* **MongoDB Assignment**

1. **Complex Filters & Projections**

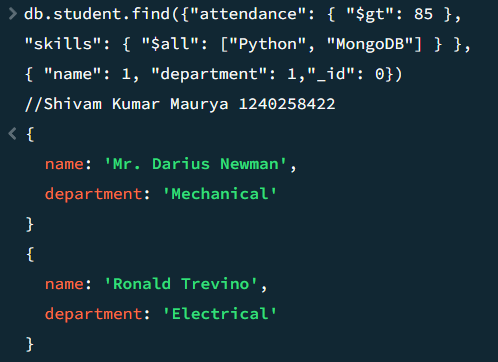
Q1. List the names and departments of students who have more than 85% attendance and are skilled in both "MongoDB" and "Python".

***Query*** *-- db.student.find({"attendance": { "$gt": 85 },*

*"skills": { "$all": ["Python", "MongoDB"] } },*

*{ "name": 1, "department": 1,"\_id": 0})*

**Output *--***

**

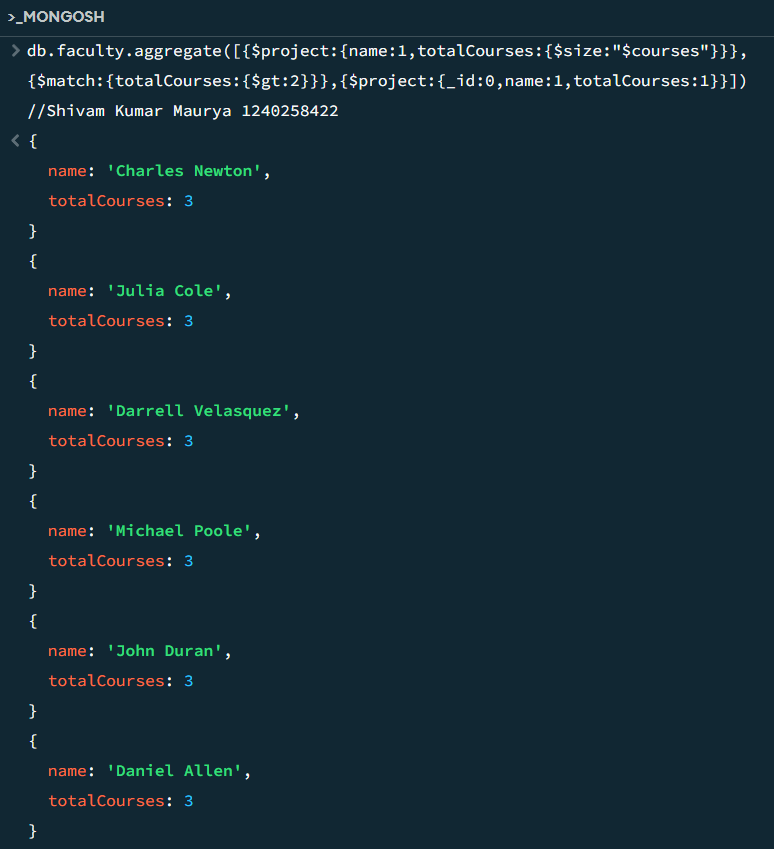
***Note—Added some new data.***

**Explanation** *–* This query will return only names and departments of students who have more than 85% attendance& theyare skilled in both MongoDB and Python.

*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.aggregate([{$project: {name: 1, totalCourses: {$size: "$courses"}}}, {$match: {totalCourses: {$gt: 2}}}, {$project: {\_id: 0, name: 1, totalCourses: 1}}])*

**Output--**

****

**Explanation**-- This query will show all faculty who are teaching more than 2 courses and will return only their names & total number of courses.

1. **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.enrollment.aggregate([{ $lookup: { from: "student", localField: "student\_id", foreignField: "\_id", as: "student\_info" }},*

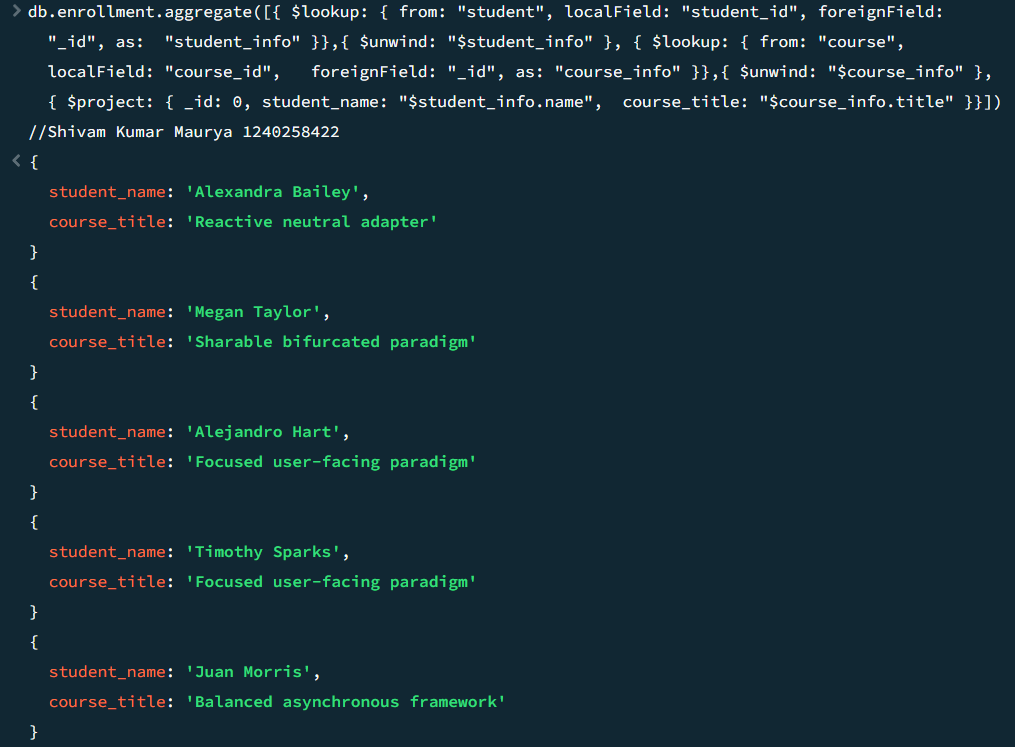
*{ $unwind: "$student\_info" }, { $lookup: { from: "course", localField: "course\_id",*

*foreignField: "\_id", as: "course\_info" }},*

*{ $unwind: "$course\_info" }, { $project: { \_id: 0, student\_name: "$student\_info.name",*

*course\_title: "$course\_info.title" }}])*

**Output—**

****

**Explantion** – This query will show each student’s name along with the course title they are enrolled in with using $lookup between enrollment, student, and course.

Q4. For each course, display the course title, number of students enrolled, and average marks

(use $group).

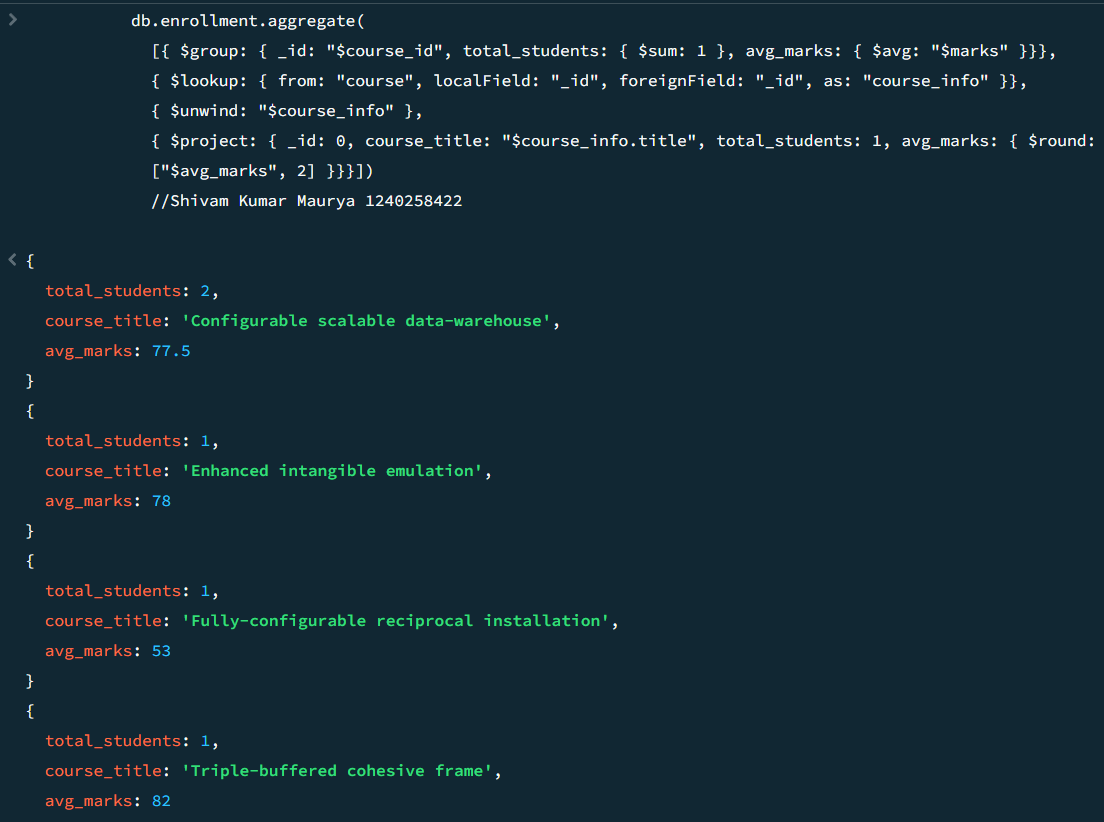
**Query**-- *db.enrollment.aggregate( [{ $group: { \_id: "$course\_id", total\_students: { $sum: 1 }, avg\_marks: { $avg: "$marks" }}},*

*{ $lookup: { from: "course", 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**—



**Explanation**—This questions query will show the course title, number of students enrolled and average marks for each course given in the data.

1. **Grouping, Sorting, and Limiting**

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

**Query –** *db.enrollment.aggregate([{$group: {\_id: "$student\_id",*

*averageMarks: {$avg: "$marks"}}}, {$sort:*

*{averageMarks: -1}}, {$limit: 3}])*

**Output--**

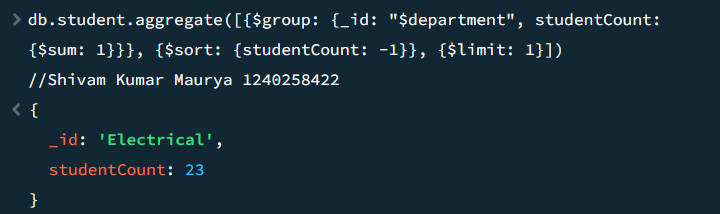
****

**Explanation --** This Query will return highest average marks of top 3 students in the given collection across all enrolled courses.

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

**Query--** *db.student.aggregate([{$group: {\_id: "$department", studentCount: {$sum: 1}}}, {$sort: {studentCount: -1}}, {$limit: 1}])*

**Output--**



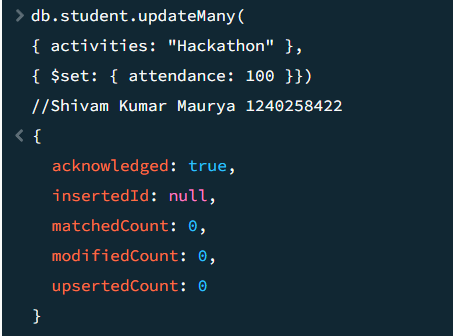
**Explanation**—Firstly this query count how many students are in each department, one by one and then display that department which is highest number of students.

1. **Update, Upsert, and Delete**

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

**Query—** *db.student.updateMany( { activities: "Hackathon" }, { $set: { attendance: 100 }})*

**Output--**

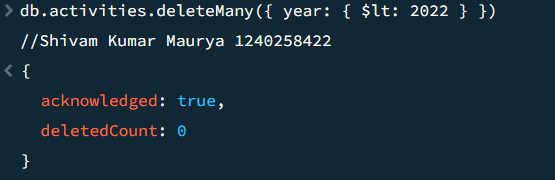
****

**Explanation--**This query is use to update attendance of only those students who won any “Hackathon” by changing to 100% .

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

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

**Output--**

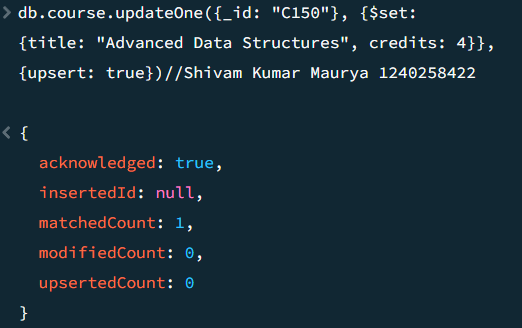


**Explanation**-- This query will delete all the records where the activity year is before 2022 of collection name activities.

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.course.updateOne({\_id: "C150"}, {$set: {title: "Advanced Data Structures", credits: 4}}, {upsert: true})*

**Output***--*

****

**Explanation** -- This query will upsert a course record for “Data Structures” with ID “C150” and credits it 4-- if it doesn’t exist &

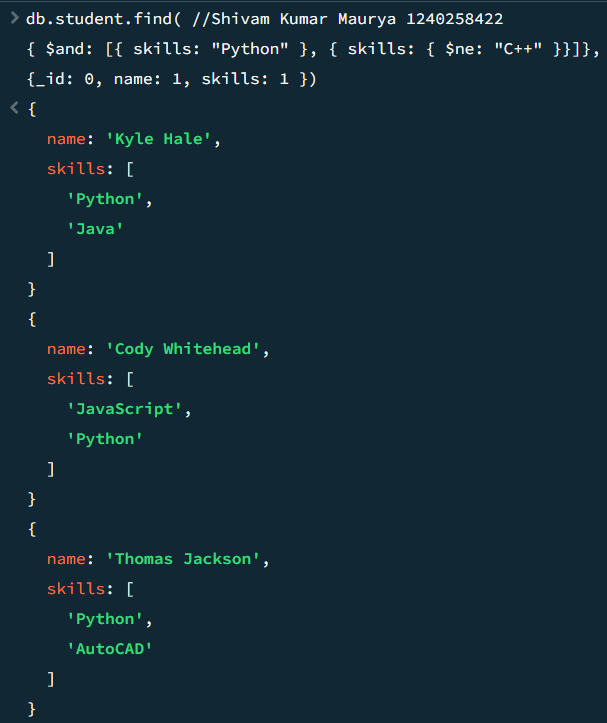
insert it otherwise update the title to "Advanced Data Structures".

**5.Array & Operator Usage**

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

**Query**-- db.student.find( { $and: [{ skills: "Python" }, { skills: { $ne: "C++" }}]},{\_id: 0, name: 1, skills: 1 })

**Output—**

****

**Explanation–** This query will give output of all students name & in skills who have “Python” but not “C++”.

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

**Query**-- *db.activites.aggregate([{ $group: { \_id: "$student\_id",*

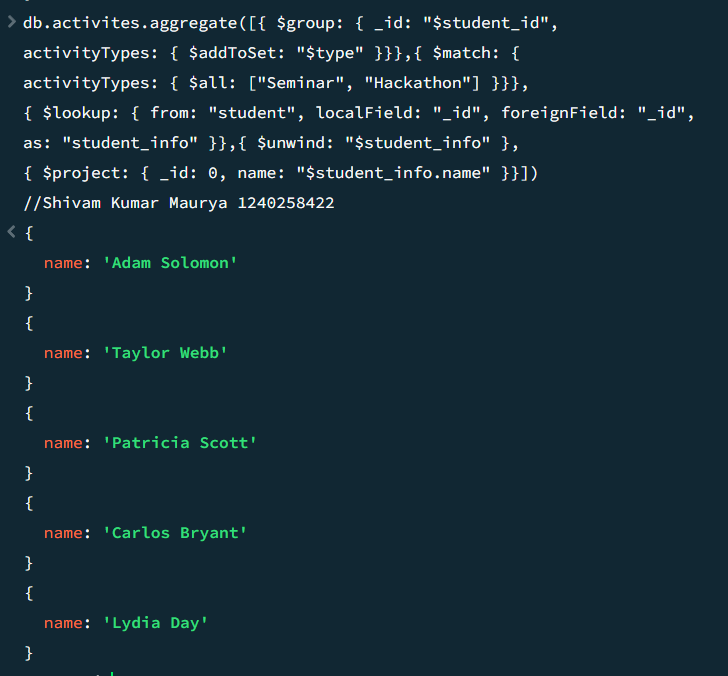
*activityTypes: { $addToSet: "$type" }}},{ $match: {*

*activityTypes: { $all: ["Seminar", "Hackathon"] }}},*

*{ $lookup: { from: "student", localField: "\_id", foreignField: "\_id",as: "student\_info" }},{ $unwind: "$student\_info" },*

*{ $project: { \_id: 0, name: "$student\_info.name" }}])*

**Output**—

**

**Explanation–** This query will return names of only those students who participated in “Seminar” and “Hackathon” both.

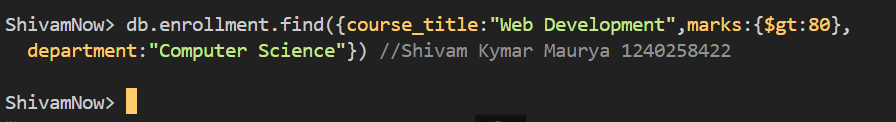
**6.Subdocuments and Nested Conditions**

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

the "Computer Science" department.

**Query**— *db.enrollment.find({course\_title: "Web Development", marks: {$gt:80},department: "Computer Science"})*

**Output—**

****

**Explanation**– This Query will find the students who scored more than 80 in “Web Development” only if they belong to the “Computer Science” department.

**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.aggregate( [{ $lookup: { from: "course", localField: "courses", foreignField: "\_id", as: "courseInfo" }},*

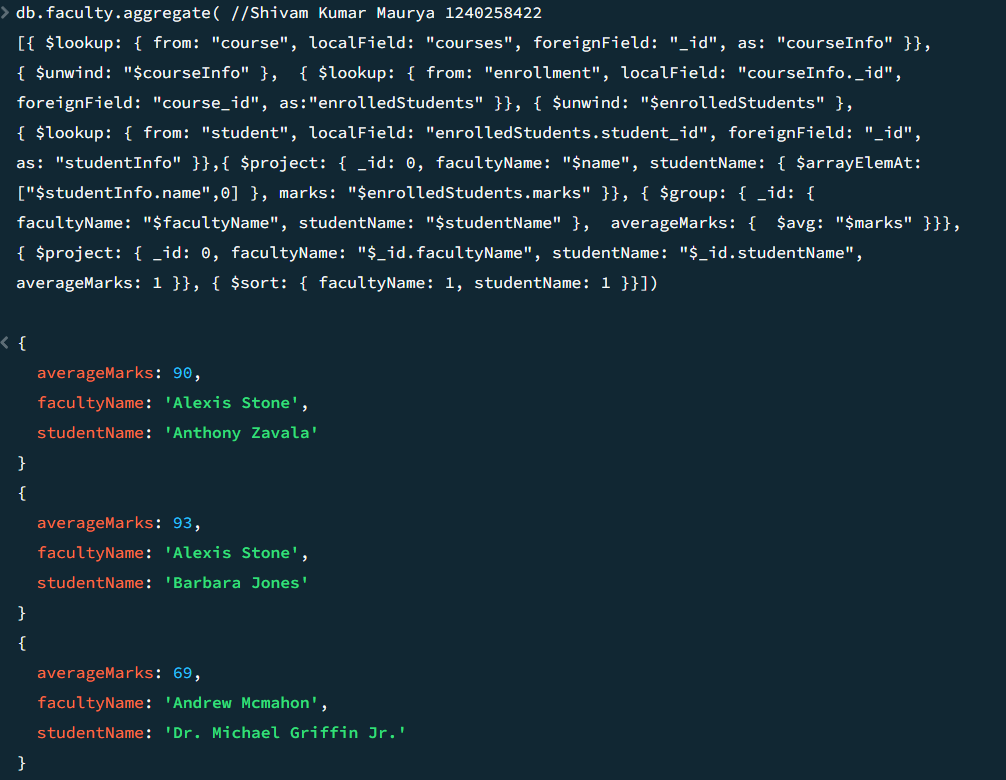
*{ $unwind: "$courseInfo" }, { $lookup: { from: "enrollment", localField: "courseInfo.\_id", foreignField: "course\_id", as:"enrolledStudents" }}, { $unwind: "$enrolledStudents" },*

*{ $lookup: { from: "student",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--**

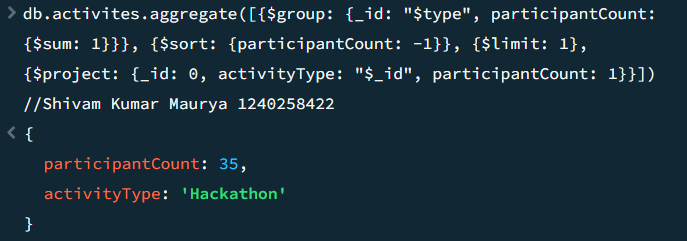


**Explanation**– This Query will list the names of all students enrolled in their courses along with average marks per students per faculty for each faculty.

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

**Query--** *db.activites.aggregate([{$group: {\_id: "$type", participantCount: {$sum: 1}}}, {$sort: {participantCount: -1}}, {$limit: 1}, {$project: {\_id: 0, activityType: "$\_id", participantCount: 1}}])*

**Output—**

****

**Explanation**–This query will show most popular activity (e.g., Hackathon, Seminar) by total number of students participated.