BABU BANARASI DAS UNIVERSITY LUCKNOW

SESSION : 2025-2026

SCHOOL OF COMPUTER APPLICATION

NOSQL and DBaas 101(NOSQL)

(BCADSN13202)

SUBMITTED BY:-

NAME:-

ADITYA SRIAVASTAVA

CLASS:- BCADS21

ROLL NO:-1240258040

SUBMITTED TO:-

MR.ANKIT VERMA

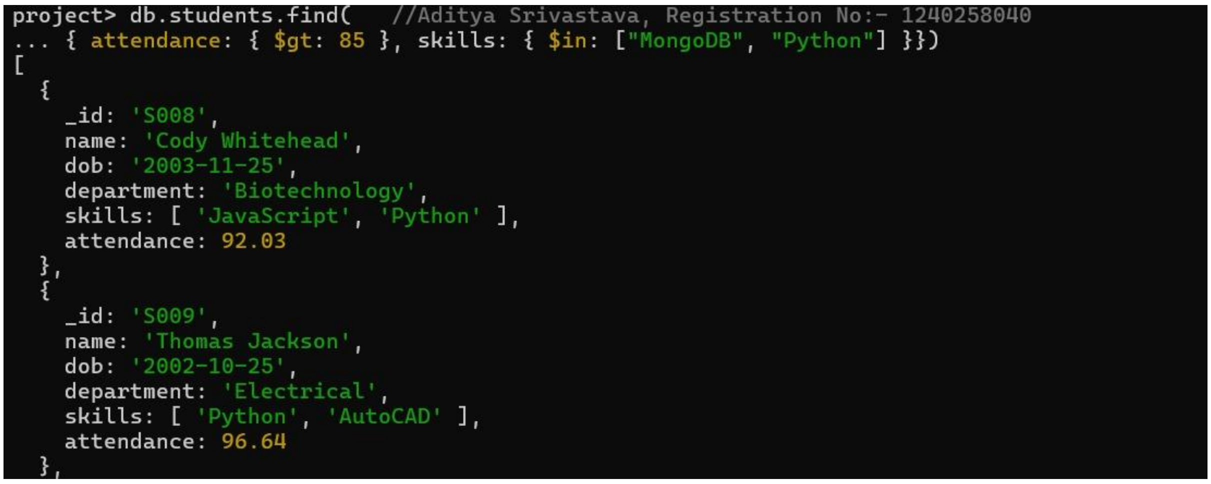
PROJECT

1. Complex Filters & Projections

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

Query: - db.students_full.find({ attendance: { $gt: 85 }, skills: { $in: ["MongoDB", "Python"] }})

Output: -



```
project> db.students.find(   //Aditya Srivastava, Registration No:- 1240258040
... { attendance: { $gt: 85 }, skills: { $in: ["MongoDB", "Python"] }})
[
  {
    _id: 'S008',
    name: 'Cody Whitehead',
    dob: '2003-11-25',
    department: 'Biotechnology',
    skills: [ 'JavaScript', 'Python' ],
    attendance: 92.03
  },
  {
    _id: 'S009',
    name: 'Thomas Jackson',
    dob: '2002-10-25',
    department: 'Electrical',
    skills: [ 'Python', 'AutoCAD' ],
    attendance: 96.64
  },
```

▪ Nothing will show up because there aren't any students who have both 'MongoDB' and 'Python' skills and more than 85% attendance.

• Use comparison operators like $gt (greater than).

• Apply array matching with $all to ensure multiple elements exist.

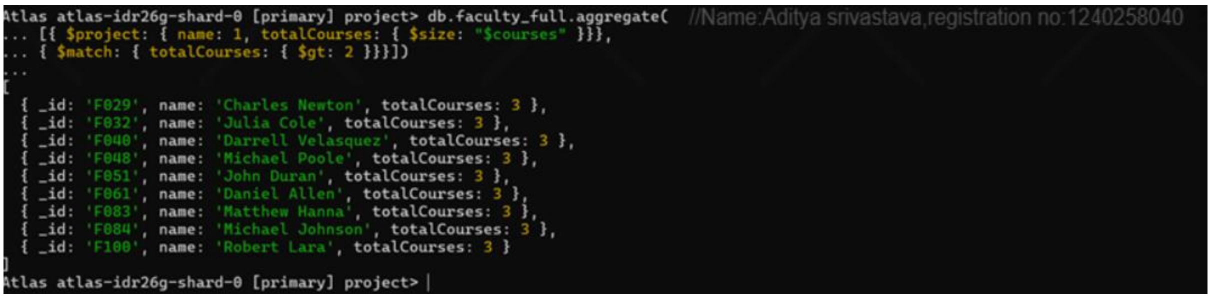
• Use projection to show only required fields.

• Build compound filters using multiple conditions.

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

Query: - db.faculty_full.aggregate([{ $project: { name: 1, totalCourses: { $size: "$courses" }}}, { $match: { totalCourses: { $gt: 2 }}}])

Output: -



```
Atlas atlas-idr26g-shard-0 [primary] project> db.faculty_full.aggregate(   //Name:Aditya srivastava,registration no:1240258040
... [{ $project: { name: 1, totalCourses: { $size: "$courses" }}},
... { $match: { totalCourses: { $gt: 2 }}}])
...
[
  { _id: 'F029', name: 'Charles Newton', totalCourses: 3 },
  { _id: 'F032', name: 'Julia Cole', totalCourses: 3 },
  { _id: 'F040', name: 'Darrell Velasquez', totalCourses: 3 },
  { _id: 'F048', name: 'Michael Poole', totalCourses: 3 },
  { _id: 'F051', name: 'John Duran', totalCourses: 3 },
  { _id: 'F061', name: 'Daniel Allen', totalCourses: 3 },
  { _id: 'F083', name: 'Matthew Hanna', totalCourses: 3 },
  { _id: 'F084', name: 'Michael Johnson', totalCourses: 3 },
  { _id: 'F100', name: 'Robert Lara', totalCourses: 3 }
]
Atlas atlas-idr26g-shard-0 [primary] project> |
```

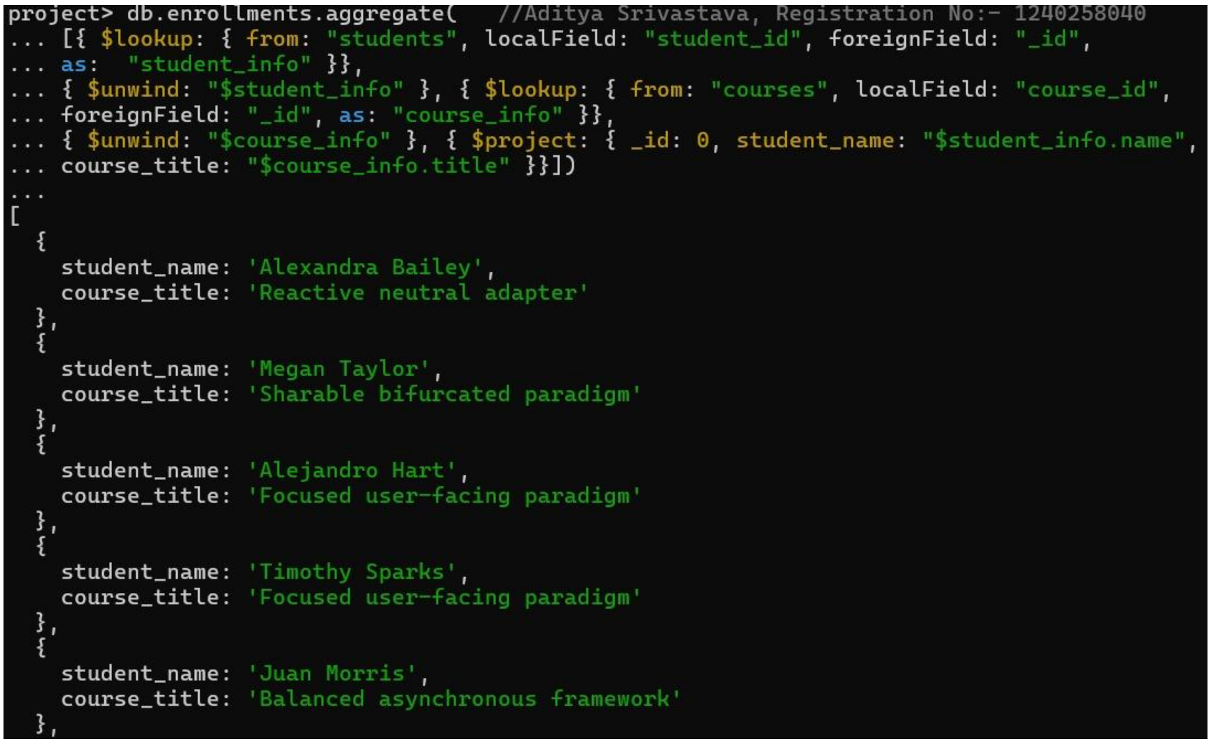
- Use $project to create computed fields.
- Use $size to count array elements.
- Combine $match after projection for conditional filtering.
- Understand aggregation pipelines.

2. Joins ($lookup) and Aggregations

Q3: - Write a query to show each student's name along with the course titles they are enrolled in (use $lookup between enrollments, students, and courses).

Query: - db.enrollments_full.aggregate([{ $lookup: { from: "students_full", localField: "student_id", foreignField: "_id", as: "student_info" }}, { $unwind: "$student_info" }, { $lookup: { from: "courses_full", localField: "course_id", foreignField: "_id", as: "course_info" }}, { $unwind: "$course_info" }, { $project: { _id: 0, student_name: "$student_info.name", course_title: "$course_info.title" }}])

Output:-



```
project> db.enrollments.aggregate(   //Aditya Srivastava, Registration No:- 1240258040
... [{ $lookup: { from: "students", localField: "student_id", foreignField: "_id",
... as:  "student_info" }},
... { $unwind: "$student_info" }, { $lookup: { from: "courses", localField: "course_id",
... foreignField: "_id", as: "course_info" }},
... { $unwind: "$course_info" }, { $project: { _id: 0, student_name: "$student_info.name",
... course_title: "$course_info.title" }}])
...
[
  {
    student_name: 'Alexandra Bailey',
    course_title: 'Reactive neutral adapter'
  },
  {
    student_name: 'Megan Taylor',
    course_title: 'Sharable bifurcated paradigm'
  },
  {
    student_name: 'Alejandro Hart',
    course_title: 'Focused user-facing paradigm'
  },
  {
    student_name: 'Timothy Sparks',
    course_title: 'Focused user-facing paradigm'
  },
  {
    student_name: 'Juan Morris',
    course_title: 'Balanced asynchronous framework'
  },
```

• Use $lookup for joins between collections.

• Combine multiple $lookups for complex relationships.

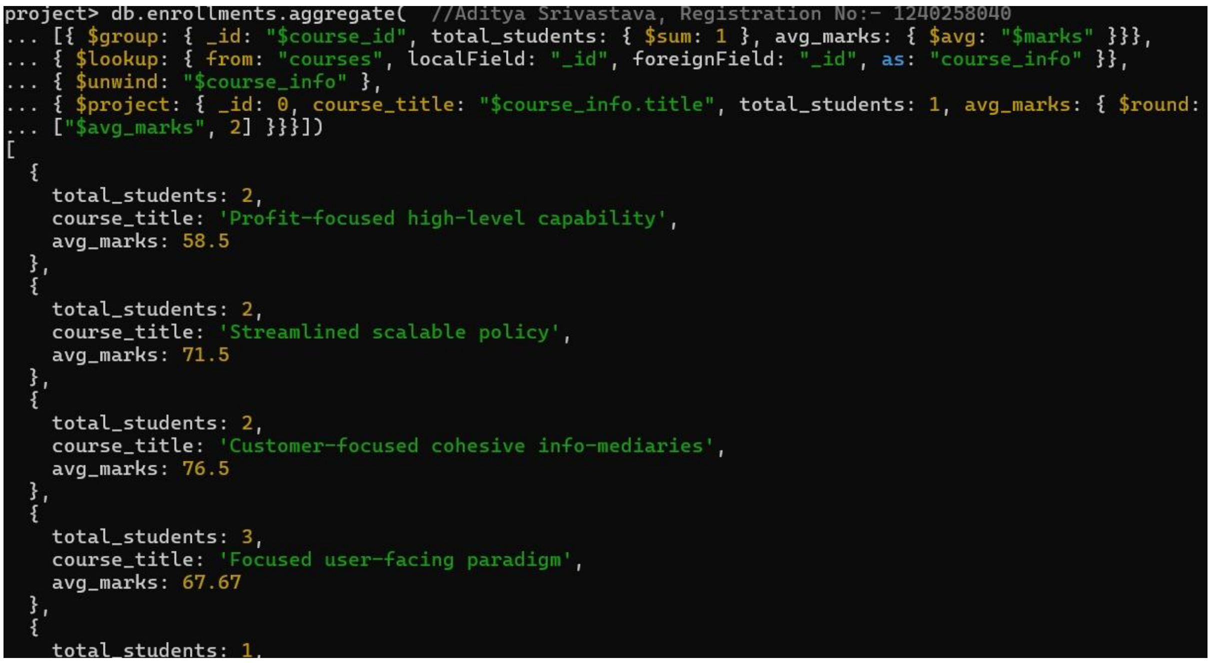
• Use $arrayElemAt to extract single values from arrays.

• Understand MongoDB’s relational-like linking.

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

Query: - db.enrollments_full.aggregate([{ $group: { _id: "$course_id", total_students: { $sum: 1 }, avg_marks: { $avg: "$marks" }}}, { $lookup: { from: "courses_full", localField: "_id", foreignField: "_id", as: "course_info" }}, { $unwind: "$course_info" }, { $project: { _id: 0, course_title: "$course_info.title", total_students: 1, avg_marks: { $round: ["$avg_marks", 2] }}}])

Output:-



```
project> db.enrollments.aggregate(  //Aditya Srivastava, Registration No:- 1240258040
... [{ $group: { _id: "$course_id", total_students: { $sum: 1 }, avg_marks: { $avg: "$marks" }}},
... { $lookup: { from: "courses", localField: "_id", foreignField: "_id", as: "course_info" }},
... { $unwind: "$course_info" },
... { $project: { _id: 0, course_title: "$course_info.title", total_students: 1, avg_marks: { $round:
... ["$avg_marks", 2] }}}])
[
  {
    total_students: 2,
    course_title: 'Profit-focused high-level capability',
    avg_marks: 58.5
  },
  {
    total_students: 2,
    course_title: 'Streamlined scalable policy',
    avg_marks: 71.5
  },
  {
    total_students: 2,
    course_title: 'Customer-focused cohesive info-mediaries',
    avg_marks: 76.5
  },
  {
    total_students: 3,
    course_title: 'Focused user-facing paradigm',
    avg_marks: 67.67
  },
  {
    total_students: 1,
```

• Use $group for summarizing data.

• Use $avg and $sum to calculate aggregates.

• $unwind helps to deconstruct arrays.

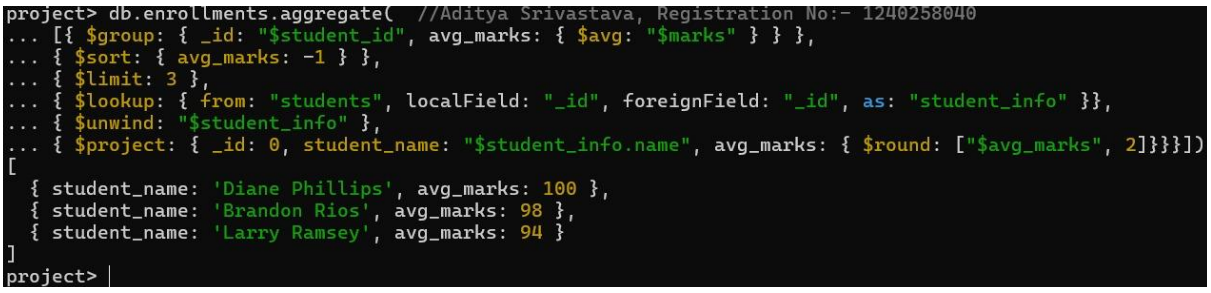
• $project to rename and structure output.

3. Grouping, Sorting, and Limiting

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

Query: - db.enrollments_full.aggregate([{ $group: { _id: "$student_id", avg_marks: { $avg: "$marks" } } }, { $sort: { avg_marks: -1 } }, { $limit: 3 }, { $lookup: { from: "students_full", localField: "_id", foreignField: "_id", as: "student_info" }}, { $unwind: "$student_info" }, { $project: { _id: 0, student_name: "$student_info.name", avg_marks: { $round: ["$avg_marks", 2] }}}])

Output:-



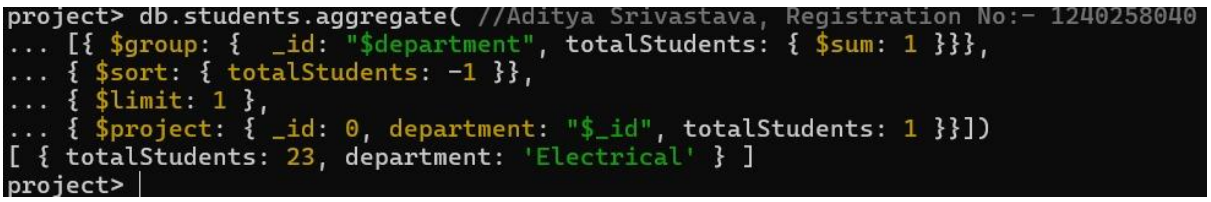
```
project> db.enrollments.aggregate(  //Aditya Srivastava, Registration No:- 1240258040
... [{ $group: { _id: "$student_id", avg_marks: { $avg: "$marks" } } },
... { $sort: { avg_marks: -1 } },
... { $limit: 3 },
... { $lookup: { from: "students", localField: "_id", foreignField: "_id", as: "student_info" }},
... { $unwind: "$student_info" },
... { $project: { _id: 0, student_name: "$student_info.name", avg_marks: { $round: ["$avg_marks", 2]}}}])
[
  { student_name: 'Diane Phillips', avg_marks: 100 },
  { student_name: 'Brandon Rios', avg_marks: 98 },
  { student_name: 'Larry Ramsey', avg_marks: 94 }
]
project>
```

- $sort sorts data in ascending/descending order.
- $limit restricts results to top records.
- $group for calculating averages.
- Combining joins with grouping.

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

Query: - db.students_full.aggregate([{ $group: { _id: "$department", totalStudents: { $sum: 1 }}}, { $sort: { totalStudents: -1 }}, { $limit: 1 }, { $project: { _id: 0, department: "$_id", totalStudents: 1 }}])

Output:-



```
project> db.students.aggregate( //Aditya Srivastava, Registration No:- 1240258040
... [{ $group: {  _id: "$department", totalStudents: { $sum: 1 }}},
... { $sort: { totalStudents: -1 }},
... { $limit: 1 },
... { $project: { _id: 0, department: "$_id", totalStudents: 1 }}])
[ { totalStudents: 23, department: 'Electrical' } ]
project> |
```

• Count items per category with $sum: 1

. • Use $sort to rank results.

• Identify top-performing or most populated groups.

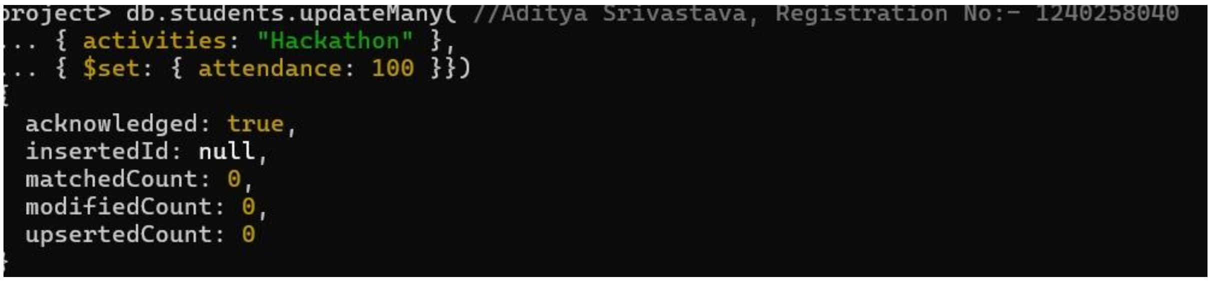
• Apply $limit to get top results.

4. Update, Upsert, and Delete

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

Query: - db.students_full.updateMany({ activities: "Hackathon" }, { $set: { attendance: 100 }})

Output:-



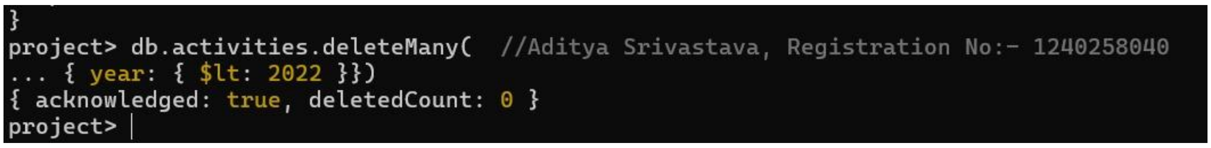
```
project> db.students.updateMany( //Aditya Srivastava, Registration No:- 1240258040
... { activities: "Hackathon" },
... { $set: { attendance: 100 }})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

- Use updateMany() for bulk updates.
- $set modifies specific fields.
- Target documents via nested fields.
- Understand bulk updates with filters.

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

Query: - db.activities_full.deleteMany({ year: { $lt: 2022 }})

Output:-



```
}
project> db.activities.deleteMany(  //Aditya Srivastava, Registration No:- 1240258040
... { year: { $lt: 2022 }})
{ acknowledged: true, deletedCount: 0 }
project> |
```

• Delete records conditionally using deleteMany().

• $lt filters by less than a value.

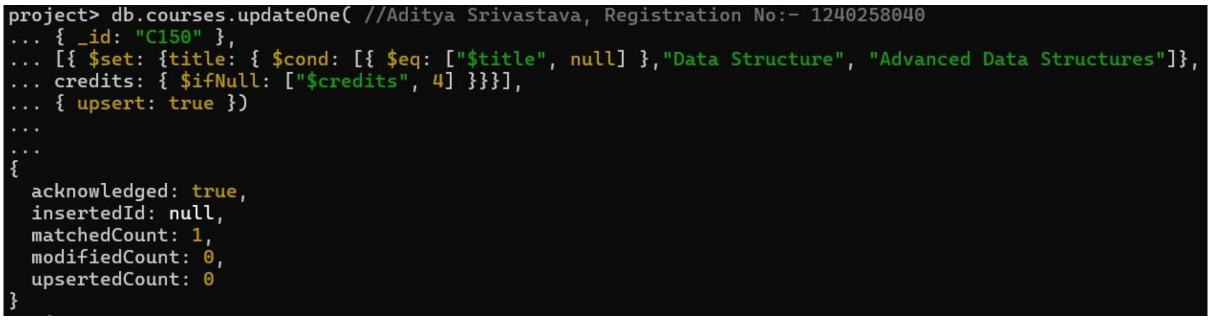
• Manage dataset cleanup.

• Apply conditional data management.

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".

Query: - db.courses_full.updateOne({ _id: "C150" }, [{ $set: { title: { $cond: [{ $eq: ["$title", null] }, "Data Structure", "Advanced Data Structures"]}, credits: { $ifNull: ["$credits": 4]}}}], { upsert: true })

Output:-



```
project> db.courses.updateOne( //Aditya Srivastava, Registration No:- 1240258040
... { _id: "C150" },
... [{ $set: {title: { $cond: [{ $eq: ["$title", null] },"Data Structure", "Advanced Data Structures"]},
... credits: { $ifNull: ["$credits", 4] }}}],
... { upsert: true })
...
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
```

• upsert: true inserts if no match is found.

• $setOnInsert applies only when inserting new data.

• $set updates fields if record exists

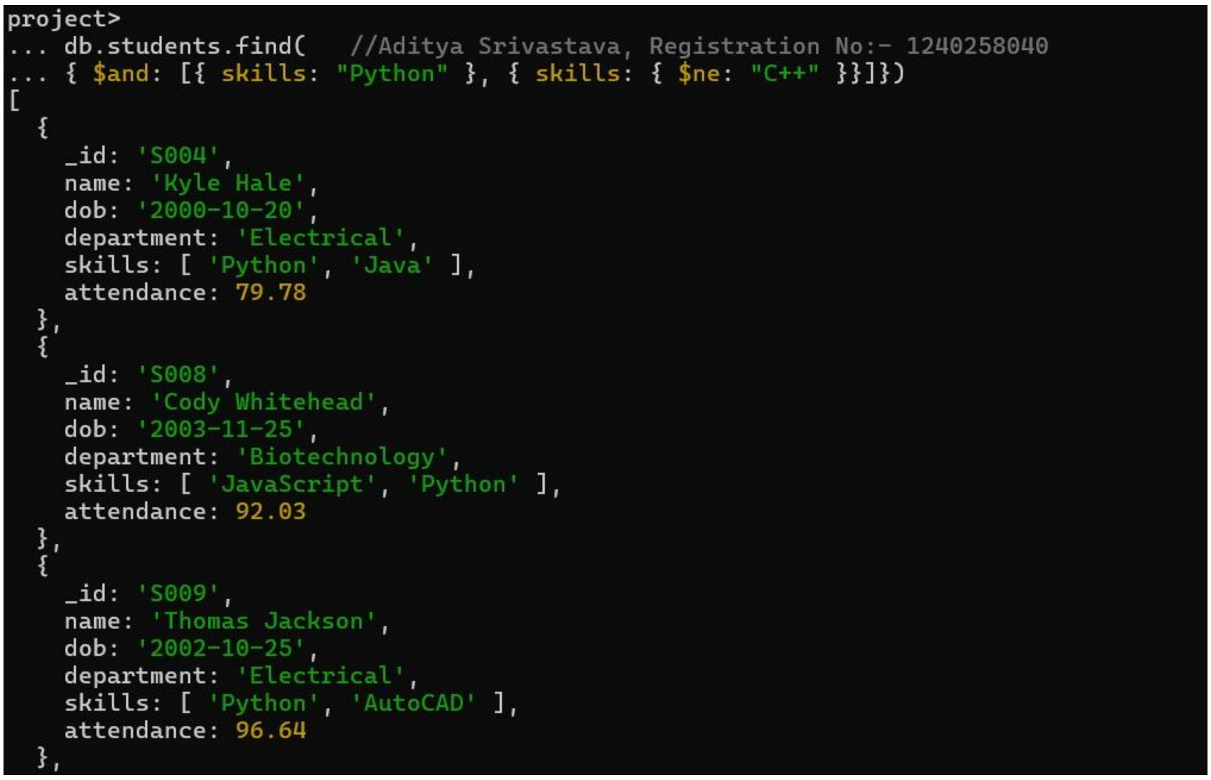
. • Handle both insert and update in one command.

5. Array & Operator Usage

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

Query: - db.students_full.find({ $and: [{ skills: "Python" }, { skills: { $ne: "C++" }}]})

Output:-



```
project>
... db.students.find(   //Aditya Srivastava, Registration No:- 1240258040
... { $and: [{ skills: "Python" }, { skills: { $ne: "C++" }}]})
[
  {
    _id: 'S004',
    name: 'Kyle Hale',
    dob: '2000-10-20',
    department: 'Electrical',
    skills: [ 'Python', 'Java' ],
    attendance: 79.78
  },
  {
    _id: 'S008',
    name: 'Cody Whitehead',
    dob: '2003-11-25',
    department: 'Biotechnology',
    skills: [ 'JavaScript', 'Python' ],
    attendance: 92.03
  },
  {
    _id: 'S009',
    name: 'Thomas Jackson',
    dob: '2002-10-25',
    department: 'Electrical',
    skills: [ 'Python', 'AutoCAD' ],
    attendance: 96.64
  },
```

- $in checks for presence in arrays.
- $nin checks for absence in arrays.

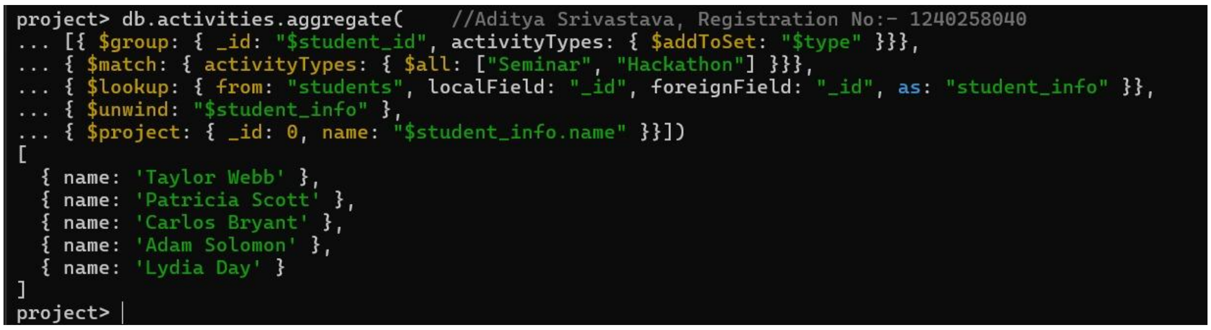
• Combine both for exclusive conditions.

• Operate effectively on array fields.

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

Query: - db.activities_full.aggregate([{ $group: { _id: "$student_id", activityTypes: { $addToSet: "$type" }}}, { $match: { activityTypes: { $all: ["Seminar", "Hackathon"] }}}, { $lookup: { from: "students_full", localField: "_id", foreignField: "_id", as: "student_info" }}, { $unwind: "$student_info" }, { $project: { _id: 0, name: "$student_info.name" }}])

Output:-



```
project> db.activities.aggregate(    //Aditya Srivastava, Registration No:- 1240258040
... [{ $group: { _id: "$student_id", activityTypes: { $addToSet: "$type" }}},
... { $match: { activityTypes: { $all: ["Seminar", "Hackathon"] }}},
... { $lookup: { from: "students", localField: "_id", foreignField: "_id", as: "student_info" }},
... { $unwind: "$student_info" },
... { $project: { _id: 0, name: "$student_info.name" }}])
[
  { name: 'Taylor Webb' },
  { name: 'Patricia Scott' },
  { name: 'Carlos Bryant' },
  { name: 'Adam Solomon' },
  { name: 'Lydia Day' }
]
project> 
```

• $all ensures all specified elements exist in an array

. • Simple array querying in MongoDB.

• Combine multiple filters in a single query.

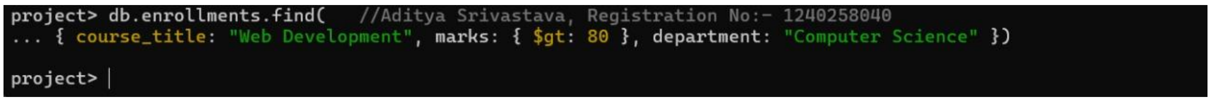
• Efficient participation tracking.

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.

Query: - db.enrollments_full.find({ course_title: "Web Development", marks: { $gt: 80 }, department: "Computer Science" })

Output:-



```
project> db.enrollments.find(    //Aditya Srivastava, Registration No:- 1240258040
... { course_title: "Web Development", marks: { $gt: 80 }, department: "Computer Science" })

project> |
```

▪ Nothing will show up because there are no students in the Computer Science department who scored more than 80 in 'Web Development'.

• Access nested fields using dot notation

. • Combine multiple field conditions.

• Query subdocuments efficiently.

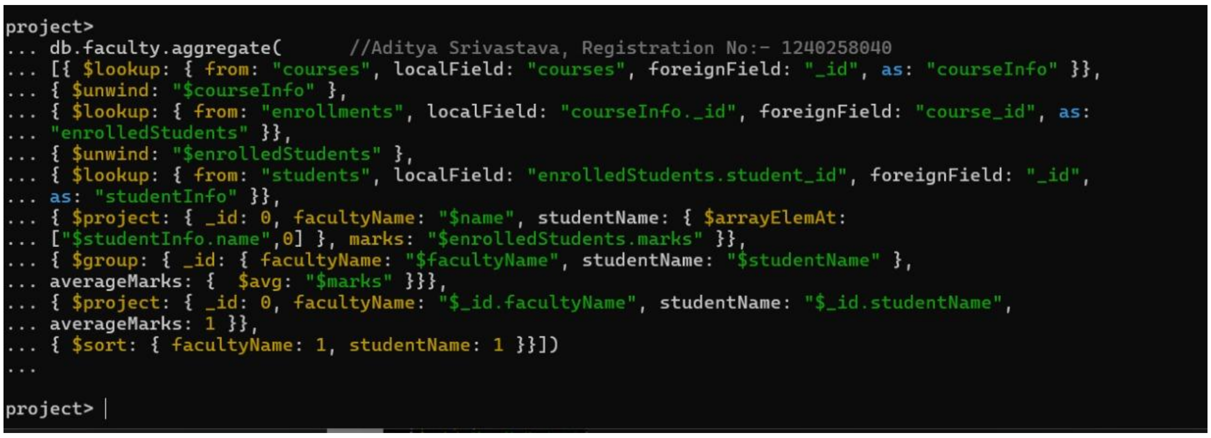
• Focused filtering by department and performance.

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.

Query: - db.faculty_full.aggregate([{ $lookup: { from: "courses_full", localField: "courses", foreignField: "_id", as: "courseInfo" }}, { $unwind: "$courseInfo" }, { $lookup: { from: "enrollments_full", localField: "courseInfo._id", foreignField: "course_id", as: "enrolledStudents" }}, { $unwind: "$enrolledStudents" }, { $lookup: { from: "students_full", 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: -



```
project>
... db.faculty.aggregate(      //Aditya Srivastava, Registration No:- 1240258040
... [{ $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 }}])
...

project> |
```

• Multi-level joins using $lookup.

• $addToSet to avoid duplicate student names.

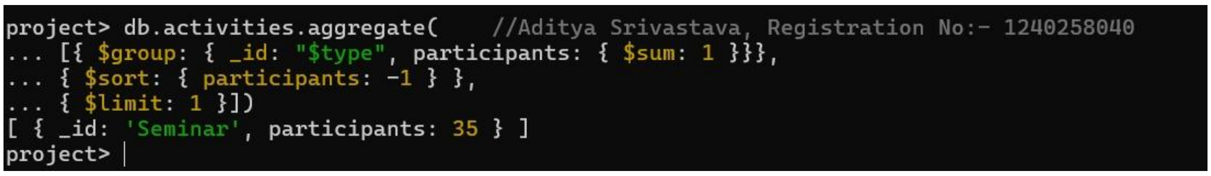
• $avg to compute average marks per faculty.

• Real-world aggregation chaining

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

Query: - db.activities_full.aggregate([{ $group: { _id: "$type", participants: { $sum: 1 }}}, { $sort: { participants: -1 } }, { $limit: 1 }])

Output: -



```
project> db.activities.aggregate(    //Aditya Srivastava, Registration No:- 1240258040
... [{ $group: { _id: "$type", participants: { $sum: 1 }}},
... { $sort: { participants: -1 } },
... { $limit: 1 }])
[ { _id: 'Seminar', participants: 35 } ]
project> 
```

• $unwind to count array elements.

• $group and $sum for totals.

• $sort to rank results.

• Identify “most popular” entities..