ADBMS Practical

Practical – 1

Create a DataBase with Example.

```
use collegeDB


db.createCollection("students")
db.createCollection("courses")



db.students.insertMany([
  {
    studentId: "S1001",
    Name: "Alice",
    email: "alicej@college.edu",
    CGPA: 7.85,
  },
  {
    studentId: "S1002",
    Name: "Bob",
    email: "bob.s@college.edu",
    gpa: 8.20,
  }
])



db.courses.insertMany([
  {
    courseId: "CS101",
    courseName: "Introduction to Programming",
    department: "Computer Science",
    credits: 3
```

```
    },

  {
    courseId: "ME201",
    courseName: "Thermodynamics",
    department: "Mechanical Engineering",
    credits: 4
  }
])
```

show collections

show db

Practical – 2

 # Create , implement , insert, remove , update , find and find one query criteria

use collegeDB

```
db.students.insertOne({
 student_id: "S101",
 name: "Amit Sharma",
 email: "amit@example.com",
 department: "Computer Science",
 year: 2
})

db.students.insertMany([
 {
  student_id: "S103",
```

```
    name: "Rahul Mehta",
    email: "rahul@example.com",
    department: "Electrical Engineering",
    year: 1
  },
  {
    student_id: "S104",
    name: "Sneha Kulkarni",
    email: "sneha@example.com",
    department: "Civil Engineering",
    year: 2
  }
])



# Remove One
db.students.deleteOne({ student_id: "S104" })
# Remove Many
db.students.deleteMany({ department: "Civil Engineering" })

#Update
db.students.updateOne(
  { student_id: "S103" },
  { $set: { year: 2 } }
)
db.students.updateMany(
  { department: "Computer Science" },
  { $set: { year: 3 } }
)

# Find
```

```
db.students.find()

#Find with criteria

db.students.find({ department: "Computer Science" })
```

Practical – 3

 # Create DataBase with suitable example implement where queies and cursors.

```
use collegeDB


db.students.insertMany([
  { student_id: "S101", name: "Amit", department: "CS", year: 2, marks: 78 },
  { student_id: "S102", name: "Priya", department: "ME", year: 3, marks: 85 },
  { student_id: "S103", name: "Rahul", department: "CS", year: 1, marks: 92 },
  { student_id: "S104", name: "Sneha", department: "EE", year: 2, marks: 67 },
  { student_id: "S105", name: "Jay", department: "CS", year: 3, marks: 88 }
])


# where

db.students.find({ $where: "this.marks > 80" })

db.students.find({ $where: "this.department == 'CS' && this.year > 2" })
```

```
#in

db.students.find({year: { $in: [2, 3] }})


#and

db.students.find({ $and: [ { department: "CS" }, { marks: { $gt: 80 } } ] })


#or

db.students.find({ $or: [ { department: "CS" }, { department: "EE" } ]})


#exists

db.students.find({ marks: { $exists: true } })


# sorting ascending

db.students.find().sort({ marks: 1 })

# sorting descending

db.students.find().sort({ marks: -1 })


#greater than

db.students.find({ marks: { $gte: 80 } })


# less than

db.students.find({ marks: { $lte: 70 } })


#Print all students name

var cursor = db.students.find()

cursor.forEach(function(doc) {

  print("Student Name: " + doc.name)

})


# cs student with > 80 marks
```

```
var cursor = db.students.find({ department: "CS", marks: { $gt: 80 } })
cursor.forEach(function(doc) {
  print(doc.name + " scored " + doc.marks)
})
```

Practical – 4

Implement Map Reduction in MongoDB

```
use collegeDB

db.students.insertMany( [
{
  "student_id": "S101",
  "name": "Amit",
  "department": "CS",
  "marks": 78
},
{
  "student_id": "S102",
  "name": “Yash",
```

```
  "department": "IT",
  "marks": 85
},
{
  "student_id": "S103",
  "name": "Nidhi",
  "department": "CS",
  "marks": 88
},
{
  "student_id": "S103",
  "name": "Tejas",
  "department": "IT",
  "marks": 92
} ] )
# Define the Map Function
var mapFunction = function() {
  emit(this.department, this.marks);
};

# Define the Reduce Function
var reduceFunction = function(department, marksArray) {
  var total = Array.sum(marksArray);
  return { total: total, count: marksArray.length };
};

# Finalize Function (Optional)
var finalizeFunction = function(department, reducedValue) {
  reducedValue.avg = reducedValue.total / reducedValue.count;
  return reducedValue;
};
```

```
# Run MapReduce
db.students.mapReduce(
  mapFunction,
  reduceFunction,
  {
    out: "avg_marks_per_department",
    finalize: finalizeFunction
  }
)

# View Results
db.avg_marks_per_department.find().pretty()
```

Practical – 5

Implement map aggregation in MongoDB

```
use collegeDB

db.students.insertOne( [
{
  "student_id": "S101",
  "name": "Amit",
  "marks": [78, 85, 90]
}
] )

# Goal: Add 5 Bonus Marks to Each Subject
```

1. Aggregation with `$map`

```
db.students.aggregate([
 {
  $project: {
   name: 1,
   originalMarks: "$marks",
   updatedMarks: {
    $map: {
     input: "$marks",
     as: "score",
     in: { $add: ["$$score", 5] }
    }
   }
  }
 }
])

# Convert Marks to Grade Labels
db.students.aggregate([
 {
  $project: {
   name: 1,
   gradeLabels: {
    $map: {
     input: "$marks",
     as: "score",
     in: {
      $cond: [
       { $gte: ["$$score", 90] }, "A",
       { $gte: ["$$score", 80] }, "B",
       { $gte: ["$$score", 70] }, "C",
```

```
          "D"
        ]
      }
    }
  }
 }
])
```

`$filter` — Select Array Elements Based on Condition

Goal: Keep only marks ≥ 80

```
db.students.aggregate([
 {
  $project: {
   name: 1,
   highMarks: {
    $filter: {
     input: "$marks",
     as: "score",
     cond: { $gte: ["$$score", 80] }
    }
   }
  }
 }
])
```