1.

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

a. 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" }
```

b. 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"
}
```

c. 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"
}
```

d. 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"
}
```

e. 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" }
```

f. 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" }
```

g. 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" }

h. 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.

a. 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().

12.

Insert one document at a time

Insert Multiple documents using batch insert.

Student_id	Student_Name	Dept_Name,	Fees	Result
101E	Ravi	IT	30000	Pass
102E	Mangesh	IT	20000	Pass
103F	Akshay	Comp	25000	Fail

Remove a document using $where.

Update a document using $where.

Upserting a document using save().