Implement Map reduces operation using MongoDB.

Problem:A king want to count the total population in his country. He can send one person to count the population. The assigned person will visit every city serially and return with the total population in the country.

> db.createCollection("MapReduce\_King")

{ "ok" : 1 }

> db.MapReduce\_King.insert({City: "Los Angels",Population: 300000})

WriteResult({ "nInserted" : 1 })

> db.MapReduce\_King.insert({City: "Texas",Population: 42000})

WriteResult({ "nInserted" : 1 })

> db.MapReduce\_King.insert({City: "Vegas",Population: 99000})

WriteResult({ "nInserted" : 1 })

> db.MapReduce\_King.insert({City: "NashVille",Population: 30000})

WriteResult({ "nInserted" : 1 })

> db.MapReduce\_King.insert({City: "EdinBurgh",Population: 900000})

WriteResult({ "nInserted" : 1 })

> var mapFunction2 = function() {emit(null, this.Population);};

> var reduceFunction2= function(Country, Population) {return Array.sum(Population);};

> db.MapReduce\_King.mapReduce(mapFunction2, reduceFunction2, {out:"Result"});

{ "result" : "Result", "ok" : 1 }

> db.Result.find();

{ "\_id" : null, "value" : 1371000 }

**2.**

Implement aggregation and indexing with following example using MongoDB

Example: In this Assignment, we are creating Student Database. Which contain the information ofstudent\_name,student\_rollno,status of a student. Here status is whether student is passed/failed by the university.

> db.createCollection("Student")

{ "ok" : 1 }

> db.Student.insert({Stud\_Name: "Aarohi", Stud\_Roll\_No: 01, Status: "Passed"})

WriteResult({ "nInserted" : 1 })

> db.Student.insert({Stud\_Name: "Vrushali", Stud\_Roll\_No: 02, Status: "Passed"})

WriteResult({ "nInserted" : 1 })

> db.Student.insert({Stud\_Name: "Monica", Stud\_Roll\_No: 03, Status: "Passed"})

WriteResult({ "nInserted" : 1 })

> db.Student.insert({Stud\_Name: "Joey", Stud\_Roll\_No: 04, Status: "Failed"})

WriteResult({ "nInserted" : 1 })

> db.Student.insert({Stud\_Name: "Srinidhi", Stud\_Roll\_No: 05, Status: "Failed"})

WriteResult({ "nInserted" : 1 })

**Craete Index:**

> db.Student.createIndex({Stud\_Roll\_No: 01})

{

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"createdCollectionAutomatically" : false,

"ok" : 1

}

> db.Student.createIndex({Stud\_Roll\_No: 04})

{

"numIndexesBefore" : 2,

"numIndexesAfter" : 3,

"createdCollectionAutomatically" : false,

"ok" : 1

}

**Get Index**

> db.Student.getIndexes()

[

{

"v" : 2,

"key" : {

"\_id" : 1

},

"name" : "\_id\_"

},

{

"v" : 2,

"key" : {

"Stud\_Roll\_No" : 1

},

"name" : "Stud\_Roll\_No\_1"

},

{

"v" : 2,

"key" : {

"Stud\_Roll\_No" : 4

},

"name" : "Stud\_Roll\_No\_4"

}

]

**Drop index**

> db.Student.dropIndexes()

{

"nIndexesWas" : 3,

"msg" : "non-\_id indexes dropped for collection",

"ok" : 1

}

> db.Student.getIndexes()

[ { "v" : 2, "key" : { "\_id" : 1 }, "name" : "\_id\_" } ]

> db.Student.createIndex({Stud\_Roll\_No: 04})

{

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"createdCollectionAutomatically" : false,

"ok" : 1

}

> db.Student.dropIndex({Stud\_Roll\_No: 04})

{ "nIndexesWas" : 2, "ok" : 1 }

> db.Student.getIndexes()

[ { "v" : 2, "key" : { "\_id" : 1 }, "name" : "\_id\_" } ]

**Aggregation:**

>db.Student.aggregate({$group:{\_id:"$Status", sum:{$sum:"$Stud\_Roll\_No"}}})

{ "\_id" : "Failed", "sum" : 9 }

{ "\_id" : "Passed", "sum" : 6 }

>db.Student.aggregate({$group:{\_id:"$Stud\_Roll\_No", avg:{$avg:"$Stud\_Roll\_No"}}})

{ "\_id" : 1, "avg" : 1 }

{ "\_id" : 2, "avg" : 2 }

{ "\_id" : 3, "avg" : 3 }

{ "\_id" : 5, "avg" : 5 }

{ "\_id" : 4, "avg" : 4 }

>db.Student.aggregate({$group:{\_id:"$Status", avg:{$avg:"$Stud\_Roll\_No"}}})

{ "\_id" : "Failed", "avg" : 4.5 }

{ "\_id" : "Passed", "avg" : 2 }

> db.Student.aggregate({$group:{\_id:"$max", max:{$max:"$Stud\_Roll\_No"}}})

{ "\_id" : null, "max" : 5 }

> db.Student.aggregate({$group:{\_id:"$min", min:{$min:"$Stud\_Roll\_No"}}})

{ "\_id" : null, "min" : 1 }

> db.Student.aggregate({$group:{\_id:"$Status", name:{$push:"$Stud\_Name"}}})

{ "\_id" : "Failed", "name" : [ "Joey", "Srinidhi" ] }

{ "\_id" : "Passed", "name" : [ "Aarohi", "Vrushali", "Monica" ] }

>db.Student.aggregate({$group:{\_id:"$Status",sum:{$sum:"$Stud\_Roll\_No"}}}, $match:{sum:{$lte: 7}}})

{ "\_id" : "Passed", "sum" : 6 }

>db.Student.aggregate({$group:{\_id:"$Status",sum:{$sum:"$Stud\_Roll\_No"}}}, {$match:{sum:{$gte: 7}}})

{ "\_id" : "Failed", "sum" : 9 }

**3.**

Implement queries using MongoDB

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Teacher\_id | Teacher\_Name | Dept\_Name, | Salary | Status |
| Pic001 | Ravi | IT | 30000 | A |
| Pic002 | Mangesh | IT | 20000 | A |
| Pic003 | Akshay | Comp | 25000 | N |

1. Create Collection, Insert data into collection,FindAll,findOne(with condition)

> db.createCollection("Teacher")

{ "ok" : 1 }

> db.Teacher.insert({Teacher\_id: "Pic001", Teacher\_Name: "Ravi", Dept\_Name: "IT", Salary:30000, Status: "A"})

WriteResult({ "nInserted" : 1 })

> db.Teacher.insert({Teacher\_id: "Pic002", Teacher\_Name: "Mangesh", Dept\_Name: "IT",Salary: 20000, Status: "A"})

WriteResult({ "nInserted" : 1 })

> db.Teacher.insert({Teacher\_id: "Pic003", Teacher\_Name: "Akshay", Dept\_Name: "Comp", Salary: 25000, Status: "N"})

WriteResult({ "nInserted" : 1 })

> db.Teacher.find();

{ "\_id" : ObjectId("63612967a2e9accf7753dbfd"), "Teacher\_id" : "Pic001", "Teacher\_Name": "Ravi", "Dept\_Name" : "IT", "Salary" : 30000, "Status" : "A" }

{ "\_id" : ObjectId("6361297aa2e9accf7753dbfe"), "Teacher\_id" : "Pic002", "Teacher\_Name" : "Mangesh", "Dept\_Name" : "IT", "Salary" : 20000, "Status" : "A" }

{ "\_id" : ObjectId("63612995a2e9accf7753dbff"), "Teacher\_id" : "Pic003", "Teacher\_Name" : "Akshay", "Dept\_Name" : "Comp", "Salary" : 25000, "Status" : "N" }

> db.Teacher.findOne();

{

"\_id" : ObjectId("63612967a2e9accf7753dbfd"),

"Teacher\_id" : "Pic001",

"Teacher\_Name" : "Ravi",

"Dept\_Name" : "IT",

"Salary" : 30000,

"Status" : "A"

}

> db.Teacher.findOne({Teacher\_id:"Pic003"});

{

"\_id" : ObjectId("63612995a2e9accf7753dbff"),

"Teacher\_id" : "Pic003",

"Teacher\_Name" : "Akshay",

"Dept\_Name" : "Comp",

"Salary" : 25000,

"Status" : "N"

}

> db.Teacher.findOne({Salary:{$eq: 30000}});

{

"\_id" : ObjectId("63612967a2e9accf7753dbfd"),

"Teacher\_id" : "Pic001",

"Teacher\_Name" : "Ravi",

"Dept\_Name" : "IT",

"Salary" : 30000,

"Status" : "A"

}

> db.Teacher.find({Salary:{$eq: 30000}});

{ "\_id" : ObjectId("63612967a2e9accf7753dbfd"), "Teacher\_id" : "Pic001", "Teacher\_Name" : "Ravi", "Dept\_Name" : "IT", "Salary" : 30000, "Status" : "A" }

1. Find teacher who is having salary greater than 50000 and status is A

>db.Teacher.find({$and:[{Salary:{$gte: 30000}},{Status:{$eq:"A"}}]}).pretty();

{

"\_id" : ObjectId("63612967a2e9accf7753dbfd"),

"Teacher\_id" : "Pic001",

"Teacher\_Name" : "Ravi",

"Dept\_Name" : "IT",

"Salary" : 30000,

"Status" : "A"

}

1. Find teacher who is having salary greater than 50000 OR status is A

> db.Teacher.find({$or:[{Salary:{$gte: 30000}},{Status:{$eq:"A"}}]}).pretty();

{

"\_id" : ObjectId("63612967a2e9accf7753dbfd"),

"Teacher\_id" : "Pic001",

"Teacher\_Name" : "Ravi",

"Dept\_Name" : "IT",

"Salary" : 30000,

"Status" : "A"

}

{

"\_id" : ObjectId("6361297aa2e9accf7753dbfe"),

"Teacher\_id" : "Pic002",

"Teacher\_Name" : "Mangesh",

"Dept\_Name" : "IT",

"Salary" : 20000,

"Status" : "A"

}

1. Display teacher info in ascending and descending order.

> db.Teacher.find().sort({key:1}).pretty();

{

"\_id" : ObjectId("63612967a2e9accf7753dbfd"),

"Teacher\_id" : "Pic001",

"Teacher\_Name" : "Ravi",

"Dept\_Name" : "IT",

"Salary" : 30000,

"Status" : "A"

}

{

"\_id" : ObjectId("6361297aa2e9accf7753dbfe"),

"Teacher\_id" : "Pic002",

"Teacher\_Name" : "Mangesh",

"Dept\_Name" : "IT",

"Salary" : 20000,

"Status" : "A"

}

{

"\_id" : ObjectId("63612995a2e9accf7753dbff"),

"Teacher\_id" : "Pic003",

"Teacher\_Name" : "Akshay",

"Dept\_Name" : "Comp",

"Salary" : 25000,

"Status" : "N"

}

> db.Teacher.find().sort({key:-1}).pretty();

{

"\_id" : ObjectId("63612967a2e9accf7753dbfd"),

"Teacher\_id" : "Pic001",

"Teacher\_Name" : "Ravi",

"Dept\_Name" : "IT",

"Salary" : 30000,

"Status" : "A"

}

{

"\_id" : ObjectId("6361297aa2e9accf7753dbfe"),

"Teacher\_id" : "Pic002",

"Teacher\_Name" : "Mangesh",

"Dept\_Name" : "IT",

"Salary" : 20000,

"Status" : "A"

}

{

"\_id" : ObjectId("63612995a2e9accf7753dbff"),

"Teacher\_id" : "Pic003",

"Teacher\_Name" : "Akshay",

"Dept\_Name" : "Comp",

"Salary" : 25000,

"Status" : "N"

}

1. Find teacher from different departments.

> db.Teacher.find({Dept\_Name: "IT"});

{ "\_id" : ObjectId("63612967a2e9accf7753dbfd"), "Teacher\_id" : "Pic001", "Teacher\_Name" : "Ravi", "Dept\_Name" : "IT", "Salary" : 30000, "Status" : "A" }

{ "\_id" : ObjectId("6361297aa2e9accf7753dbfe"), "Teacher\_id" : "Pic002", "Teacher\_Name" : "Mangesh", "Dept\_Name" : "IT", "Salary" : 20000, "Status" : "A" }

> db.Teacher.find({Dept\_Name: "Comp"});

{ "\_id" : ObjectId("63612995a2e9accf7753dbff"), "Teacher\_id" : "Pic003", "Teacher\_Name" : "Akshay", "Dept\_Name" : "Comp", "Salary" : 25000, "Status" : "N" }

1. Update dept\_name to ETC of techer\_id= Pic002

>db.Teacher.update({Teacher\_id: "Pic002"},{$set:{Dept\_Name:"ENTC"}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.Teacher.find()

{ "\_id" : ObjectId("63612967a2e9accf7753dbfd"), "Teacher\_id" : "Pic001", "Teacher\_Name" : "Ravi", "Dept\_Name" : "IT", "Salary" : 30000, "Status" : "A" }

{ "\_id" : ObjectId("6361297aa2e9accf7753dbfe"), "Teacher\_id" : "Pic002", "Teacher\_Name" : "Mangesh", "Dept\_Name" : "ENTC", "Salary" : 20000, "Status" : "A" }

{ "\_id" : ObjectId("63612995a2e9accf7753dbff"), "Teacher\_id" : "Pic003", "Teacher\_Name" : "Akshay", "Dept\_Name" : "Comp", "Salary" : 25000, "Status" : "N" }

1. Increment the salary of teacher by 10000 who is having Status A

> db.Teacher.updateMany({},{$inc: {Salary: 10000}})

{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }

> db.Teacher.updateMany({Status: "A"},{$inc: {Salary: 10000}})

{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }

> db.Teacher.find()

{ "\_id" : ObjectId("63612967a2e9accf7753dbfd"), "Teacher\_id" : "Pic001", "Teacher\_Name" : "Ravi", "Dept\_Name" : "IT", "Salary" : 50000, "Status" : "A" }

{ "\_id" : ObjectId("6361297aa2e9accf7753dbfe"), "Teacher\_id" : "Pic002", "Teacher\_Name" : "Mangesh", "Dept\_Name" : "ENTC", "Salary" : 40000, "Status" : "A" }

{ "\_id" : ObjectId("63612995a2e9accf7753dbff"), "Teacher\_id" : "Pic003", "Teacher\_Name" : "Akshay", "Dept\_Name" : "Comp", "Salary" : 35000, "Status" : "N" }

1. Delete teacher of teacher\_id=Pic001

> db.Teacher.deleteOne({Teacher\_id: "Pic001"})

{ "acknowledged" : true, "deletedCount" : 1 }

> db.Teacher.find()

{ "\_id" : ObjectId("6361297aa2e9accf7753dbfe"), "Teacher\_id" : "Pic002", "Teacher\_Name" : "Mangesh", "Dept\_Name" : "ENTC", "Salary" : 40000, "Status" : "A" }

{ "\_id" : ObjectId("63612995a2e9accf7753dbff"), "Teacher\_id" : "Pic003", "Teacher\_Name" : "Akshay", "Dept\_Name" : "Comp", "Salary" : 35000, "Status" : "N" }

**4.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student\_id | Student\_Name | Dept\_Name, | Fees | Result |
| 101E | Ravi | IT | 30000 | Pass |
| 102E | Mangesh | IT | 20000 | Pass |
| 103F | Akshay | Comp | 25000 | Fail |

* Insert one document at a time

> db.Chit4.insert({Student\_id: "101E", Student\_Name: "Ravi", Dept\_Name: "IT", Fees: 30000, Result: "Pass"})

WriteResult({ "nInserted" : 1 })

* Insert Multiple documents using batch insert.

> var student= [{Student\_id: "102E", Student\_Name: "Mangesh", Dept\_Name: "IT", Fees: 20000, Result: "Pass"},{Student\_id: "103F", Student\_Name: "Akshay", Dept\_Name: "Comp", Fees: 25000, Result: "Fail"}]

> db.Chit4.insert(student)

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 2,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

Or

>db.Chit4.insertMany([{Student\_id: "102E", Student\_Name: "Mangesh", Dept\_Name: "IT", Fees: 20000, Result: "Pass"},{Student\_id: "103F", Student\_Name: "Akshay", Dept\_Name: "Comp", Fees: 25000, Result: "Fail"}])

> db.Chit4.find()

{ "\_id" : ObjectId("63614089a2e9accf7753dc02"), "Student\_id" : "101E", "Student\_Name" : "Ravi", "Dept\_Name" : "IT", "Fees" : 30000, "Result" : "Pass" }

{ "\_id" : ObjectId("636140dea2e9accf7753dc03"), "Student\_id" : "102E", "Student\_Name" : "Mangesh", "Dept\_Name" : "IT", "Fees" : 20000, "Result" : "Pass" }

{ "\_id" : ObjectId("636140dea2e9accf7753dc04"), "Student\_id" : "103F", "Student\_Name" : "Akshay", "Dept\_Name" : "Comp", "Fees" : 25000, "Result" : "Fail" }

* Remove a document using $where.

> db.Chit4.remove({$where: function() {return (this.Student\_id == "101E")}});

WriteResult({ "nRemoved" : 1 })

* Update a document using $where.
* Upserting a document using save().

> db.Chit4.save({Student\_id: "106A", Student\_Name: "Ryan", Dept\_Name: "IT", Fees: 40000, Result: "Pass"})

WriteResult({ "nInserted" : 1 })

> db.Chit4.find();

{ "\_id" : ObjectId("636140dea2e9accf7753dc03"), "Student\_id" : "102E", "Student\_Name" : "Mangesh", "Dept\_Name" : "IT", "Fees" : 20000, "Result" : "Pass" }

{ "\_id" : ObjectId("636140dea2e9accf7753dc04"), "Student\_id" : "103F", "Student\_Name" : "Akshay", "Dept\_Name" : "Comp", "Fees" : 25000, "Result" : "Fail" }

{ "\_id" : ObjectId("6365075025d332a3c95283bd"), "Student\_id" : "106A", "Student\_Name" : "Ryan", "Dept\_Name" : "IT", "Fees" : 40000, "Result" : "Pass" }

>

**5.**

Implement aggregation and indexing (all three) with example using MongoDB

**Indexing:**

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

{

"\_id" : ObjectId("631ecd5bb34e1a691abbaafc"),

"Stud\_id" : 1,

"Stud\_Name" : "Rachel",

"Stud\_Age" : 18

}

{

"\_id" : ObjectId("631ecd9db34e1a691abbaafd"),

"Stud\_id" : 2,

"Stud\_Name" : "Lily",

"Stud\_Age" : 19

}

{

"\_id" : ObjectId("631ecee1b34e1a691abbaafe"),

"Stud\_id" : 4,

"Stud\_Name" : "Emily",

"Stud\_Age" : 19

}

{

"\_id" : ObjectId("631ecf39b34e1a691abbaaff"),

"Stud\_id" : 7,

"Stud\_Name" : "Joey",

"Stud\_Age" : 21

}

{

"\_id" : ObjectId("631ecf7ab34e1a691abbab00"),

"Stud\_id" : 3,

"Stud\_Name" : "Josh",

"Stud\_Age" : 18

}

{

"\_id" : ObjectId("631ecf91b34e1a691abbab01"),

"Stud\_id" : 5,

"Stud\_Name" : "Christine",

"Stud\_Age" : 18

}

{

"\_id" : ObjectId("631ecfa2b34e1a691abbab02"),

"Stud\_id" : 6,

"Stud\_Name" : "Samantha",

"Stud\_Age" : 20

}

{

"\_id" : ObjectId("631ecfb4b34e1a691abbab03"),

"Stud\_id" : 8,

"Stud\_Name" : "Ernie",

"Stud\_Age" : 17

}

**1. Create Index:**

> db.Student.createIndex({Stud\_id: 01})

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

> db.Student.createIndex({Stud\_id: 06})

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 2,

"numIndexesAfter" : 3,

"ok" : 1

}

**2. Get Index:**

> db.Student.getIndexes()

[

{

"v" : 2,

"key" : {

"\_id" : 1

},

"name" : "\_id\_",

"ns" : "Aishwarya.Student"

},

{

"v" : 2,

"key" : {

"Stud\_id" : 1

},

"name" : "Stud\_id\_1",

"ns" : "Aishwarya.Student"

},

{

"v" : 2,

"key" : {

"Stud\_id" : 6

},

"name" : "Stud\_id\_6",

"ns" : "Aishwarya.Student"

}

]

**3. Drop Index:**

> db.Student.dropIndex({Stud\_id: 6})

{ "nIndexesWas" : 3, "ok" : 1 }

**Aggregation Functions:**

> db.createCollection("Employee")

{ "ok" : 1 }

> db.Employee.insert({Emp\_id: 01, Emp\_Name: "Emily", Emp\_Sal: 10000, Emp\_Dept: "IT"});

WriteResult({ "nInserted" : 1 })

> db.Employee.insert({Emp\_id: 02, Emp\_Name: "Christine", Emp\_Sal: 15000, Emp\_Dept: "IT"});

WriteResult({ "nInserted" : 1 })

> db.Employee.insert({Emp\_id: 03, Emp\_Name: "Alexa", Emp\_Sal: 25000, Emp\_Dept: "Comp"});

WriteResult({ "nInserted" : 1 })

> db.Employee.insert({Emp\_id: 04, Emp\_Name: "Albert", Emp\_Sal: 10000, Emp\_Dept: "Comp"});

WriteResult({ "nInserted" : 1 })

> db.Employee.insert({Emp\_id: 05, Emp\_Name: "Annabella", Emp\_Sal: 100000, Emp\_Dept: "IT"});

WriteResult({ "nInserted" : 1 })

> db.Employee.insert({Emp\_id: 06, Emp\_Name: "Joey", Emp\_Sal: 30000, Emp\_Dept: "Civil"});

WriteResult({ "nInserted" : 1 })

> db.Employee.insert({Emp\_id: 07, Emp\_Name: "Emma", Emp\_Sal: 30000, Emp\_Dept: "Mech"});

WriteResult({ "nInserted" : 1 })

> db.Employee.insert({Emp\_id: 08, Emp\_Name: "Ariana", Emp\_Sal: 20000, Emp\_Dept: "ENTC"});

WriteResult({ "nInserted" : 1 })

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

{

"\_id" : ObjectId("631ed50db34e1a691abbab05"),

"Emp\_id" : 1,

"Emp\_Name" : "Emily",

"Emp\_Sal" : 10000,

"Emp\_Dept" : "IT"

}

{

"\_id" : ObjectId("631ed532b34e1a691abbab06"),

"Emp\_id" : 2,

"Emp\_Name" : "Christine",

"Emp\_Sal" : 15000,

"Emp\_Dept" : "IT"

}

{

"\_id" : ObjectId("631ed54ab34e1a691abbab07"),

"Emp\_id" : 3,

"Emp\_Name" : "Alexa",

"Emp\_Sal" : 25000,

"Emp\_Dept" : "Comp"

}

{

"\_id" : ObjectId("631ed55eb34e1a691abbab08"),

"Emp\_id" : 4,

"Emp\_Name" : "Albert",

"Emp\_Sal" : 10000,

"Emp\_Dept" : "Comp"

}

{

"\_id" : ObjectId("631ed581b34e1a691abbab09"),

"Emp\_id" : 5,

"Emp\_Name" : "Annabella",

"Emp\_Sal" : 100000,

"Emp\_Dept" : "IT"

}

{

"\_id" : ObjectId("631ed5a5b34e1a691abbab0a"),

"Emp\_id" : 6,

"Emp\_Name" : "Joey",

"Emp\_Sal" : 30000,

"Emp\_Dept" : "Civil"

}

{

"\_id" : ObjectId("631ed5c6b34e1a691abbab0b"),

"Emp\_id" : 7,

"Emp\_Name" : "Emma",

"Emp\_Sal" : 30000,

"Emp\_Dept" : "Mech"

}

{

"\_id" : ObjectId("631ed5e4b34e1a691abbab0c"),

"Emp\_id" : 8,

"Emp\_Name" : "Ariana",

"Emp\_Sal" : 20000,

"Emp\_Dept" : "ENTC"

}

**1. Sum:**

> db.Employee.aggregate({$group:{\_id:"$Emp\_Dept" ,sum:{$sum:"$Emp\_Sal"}}})

{ "\_id" : "ENTC", "sum" : 20000 }

{ "\_id" : "Mech", "sum" : 30000 }

{ "\_id" : "Comp", "sum" : 35000 }

{ "\_id" : "Civil", "sum" : 30000 }

{ "\_id" : "IT", "sum" : 125000 }

**2.Average :**

> db.Employee.aggregate({$group:{\_id : "$Emp\_Dept, avg :{$avg : "$Emp\_Sal"}}})

{ "\_id" : "ENTC", "sum" : 20000 }

{ "\_id" : "Mech", "sum" : 30000 }

{ "\_id" : "Comp", "sum" : 17500 }

{ "\_id" : "Civil", "sum" : 30000 }

{ "\_id" : "IT", "sum" : 41666.6666666 }

**3.Maximum :**

> db.Employee.aggregate({$group:{\_id : "$max”, max:{$max : "$Emp\_Sal"}}})

{ "\_id" : “IT”, "max" : 100000 }

**4.Min :**

> db.Employee.aggregate({$group:{\_id : "$min”, min:{$min : "$Emp\_Sal"}}})

{ "\_id" : "ENTC", "sum" : 20000 }

**5.Push :**

> db.Employee.aggregate({$group:{\_id : "$Emp\_Dept", name :{$push : "$Emp\_Name"}}})

{ "\_id" : "null", " Emp\_Name" : [ "Emily"]}

{ "\_id" : "IT", " Emp\_Name" : [ "Emily", "Christine", "Anabella" ] }

{ "\_id" : "Comp", " Emp\_Name" : [ "Alexa", "Albert" ] }

{ "\_id" : "Civil", " Emp\_Name" : [ "Joey" ] }

{ "\_id" : "Mech", " Emp\_Name" : [ "Emma" ] }

{ "\_id" : "ENTC", " Emp\_Name" : [ "Ariana" ] }

**6. Match:**

> db.Employee.aggregate({$group:{\_id:"$Emp\_Dept" ,sum:{$sum:"$Emp\_Sal"}}},{$match:{sum:{$gte:120000}}})

{ "\_id" : "IT", "sum" : 125000 }

> db.Employee.aggregate({$group:{\_id:"$Emp\_Dept" ,sum:{$sum:"$Emp\_Sal"}}},{$match:{sum:{$lte:120000}}})

{ "\_id" : "ENTC", "sum" : 20000 }

{ "\_id" : "Mech", "sum" : 30000 }

{ "\_id" : null, "sum" : 10000 }

{ "\_id" : "Comp", "sum" : 35000 }

{ "\_id" : "Civil", "sum" : 30000 }

**6.**

Execute at least 10 queries on following database that demonstrates following querying

techniques:

Book(Book\_id,Book\_Name,Author,Price,No\_of\_Pages)

* Display all books from the collection.

> db.Book.insertMany([{Book\_Name: "Wings of Fire", Author: "ABC", Price: 350, No\_of\_Pages: 300},{Book\_Name: "Hello its Me", Author: "ABCD", Price: 4550, No\_of\_Pages: 600},{Book\_Name: "Aishwarya", Author: "ABCDE", Price: 550, No\_of\_Pages: 200}])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("6361512da2e9accf7753dc05"),

ObjectId("6361512da2e9accf7753dc06"),

ObjectId("6361512da2e9accf7753dc07")

]

}

> db.Book.find()

{ "\_id" : ObjectId("6361512da2e9accf7753dc05"), "Book\_Name" : "Wings of Fire", "Author" : "ABC", "Price" : 350, "No\_of\_Pages" : 300 }

{ "\_id" : ObjectId("6361512da2e9accf7753dc06"), "Book\_Name" : "Hello its Me", "Author" : "ABCD", "Price" : 4550, "No\_of\_Pages" : 600 }

{ "\_id" : ObjectId("6361512da2e9accf7753dc07"), "Book\_Name" : "Aishwarya", "Author" : "ABCDE", "Price" : 550, "No\_of\_Pages" : 200 }

* Display one book using findOne

> db.Book.findOne()

{

"\_id" : ObjectId("6361512da2e9accf7753dc05"),

"Book\_Name" : "Wings of Fire",

"Author" : "ABC",

"Price" : 350,

"No\_of\_Pages" : 300

}

> db.Book.findOne({Book\_Name:"Aishwarya"})

{

"\_id" : ObjectId("6361512da2e9accf7753dc07"),

"Book\_Name" : "Aishwarya",

"Author" : "ABCDE",

"Price" : 550,

"No\_of\_Pages" : 200

}

* Display all books having price greater than 300 using $gt.

> db.Book.find({Price:{$gt: 350}})

{ "\_id" : ObjectId("6361512da2e9accf7753dc06"), "Book\_Name" : "Hello its Me", "Author" : "ABCD", "Price" : 4550, "No\_of\_Pages" : 600 }

{ "\_id" : ObjectId("6361512da2e9accf7753dc07"), "Book\_Name" : "Aishwarya", "Author" : "ABCDE", "Price" : 550, "No\_of\_Pages" : 200 }

* Display all books having price less than 300 using $lt AND No\_of\_pages greater than 1000 using $gt.

> db.Book.find({$and: [{Price:{$gt: 350}},{No\_of\_Pages:{$gt:200}}]})

{ "\_id" : ObjectId("6361512da2e9accf7753dc06"), "Book\_Name" : "Hello its Me", "Author" : "ABCD", "Price" : 4550, "No\_of\_Pages" : 600 }

* Display all books having price less than or equal to 300 using $lte ORNo\_of\_pages greater than or equal to 1000 using $gte.

> db.Book.find({$or:[{Price:{$lte: 350}},{No\_of\_Pages:{$gte: 1000}}]}).pretty();

{

"\_id" : ObjectId("6361512da2e9accf7753dc05"),

"Book\_Name" : "Wings of Fire",

"Author" : "ABC",

"Price" : 350,

"No\_of\_Pages" : 300

}

> db.Book.find({$or:[{Price:{$lte: 350}},{No\_of\_Pages:{$gte: 500}}]}).pretty();

{

"\_id" : ObjectId("6361512da2e9accf7753dc05"),

"Book\_Name" : "Wings of Fire",

"Author" : "ABC",

"Price" : 350,

"No\_of\_Pages" : 300

}

{

"\_id" : ObjectId("6361512da2e9accf7753dc06"),

"Book\_Name" : "Hello its Me",

"Author" : "ABCD",

"Price" : 4550,

"No\_of\_Pages" : 600

}

* Use $not.

> db.Book.find({Price:{$not:{$lte: 400}}}).pretty();

{

"\_id" : ObjectId("6361512da2e9accf7753dc06"),

"Book\_Name" : "Hello its Me",

"Author" : "ABCD",

"Price" : 4550,

"No\_of\_Pages" : 600

}

{

"\_id" : ObjectId("6361512da2e9accf7753dc07"),

"Book\_Name" : "Aishwarya",

"Author" : "ABCDE",

"Price" : 550,

"No\_of\_Pages" : 200

}

* Accept a Null value in a document.

> db.Book.insert({Book\_Name: "Random", Author: "Ariana", Price: 700, No\_of\_Pages: ""})

WriteResult({ "nInserted" : 1 })

> db.Book.find();

{ "\_id" : ObjectId("6361512da2e9accf7753dc05"), "Book\_Name" : "Wings of Fire", "Author" : "ABC", "Price" : 350, "No\_of\_Pages" : 300 }

{ "\_id" : ObjectId("6361512da2e9accf7753dc06"), "Book\_Name" : "Hello its Me", "Author" : "ABCD", "Price" : 4550, "No\_of\_Pages" : 600 }

{ "\_id" : ObjectId("6361512da2e9accf7753dc07"), "Book\_Name" : "Aishwarya", "Author" : "ABCDE", "Price" : 550, "No\_of\_Pages" : 200 }

{ "\_id" : ObjectId("636509d625d332a3c95283bf"), "Book\_Name" : "Random", "Author" : "Ariana", "Price" : 700, "No\_of\_Pages" : "" }

>

* Find all books whose name starts with ‘b’ using $regex.

> db.Book.find({Book\_Name: {$regex: /ing/i}})

{ "\_id" : ObjectId("6361512da2e9accf7753dc05"), "Book\_Name" : "Wings of Fire", "Author" : "ABC", "Price" : 350, "No\_of\_Pages" : 300 }

> db.Book.find({Book\_Name: {$regex: /rya/i}})

{ "\_id" : ObjectId("6361512da2e9accf7753dc07"), "Book\_Name" : "Aishwarya", "Author" : "ABCDE", "Price" : 550, "No\_of\_Pages" : 200 }

**7.**

Execute at least 10 queries on any suitable MongoDB database that demonstrates following:

Mobile\_Specs(Mobile\_Name,RAM,Price,Camera)

> db.createCollection("Mobile\_Specs")

{ "ok" : 1 }

> db.Mobile\_Specs.insertMany([{Mobile\_Name: "Realme", RAM: 16, Price: 15000, Camera: 17},{Mobile\_Name: "Oppo", RAM: 32, Price: 20000, Camera: 45},])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("63650ace25d332a3c95283c0"),

ObjectId("63650ace25d332a3c95283c1")

]

}

> db.Mobile\_Specs.insertMany([{Mobile\_Name: "Redmi", RAM: 64, Price: 18000, Camera: 12},{Mobile\_Name: "Poco", RAM: 32, Price: 27000, Camera: 68},{Mobile\_Name: "Iphone", RAM: 512, Price: 70000, Camera: 90},{Mobile\_Name: "Realme", RAM: 32, Price: 25000, Camera: 40},{Mobile\_Name: "Itel", RAM: 16, Price: 12000, Camera: 12}])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("63650b9525d332a3c95283c2"),

ObjectId("63650b9525d332a3c95283c3"),

ObjectId("63650b9525d332a3c95283c4"),

ObjectId("63650b9525d332a3c95283c5"),

ObjectId("63650b9525d332a3c95283c6")

]

}

> db.Mobile\_Specs.find();

{ "\_id" : ObjectId("63650ace25d332a3c95283c0"), "Mobile\_Name" : "Realme", "RAM" : 16, "Price" : 15000, "Camera" : 17 }

{ "\_id" : ObjectId("63650ace25d332a3c95283c1"), "Mobile\_Name" : "Oppo", "RAM" : 32, "Price" : 20000, "Camera" : 45 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c2"), "Mobile\_Name" : "Redmi", "RAM" : 64, "Price" : 18000, "Camera" : 12 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c3"), "Mobile\_Name" : "Poco", "RAM" : 32, "Price" : 27000, "Camera" : 68 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c4"), "Mobile\_Name" : "Iphone", "RAM" : 512, "Price" : 70000, "Camera" : 90 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c5"), "Mobile\_Name" : "Realme", "RAM" : 32, "Price" : 25000, "Camera" : 40 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c6"), "Mobile\_Name" : "Itel", "RAM" : 16, "Price" : 12000, "Camera" : 12 }

* Find all mobiles which have 16GB RAM using $where.

> db.Mobile\_Specs.find({$where: function() {return (this.RAM == "16")}});

{ "\_id" : ObjectId("63650ace25d332a3c95283c0"), "Mobile\_Name" : "Realme", "RAM" : 16, "Price" : 15000, "Camera" : 17 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c6"), "Mobile\_Name" : "Itel", "RAM" : 16, "Price" : 12000, "Camera" : 12 }

* Limit the display records to 5.

> db.Mobile\_Specs.find().limit(5)

{ "\_id" : ObjectId("63650ace25d332a3c95283c0"), "Mobile\_Name" : "Realme", "RAM" : 16, "Price" : 15000, "Camera" : 17 }

{ "\_id" : ObjectId("63650ace25d332a3c95283c1"), "Mobile\_Name" : "Oppo", "RAM" : 32, "Price" : 20000, "Camera" : 45 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c2"), "Mobile\_Name" : "Redmi", "RAM" : 64, "Price" : 18000, "Camera" : 12 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c3"), "Mobile\_Name" : "Poco", "RAM" : 32, "Price" : 27000, "Camera" : 68 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c4"), "Mobile\_Name" : "Iphone", "RAM" : 512, "Price" : 70000, "Camera" : 90 }

* Sort the mobiles in ascending order in price.

> db.Mobile\_Specs.find().sort({Price: 1})

{ "\_id" : ObjectId("63650b9525d332a3c95283c6"), "Mobile\_Name" : "Itel", "RAM" : 16, "Price" : 12000, "Camera" : 12 }

{ "\_id" : ObjectId("63650ace25d332a3c95283c0"), "Mobile\_Name" : "Realme", "RAM" : 16, "Price" : 15000, "Camera" : 17 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c2"), "Mobile\_Name" : "Redmi", "RAM" : 64, "Price" : 18000, "Camera" : 12 }

{ "\_id" : ObjectId("63650ace25d332a3c95283c1"), "Mobile\_Name" : "Oppo", "RAM" : 32, "Price" : 20000, "Camera" : 45 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c5"), "Mobile\_Name" : "Realme", "RAM" : 32, "Price" : 25000, "Camera" : 40 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c3"), "Mobile\_Name" : "Poco", "RAM" : 32, "Price" : 27000, "Camera" : 68 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c4"), "Mobile\_Name" : "Iphone", "RAM" : 512, "Price" : 70000, "Camera" : 90 }

* Sort the mobiles in descending order of RAM.

> db.Mobile\_Specs.find().sort({RAM: -1})

{ "\_id" : ObjectId("63650b9525d332a3c95283c4"), "Mobile\_Name" : "Iphone", "RAM" : 512, "Price" : 70000, "Camera" : 90 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c2"), "Mobile\_Name" : "Redmi", "RAM" : 64, "Price" : 18000, "Camera" : 12 }

{ "\_id" : ObjectId("63650ace25d332a3c95283c1"), "Mobile\_Name" : "Oppo", "RAM" : 32, "Price" : 20000, "Camera" : 45 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c3"), "Mobile\_Name" : "Poco", "RAM" : 32, "Price" : 27000, "Camera" : 68 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c5"), "Mobile\_Name" : "Realme", "RAM" : 32, "Price" : 25000, "Camera" : 40 }

{ "\_id" : ObjectId("63650ace25d332a3c95283c0"), "Mobile\_Name" : "Realme", "RAM" : 16, "Price" : 15000, "Camera" : 17 }

{ "\_id" : ObjectId("63650b9525d332a3c95283c6"), "Mobile\_Name" : "Itel", "RAM" : 16, "Price" : 12000, "Camera" : 12 }

* Skip the first 5 records using cursor while displaying.

> var mycursor=db.Mobile\_Specs.find().skip(5).pretty();

> mycursor;

{

"\_id" : ObjectId("63650b9525d332a3c95283c5"),

"Mobile\_Name" : "Realme",

"RAM" : 32,

"Price" : 25000,

"Camera" : 40

}

{

"\_id" : ObjectId("63650b9525d332a3c95283c6"),

"Mobile\_Name" : "Itel",

"RAM" : 16,

"Price" : 12000,

"Camera" : 12

}

**8.**

Implement aggregation and indexing with suitable example using MongoDB (Same AS Chit 5)

**9.**

**Example** : In this Assignment, we are creating Teacher Database. Which contain the information of Teacher\_id, name of a teacher, department of a teacher, salary and status of a teacher. Here status is whether teacher is approved by the university or not. (It’s Same As Chit 3)

1. Implement Map reduces operation using MongoDB.

Problem: College student data (FE,SE,TE,BE)

> db.createCollection("Students")

{ "ok" : 1 }

> db.Students.insert({Stud\_Name: "Christine", Stud\_Year: "SE", Pending\_Fees: 25000});

WriteResult({ "nInserted" : 1 })

> db.Students.insert({Stud\_Name: "Sydney", Stud\_Year: "TE", Pending\_Fees: 40000});

WriteResult({ "nInserted" : 1 })

> db.Students.insert({Stud\_Name: "Chandler", Stud\_Year: "FE", Pending\_Fees: 7000});

WriteResult({ "nInserted" : 1 })

> db.Students.insert({Stud\_Name: "Joshua", Stud\_Year: "TE", Pending\_Fees: 30000});

WriteResult({ "nInserted" : 1 })

> db.Students.insert({Stud\_Name: "Jeremy", Stud\_Year: "SE", Pending\_Fees: 20000});

WriteResult({ "nInserted" : 1 })

> db.Students.insert({Stud\_Name: "Joey", Stud\_Year: "FE", Pending\_Fees: 37000});

WriteResult({ "nInserted" : 1 })

> db.Students.insert({Stud\_Name: "Mary", Stud\_Year: "SE", Pending\_Fees: 44000});

WriteResult({ "nInserted" : 1 })

> db.Students.insert({Stud\_Name: "Martha", Stud\_Year: "BE", Pending\_Fees: 50000});

WriteResult({ "nInserted" : 1 })

> db.Students.insert({Stud\_Name: "Monica", Stud\_Year: "BE", Pending\_Fees: 70000});

WriteResult({ "nInserted" : 1 })

> var mapFunction1 =function() {emit(this.Stud\_Year, this.Pending\_Fees);};

> var reduceFunction1 =function (keyStud\_Year, Pending\_Fees){return Array.sum(Pending\_Fees);};

>db.Student.mapReduce(mapFunction1,reduceFunction1, {out:"Pending\_Fees\_List"})

{ "result" : "Pending\_Fees\_List", "ok" : 1 }

> db.Pending\_Fees\_List.find();

{ "\_id" : "SE", "value" : 89000 }

{ "\_id" : "FE", "value" : 44000 }

{ "\_id" : "TE", "value" : 70000 }

{ "\_id" : "BE", "value" : 120000 }

**10.**

Consider the following database:

Employee (emp\_no, name, skill, pay rate)

Insert one document at a time

> db.Employee.insert({Emp\_No: 1011, Name: "Srinidhi", Skill: "Developer", Pay\_Rate: 20000})

WriteResult({ "nInserted" : 1 })

Insert Multiple documents using batch insert.

> db.Employee.insertMany([{Emp\_No: 1012, Name: "Ovi", Skill: "Tester", Pay\_Rate: 25000},{Emp\_No: 1013, Name: "Sanchi", Skill: "Analyst Trainee", Pay\_Rate: 35000},{Emp\_No: 1014, Name: "Suresh", Skill: "Assistent Trainee", Pay\_Rate: 22000},{Emp\_No: 1015, Name: "Girish", Skill: "Programmer", Pay\_Rate: 29000}])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("636525c125d332a3c95283d1"),

ObjectId("636525c125d332a3c95283d2"),

ObjectId("636525c125d332a3c95283d3"),

ObjectId("636525c125d332a3c95283d4")

]

}

> db.Employee.find()

{ "\_id" : ObjectId("636524d025d332a3c95283d0"), "Emp\_No" : 1011, "Name" : "Srinidhi", "Skill" : "Developer", "Pay\_Rate" : 20000 }

{ "\_id" : ObjectId("636525c125d332a3c95283d1"), "Emp\_No" : 1012, "Name" : "Ovi", "Skill" : "Tester", "Pay\_Rate" : 25000 }

{ "\_id" : ObjectId("636525c125d332a3c95283d2"), "Emp\_No" : 1013, "Name" : "Sanchi", "Skill" : "Analyst Trainee", "Pay\_Rate" : 35000 }

{ "\_id" : ObjectId("636525c125d332a3c95283d3"), "Emp\_No" : 1014, "Name" : "Suresh", "Skill" : "Assistent Trainee", "Pay\_Rate" : 22000 }

{ "\_id" : ObjectId("636525c125d332a3c95283d4"), "Emp\_No" : 1015, "Name" : "Girish", "Skill" : "Programmer", "Pay\_Rate" : 29000 }

Remove a document using $where.

> db.Employee.remove({$where: function() {return (this.Emp\_No == "1013")}});

WriteResult({ "nRemoved" : 1 })

Update a document using $where.

Upserting a document using save().

> db.Employee.save({Emp\_No: 1016, Name: "Nilisha", Skill: "Senior Developer", Pay\_Rate: 40000})

WriteResult({ "nInserted" : 1 })

**11.**

Position (posting\_no, skill)

Duty\_allocation (posting\_no, emp\_no, day, shift(day/night))

Insert one document at a time

Insert Multiple documents using batch insert.

Remove a document using $where.

Update a document using $where.

Upserting a document using save().

(Same As Chit 4 and Chit 11)

**12.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student\_id | Student\_Name | Dept\_Name, | Fees | Result |
| 101E | Ravi | IT | 30000 | Pass |
| 102E | Mangesh | IT | 20000 | Pass |
| 103F | Akshay | Comp | 25000 | Fail |

Insert one document at a time

Insert Multiple documents using batch insert.

Remove a document using $where.

Update a document using $where.

Upserting a document using save(). (Same As Chit 4)