: “kaushal kashyap”, “age”:12})

\*- db.stu.insertOne({“rno”: 1, “name”: “kaushal kashyap”, “age”:12})

= insertOne will return the object id of newly inserted data.

\*- db.stu.insertMany([{“rno” : 7 , “name” : “monica”}, {“rno” : 2 , “name”: “sonica”, “class” : “7th”}])

= inserting more than one documents(data) at a same time.

= this save work exact same as insert . It also use for inserting data.

\* db.stu.save({‘rno’:5,’name’:’Kaushal Kashyap’, ‘city’:’patna’})

\*db.stu.save({‘\_id’:ObjectId(“63233fa3ce”)})

Viola! Data inserted……

###################################################################################################################

commands:

1,- show dbs -it is basically for showing all the database inside that repository.

2.- db.stu.find() -it is to retrieve and show the data.

db.stu.find().pretty(); it also retrieve and show widely the documents(data). Or can say (“formatted Output”).

3.- db.dropdatabase() - will delete the whole database. But before deleting first check in which database you are in by using keyword ‘ db’.

4.- db.createCollection(“stu”) - to create a collection “stu”.

5.- show collections - it will show all the collection inside that database

6.- db.createCollection(“stu2”,{“capped”: true, size: 5000, max : 5}) = it follows last in first out crieteria.

7- db.stu.drop() = dropping the collection “stu”.

#####################################################################################################################

Update documents

# FOR A SINGLE DATA UPDATION USE UPDATEONE.

\* db.stu.updateOne({'EID': 1025},{$set:{'ADDRESS' : 'B302 PRAGYA APARTMENTS, DWARKA,DELHI','PHONE':9987456521}});

# FOR BELOW, IT IS SAYING THAT CHECK WHRER 'DEPT' IS 'HR' THEN SET 'DESI' TO SR. ASSOCIATE.

# IT WILL UPDATE ALL WHERE DEPT IS HR.

\* db.salary.updateMany({$and: [{"DEPT" :"HR"},{"DESI" :"ASSOCIATE"}]},

{$set:{"DESI" : "SR. ASSOCIATE"}});

\* db.salary.updateMany({$and: [{"DEPT" :"HR"},{"DESI" : "ASSOCIATE"}]},{$set:{"DESI" : "SR. ASSOCIATE"}});

# BELOW WITHOUT GIVING ANY CONDITION. UPDATE THE WHOLE COLUMN "CLASS" TO 8TH.

\* db.stu2.updateMany({},{"$set" : {"class" : "8th"}});

\* db.stu.replaceOne({"rno" : 2},{"rno" : 3 ,"name" : "ajay kumar","age" : 16});

\*

#####################################################################################################################

DELETE DOCUMENTS

\* db.stu.deleteOne({'rno': 5}); -- it will delete only one document specified under the condition.

\* db.emp.deleteOne({“name” : “ravi”});

\* db.collection.deleteMany({'dept':'HR'}) -- it will delete all the document where dept is hr.

\* db.stu.deleteMany({}) -- it will delete all the documents inside that collection.

\* db.sal.remove({“dept”: “MIS”}) --- it work similar as deleteMany().

\* db.sal.remove({“dept”: “ HR”},1) – it work similar as deleteOne().

\* db.sal.remove({“dept”: “ HR”},true) -- it work similar as deleteOne()..

###################################################################################

Operator in mongodb

8- Conditional statement . Like Inside a emp collection I want to see all “Hr” department employee details.

For that:

db.sal.find({“dept” : “HR”})

9- MORE THAN ONE CONDITION . THEN USE $ AND OPERATOR.

db.sal.find({ $and: [{ “dept” : “ops” }, { “desi” : “manager”}]})

db.sal.find({$and: [ {“dept”: “temp”},{“desi”: “associate”}]})

through this we can give as many condition as we want.

10- USING “$ OR” OPERATOR FOR CONDITIONAL QUESTION.

db.sal.find({$or : [{“dept” : “MIS”},{“desig” : “Associate”}]})

$or means either of any one condition or all condition is true.

\* $and operator

db.sal.find({$and: [{‘dept’ : ‘ops’},{‘desi’:’Manager’}]})

db.sal.find({‘dept’ : ‘hr’, ‘desi’ : ‘manager’})

db.sal.find({‘dept’ : ‘ops’, ‘desi’ : ‘manager’})

( $or operator;

db.sal.find({$or: [{‘dept’ : ‘hr’},{‘dept’: ‘IT’}]});

db.sal.find({$or: [{‘dept’ : ‘hr’},{‘dept’: ‘mis’}]});

db.sal.find({$or: [{ “dept” : “ops”},{“dept” : “hr”}]});

db.sal.find({$and: [{‘desi’: ‘associate’}.{$or: [{‘dept’ : “ops”},{‘dept’: “hr”}]}]})

\* Not Equals to :-

db.sal.find({“dept”: {$not : {$eq : “temp”}}}) – mean other which dept!= temp

\* db.sal.find({$nor: [{“dept”: “temp”},{“desi” : “associate”}]})

means neither department should be temp nor the desi should be associate.

###################################################################################

Comparison operator

\* $eq = equals to.

db.stu.find({“dept” : {$eq : “mis”}});

\* db.stu.find({$and : [{“dept”: {$eq : “OPS”}},{$eq : {‘manager’}}]})

\* db.emp.find({$or: [{“key” : “value”},{“key” : “value”})

\* db.emp.find({“DEPT”: {$NOT: {$eq: “hr”}}});

\* db.sal.find({$nor: [{“desi”: “associate”}, {“dept”: “hr”}]});

\* eq( equals to ):- $eq: Equality Operator same as where dept = ‘hr’ in SQL

\* db.sal.find({"DEPT" :{$eq : "HR"}});

\* db.sal.find({"DEPT" : "HR"});

\* - $ne: Not equal to Operator same as where dept <> ‘hr’ in SQL

-- db.sal.find({"DEPT" :{$ne : "HR"}});

\* $lt: less than Operator same as where salary< 100000 in SQL

Syntax: {<key>:{$lt:<value>}}

Eg: > db.sal.find({"SALARY" :{$lt : 50000}});

\* $lte: less than or equal to Operator same as where salary<= 50000 in SQL

Syntax: {<key>:{$lte:<value>}}

Eg: > db.sal.find({"SALARY" :{$lte : 50000}});

\* $gt: greater than or equal to Operator same as where salary>= 50000 in SQL

Syntax: {<key>:{$gt:<value>}}

Eg: > db.sal.find({"SALARY" :{$gt : 50000}});

\* $gte: greater than or equal to Operator same as where salary >= 50000 in SQL

Syntax: {<key>:{$gte:<value>}}

Eg: > db.sal.find({"SALARY" :{$gte : 50000}});

\* $in: represents values in an array same as where dept in (‘hr’, ‘it’, ’admin’)

Syntax: {<key>:{$in:[<value1>, <value2>,......<valueN>]}}

Eg: > db.sal.find({"DEPT" :

{$in :

["HR", "IT", "TEMP"]

}

});

\* $nin: represents values not in an array same as where NOT dept in (‘hr’, ‘it’,

’admin’)

Syntax: {<key>:{$nin:[<value1>, <value2>,......<valueN>]}}

Eg: > db.sal.find({"DEPT" :{$nin : ["HR", "IT", "TEMP“,”ADMIN”]}});

Operators Examples:

\* db.orders.find({"Category" : {"$eq" : "Technology"},"Sub-Category" : {"$eq" : "Phones"}}).count()

> db.salary.find({"$and" : [{"DEPT" : {"$eq" : "HR"}}, {"DESI" : {"$eq" : "ASSOCIATE"}}]})

> db.salary.find({"$and" : [{"DEPT" : "HR"}, {"DESI" : "ASSOCIATE"}]})

> db.salary.find({"$and" : [{"$or" : [{"DEPT" : "HR"}, {"DEPT" : "MIS"}]},{"$or" : [{"DESI" :

"MANAGER"}, {"DESI" : "SR. ASSOCIATE"}]}]})

> db.salary.find({"SALARY" :{"$not" : {"$gt" : 100000}}})

-- Importing Data

d: Specifies what database to use. We used the demo database.

-c: Specifies what collection to use. We used a sal collection.

--type: Specifies the type of file to import. json, csv, or tsv. We are using csv

--headerline: Specifies that the first row in our csv file should be the field names.

--drop: Specifies that we want to drop the collection before importing documents to avoid

duplicate documents.

To import a csv [file:-](../../../../../-)

>**mongoimport -d demo -c sal --type csv --file Salary.csv --headerline --drop**

**>mongoimport -d demo2 -c sal --type csv --file C:\Users\Raj\Desktop\Salary.csv --**

**headerline --drop**

# To get the count of the records:

\* db.stu.find().count()

**Projection**

**projection are for filtering data to display or should act on it..**

For better understanding looks this way:

db.collection.field({condition},{projection})

\* db.sal.find({},{“EID” : 1, “dept”: 1, “salary”: 1})

\* db.sal.find({},{“desi”: 0 })

suppose I want to see only their eid, salary, and should belongs to hr desi..

\* db.sal.find({“dept” : “hr”), {“EID”: 1, “salary”: 1})

**Limiting the documents**

\* db.sal.find().limit(10);

**Skipping the documents;**

\* db.sal.find().limit().skip()

**Sorting the documents**

**\*** db.sal.find().sort({“salary”: 1})

- sorting on the basis of desi hr in salary.

\* db.sal.find({“desi”: “hr”}).sort({“salary”: 1}) -- in ascending order

\* db.sal.find({“desi”: “hr”}).sort({“salary”: -1}) -- in descending order

\* db.sal.find({“dept”: “hr”},{“desi”: 0}).sort({“salary”: 1}).limit(3) -- sort on the basis of dept=hr.

###################################################################################

**Indexing**

\* db.emp.createIndex({“city”: 1})

to see the index -

db.emp.getIndexes()

\* compound index

db.emp.createIndex({“city”: 1, “addr”: 1}); -- compound is basically for indexing more than 1 column.

\* time to leave (ttl) –db.emp.createIndex({“EId ”: 1},{expiredafterseconds: 600})-- after that 600s index will automatically be destroyed.

\* unique indexing

db.sal.createIndex({“EID”: 1, }, {unique: true})

Note: Through indexing you can stop duplicacy in database.

Dropping the indexes:

\* db.emp.dropIndexes({“emp”: -1}) – condition dropping

\* db.emp.dropIndexes({}) -- to drop all the indexes.

Note: a collection can have maximum 64 indexes.

###################################################################################

**Aggregation(“ brought together ”)**

db.sal.aggregate([{$group: { \_id : “dept”, “TC”: {$sum : “$salary”}}}]);

here we are gouping the department on the basis of sum of salary.

\* db.sal.aggregate([{$ group: {\_id : “$dept”, “TC”: {$sum: “$salary”},”tm”: {$sum: 1}}}])

\* db.sal.aggregate([{$ group: {\_id : “$dept”, “TC”: {$sum: “$salary”},”tm”: {$sum: 1},”avgsal”: {“avg”: “$salary”} }}])

db.sal.aggregate([{$ group: {\_id : “$dept”, “TC”: {$sum: “$salary”},”tm”: {$sum: 1},”avgsal”: {“avg”: “$salary”} ,”maxsal”:{$max:”$salary”},”Minsal”: {“$min”: “$salary”}}}])