University Management System

Submitted for

Advance Database

Submitted by:

(502304202) Manmeet Kour

Submitted to-

Dr. Nitin Arvind Shelke

L M Thapar School of Management

Jan-May 2024

INDEX

Problem Statement

University management system

This university management advanced DBMS project aims to develop a robust database system for managing diverse university operations, including student enrollment, course scheduling, faculty assignments, and grade tracking. The system facilitates efficient data retrieval and manipulation, supports complex queries, and provides insightful reports to aid decision-making.

Key Features:

1. Student and Faculty Management
2. Course and Department Management
3. Academic Records
4. Grades Tracking

1. Creation of the Database

```
> use university
switched to db university
>
```

Creating Collections

```
> db.createCollection("students")
{ "ok" : 1 }
> db.createCollection("faculty")
{ "ok" : 1 }
> db.createCollection("courses")
{ "ok" : 1 }
> db.createCollection("departments")
{ "ok" : 1 }
> db.createCollection("grades")
```

Inserting data into students

```
> db.students.insertOne({
... name: "Sheeru bansal",
... student_id: "01",
... department: "Civil engineering",
... semester: 3,
... age: 20})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("663e1e9c70351776199c3ec3")
}
```

```
>
> db.students.insertMany([
...     {
...         name: "Shruti Sharma",
...         student_id: "02",
...         department: "Civil Engineering",
...         semester: 2,
...         age: 19,
...         courses: [
...             { course_id: "CSCI101"},
...             { course_id: "MATH201"},
...             { course_id: "PHYS202"}
...         ]
...     },
...
...     {
...         student_id: "04",
...         name: "Reeti Rana",
...         department: "Civil engineering",
...         semester: 1,
...         age: 22,
...         courses: [
...             { course_id: "CSCI101"},
...             { course_id: "MATH201"},
...             { course_id: "PHYS202"}
...         ]
...     },
...
...     {
...         student_id: "05",
...         name: "Prashant Sharma",
...         department: "Electrical Engineering",
...         semester: 2,
...         age: 18,
...         courses: [
...             { course_id: "EE101"},
...             { course_id: "MATH201"},
...             { course_id: "PHYS202"}
...         ]
...     },
...
```

```
...
...     {
...         student_id: "06",
...         name: "Varun Singh",
...         department: "Mechanical Engineering",
...         semester: 1,
...         age: 20,
...         courses: [
...             { course_id: "MECH101"},
...             { course_id: "MATH201"},
...             { course_id: "PHYS202"}
...         ]
...     },
...
...     {
...         student_id: "07",
...         name: "Aaman Sidhu",
...         department: "Computer Science",
...         semester: 2,
...         age: 20,
...         courses: [
...             { course_id: "CSCI101"},
...             { course_id: "MATH201"},
...             { course_id: "PHYS202"}
...         ]
...     },
...
...     {
...         student_id: "08",
...         name: "Arushi Verma",
...         department: "Computer Science",
...         semester: 1,
...         age: 19,
...         courses: [
...             { course_id: "CSCI101"},
...             { course_id: "MATH201"},
...             { course_id: "PHYS202"}
...         ]
...     },
```

```
...
...     {
...         student_id: "09",
...         name: "Ritik kumar",
...         department: "Electrical Engineering",
...         semester: 2,
...         age: 18,
...         courses: [
...             { course_id: "EE101"},
...             { course_id: "CSCI101"},
...             { course_id: "MATH201"}
...         ]
...     },
...
...     {
...         student_id: "10",
...         name: "Aryan Sehgal",
...         department: "Mechanical Engineering",
...         semester: 1,
...         age: 18,
...         courses: [
...             { course_id: "MECH101"},
...             { course_id: "MATH201"},
...             { course_id: "PHYS202"}
...         ]
...     }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("663e272670351776199c3ec4"),
                ObjectId("663e272670351776199c3ec5"),
                ObjectId("663e272670351776199c3ec6"),
                ObjectId("663e272670351776199c3ec7"),
                ObjectId("663e272670351776199c3ec8"),
                ObjectId("663e272670351776199c3ec9"),
                ObjectId("663e272670351776199c3eca"),
                ObjectId("663e272670351776199c3ecb")
        ]
}
```

```
> db.students.insertOne({
...     name: "Karam Virk",
...     student_id: "02",
...     department: "Computer Science",
...     semester: 2,
...     age: 19,
...     courses: [
...         { course_id: "CSCI101"},
...         { course_id: "MATH201"},
...         { course_id: "PHYS202"}
...     ]
... })
{
        "acknowledged" : true,
        "insertedId" : ObjectId("663e28da70351776199
}
```

Inserting Data into Departments

```
> db.departments.insertMany([
...     { dp_id: "CE",
...       name: "Civil Engineering",
...       faculty: "Engineering",
...       location: "Building B" },
...
...     { dp_id: "SC",
...       name: "Mathematics",
...       faculty: "Science",
...       location: "Building A" },
...
...     { dp_id: "CS",
...       name: "Computer Science",
...       faculty: "Engineering",
...       location: "Building A" },
...
...     { dp_id: "MATH",
...       name: "Mathematics",
...       faculty: "Science",
...       location: "Building B" },
...
...     { dp_id: "PHY",
...       name: "Physics",
...       faculty: "Science",
...       location: "Building C" },
...
...     { dp_id: "EE",
...       name: "Electrical Engineering",
...       faculty: "Engineering",
...       location: "Building D" }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("663e2ba270351776199c3ecd")
                ObjectId("663e2ba270351776199c3ece")
                ObjectId("663e2ba270351776199c3ecf")
                ObjectId("663e2ba270351776199c3ed0")
                ObjectId("663e2ba270351776199c3ed1")
                ObjectId("663e2ba270351776199c3ed2")
```

Inserting Data into courses

```
> db.courses.insertMany([
...     {
...         c_code: "CS101",
...         name: "Introduction to Computer Science",
...         instructor: "Aditya Kumar",
...         department: "Computer Science",
...         credits: 3
...     },
...     {
...         c_code: "MAT102",
...         name: "Calculus",
...         instructor: "Sandeep Goyal",
...         department: "Mathematics",
...         credits: 4
...     },
...     {
...         c_code: "CE103",
...         name: "Intro to Civil Engineering",
...         instructor: "Karminder Singh",
...         department: "Civil Engineering",
...         credits: 3
...     },
...     {
...         c_code: "PHYS104",
...         name: "Physics",
...         instructor: "Piyush Verma",
...         department: "PHY",
...         credits: 4
...     },
...     {
...         c_code: "EE105",
...         name: "Introduction to Electrical Engineering",
...         instructor: "Inderjit Kaur",
...         department: "EE",
...         credits: 3
...     }
... ])
{
        "acknowledged" : true,
```

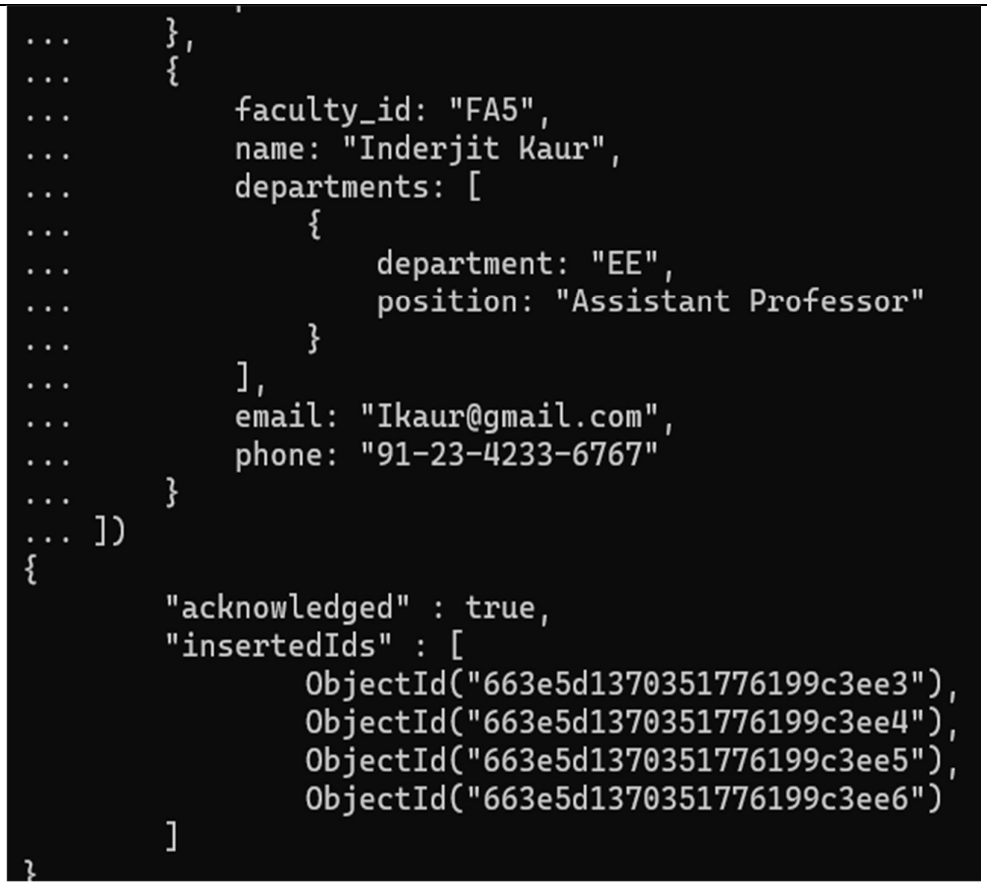
Inserting data into grades

```
> db.grades.insertMany([
...     { student_id: "01", c_code: "CE103", grade: "A" },
...     { student_id: "02", c_code: "CE103", grade: "B" },
...     { student_id: "03", c_code: "CS101", grade: "A" },
...     { student_id: "04", c_code: "CE103", grade: "C" },
...     { student_id: "05", c_code: "EE105", grade: "A+" }
...     { student_id: "06", c_code: "PHYS104", grade: "B" }
...     { student_id: "07", c_code: "CS101", grade: "C" },
...     { student_id: "08", c_code: "CS101", grade: "B+" }
...     { student_id: "09", c_code: "EE105", grade: "A+" }
...     { student_id: "10", c_code: "PHYS104", grade: "B+"
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("663e342870351776199c3ed8"),
                ObjectId("663e342870351776199c3ed9"),
                ObjectId("663e342870351776199c3eda"),
                ObjectId("663e342870351776199c3edb"),
                ObjectId("663e342870351776199c3edc"),
                ObjectId("663e342870351776199c3edd"),
                ObjectId("663e342870351776199c3ede"),
                ObjectId("663e342870351776199c3edf"),
                ObjectId("663e342870351776199c3ee0"),
                ObjectId("663e342870351776199c3ee1")
```

Inserting data into faculty

```
> db.faculty.insertOne({
...     faculty_id: "FA1",
...     name: "Aditya Kumar",
...     departments: [
...         {
...             department: "Computer Science",
...             position: "Professor"
...         }
...     ],
...     email: "Aditya.kumar@gmail.com",
...     phone: "91-22-1234-5678"
... })
{
        "acknowledged" : true,
        "insertedId" : ObjectId("663e55ed70351776199c3ee2")
```

```
> db.faculty.insertMany([
...     {
...         faculty_id: "FA2",
...         name: "Sandeep Goyal",
...         departments: [
...             {
...                 department: "Mathematics",
...                 position: "Professor"
...             }
...         ],
...         email: "Sandeep.goyal@gmail.com",
...         phone: "91-21-2457-6792"
...     },
...     {
...         faculty_id: "FA3",
...         name: "Karminder Singh",
...         departments: [
...             {
...                 department: "Civil Engineering",
...                 position: "Assistant Professor"
...             }
...         ],
...         email: "Ksingh@gmail.com",
...         phone: "91-23-4467-8369"
...     },
...     {
...         faculty_id: "FA4",
...         name: "Piyush Verma",
...         departments: [
...             {
...                 department: "PHY",
...                 position: "Professor"
...             }
...         ],
...         email: "PVerma@gmail.com",
...         phone: "91-267-4532-6767"
```


... },
... {
... faculty_id: "FA5",
... name: "Inderjit Kaur",
... departments: [
... {
... department: "EE",
... position: "Assistant Professor"
... }
...],
... email: "Ikaur@gmail.com",
... phone: "91-23-4233-6767"
... }
...])
{
"acknowledged" : true,
"insertedIds" : [
ObjectId("663e5d1370351776199c3ee3"),
ObjectId("663e5d1370351776199c3ee4"),
ObjectId("663e5d1370351776199c3ee5"),
ObjectId("663e5d1370351776199c3ee6")
]
}

MongoDB Query

1.1 Beginner

1. **Find all courses**

   db.courses.find().pretty()

2. **Find all courses with more than 3 credits**

   db.courses.find({ credits: { $gt: 3 } })

3. **Update student_id**

   db.students.updateOne(

   { name: "Karam Virk", student_id: "02" },

   { $set: { student_id: "03" } })

4. **Find all courses in a specific department**

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

5. **Find all grades for a specific course**

   db.grades.find({ c_code: "CS101" })

6. **Find all students with a specific grade**

   db.grades.find({ grade: "A" })

7. **Find all faculty members who are professors**

   db.faculty.find({ "departments.position": "Professor" })

8. **Find the oldest student**

   db.students.find().sort({ age: 1 }).limit(1)

9. **Find the average age of all students**

   db.students.aggregate([

   { $group: { _id: null, avgAge: { $avg: "$age" } } }

   ])

10. **Find the total number of courses offered**

    db.courses.count()

11. **Find all courses with instructors who are also professors**

    db.courses.find({ instructor: { $in: db.faculty.distinct("name", {

    "departments.position": "Professor" }) } })

12. **Update the age**

    db.students.updateOne({ student_id: "07" },{ $set: { age: 22 } })

13. **Find all courses offered by the Engineering faculty**

    db.courses.find({ department: { $in: db.departments.distinct("name", { faculty: "Engineering" }) } })

14. **Find courses with more than 1 credits**

    db.courses.distinct("c_code", { credits: { $gt:1} })

15. **Retrieve the grade for student ID '07' in the course 'CS101' from the grades collection**

    db.grades.findOne({ student_id: "07", c_code: "CS101" })

Note: Grades collection modification: (Add new document for grade point).

```
> db.grades.updateMany(
...     {},
...     [
...
...         {
...             $set: {
...                 grade_point: {
...                     $switch: {
...                         branches: [
...                             { case: { $eq: ["$grade", "A+"] }, then: 10 },
...                             { case: { $eq: ["$grade", "A"] }, then: 9 },
...                             { case: { $eq: ["$grade", "B+"] }, then: 8 },
...                             { case: { $eq: ["$grade", "B"] }, then: 7 },
...                             { case: { $eq: ["$grade", "C"] }, then: 6 }
...                         ]
...                     }
...                 }
...             }
...         }
...     ]
... )
{ "acknowledged" : true, "matchedCount" : 10, "modifiedCount" : 10 }
> db.grades.find().pretty()
{
        "_id" : ObjectId("663e342870351776199c3ed8"),
        "student_id" : "01",
        "c_code" : "CE103",
        "grade" : "A",
        "grade_point" : 9
```

Note: Courses collection modification (added new document (array) namely skills)

```
> db.courses.updateMany(
...     { department: "Computer Science" },
...     { $set: { skills: ["Python", "Java", "C++"] } }
... );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
> db.courses.updateMany(
...     { department: "Mathematics" },
...     { $set: { skills: ["Differential Calculus", "Integral Calculus", "Multivariable Calculus"] } }
... );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
> db.courses.updateMany(
...     { department: "Civil Engineering" },
...     { $set: { skills: ["Structural Analysis and Design", "Construction Management", "Geotechnical Engineering"] }
... );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
> db.courses.updateMany(
...     { department: "PHY" },
...     { $set: { skills: ["Quantum Mechanics", "Particle Physics", "Thermodynamics"] } }
... );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
> db.courses.updateMany(
...     { department: "EE" },
...     { $set: { skills: ["Circuit Analysis and Design", "Control Systems", "Signal Processing"] } }
... );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.courses.find().pretty()
{
        "_id" : ObjectId("663e301670351776199c3ed3"),
        "c_code" : "CS101",
        "name" : "Introduction to Computer Science",
        "instructor" : "Aditya Kumar",
        "department" : "Computer Science",
        "credits" : 3,
        "skills" : [
                "Python",
                "Java",
```

```
{
        "_id" : ObjectId("663e301670351776199c3ed4"),
        "c_code" : "MAT102",
        "name" : "Calculus",
        "instructor" : "Sandeep Goyal",
        "department" : "Mathematics",
        "credits" : 4,
        "skills" : [
                "Differential Calculus",
                "Integral Calculus",
                "Multivariable Calculus"
        ]
}
```

1.2 Medium

16. **Find faculty members with a phone number starting with "91-23"**

```
db.faculty.find({ phone: { $regex: /^91-23/ } })
```

17. **Find students who are aged between 18 and 20:**

```
db.students.find({ age: { $gte: 18, $lte: 20 } })
```

18. **Find courses with an instructor whose name starts with "A"**

```
db.courses.find({ instructor: { $regex: /^A/ } })
```

19. **Find courses with an instructor whose name does not contain "Verma" or "Kumar"**

```
db.courses.find({ instructor: { $not: /Verma|Kumar/ } })
```

20. **Find all students who are in semester 2 and not in the mathematics department**

```
db.students.find({ semester: 2, department: { $ne: "Mathematics" } })
```

21. **Find all students who are enrolled in the "Computer Science" department and have at least 'MATH201' and PHYS202 course id with them**

```
db.students.find({ department: "Computer Science", courses: {
$elemMatch: { course_id: { $in: ["MATH201", "PHYS202"] } } } })
```

22. **Find all courses taught by instructors who are not in the "Engineering" department**

```
db.courses.find({
instructor: {
$not: {
$in: db.faculty.distinct("name", { "departments.department":
"Engineering" })
}
}
})
```

23. **Find all students who are not enrolled in any courses**

```
db.students.find({ courses: { $exists: true, $eq: [] } })
```

24. Find all faculty members who are not located in "Building A" or "Building B"

```
db.faculty.find({
$nor: [
{ "departments.location": "Building A" },
{ "departments.location": "Building B" }
]
})
```

25. Find all courses with less than 3 credits or taught by instructors with names starting with "S"

```
db.courses.find({
$or: [
{ credits: { $lt: 3 } },
{ instructor: { $regex: /^S/ } }
]
})
```

26. Find all students enrolled in the "Computer Science" department and either aged less than 20 or have a semester greater than 1

```
db.students.find({
$and: [
{ department: "Computer Science" },
{ $or: [
{ age: { $lt: 20 } },
{ semester: { $gt: 1 } }
]}
]
})
```

27. Find all grades where the grade is either "A" or "B+"

```
db.grades.find({ grade: { $in: ["A", "B+"] } })
```

28. Find all grades with a grade point less than 7 or for course code "PHYS104"

```
db.grades.find({
$or: [
{ grade_point: { $lt: 7 } },
{ c_code: "PHYS104" }
]
})
```

29. Find courses offered by departments other than "Civil Engineering" and "Electrical Engineering" with names containing the word 'Engineering'

```
db.courses.find({
$and: [
{ department: { $nin: ["Civil Engineering", "Electrical Engineering"] } },
{ name: { $regex: /Engineering/i } }
]
})
```

30. Find courses offered by departments with at least 1 credits and taught by instructors with names starting with 'A'

```
db.courses.find({
$and: [
{ credits: { $gte: 1 } },
{ instructor: { $regex: /^A/i } }
]
})
```

1.3 Advance

31. Update the email domains of faculty members with IDs FA1 and FA3 to end with '.edu'

```
db.faculty.updateMany(
{ $or: [ { faculty_id: "FA1" }, { faculty_id: "FA3" } ] },

[{
$set: {
"email": {
$concat: [
{ $arrayElemAt: [{ $split: ["$email", "@"] }, 0] },
"@",
"edu"]}}}])
```

32. Find courses offered by departments in the Building B location

```
db.courses.aggregate([
{
$lookup: {
from: "departments",
localField: "department",
foreignField: "name",
as: "departmentDetails"
}
},
{
$unwind: "$departmentDetails"
},
{
$match: { "departmentDetails.location": "Building B" }
}])
```

33. Find all courses with a description field and either have 3 credits or are taught by the mathematics department:

```
db.courses.find({ $or: [{ description: { $exists: true }},{ department:
"Mathematics" }],$or: [{ credits: 3 },{ department:"Mathematics" }]})
```

34. Find all courses with credits not divisible by 2

```
db.courses.find({ credits: { $not: { $mod: [2, 0] } } })
```

35. Find the course with the highest number of enrolled students

```
db.students.aggregate([
{ $unwind: "$courses" },
{ $group: { _id: "$courses.course_id", count: { $sum: 1 } } },

{ $sort: { count: -1 } },
{ $limit: 1 }
])
```

36. Find the department with the most number of students

```
db.students.aggregate([
{ $group: { _id: "$department", count: { $sum: 1 } } },
{ $sort: { count: -1 } },
{ $limit: 1 }
])
```

37. Calculate Average Grade Point by Department

```
db.grades.aggregate([
{ $lookup: { from: "students", localField: "student_id", foreignField:
"student_id", as: "student" } },
{ $unwind: "$student" },
{ $group: { _id: "$student.department", avgGradePoint: { $avg:
"$grade_point" } } }
])
```

38. Skills provided by each course

```
db.courses.aggregate([
{
$group: {
_id: "$name",
skills: { $push: "$skills" }}}])
```

39. Find the departments where students have the highest grade points

```
db.grades.aggregate([
{ $group: {
_id: "$student_id",
maxGradePoint: { $max: "$grade_point" }
}
},
{ $lookup: {
from: "students",
localField: "_id",

foreignField: "student_id",
as: "student"
}
},
{ $unwind: "$student" },
{ $group: {
_id: "$student.department",
maxGradePoint: { $max: "$maxGradePoint" }
}
},
{ $sort: { maxGradePoint: -1 } }
])
```

40. Retrieve the list of courses taught by professors with their respective departments

```
db.faculty.aggregate([
{ $unwind: "$departments"},
{$lookup: {
from: "courses",
localField: "departments.department",
foreignField: "department",
as: "courses_taught"
} },
{$project: {
_id: 0,
professor_name: "$name",
department: "$departments.department",
courses_taught: "$courses_taught.name"}}])
```

41. Retrieve the list of students along with their departments who have achieved the highest-grade point in each department

```
db.students.aggregate([
{
$lookup: {
from: "grades",

localField: "student_id",
foreignField: "student_id",
as: "student_grades"
}
},
{
$group: {
_id: "$department",
highest_grade_point: { $max: "$student_grades.grade_point" },
students: {
$push: {
name: "$name",
student_id: "$student_id",
semester: "$semester",
age: "$age"}}}},
{
$project: {
_id: 0,
department: "$_id",
highest_grade_point: 1,
students: {
$filter: {
input: "$students",
as: "student",
```

```
cond: { $eq: ["$$student.student_grades.grade_point",
"$highest_grade_point"] }}}}}])
```

42. Retrieve the list of courses along with the total number of students enrolled in each

```
db.students.aggregate([
{
$unwind: "$courses"},
{
$group: {

_id: "$courses.course_id",
course_name: { $first: "$courses.course_name" },
total_students_enrolled: { $sum: 1 }}}])
```

43. The average age of students in each department.

```
db.students.aggregate([
{
$group: {
_id: "$department",
average_age: { $avg: "$age" }}}])
```

44. The list of faculty members along with their email addresses and the departments they belong to

```
db.faculty.aggregate([
{
$unwind: "$departments"
},
{
$project: {
_id: 0,
name: 1,
email: 1,
department: "$departments.department"}}])
```

45. The names of courses along with their corresponding departments

```
db.courses.aggregate([
{
$project: {
_id: 0,
course_name: "$name",
department: 1}}])
```

The End