Assignment

Feb19/ DBT/ 129

Database Technologies

Diploma in Advance Computing

February 2019

**MongoDB**

USE ***EMP***collection.

|  |
| --- |
| 1. Display all databases. |
| show dbs;  show databases; |
|  |
| 1. Display the current database. |
| db  db.getName(); |
|  |
| 1. Display all collection. |
| db.getCollectionNames(); |
|  |
| 1. Display the current version of MongoDB. |
| version(); |
|  |
| 1. Display the current host details. |
| hostname(); |
|  |
| 1. Get the current ip address and the port number. |
| db.getMongo(); |
|  |
| 1. Display all documents from EMP collection. |
| db.emp.find();  db.getCollection('emp').find(); |
|  |
| 1. Display first 5 documents from EMP collection. |
| db.emp.find().limit(5); |
|  |
| 1. Display employee name, and his address from EMP collection. |
| db.emp.find({}, { ename: true, address: true }) |
|  |
| 1. Display all building and coord details of all employee from EMP collection. |
| db.getCollection('emp').find({}, { \_id: false, "address.building": true, "address.coord": true }); |
|  |
| 1. Display all documents who are staying in building number “2780”. |
| db.getCollection('emp').find({"address.building":"2780"}); |
|  |
| 1. Display all female employee documents. |
| db.getCollection('emp').find({ gender: "female" }); |
|  |
| 1. Display all employee working in department number 40. |
| db.getCollection('emp').find({ deptno: 40 }); |
|  |
| 1. Enter 5 documents in EMP collection in the following format.   empid:number, ename:str, address:{ building:str,"coord" : [number,number], street:str, zipcode:number }, isActive : bool, gender:char, canVote:bool, canDrive:bool, favouriteColor[,..], favouriteFruit[,..], aadhar:str, job:str, mgr:number, hiredate:date, sal:number, comm:number, deptno:number |
| db.emp.insertMany( [ {}, {}, {}, {}, {} ] ) |
|  |
| 1. Count total documents in EMP collection. |
| db.emp.countDocuments({}) |
|  |
| 1. Display ename, sal, comm fields from the collection, who are getting some comm. |
| db.getCollection('emp').find( {comm: { $ne: null} }, {ename: true, sal: true, comm: true}) |
|  |
| 1. Count the documents of ‘Computer Programmer’ |
| db.emp.countDocuments({ job: "Computer Programmer" } ) |
|  |
| 1. Display ename, job, and salary fields from EMP collection in ascending order of ename. |
| db.emp.find({}, { ename: true, job: true, sal: true }).sort( { ename: 1 } ) |
|  |
| 1. Display all documents between 5 and 10. |
| db.emp.find().skip(5).limit(5) |
|  |
| 1. Display the last document. |
| db.emp.find().skip(db.emp.countDocuments({}) - 1); |
|  |
| 1. Display all employee *ename, job,* and *sal* field who are working either as ‘manager’ or ‘Computer Programmer’ |
| db.emp.find({ $or:[{job:'manager'}, {job: 'Computer Programmer'} ]}, {ename: true, job: true, sal: true}) |
|  |
| 1. Get all employee whose salary is between 2000 and 4000. |
| db.emp.find({ $and: [ { sal : { $gt : 2000 } }, {sal : { $lt:4000 } } ] }); |
|  |
| 1. Display all distinct job from EMP collection. |
| db.emp.distinct("job") |
|  |
| 1. Display all distinct job from EMP collection whose salary in more than 5000. |
| db.emp.distinct("job", { sal: { $gt: 5000 } } ) |
|  |
| 1. Display all distinct job who are not getting commission. |
| db.emp.distinct("job", { comm: { $ne: null } } ) |
|  |
| 1. Display all documents from EMP collection using aggregate. |
| db.emp.aggregate ([]) |
|  |
| 1. Display all documents whose jab is ‘manager’ using aggregation. |
| db.emp.aggregate ([{$match:{job:'manager'}}]) |
|  |
| 1. Display all documents job is either ‘manager’ or ’salesman’ using aggregation. |
| db.emp.aggregate ([{$match:{$or:[{job:'manager'}, {job:'salesman'}]}}]) |
|  |
| 1. Display sum of salary. |
| db.emp.aggregate ([{$group: {\_id:null, total :{$sum:'$sal'}}}]) |
|  |
| 1. Display sum of salary jobwise. |
| db.emp.aggregate ([{$group: {\_id:'$job', total :{$sum:'$sal'}}}]) |
|  |
| 1. Display all distinct job who are not getting commission. |
| db.emp.distinct("job", { comm: { $eq: null } } ) |
|  |
| 1. Display the count of employees working for every job. |
| db.emp.aggregate ([{$group: {\_id:'$job', count :{$sum:1}}}]) |
|  |
| 1. Export EMP collection in JSON format. (filename “D:\e.json”) |
| mongoexport --host 192.168.100.20 --port 27017 --collection emp --db db1 --out "d:\e.json" |
|  |
| 1. Increase the salary of all employees by Rs. 1000 and print their employee name, sal and increased salary as “New Salary”. |
| db.emp.aggregate([{$project:{\_id: false, ename: true, sal: true, "New Salary": { $add: ['$sal',1000] } } }]) |
|  |
| 1. Find all documents whose commission is **null** and replace the null value with ‘NA’. |
| db.emp.aggregate ([{$project:{comm : {$ifNull:['$comm', 'NA']}}}]) |
|  |
| 1. Display all employee name is uppercase. |
| db.emp.aggregate([{$project: { ename : { $toUpper : '$ename'}}}]) |
|  |
| 1. Display all employee name is lowercase. |
| db.emp.aggregate([{$project: { ename : { $toLower : '$ename'}}}]) |
|  |
| 1. Print employee name and job in the given format.   jack----accountant |
| db.emp.aggregate([{$project: { ename : { $concat : [ '$ename', '----', '$job']}}}]) |
|  |
| 1. Display ename, job, sal and comm whose comm is null, replace comm with ‘NA’ if null. |
| db.emp.aggregate([{$match:{ comm: {$eq: null} }}, {$project: {\_id: false, ename: true, job: true, sal: true, "New Commission": {$ifNull: ['$comm', 'NA']} }}]) |
|  |
| 1. Compute “Gross Salary” by adding sal and commission, if commission is null replace it with 0. |
| db.emp.aggregate([{$project: {sal: true, comm: true, "Gross Salary": {$add: ['$sal', {$ifNull: ['$comm', 0]}]}}}]) |
|  |
| 1. Display all ename, and isActive whose gender is ‘female’ and isActive is true. |
| db.emp.aggregate([ {$match: {isActive: true, gender: 'female'}}, {$project: {\_id:false, ename:true, isActive: true, gender: true}}]) |
|  |
| 1. Display all ename, isActive, gender, salary, and commission whose gender is ‘female’ and isActive is true and also computer the addition of sal and commission. If commission is null replace it with zero. |
| db.getCollection('emp').aggregate([ {$match:{isActive: true, gender: 'female'}},{$project:{\_id:false, ename:true, isActive: true, gender: true, sal: true, comm:true, a: {$add:["$sal", {$ifNull: ['$comm',0]}]}}}]) |
|  |
| 1. Update isActive to true whose ename is ‘jack’. |
| db.emp.update({ename: 'jack'},{ $set: {isActive: true}}) |
|  |
| 1. Update sal to 6500 for ObjectId (give any ObjectId from the collection). |
| db.emp.updateOne({"\_id" : ObjectId("5c0a079b510ec014c6a1ef1f") }, { $set :{ sal: 6500 } }) |
|  |
| 1. Create a new collection *EMPL* with following fields (\_id, firstName, lastName, and salary) and add 10 documents *(Note: \_id must start with the value 1 and so on in the sequence)*. |
| db.empl.insert([{\_id:1, …}, {\_id:2, …}, {\_id:3, …}, {\_id:4, …}, {\_id:5, …}, {\_id:6, …}, {\_id:7, …}, {\_id:8, …}, {\_id:9, …}, {\_id:10, …}]) |
|  |
| 1. Increment the salary of those employees whose salary in more than 6500Rs. by 1000Rs.. Only for the first document. |
| db.emp.update({sal:{$gt:6500}}, { $inc: {sal: 1000}}) |
|  |
| 1. Increment the salary of those employees whose salary in more than 6500Rs. by 1000Rs.. For all employees. |
| db.emp.updateMany({ sal: {$gt: 6500} }, { $inc: {sal: 1000} }) |
|  |
| 1. Delete all employee document whose salary is more than 8000Rs. |
| db.emp.deleteMany({ sal : {$gt: 8000} }) |
|  |
| 1. Delete all document from EMPL collection. |
| db.empl.remove({})  db.empl.deleteMany({}) |
|  |
| 1. Display ename, favourite fruit of all employees who likes only one fruit. |
| db.emp.aggregate([ {$match: {favouriteFruit :{$size: 1}}},{$project: {\_id:false, ename:true, favouriteFruit: true} } ]) |
|  |
| 1. Display all documents for those employees who like more than 2 color. |
| db.emp.aggregate([ {$project: {\_id: false, x: {$size: '$favouriteColor'}, ename: true, favouriteColor: true}}, {$match: {x: {$gt: 2} } } ]) |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |