**use**

**mongdb**

## **Mongodb**

**Example 19**.

Create Database DYPIT using MongoDB

Create following Collections

**Teachers(Tname,dno,dname,experience,salary,date\_of\_joining )**

**Students(Sname,roll\_no,class)**

db.createCollection('Teachers')

db.createCollection('Students')

Teachers(Tname,dno,dname,experience,salary,date\_of\_joining )

db.Teachers.insertOne({

'Tname': 'Sunita',

'dno': 1,

'dname': 'Computer',

'experience':11,

'salary':10001,

'date\_of\_joining':'1/1/2001'

})

db.Teachers.insertMany([{

'Tname': 'Rajesh',

'dno': 1,

'dname': 'IT',

'experience':11,

'salary':10001,

'date\_of\_joining':'1/1/2001'

},

{

'Tname': 'Praveen',

'dno': 2,

'dname': 'Computer',

'experience':5,

'salary':100011,

'date\_of\_joining':'2/2/2012'

},

{

'Tname': 'Rucha',

'dno': 3,

'dname': 'E&TC',

'experience':17,

'salary':200001,

'date\_of\_joining':'9/6/1996'

},

{

'Tname': 'Minal',

'dno': 2,

'dname': 'IT',

'experience':7,

'salary':10002,

'date\_of\_joining':'1/1/2011'

},

{

'Tname': 'Rupesh',

'dno': 2,

'dname': 'E&TC',

'experience':7,

'salary':10002,

'date\_of\_joining':'1/1/2011'

}

])

**Student :**

db.createCollection(‘Students’)

db.Students.insertMany([{

'Sname': 'Isha',

'roll\_no': 1,

'class': 'Computer'

},

{

'Sname': 'Avadhut',

'roll\_no': 2,

'class': 'E&TC'

},

{

'Sname': 'Ritul',

'roll\_no': 3,

'class': 'IT'

},

{

'Sname': 'Rushikesh',

'roll\_no': 4,

'class': 'IT'

}

])

**1)Find the information about all teachers**

db.Teachers.find().pretty()

**2)Find the information about all teachers of computer department.**

db.Teachers.find({'dname':'Computer'}).pretty()

**3)Find the information about all teachers of computer,IT,and e&TC** department

db.Teachers.find({'dname':'IT'} , {'dname':'ENTC'},{'dname':'Computer'}).pretty()

**4)department having salary greate than or equl to 10000/-**

db.Teachers.find({'salary':{$gte:10000}}).pretty()

**5)Find the student information having roll\_no = 2 or Sname=xyz**

1. db.Students.find({$or:[{'roll\_no':2},{'Sname':'avadhut'}]}).pretty()

**6)Update the experience of teacher-praveen to 10years, if the entry is not available in database consider the entry as new entry.**

db.Teachers.insert({

'Tname': 'Praveen',

'dno': 3,

'dname': 'E&TC',

'experience':10,

'salary':5001,

'date\_of\_joining':'1/1/2021'

})

db.Teachers.updateOne({Tname:'Praveen'}, {$set:{experience:10}})

**7)Update the deparment of all the teachers working in IT deprtment to COMP**

db.Teachers.updateMany({dname:'IT'}, {$set:{dname:'Computer'}})

**8)find the teachers name and their experience from teachers collection**

db.Teachers.find({},{dname:0,dno:0,salary:0,date\_of\_joining:0}).pretty()

db.Teachers.find({},{dno:0,dname:0,salary:0,date\_of\_joining:0})

**9)Using Save() method insert one entry in department collection**

db.Teachers.save({

'Tname': 'Rajesh',

'dno': 1,

'dname': 'Computer',

'experience':8,

'salary':50001,

'date\_of\_joining':'1/1/2019'

})

**10)Using Save() method change the dept of teacher Rajesh to IT 11.**

db.Teachers.save({

'Tname': 'Rajesh',

'dno': 1,

'dname': 'Computer',

'experience':8,

'salary':50001,

'date\_of\_joining':'1/1/2019'

})

WriteResult({ "nInserted" : 1 })

**11)Delete all the doccuments from teachers collection having IT dept.**

**display with pretty() method, the first 3 doccuments in teachers collection in ascending order.**

db.Teachers.find().sort({dno:1}).limit(3).pretty()

Mongo last 2

Map reduce

**Problem 20 :-**

1.Create Database DYPIT

use DYPIT;

1. Create following Collections

**Teachers(Tname,dno,dname,experience,salary,date\_of\_joining ) Students(Sname,roll\_no,class)**

3)Find the information about two teachers

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

1. Find the information about all teachers of computer department

db.Teachers.cfind({dname:'Computer'}).pretty()

1. Find the information about all teachers of computer,IT,and e&TC department
2. Find the information about all teachers of computer,IT,and E&TC department having salary greate than or equl to 25000/-

db.Teachers.find({'salary':{$gte:25000}}).pretty()

1. Find the student information having roll\_no = 25 or Sname=xyz
2. db.Students.find({$or:[{'roll\_no':2},{'Sname':'avadhut'}]}).pretty()
3. Update the experience of teacher-praveen to 10years, if the entry is not available in database consider the entry as new entry.

db.Teachers.insert({

... 'Tname': 'Praveen',

... 'dno': 3,

... 'dname': 'E&TC',

... 'experience':10,

... 'salary':5001,

... 'date\_of\_joining':'1/1/2021'

... })

db.Teachers.updateOne({Tname:'Praveen'}, {$set:{experience:10}})

1. Update the deparment of all the teachers working in IT deprtment to COMP

db.Teachers.updateMany({dname:'IT'}, {$set:{dname:'Computer'}})

1. find the teachers name and their experience from teachers collection

db.Teachers.find({},{dname:0,dno:0,salary:0,date\_of\_joining:0}).pretty()

db.Teachers.find({},{dno:0,dname:0,salary:0,date\_of\_joining:0})

1. Using Save() method insert one entry in department collection 13. Delete all the doccuments from teachers collection having IT dept.

db.Teachers.save({

'Tname': 'Rajesh',

'dno': 1,

'dname': 'Computer',

'experience':8,

'salary':50001,

'date\_of\_joining':'1/1/2019'

})

**14. display with pretty() method, the first 5 documents in teachers collection in ascending order**

db.Teachers.find().sort({dno:1}).limit(5).pretty()

**Example 21 : Create Database DYPIT using MongoDB**

**Create following Collections**

**Teachers(Tname,dno,dname,experience,salary,date\_of\_joining )**

**Students(Sname,roll\_no,class)**

Use DYPIT

Create following Collections

db.createCollection('Teachers')

Teachers(Tname,dno,dname,experience,salary,date\_of\_joining )

db.Teachers.insertOne({

'Tname': 'Sunita',

'dno': 1,

'dname': 'Computer',

'experience':11,

'salary':10001,

'date\_of\_joining':'1/1/2001'

})

db.Teachers.insertMany([{

'Tname': 'Rajesh',

'dno': 1,

'dname': 'IT',

'experience':11,

'salary':10001,

'date\_of\_joining':'1/1/2001'

},

{

'Tname': 'Praveen',

'dno': 2,

'dname': 'Computer',

'experience':5,

'salary':100011,

'date\_of\_joining':'2/2/2012'

},

{

'Tname': 'Rucha',

'dno': 3,

'dname': 'E&TC',

'experience':17,

'salary':200001,

'date\_of\_joining':'9/6/1996'

},

{

'Tname': 'Minal',

'dno': 2,

'dname': 'IT',

'experience':7,

'salary':10002,

'date\_of\_joining':'1/1/2011'

},

{

'Tname': 'Rupesh',

'dno': 2,

'dname': 'E&TC',

'experience':7,

'salary':10002,

'date\_of\_joining':'1/1/2011'

}

])

Students(Sname,roll\_no,class)

db.createCollection(‘Students’)

db.Students.insertMany([{

'Sname': 'Isha',

'roll\_no': 1,

'class': 'Computer'

},

{

'Sname': 'Avadhut',

'roll\_no': 2,

'class': 'E&TC'

},

{

'Sname': 'Ritul',

'roll\_no': 3,

'class': 'IT'

},

{

'Sname': 'Rushikesh',

'roll\_no': 4,

'class': 'IT'

}

])

**1)Find the information about all teachers.**

db.Teachers.find().pretty()

**2)Find the average salary teachers of computer department**

db.Teachers.aggregate([{$match:{"dname":"Computer"}},{$group : {\_id : "$dname", salary\_maximum : {$avg : "$salary"}}}])

**3)Find the minimum and maximum salary of e&TC department teachers**

db.Teachers.aggregate([{$match:{"dname":"E&TC"}},{$group : {\_id : "$dname", salary\_maximum : {$max : "$salary"}, salary\_minimum:{$min : "$salary"}}}])

**4)Find the information about all teachers of computer,IT,and E&TC department having salary greate than or equl to 10000/-**

db.Teachers.find({'salary':{$gte:10000}}).pretty()

**5)Find the student information having roll\_no = 2 or Sname=xyz**

db.Students.find({$or:[{'roll\_no':2},{'Sname':'avadhut'}]}).pretty()

**6)Update the experience of teacher-praveen to 10years, if the entry is not available in database consider the entry as new entry.**

db.Teachers.insert({

'Tname': 'Praveen',

'dno': 3,

'dname': 'E&TC',

'experience':10,

'salary':5001,

'date\_of\_joining':'1/1/2021'

})

db.Teachers.updateOne({Tname:'Praveen'}, {$set:{experience:10}})

**7)Update the deparment of all the teachers working in IT deprtment to COMP**

db.Teachers.updateMany({dname:'IT'}, {$set:{dname:'Computer'}})

**8)find the teachers name and their experience from teachers collection.**

db.Teachers.find({},{dname:0,dno:0,salary:0,date\_of\_joining:0}).pretty()

**9)Using Save() method insert one entry in department collection 10. Find the total salary all teachers.**

db.Teachers.aggregate([{$group : {\_id : "", total\_salary : {$sum : "$salary"}}}])

**Problem Statement 22:**

**1)Create Database DYPIT using MongoDB**

**2)Create following Collections**

**Teachers(Tname,dno,dname,experience,salary,date\_of\_joining ) Students(Sname,roll\_no,class)**

**3)Display the department wise average salary**

db.Teachers.aggregate([{$group : {\_id : "$dname", salary\_avarage : {$avg : "$salary"}}}])

**4)display the no. Of employees working in each department**

db.Teachers.aggregate( [ { $unwind: "$dname" }, { $sortByCount: "$dname" } ] )

**5)Display the department wise total salary of departments having total salary greater than or equals to 50000/-**

db.Teachers.find({'salary':{$gte:50000}}).pretty()

**6)Write the queries using the different operators like max, min. Etc.**

db.Teachers.aggregate([{$match:{"dname":"E&TC"}},{$group : {\_id : "$dname", salary\_maximum : {$max : "$salary"}, salary\_minimum:{$min : "$salary"}}}])

**7)Create unique index on any field for above given collections**

db.Teachers.createIndex({Tname:1}, {unique:true})

**8)Create compound index on any fields for above given collections 7. Show all the indexes created in the database DYPIT**

db.Teachers.getIndexes()

**9)Show all the indexes created in above collections.**

db.Teachers.getIndexes()

**Problem Statement 23:**

Create index and fire queries with MongoDB

**1)Import zip.json.**

mongoimport --dbsai --collection zip --file C:\Users\OMKAR\Desktop\zips.json

**2)Create single field, composite and multikey indexes.**

db.zip.createIndex( {'ind' : 1 });

db.zip.createIndex( {'index' : 2, 'name' : 4 })

db.zip.createIndex( {'index' : 2, 'name' : 4 },{'indw' : 3 })

**Fire queries given below again and write your analysis.**

3)Display all cities having population above 1600.

db.zip.find( { population : {$gt : 1600} })

4**)Display all cities in state “KS”.**

db.zip.find( {state : 'KS'}, {city :1});

**5)Display location of city "TIMKEN"**

db.zip.find( {city : 'TIMKEN'}, {location :1})

**Problem Statement 24 :-**

**Design and Implement following query using MongoDB**

**1)Create a collection called ‘games’.**

db.createCollection(‘games’);

**2)Add 5 games to the database. Give each document the following properties:**

name, gametype, rating (out of 100)

db.games.insertMany([{

'name': 'life',

'gametype': 'joke',

'rating': 100

},

{

'name': 'Crypto',

'gametype': 'Luck',

'rating': 10

},

{

'name': 'Solitare',

'gametype': 'card',

'rating': 80

},

{

'name': 'Pubg',

'gametype': 'FPS',

'rating': 80

},

{

'name': 'GTA',

'gametype': 'open\_world',

'rating': 75

}])

**3)Write a query that returns all the games**

db.games.find().pretty()

**4)Write a query that returns the 3 highest rated games.**

db.games.find().sort({rating:-1}).limit(3).pretty()

**5)Update your two favourite games to have two achievements called ‘Game Master’ and ‘Speed Demon’.**

db.games.updateOne({name:"GTA"}, {$set:{achievements:"Game-master,Speed-daemon"}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

db.games.updateOne({name:"life"},

... {$set:{achievements:"Game-master","Speed-daemon"}})

**6)Write a query that returns all the games that have both the ‘Game Maser’ and the ‘Speed Demon’ achievements.**

db.games.find({"achievements":"Game-master,Speed-daemon"}).pretty()

**7)Write a query that returns only games that have achievements.**

**Problem Statement 25**

**Using MapReduce in mongodb solve following queries on given below collection.**

{

“id” : 0,

“name” : “Leanne Flinn”,

“email” : “leanne.flinn@unilogic.com”,

“work” :”Unilogic” ,

“age” :27

“gender” :”Male”

“Salary” :16660

“hobbies”:”Acrobatics,Photography,Papier-Mache” }

db.prac25.insertMany(

... [

... { id : 0, name : 'Larry Page', email : 'krishna@gmail.com', work :'SDE-1' , age :26, gender :'Male', Salary :16660, hobbies :'singing' },

... { id : 1, name : 'Elon musk', email : 'elon@gmail.com', work :'SDE-2' , age :36, gender :'Male', Salary :196660, hobbies :'wrapper' },

... { id : 2, name : 'sarvesh', email : 'sarvesh@gmail.com', work :'Developer' , age : 29, gender :'Female', Salary : 898896, hobbies :'ceater' },

... { id : 3, name : 'chetan', email : 'chetan@gmail.com', work :'engineer' , age :20, gender :'Male', Salary :16660, hobbies :'singing' },

... { id : 4, name : 'sanket', email : 'sanket@gmail.com', work :'manager' , age :26, gender :'Female', Salary :89656, hobbies :'sleeping' }

... ]

... )

1. Get the count of Males and Females

db.prac25.aggregate([

{

"$group": {

"\_id": "$gender",

"count": { "$sum": 1 }

}

}

])

1. Count the number of users in each hobby

db.prac25.aggregate([

{$match : { } },

{$group : {\_id :'$hobbies', sumOfHobby : {$sum :1 } } }

])

* + 1. Get the count of Males and Females
    2. Count the number of users in each hobby

**Problem Statement 26**

**Using MapReduce in mongodb solve following queries on given below collection.**

* 1. Import zip.json.
  2. Find total population in each state.

Import zip.json.

mongoimport --dbsai --collection zip --file C:\Users\OMKAR\Desktop\zips.json

Find total population in each state.

db.zip.mapReduce( function() {emit(this.state,this.pop);}, function(key,value){return Array.sum(value)}, { query:{state:"MA"},out:"state\_pop\_totals"});

db.state\_pop\_totals.find();

**Problem Statement : 27**

Create a database called ‘library’, create a collection called ‘books’.find the number of books having pages less 250 pages and consider ad small book and greater than 250 consider as Big book using Map Reduce function.

use library

db.books.insertOne({name : "Understanding JAVA", pages : 100})

db.books.insertOne({name : "Understanding JSON", pages : 200})

db.books.insertMany([

... {name : "Understanding XML", pages : 300},

... {name : "Understanding Web Services", pages : 400},

... {name : "Understanding Axis2", pages : 150}

... ])

var map = function() {

... var category;

... if ( this.pages >= 250 )

... category = 'Big Books';

... else

... category = "Small Books";

... emit(category, {name: this.name});

... }

var reduce = function(key, values) {

... ... var sum = 0;

... ... values.forEach(function(doc) {

... ... sum += 1;

... ... });

... ... return {books: sum};

... ... }

var count = db.books.mapReduce(map, reduce, {out: "book\_results"})

db[count.result].find()