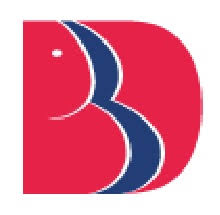
**BABU BANARASI UNIVERSITY**



**SESSION 2025-26**

**BCADS24**

Name: Neetika Srivastava

Batch: BCA(DS&AI)

Roll no:63

Subject: NO SQL and DBase 101

Subject Code: BCADSN13202

Submitted to: Signature:

Mr. Ankit Verma

**INDEX**

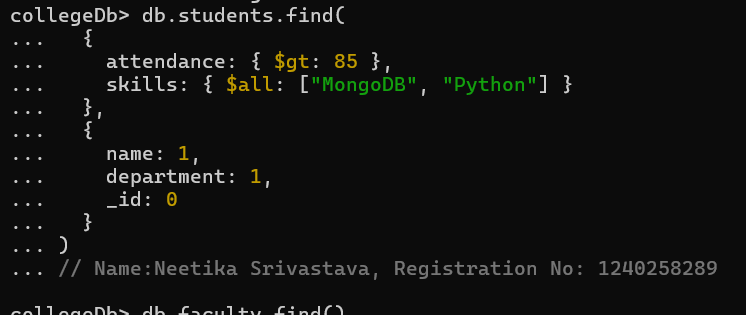
|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Name of Experiment** | **Faculty Signature** | **Remarks** |
| **1** | Complex Filters & Projections |  |  |
| **2** | Joins($lookup) and Aggregations |  |  |
| **3** | Grouping, Sorting, and Limiting |  |  |
| **4** | Update, Upsert, and Delete |  |  |
| **5** | Array & Operator Usage |  |  |
| **6** | Subdocuments and Nested Conditions |  |  |
| **7** | Advanced Aggregation (Challenge Level) |  |  |

**1.Complex Filters & Projections**

**Question1.** List the names and departments of students who have more than 85% attendance and are skilled in both "MongoDB" and "Python".

**Answer (Actual Query)**

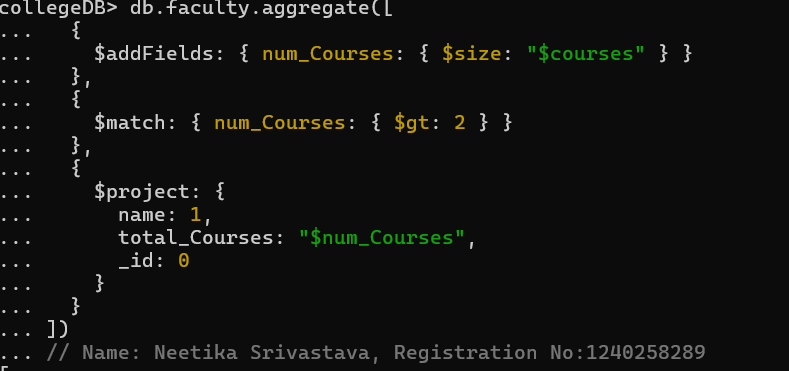
**Output:**

****

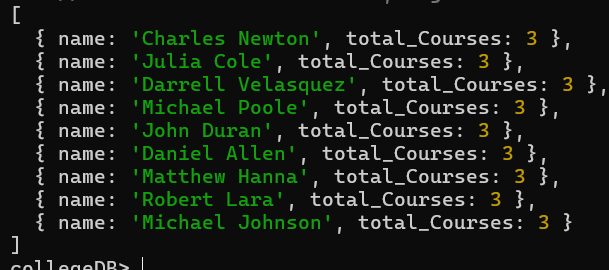
* Explanation:
* **attendance: {$gt: 85}**  
  This condition checks that the student's attendance percentage is greater than 85%.
* **skills: {$all: ["MongoDB", "Python"]}**  
  This condition ensures the student's "skills" array contains both "MongoDB" and "Python".
* Only students meeting both conditions (attendance > 85 and both skills present) will be listed with their respective names and departments.

**Question2.** Show all faculty who are teaching more than 2 courses. Display their names and the total number of courses they teach.

**Answer (Actual Query)**



**Output:**

****

* Explanation:
* Count Courses for Each Faculty: The $addFields stage creates a new field called num\_Courses that counts how many items (courses) are in the courses array for each faculty member.
* Filter Faculty with More Than 2 Courses: The $match stage selects only faculty members whose num\_Courses field is greater than 2, meaning they teach at least three courses.​
* Display Only Names and Counts: The $project stage shows only the faculty member's name and the number of courses they teach (total\_Courses), hiding their ID.

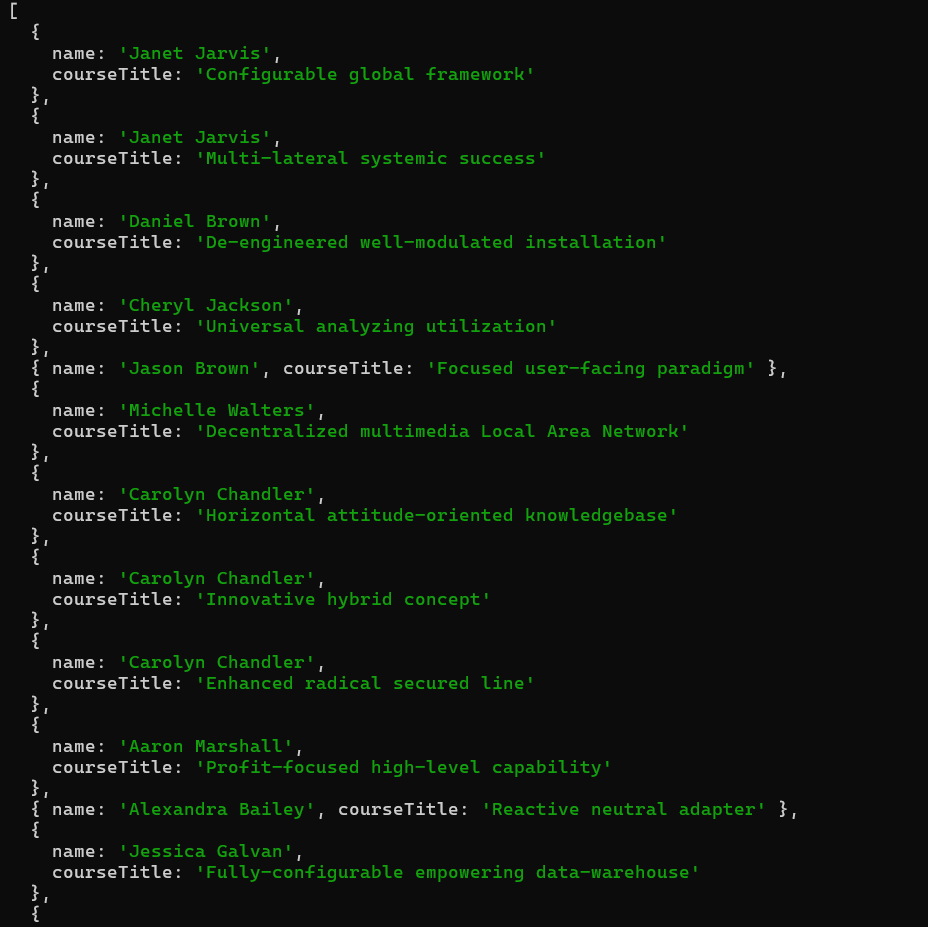
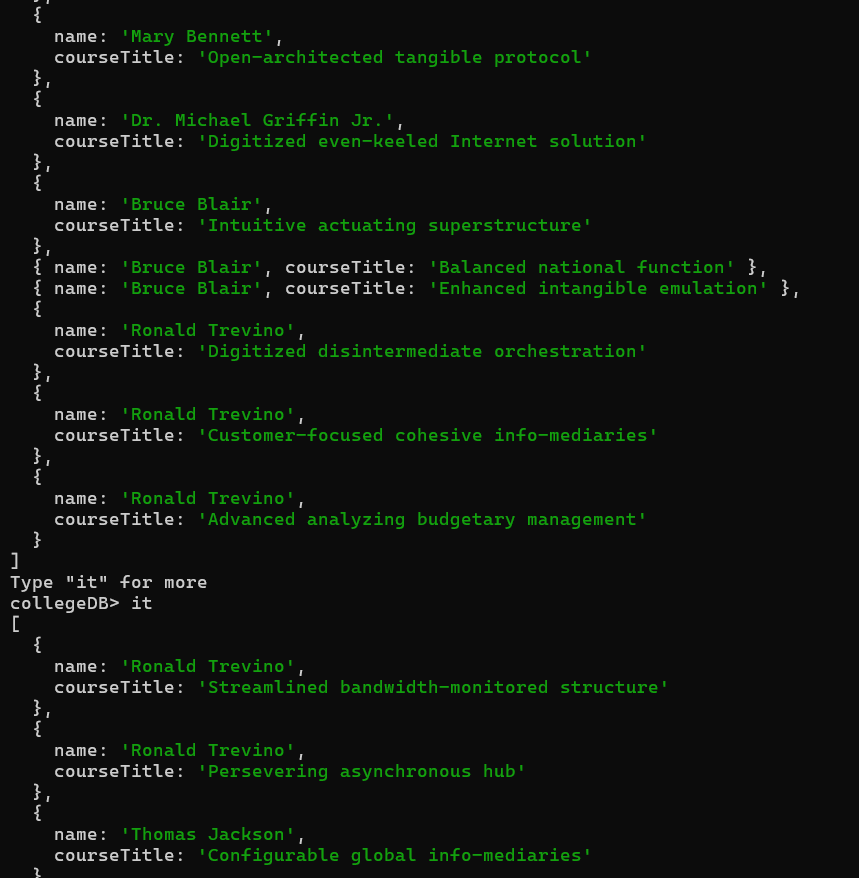
**2. Joins ($lookup) and Aggregations**

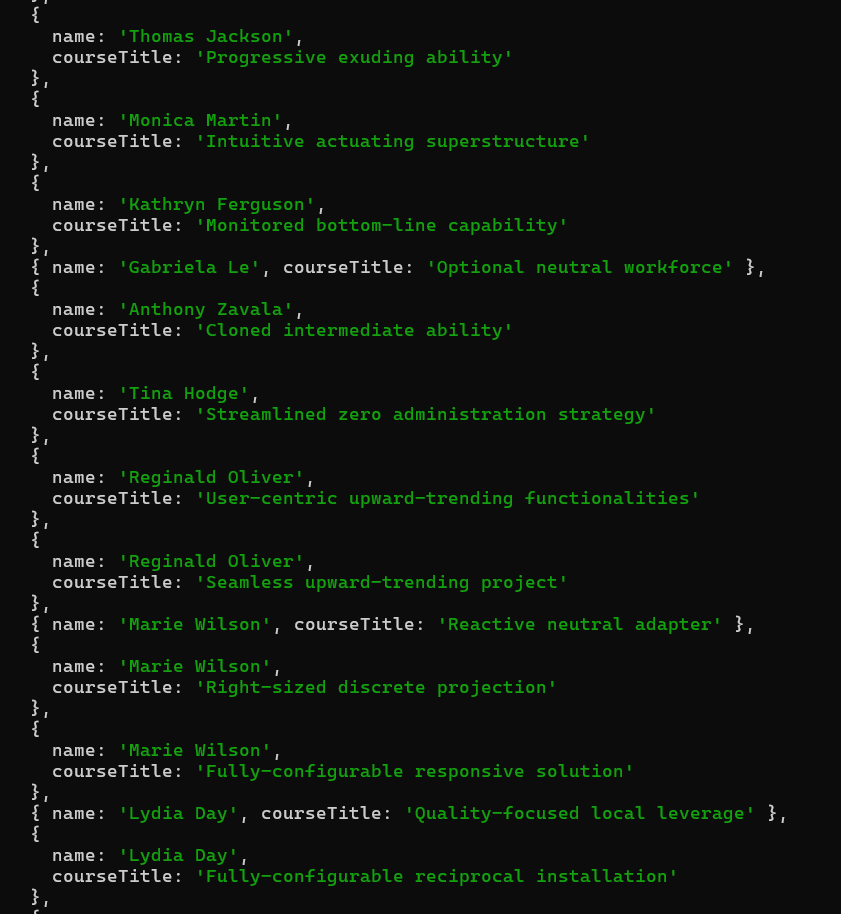
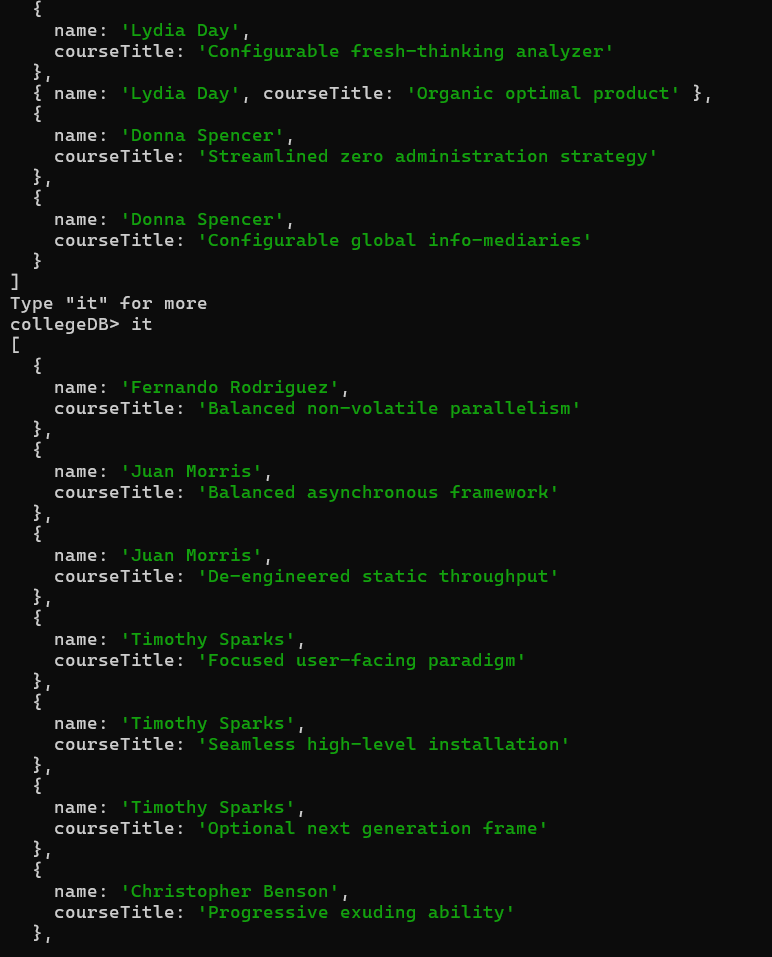
**Question3.** Write a query to show each student’s name along with the course titles they are enrolled in (use $lookup between enrollments, students, and courses).

**Answer (Actual Query)**



**Output:**

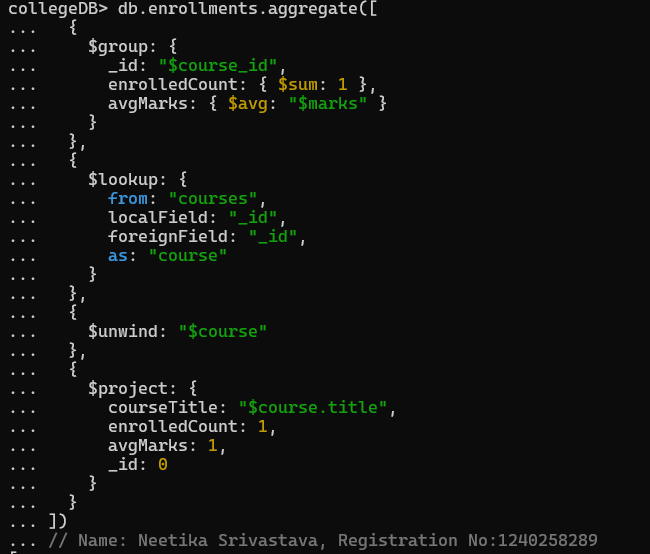
 

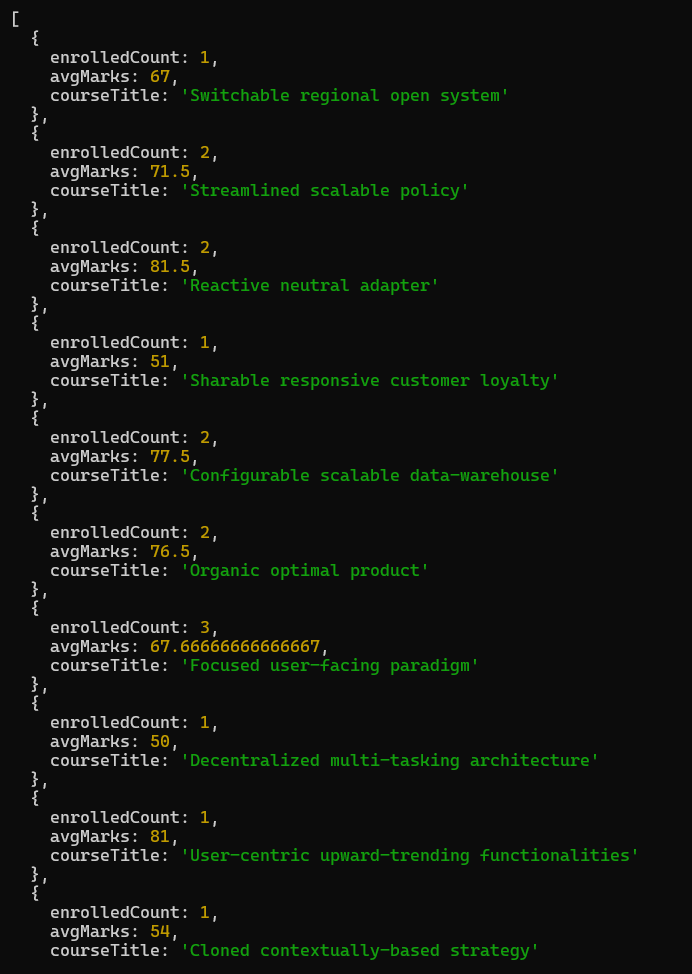
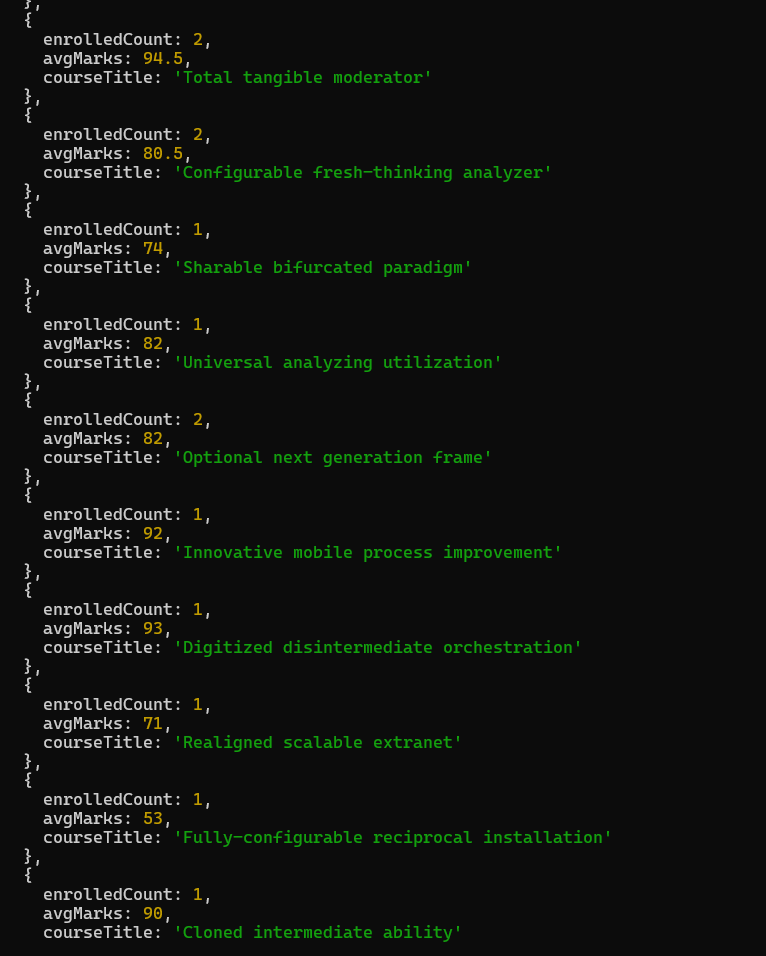
* Explanation:
* **First Lookup:**  
  The query connects each student with their enrollment records (using the student's ID), so you know which courses they signed up for.
* **Second Lookup:**  
  It then connects these enrollments to course records (using the course ID) to get the actual course titles.
* Finally shows the course titles next to each student’s name.

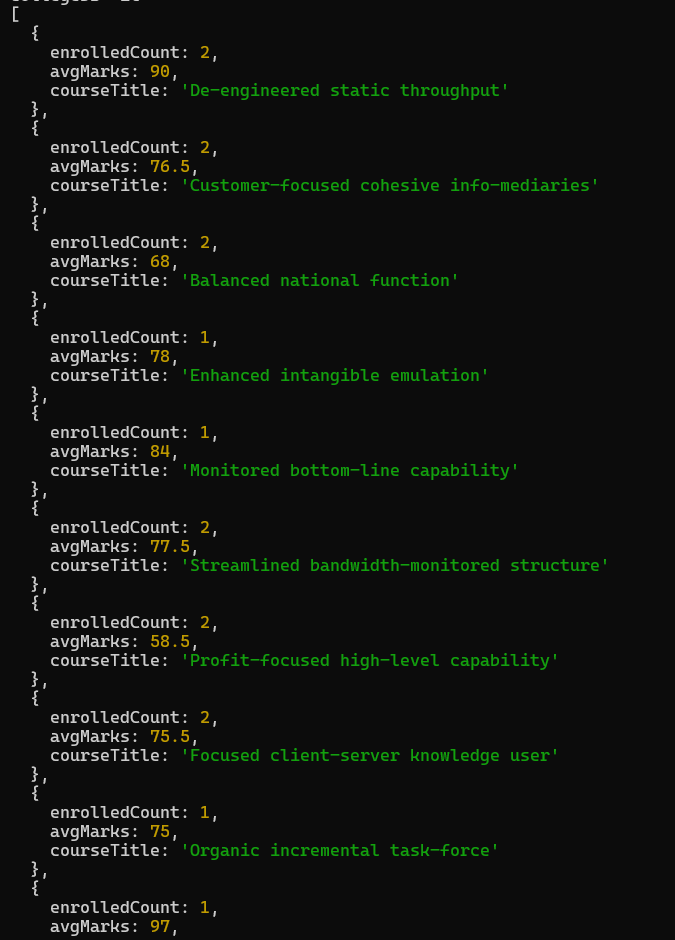
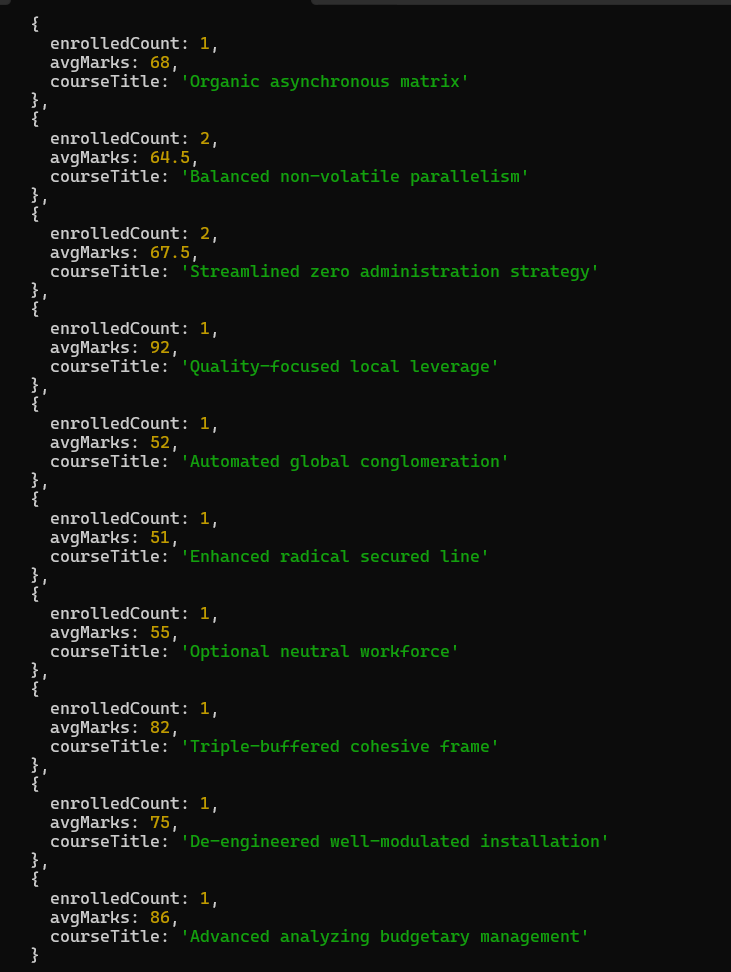
**Question4.** For each course, display the course title, number of students enrolled, and average marks (use $group).

**Answer (Actual Query)**

****

**Output:**

** **

** **

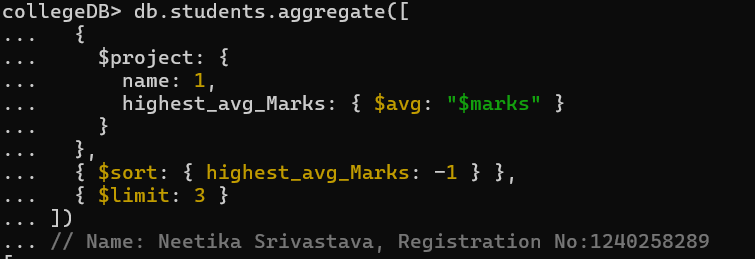
Explanation:

* **Grouping:** grouping all student enrollments by course, counting the number of students enrolled and calculating the average marks each course.
* **Course Lookup:**  
  It then looks up each course’s details using $lookup, joining the grouped information with course data.
* Finally shows the avgMarks with courseTitle.

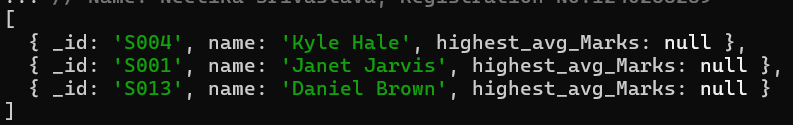
**3. Grouping, Sorting, and Limiting**

**Question5.** Find the top 3 students with the highest average marks across all enrolled courses.

**Answer (Actual Query)**



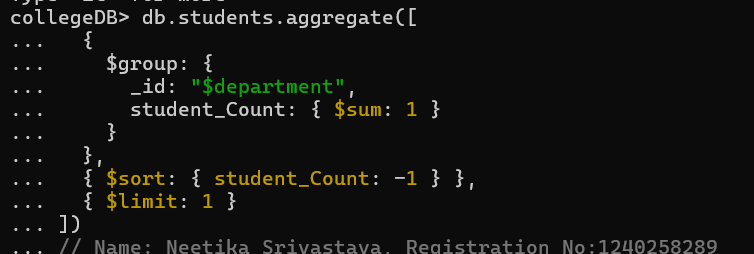
**Output:**



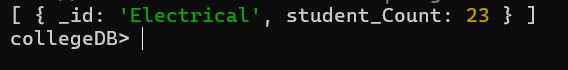
* Explanation:
* **Calculate Average Marks:**  
  For each student, it calculates the average’s marks across all courses with $avg.
* **Sort in Descending Order:**  
  Arrange the average marks using $sort.
* **Take Top 3:**  
  Finally, only the top 3 students (only highest averages) are shown using $limit.

**Question6.** Count how many students are in each department. Display the department with the highest number of students.

**Answer (Actual Query)**



Output:

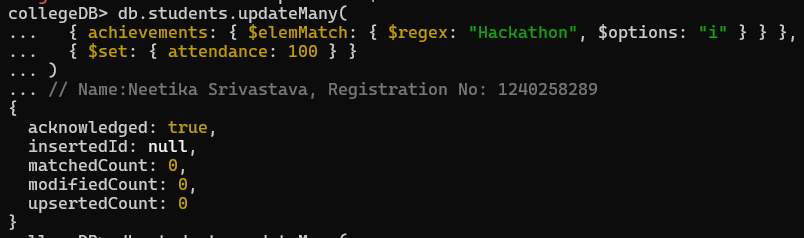


* Explanation:
* Group all students by their department name. For each department, it collects all the students who belong to it.
* Arrange the department with the most students appears at the top (student\_Count: -1 means descending order) using $Sort.
* Finally, $limit: 1 is used to keep only the top department—the one with the most students.

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

**Question7.** Update attendance to 100% for all students who won any "Hackathon".

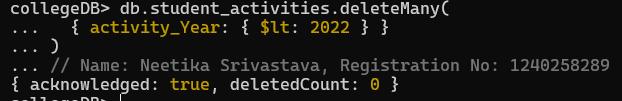
**Output:**

****

* Explanation:
* The query uses updateMany to affect all student records that match the condition.​
* Inside the filter, $elemMatch is used to look for any achievement entry containing the text "Hackathon" (case-insensitive due to $options: "i").​
* For all matching students, the operation sets their attendance field to 100 using the $set update operator.

**Question8.** Delete all student activity records where the activity year is before 2022.

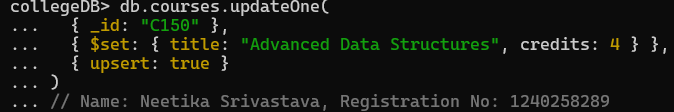
**Output:**

****

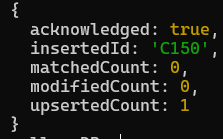
* Explanation:
  + **Target the Collection**: It runs on the student\_activities collection and uses the deleteMany operations.​
  + **Set the Condition**: The condition {activity\_Year: {$lt: 2022}} means “activity year is less than 2022.” So, it matches every record with an activity year before 2022.​
* **Delete the Records**: All records that match the condition are deleted from the collection in a single step.​

**Question9.** 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".

**Answer (Actual Query)**

****

**Output:**

****

* Explanation:
* **What "Upsert" Means:** "Upsert" in MongoDB is a special option that means "update if found, insert if not found".​
* **Find or Insert Logic:** The query looks for a course with the ID "C150". If it finds this record, it updates the title to "Advanced Data Structures" and the credits to 4.​

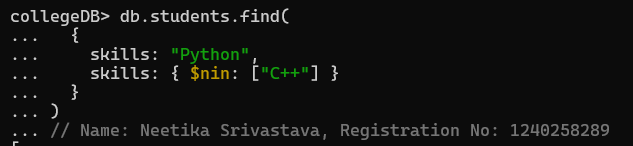
If no record is found with that ID, it creates (inserts) a new course document with ID "C150", title "Advanced Data Structures", and credits 4.​

* **How the Query Works:** The {$set: {...}} part sets the values whether updating or inserting. The upsert: truly makes sure the insert happens if the document isn't found.​

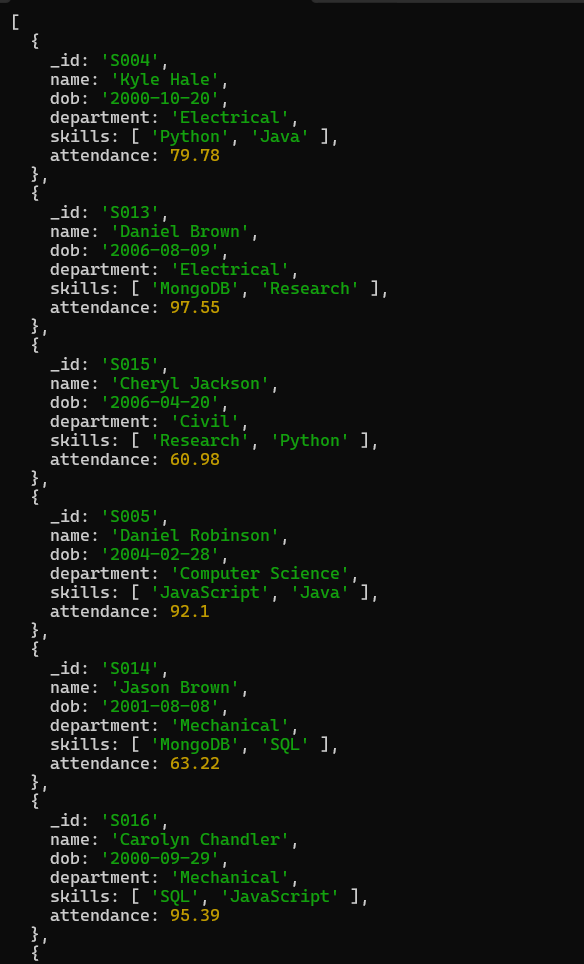
**5. Array & Operator Usage**

**Question10.** Find all students who have "Python" as a skill but not "C++".

**Answer (Actual Query)**

****

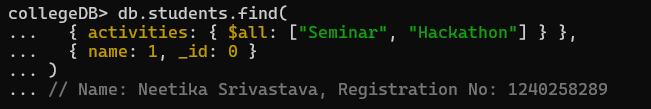
**Output:**

**  **

* Explanation:
* The condition skills: "Python" ensures the student’s skills array contains "Python".
* The part skills: {$nin: ["C++"]} means exclude any student whose skills array contains "C++".
* Only students who have "Python" in their skill set AND do not have "C++" in their skill set will be returned.

**Question11.** Return names of students who participated in "Seminar" and "Hackathon" both.

**Output:**

****

* Explanation:
* **Check Both Activities**: The query uses activities: {$all: ["Seminar", "Hackathon"]} to search for student documents where the activities array contains both the values "Seminar" and "Hackathon".​
* **Return Only Student Names**: The part {name: 1, \_id: 0} means the query will only show the student's name in the result, not their database ID.​

**6. Subdocuments and Nested Conditions**

**Question12.** Find students who scored more than 80 in "Web Development" only if they belong to the "Computer Science" department.

**Output:**

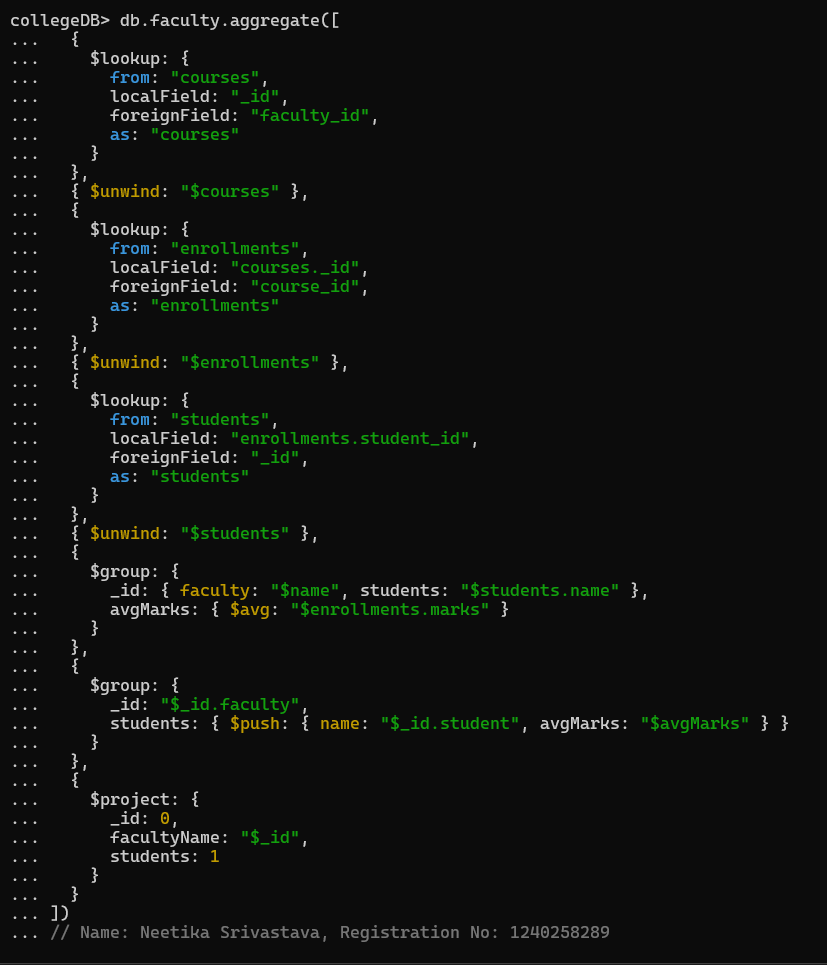
****

* Explanation:
* The query checks that the student’s department is "Computer Science".​
* Inside each student document, it looks for marks where the course is "Web Development" and the score is greater than 80 using $elemMatch.​
* Only students meeting both these conditions will be shown in the results.

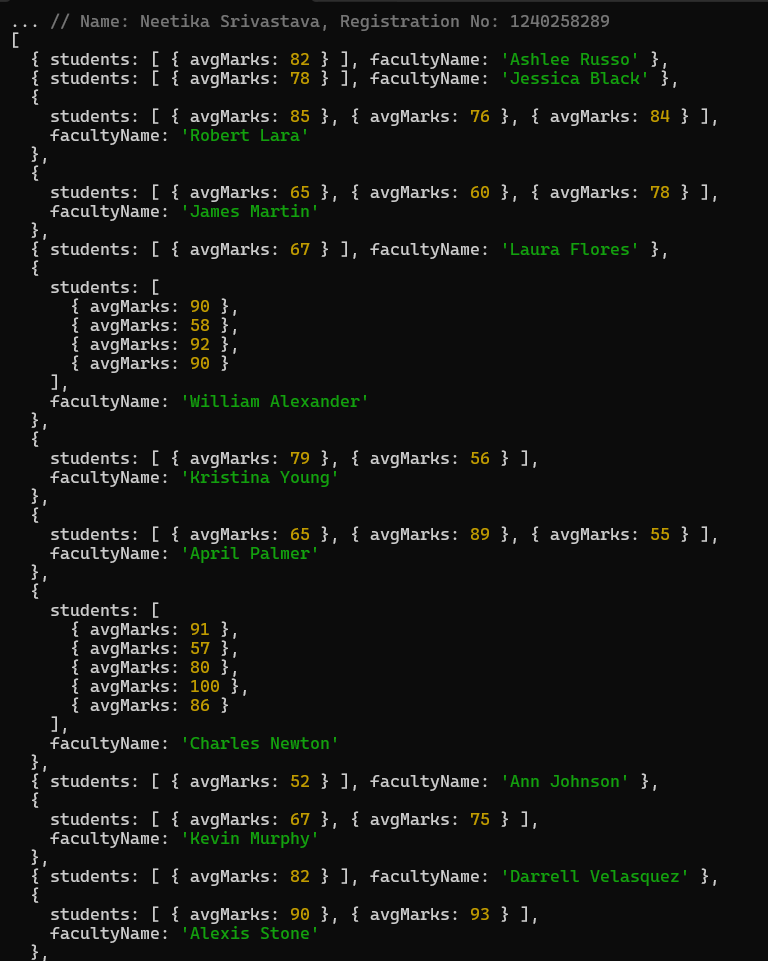
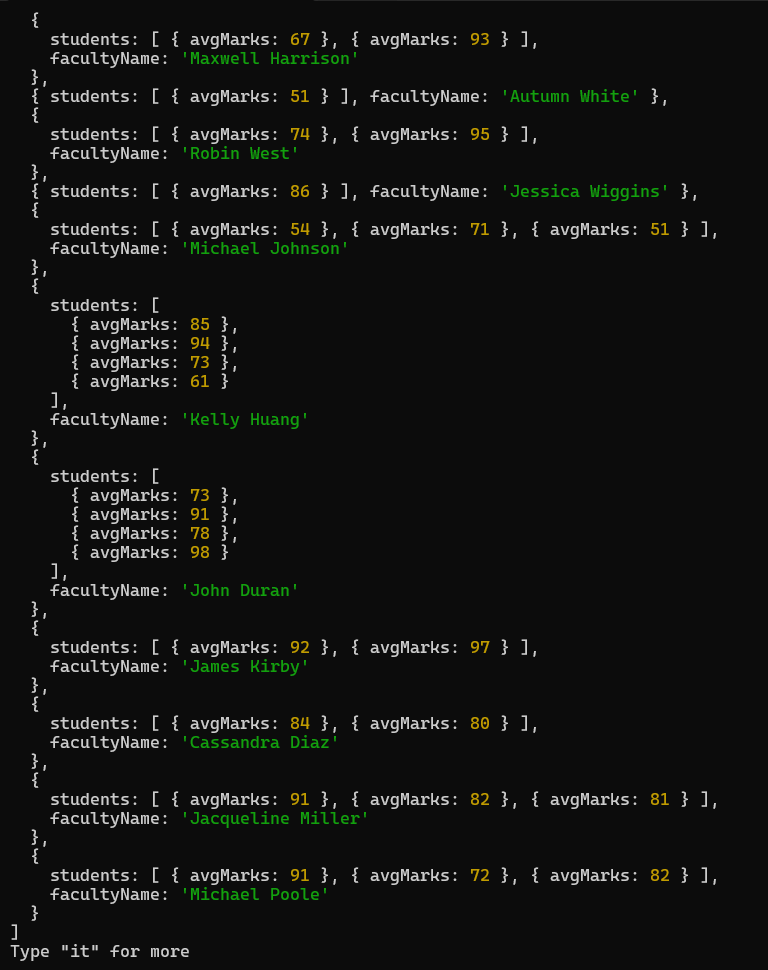
**7. Advanced Aggregation (Challenge Level)**

**Question13.** For each faculty member, list the names of all students enrolled in their courses along with average marks per student per faculty.

**Answer (Actual Query)**

****

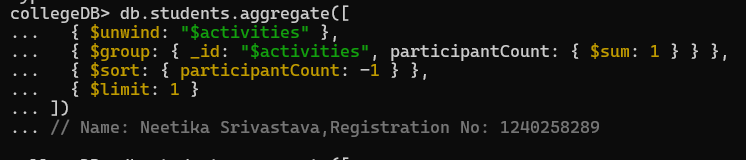
**Output:**

**  **

* Explanation:
* **Connect Faculty and Courses**: The query first matches each faculty member to the courses they teach using $lookup
* **Connect Courses and Enrollments:** Another $unwind followed by $lookup connects each course with its enrollments, allowing access to which students are enrolled in which course.
* **Get Student Names**: Next, it looks up each student's actual name from the student’s collection.
* **Calculate and List:** Lists all students' names and calculates the average marks per student in that faculty's courses.
* **Show Results**: Finally, it displays each faculty's name, all student names in their courses, and the average marks.

**Question14.** Show the most popular activity type (e.g., Hackathon, Seminar, etc.) by number of student participants.

**Output:**

****

* Explanation:
* **Separate All Activities**: The $unwind stage takes the activities array from each student and makes a new entry for each activity, so every activity gets its own line.
* **Count Each Activity**: Next, $group collects all entries of the same activity and counts them. This tells us how many times each activity was done, across all students.
* **Sort by Most Popular**: $sort puts the activities in order, arrange it (highest to lowest).
* **Show Only the Top Activity:** $limit: 1 means we only see the activity with the highest count—the most popular.

**Conclusion:** All the queries show how MongoDB is used to manage data for students, faculty, courses, enrollnment, and activities in a collegeDB.​​

* With basic commands, you can find, update, count, and delete records using simple and clear conditions (like finding skills, deleting old records, updating attendance, and more).​​
* Aggregation lets you group, filter, and sort data, so you can answer questions like which department has the most students, which activity is most popular, or which faculty members teach the most courses.​​
* Array operators quickly find students with certain skills or who did specific activities, while update and delete commands help keep your data organized and up-to-date.