**Mongo-DB PROJECT**

**Data sets (json) Used in this:**

● students

● faculty

● courses

● enrollments

● Activities

* **What is the Project about ? :**

We haveLoaded all the Datasets in MongoDB Database .

Here We have given the **Question** and we have to perform **Queries** and we will be providing the answers by doing in the Shell (Mongosh) and paste here the solution.

**Tech Stack Used:**

* MongoDB (NoSQL Database)
* MongoDB Query Language (MQL)
* Aggregation Pipeline, $lookup, $group
* Word for Report Project.
* **Learnings**
* Applied CRUD & Aggregation operations
* Used $lookup for joins between collections
* Filtered, grouped & analyzed data
* Worked with real-life datasets (students, faculty, courses)

**Submitted By: Submitted To:**

**Kanchan Kapri** Rag: No 1240258215 (69) Mr. Ankit Verma

**Submission Date:** 15 / 10 / 2025 (Faculty)

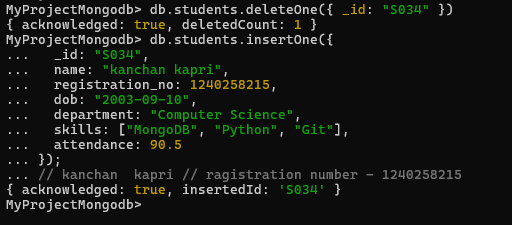
**PROJECT**

**1. Complex Filters & Projections**

**Q1.** List the names and departments of students who have more than 85% attendance and are skilled in both **"MongoDB"** and **"Python"**.

**Solution:**

**Step 1:** We do not Find Any document maybe no document was there which was matching to this question. But we created one record which has same requiornments then we had instead it to students.

* db.students.deleteOne({ \_id: "S034" })
* db.students.insertOne({

... \_id: "S034",

... name: "kanchan kapri",

... registration\_no: 1240258215,

... dob: "2003-09-10",

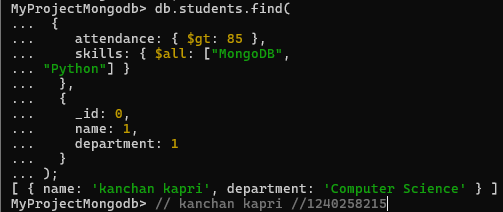
... department: "Computer Science",

... skills: ["MongoDB", "Python", "Git"],

... attendance: 90.5

... });

**Step 2:** So now it has given the Output , and resulting student is same that we have creted as my Name.

****

* db.students.find(

... {

... attendance: { $gt: 85 },

... skills: { $all: ["MongoDB",

... "Python"] }

... },

... {

... \_id: 0,

... name: 1,

... department: 1

... }

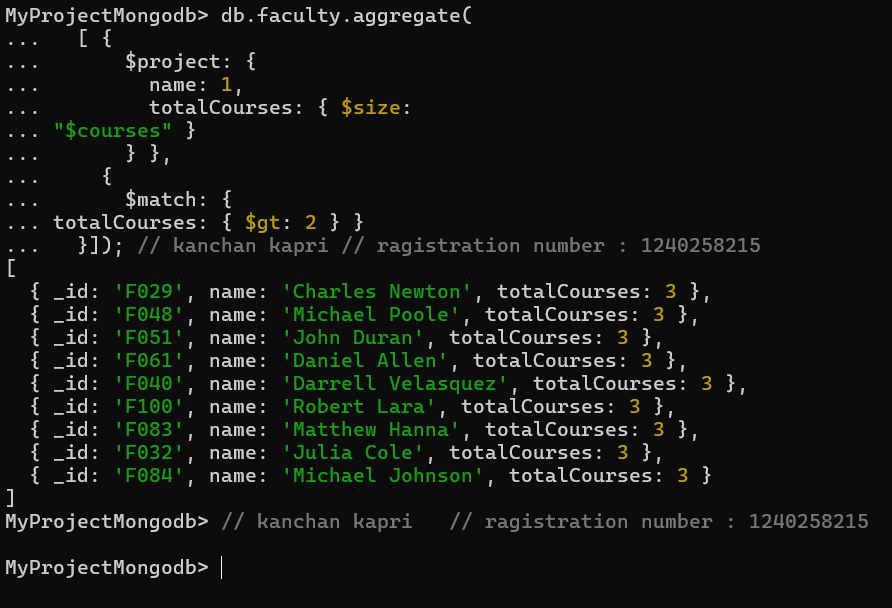
... );

[ { name: 'kanchan kapri', department: 'Computer Science' } ]

**Q2. Show all faculty who are teaching more than 2 courses. Display their names and the total number of courses they teach.**

db.faculty.aggregate( // question 2 .... //

... [ {

****... $project: {

... name: 1,

... totalCourses: { $size:

... "$courses" }

... } },

... {

... $match: {

... totalCourses: { $gt: 2 } }

... }]); db.faculty.aggregate(

... [ {

... $project: {

... name: 1,

... totalCourses: { $size:

... "$courses" }

... } },

... {

... $match: {

... totalCourses: { $gt: 2 } }

... }]);

**2. Joins ($lookup) and Aggregations**

**Q3. Show each student’s name along with the course titles they are enrolled in (use $lookup between enrollments, students, and courses).**

db.enrollments.aggregate( [

... {

... $lookup: {

... from: "students",

... localField: "student\_id",

... foreignField: "\_id",

... as: "studentInfo"

... }

... },

... {

... $lookup: {

... from: "courses",

... localField: "course\_id",

... foreignField: "\_id",

... as: "courseInfo"

... }

... },

... {

... $project: {

... \_id: 0,

... studentName: { $arrayElemAt: ["$studentInfo.name", 0] },

... courseTitles: "$courseInfo.title"

... }

... }

... ]

... );

**Q4. For each course, display the course title, number of students enrolled, and average marks (use $group).**

db.enrollments.aggregate( [

... {

... $group: {

... \_id: "$course\_id",

... totalStudents: { $sum: 1 },

... averageMarks: { $avg: "$marks" }

... }

... },

... {

... $lookup: {

... from: "courses",

... localField: "\_id",

... foreignField: "\_id",

... as: "courseInfo"

... }

... },

... {

... $project: {

... \_id: 0,

... courseTitle: { $arrayElemAt: ["$courseInfo.title", 0] },

... totalStudents: 1,

... averageMarks: 1

... }

... }

... ]

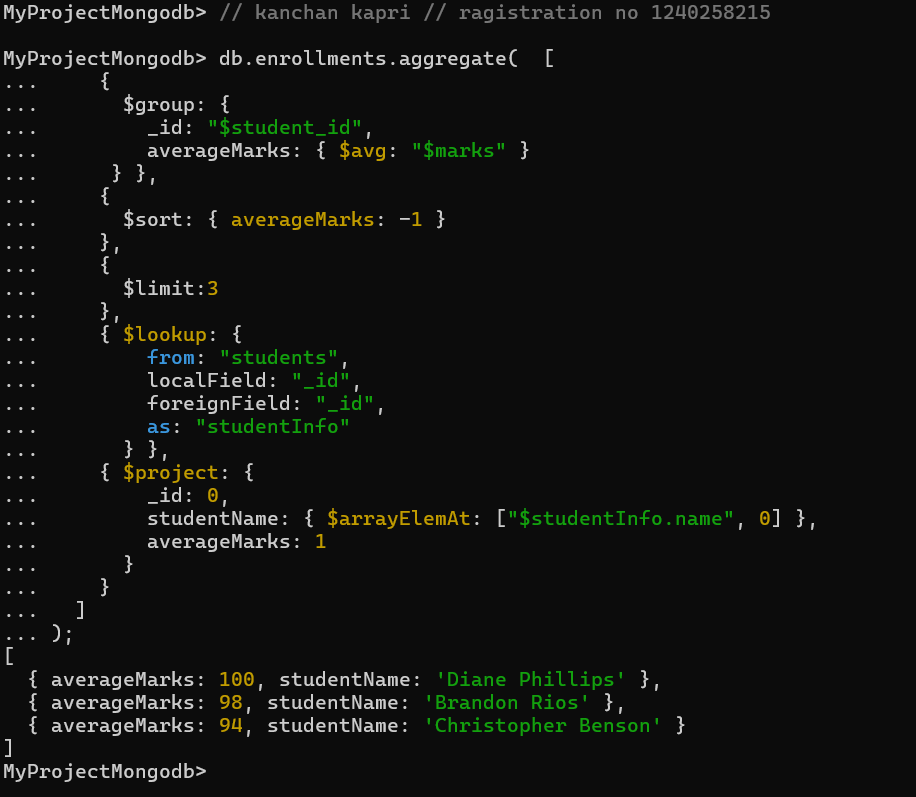
... );

**3. Grouping, Sorting, and Limiting**

**Q5. Find the top 3 students with the highest average marks across all enrolled courses.**

db.enrollments.aggregate( [

... {

... $group: {

... \_id: "$student\_id",

... averageMarks: { $avg: "$marks" }

... } },

... {

... $sort: { averageMarks: -1 }

... },

... {

... $limit:3

... },

... { $lookup: {

... from: "students",

... localField: "\_id",

... foreignField: "\_id",

... as: "studentInfo"

... } },

... { $project: {

... \_id: 0,

... studentName: { $arrayElemAt: ["$studentInfo.name", 0] },

... averageMarks: 1

... }

... }

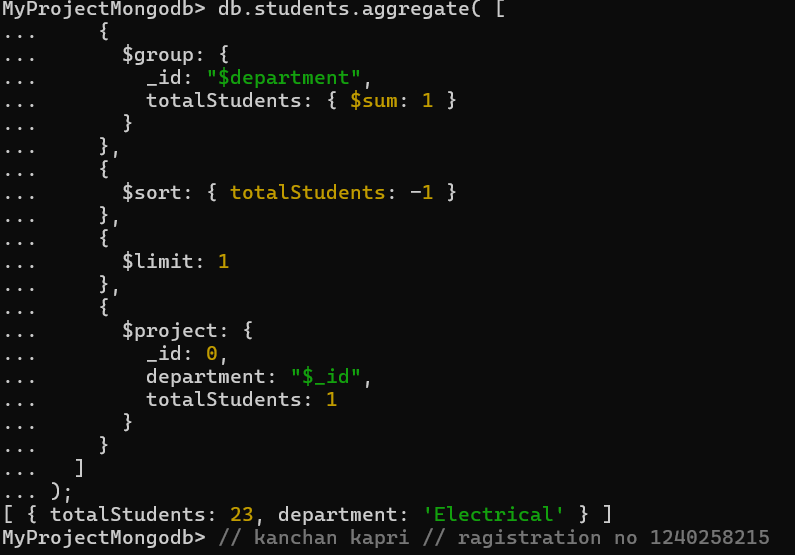
... ]

... );

**Q6. Count how many students are in each department. Display the department with the highest number of students.**

db.students.aggregate( [

... {

... $group: {

... \_id: "$department",

... totalStudents: { $sum: 1 }

... }

... },

... {

... $sort: { totalStudents: -1 }

... },

... {

... $limit: 1

... },

... {

... $project: {

... \_id: 0,

... department: "$\_id",

... totalStudents: 1

... }

... }

... ]

... );

**4. Update, Upsert, and Delete**

**Q7. Update attendance to 100% for all students who won any "Hackathon".**

db.students.updateMany(

... {

... activities: "Hackathon"

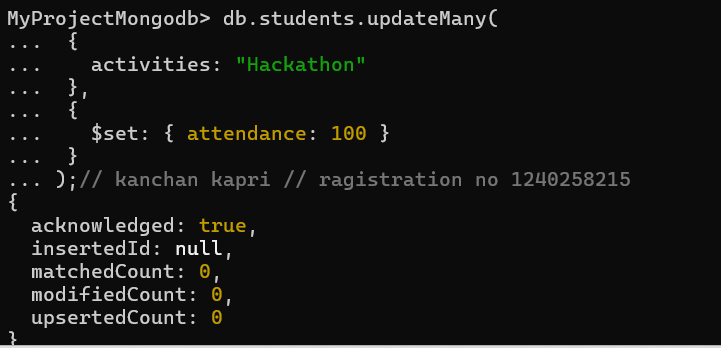
... },

... {

... $set: { attendance: 100 }

... }

... );



**Q8. Delete all student activity records where the activity year is before 2022.**

db.activities.deleteMany(

... {

... year: { $lt: 2022 }

... }

... );



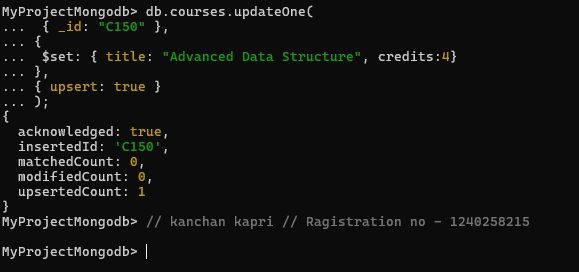
**Q9. Upsert a course record for "Data Structures" with ID "C150" and credits 4—if it doesn’t exist, insert it; otherwise update its title to "Advanced Data Structures".**

db.courses.updateOne(

... { \_id: "C150" },

... { $set: { title: "Advanced Data Structures", credits: 4 } },

... { upsert: true }

... )

...

{

acknowledged: true,

insertedId: null,

matchedCount: 1,

modifiedCount: 1,

upsertedCount: 0

}

**5. Array & Operator Usage**

**Q10. Find all students who have "Python" as a skill but not "C++".**



db.students.find(

... {

... $and: [

... { skills: "Python" },

... { skills: { $ne: "C++" } }

... ]

... },

... {

... \_id: 0,

... name: 1,

... skills: 1

... }

... );

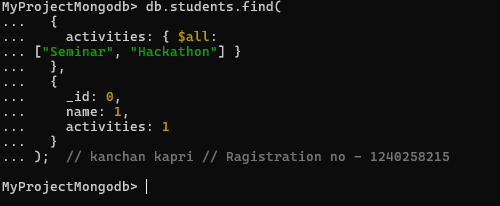
**Q11. Return names of students who participated in "Seminar" and "Hackathon" both.**

db.students.find(

... {

... activities: { $all:

... ["Seminar", "Hackathon"] }

... },

... {

... \_id: 0,

... name: 1,

... activities: 1

... }

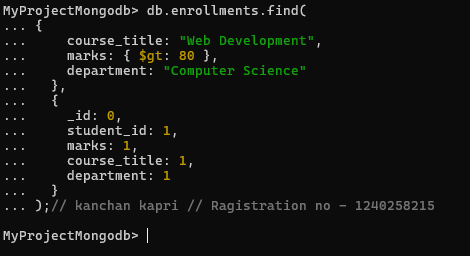
... );

**6. Subdocuments and Nested Conditions**

**Q12. Find students who scored more than 80 in "Web Development" only if they belong to the "Computer Science" department.**

db.enrollments.find(

... {

... course\_title: "Web Development",

... marks: { $gt: 80 },

... department: "Computer Science"

... },

... {

... \_id: 0,

... student\_id: 1,

... marks: 1,

... course\_title: 1,

... department: 1

... }

... );

**7. Advanced Aggregation (Challenge Level)**

**Q13. For each faculty member, list the names of all students enrolled in their courses along with average marks per student per faculty.**

db.faculty.aggregate( [

... {

... $lookup: {

... from: "courses",

... localField: "courses",

... foreignField: "\_id",

... as: "courseInfo"

... }

... },

... { $unwind: "$courseInfo" },

... {

... $lookup: {

... from: "enrollments",

... localField: "courseInfo.\_id",

... foreignField: "course\_id",

... as: "enrolledStudents"

... }

... },

... { $unwind: "$enrolledStudents" },

... {

... $lookup: {

... from: "students",

... localField: "enrolledStudents.student\_id",

... foreignField: "\_id",

... as: "studentInfo"

... }

... },

... {

... $project: {

... \_id: 0,

... facultyName: "$name",

... studentName: { $arrayElemAt:

... ["$studentInfo.name",0] },

... marks: "$enrolledStudents.marks"

... }

... },

... {

... $group: {

... \_id: { facultyName: "$facultyName", studentName:

... "$studentName" },

... averageMarks: { $avg: "$marks" }

... }

... },

... {

... $project: {

... \_id: 0,

... facultyName: "$\_id.facultyName",

... studentName: "$\_id.studentName",

... averageMarks: 1

... }

... },

... {

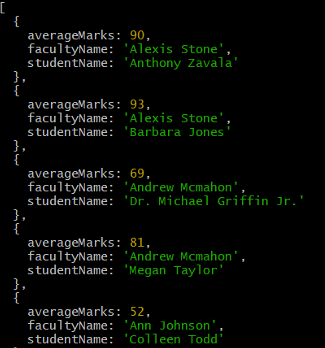
... $sort: { facultyName: 1, studentName: 1 }

... }

... ]

... );

**Output :**

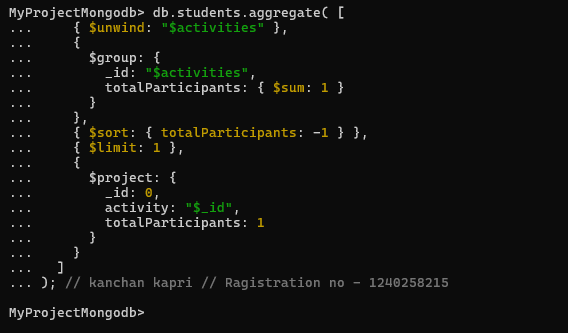


**Q14. Show the most popular activity type (e.g., Hackathon, Seminar, etc.) by number of student participants.**

db.students.aggregate( [

... { $unwind: "$activities" },

... {

... $group: {

... \_id: "$activities",

... totalParticipants: { $sum: 1 }

... }

... },

... { $sort: { totalParticipants: -1 } },

... { $limit: 1 },

... {

... $project: {

... \_id: 0,

... activity: "$\_id",

... totalParticipants: 1

... }

... }

... ]

... );

--------------------------------------------------------End of Project----------------------------------------------------