BABU BANARASI UNIVERSITY

SESSION 2025-26

Name: Neetika Srivastava

Batch: BCA(DS&AI)

Roll no:63

Subject: NO SQL and DBase 101

Subject Code: BCADSN13202

Submitted to:

Mr. Ankit Verma

Signature:

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:

```
collegeDb> db.students.find(
...   {
...     attendance: { $gt: 85 },
...     skills: { $all: ["MongoDB", "Python"] }
...   },
...   {
...     name: 1,
...     department: 1,
...     _id: 0
...   }
... )
... // Name:Neetika Srivastava, Registration No: 1240258289
```

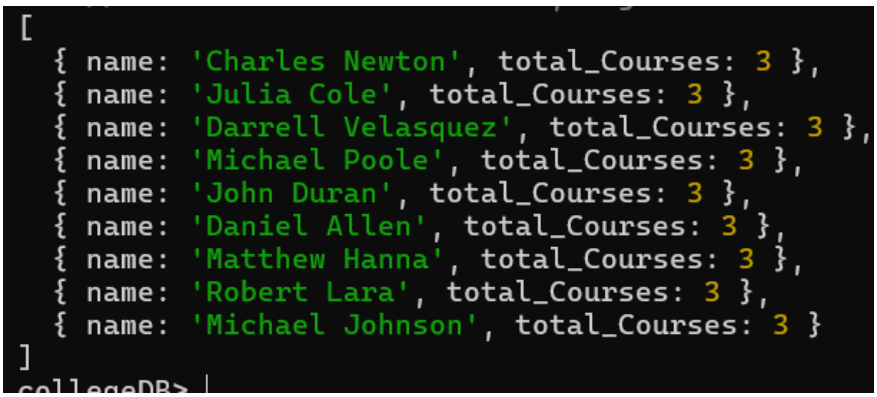
- Explanation:
  - **attendance: {$gt: 85}**
    This condition checks that the student's attendance percentage is greater than 85%.
  - **skills: {$all: ["MongoDB", "Python"]}**
    This condition ensures the student's "skills" array contains both "MongoDB" and "Python".
- Only students meeting both 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)

```
collegeDB> db.faculty.aggregate([
...   {
...     $addFields: { num_Courses: { $size: "$courses" } }
...   },
...   {
...     $match: { num_Courses: { $gt: 2 } }
...   },
...   {
...     $project: {
...       name: 1,
...       total_Courses: "$num_Courses",
...       _id: 0
...     }
...   }
... ])
... // Name: Neetika Srivastava, Registration No:1240258289
```

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)

```
collegeDB> db.students.aggregate([
...   {
...     $lookup: {
...       from: "enrollments",
...       localField: "_id",
...       foreignField: "student_id",
...       as: "enrollments"
...     }
...   },
...   {
...     $unwind: "$enrollments"
...   },
...   {
...     $lookup: {
...       from: "courses",
...       localField: "enrollments.course_id",
...       foreignField: "_id",
...       as: "course"
...     }
...   },
...   {
...     $unwind: "$course"
...   },
...   {
...     $project: {
...       name: 1,
...       courseTitle: "$course.title",
...       _id: 0
...     }
...   }
... ])
... // Name: Neetika Srivastava, Registration No:1240258289
```

Output:

```
[
  {
    name: 'Janet Jarvis',
    courseTitle: 'Configurable global framework'
  },
  {
    name: 'Janet Jarvis',
    courseTitle: 'Multi-lateral systemic success'
  },
  {
    name: 'Daniel Brown',
    courseTitle: 'De-engineered well-modulated installation'
  },
  {
    name: 'Cheryl Jackson',
    courseTitle: 'Universal analyzing utilization'
  },
  { name: 'Jason Brown', courseTitle: 'Focused user-facing paradigm' },
  {
    name: 'Michelle Walters',
    courseTitle: 'Decentralized multimedia Local Area Network'
  },
  {
    name: 'Carolyn Chandler',
    courseTitle: 'Horizontal attitude-oriented knowledgebase'
  },
  {
    name: 'Carolyn Chandler',
    courseTitle: 'Innovative hybrid concept'
  },
  {
    name: 'Carolyn Chandler',
    courseTitle: 'Enhanced radical secured line'
  },
  {
    name: 'Aaron Marshall',
    courseTitle: 'Profit-focused high-level capability'
  },
  { name: 'Alexandra Bailey', courseTitle: 'Reactive neutral adapter' },
  {
    name: 'Jessica Galvan',
    courseTitle: 'Fully-configurable empowering data-warehouse'
  },
  {
```

```
  {
    name: 'Mary Bennett',
    courseTitle: 'Open-architected tangible protocol'
  },
  {
    name: 'Dr. Michael Griffin Jr.',
    courseTitle: 'Digitized even-keeled Internet solution'
  },
  {
    name: 'Bruce Blair',
    courseTitle: 'Intuitive actuating superstructure'
  },
  { name: 'Bruce Blair', courseTitle: 'Balanced national function' },
  { name: 'Bruce Blair', courseTitle: 'Enhanced intangible emulation' },
  {
    name: 'Ronald Trevino',
    courseTitle: 'Digitized disintermediate orchestration'
  },
  {
    name: 'Ronald Trevino',
    courseTitle: 'Customer-focused cohesive info-mediaries'
  },
  {
    name: 'Ronald Trevino',
    courseTitle: 'Advanced analyzing budgetary management'
  }
]
Type "it" for more
collegeDB> it
[
  {
    name: 'Ronald Trevino',
    courseTitle: 'Streamlined bandwidth-monitored structure'
  },
  {
    name: 'Ronald Trevino',
    courseTitle: 'Persevering asynchronous hub'
  },
  {
    name: 'Thomas Jackson',
    courseTitle: 'Configurable global info-mediaries'
```

```
  {
    name: 'Thomas Jackson',
    courseTitle: 'Progressive exuding ability'
  },
  {
    name: 'Monica Martin',
    courseTitle: 'Intuitive actuating superstructure'
  },
  {
    name: 'Kathryn Ferguson',
    courseTitle: 'Monitored bottom-line capability'
  },
  { name: 'Gabriela Le', courseTitle: 'Optional neutral workforce' },
  {
    name: 'Anthony Zavala',
    courseTitle: 'Cloned intermediate ability'
  },
  {
    name: 'Tina Hodge',
    courseTitle: 'Streamlined zero administration strategy'
  },
  {
    name: 'Reginald Oliver',
    courseTitle: 'User-centric upward-trending functionalities'
  },
  {
    name: 'Reginald Oliver',
    courseTitle: 'Seamless upward-trending project'
  },
  { name: 'Marie Wilson', courseTitle: 'Reactive neutral adapter' },
  {
    name: 'Marie Wilson',
    courseTitle: 'Right-sized discrete projection'
  },
  {
    name: 'Marie Wilson',
    courseTitle: 'Fully-configurable responsive solution'
  },
  { name: 'Lydia Day', courseTitle: 'Quality-focused local leverage' },
  {
    name: 'Lydia Day',
    courseTitle: 'Fully-configurable reciprocal installation'
  },
  {
```

```
  {
    name: 'Lydia Day',
    courseTitle: 'Configurable fresh-thinking analyzer'
  },
  { name: 'Lydia Day', courseTitle: 'Organic optimal product' },
  {
    name: 'Donna Spencer',
    courseTitle: 'Streamlined zero administration strategy'
  },
  {
    name: 'Donna Spencer',
    courseTitle: 'Configurable global info-mediaries'
  }
]
Type "it" for more
collegeDB> it
[
  {
    name: 'Fernando Rodríguez',
    courseTitle: 'Balanced non-volatile parallelism'
  },
  {
    name: 'Juan Morris',
    courseTitle: 'Balanced asynchronous framework'
  },
  {
    name: 'Juan Morris',
    courseTitle: 'De-engineered static throughput'
  },
  {
    name: 'Timothy Sparks',
    courseTitle: 'Focused user-facing paradigm'
  },
  {
    name: 'Timothy Sparks',
    courseTitle: 'Seamless high-level installation'
  },
  {
    name: 'Timothy Sparks',
    courseTitle: 'Optional next generation frame'
  },
  {
    name: 'Christopher Benson',
    courseTitle: 'Progressive exuding ability'
  },
```

- 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.

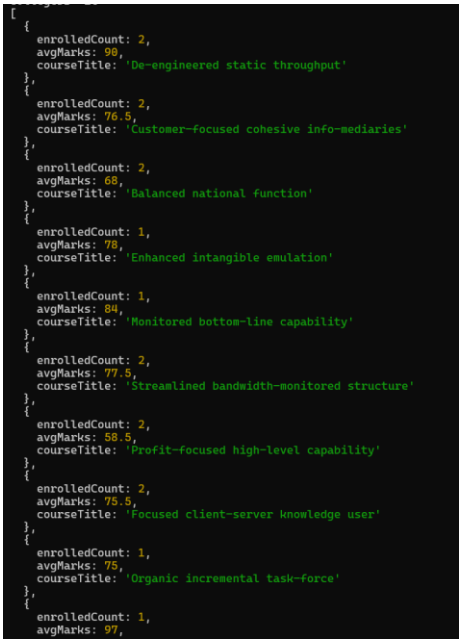
Question4. For each course, display the course title, number of students enrolled, and average marks (use $group).

Answer (Actual Query)

```
collegeDB> db.enrollments.aggregate([
...   {
...     $group: {
...       _id: "$course_id",
...       enrolledCount: { $sum: 1 },
...       avgMarks: { $avg: "$marks" }
...     }
...   },
...   {
...     $lookup: {
...       from: "courses",
...       localField: "_id",
...       foreignField: "_id",
...       as: "course"
...     }
...   },
...   {
...     $unwind: "$course"
...   },
...   {
...     $project: {
...       courseTitle: "$course.title",
...       enrolledCount: 1,
...       avgMarks: 1,
...       _id: 0
...     }
...   }
... ])
... // Name: Neetika Srivastava, Registration No:1240258289
```

Output:

```
[
  {
    enrolledCount: 1,
    avgMarks: 67,
    courseTitle: 'Switchable regional open system'
  },
  {
    enrolledCount: 2,
    avgMarks: 71.5,
    courseTitle: 'Streamlined scalable policy'
  },
  {
    enrolledCount: 2,
    avgMarks: 81.5,
    courseTitle: 'Reactive neutral adapter'
  },
  {
    enrolledCount: 1,
    avgMarks: 51,
    courseTitle: 'Sharable responsive customer loyalty'
  },
  {
    enrolledCount: 2,
    avgMarks: 77.5,
    courseTitle: 'Configurable scalable data-warehouse'
  },
  {
    enrolledCount: 2,
    avgMarks: 76.5,
    courseTitle: 'Organic optimal product'
  },
  {
    enrolledCount: 3,
    avgMarks: 67.66666666666667,
    courseTitle: 'Focused user-facing paradigm'
  },
  {
    enrolledCount: 1,
    avgMarks: 50,
    courseTitle: 'Decentralized multi-tasking architecture'
  },
  {
    enrolledCount: 1,
    avgMarks: 81,
    courseTitle: 'User-centric upward-trending functionalities'
  },
  {
    enrolledCount: 1,
    avgMarks: 54,
    courseTitle: 'Cloned contextually-based strategy'
  },
  {
    enrolledCount: 2,
    avgMarks: 94.5,
    courseTitle: 'Total tangible moderator'
  },
  {
    enrolledCount: 2,
    avgMarks: 80.5,
    courseTitle: 'Configurable fresh-thinking analyzer'
  },
  {
    enrolledCount: 1,
    avgMarks: 74,
    courseTitle: 'Sharable bifurcated paradigm'
  },
  {
    enrolledCount: 1,
    avgMarks: 82,
    courseTitle: 'Universal analyzing utilization'
  },
  {
    enrolledCount: 2,
    avgMarks: 82,
    courseTitle: 'Optional next generation frame'
  },
  {
    enrolledCount: 1,
    avgMarks: 92,
    courseTitle: 'Innovative mobile process improvement'
  },
  {
    enrolledCount: 1,
    avgMarks: 93,
    courseTitle: 'Digitized disintermediate orchestration'
  },
  {
    enrolledCount: 1,
    avgMarks: 71,
    courseTitle: 'Realigned scalable extranet'
  },
  {
    enrolledCount: 1,
    avgMarks: 53,
    courseTitle: 'Fully-configurable reciprocal installation'
  },
  {
    enrolledCount: 1,
    avgMarks: 90,
    courseTitle: 'Cloned intermediate ability'
  }
```



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)

```
collegeDB> db.students.aggregate([
...   {
...     $project: {
...       name: 1,
...       highest_avg_Marks: { $avg: "$marks" }
...     }
...   },
...   { $sort: { highest_avg_Marks: -1 } },
...   { $limit: 3 }
... ])
... // Name: Neetika Srivastava, Registration No:1240258289
```

Output:

```
[
  { _id: 'S004', name: 'Kyle Hale', highest_avg_Marks: null },
  { _id: 'S001', name: 'Janet Jarvis', highest_avg_Marks: null },
  { _id: 'S013', name: 'Daniel Brown', highest_avg_Marks: null }
]
```

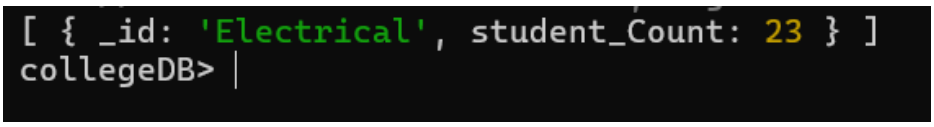
- Explanation:
  - **Calculate Average Marks:**
    For each student, it calculates the average's marks across all courses with $avg.
  - **Sort in Descending Order:**
    Arrange the average marks using $sort.
  - **Take Top 3:**
    Finally, only the top 3 students (only highest averages) are shown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

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

Answer (Actual Query)

```
collegeDB> db.students.aggregate([
...   {
...     $group: {
...       _id: "$department",
...       student_Count: { $sum: 1 }
...     }
...   },
...   { $sort: { student_Count: -1 } },
...   { $limit: 1 }
... ])
... // Name: Neetika Srivastava, Registration No:1240258289
```

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:

```
collegeDB> db.students.updateMany(
...   { achievements: { $elemMatch: { $regex: "Hackathon", $options: "i" } } },
...   { $set: { attendance: 100 } }
... )
... // Name:Neetika Srivastava, Registration No: 1240258289
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

- 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:

```
collegeDB> db.student_activities.deleteMany(
...   { activity_Year: { $lt: 2022 } }
... )
... // Name: Neetika Srivastava, Registration No: 1240258289
{ acknowledged: true, deletedCount: 0 }
```

- 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)

```
collegeDB> db.courses.updateOne(
...   { _id: "C150" },
...   { $set: { title: "Advanced Data Structures", credits: 4 } },
...   { upsert: true }
... )
... // Name: Neetika Srivastava, Registration No: 1240258289
```

Output:

```
{
  acknowledged: true,
  insertedId: 'C150',
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
```

- 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.

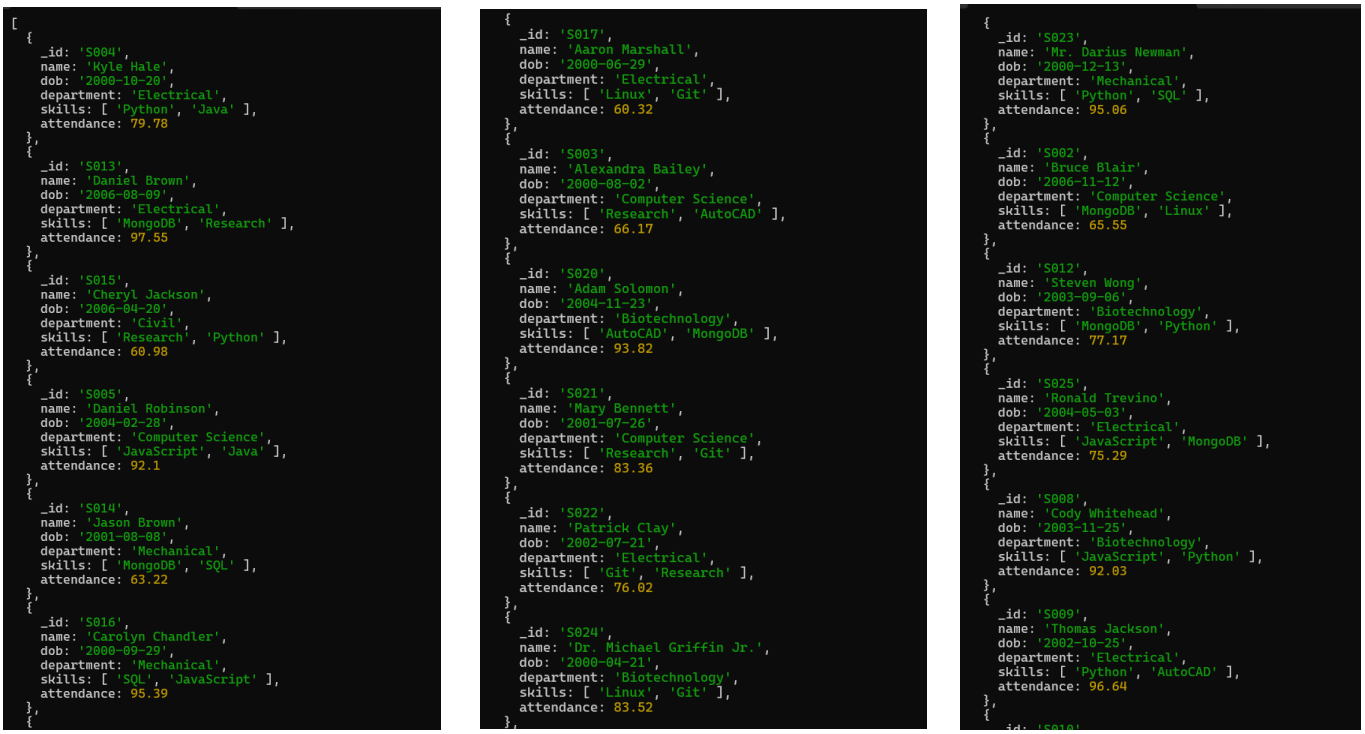
5. Array & Operator Usage

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

Answer (Actual Query)

```
collegeDB> db.students.find(
...   {
...     skills: "Python",
...     skills: { $nin: ["C++"] }
...   }
... )
... // Name: Neetika Srivastava, Registration No: 1240258289
```

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:

```
collegeDB> db.students.find(
...   { activities: { $all: ["Seminar", "Hackathon"] } },
...   { name: 1, _id: 0 }
... )
... // Name: Neetika Srivastava, Registration No: 1240258289
```

- 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:

```
collegeDB> db.students.find({
...   department: "Computer Science",
...   marks: { $elemMatch: { course: "Web Development", score: { $gt: 80 } } }
... })
... // Name: Neetika Srivastava, Registraation No: 1240258289
```

- 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)

```
collegeDB> db.faculty.aggregate([
...   {
...     $lookup: {
...       from: "courses",
...       localField: "_id",
...       foreignField: "faculty_id",
...       as: "courses"
...     }
...   },
...   { $unwind: "$courses" },
...   {
...     $lookup: {
...       from: "enrollments",
...       localField: "courses._id",
...       foreignField: "course_id",
...       as: "enrollments"
...     }
...   },
...   { $unwind: "$enrollments" },
...   {
...     $lookup: {
...       from: "students",
...       localField: "enrollments.student_id",
...       foreignField: "_id",
...       as: "students"
...     }
...   },
...   { $unwind: "$students" },
...   {
...     $group: {
...       _id: { faculty: "$name", students: "$students.name" },
...       avgMarks: { $avg: "$enrollments.marks" }
...     }
...   },
...   {
...     $group: {
...       _id: "$_id.faculty",
...       students: { $push: { name: "$_id.student", avgMarks: "$avgMarks" } }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       facultyName: "$_id",
...       students: 1
...     }
...   }
... ])
... // Name: Neetika Srivastava, Registration No: 1240258289
```

Output:

```
... // Name: Neetika Srivastava, Registration No: 1240258289
[
  { students: [ { avgMarks: 82 } ], facultyName: 'Ashlee Russo' },
  { students: [ { avgMarks: 78 } ], facultyName: 'Jessica Black' },
  {
    students: [ { avgMarks: 85 }, { avgMarks: 76 }, { avgMarks: 84 } ],
    facultyName: 'Robert Lara'
  },
  {
    students: [ { avgMarks: 65 }, { avgMarks: 60 }, { avgMarks: 78 } ],
    facultyName: 'James Martin'
  },
  { students: [ { avgMarks: 67 } ], facultyName: 'Laura Flores' },
  {
    students: [
      { avgMarks: 90 },
      { avgMarks: 58 },
      { avgMarks: 92 },
      { avgMarks: 90 }
    ],
    facultyName: 'William Alexander'
  },
  {
    students: [ { avgMarks: 79 }, { avgMarks: 56 } ],
    facultyName: 'Kristina Young'
  },
  {
    students: [ { avgMarks: 65 }, { avgMarks: 89 }, { avgMarks: 55 } ],
    facultyName: 'April Palmer'
  },
  {
    students: [
      { avgMarks: 91 },
      { avgMarks: 57 },
      { avgMarks: 80 },
      { avgMarks: 100 },
      { avgMarks: 86 }
    ],
    facultyName: 'Charles Newton'
  },
  { students: [ { avgMarks: 52 } ], facultyName: 'Ann Johnson' },
  {
    students: [ { avgMarks: 67 }, { avgMarks: 75 } ],
    facultyName: 'Kevin Murphy'
  },
  { students: [ { avgMarks: 82 } ], facultyName: 'Darrell Velasquez' },
  {
    students: [ { avgMarks: 90 }, { avgMarks: 93 } ],
    facultyName: 'Alexis Stone'
  },
```

```
  {
    students: [ { avgMarks: 100 }, { avgMarks: 80 } ],
    facultyName: 'Thomas Lewis'
  },
  {
    students: [ { avgMarks: 92 }, { avgMarks: 56 }, { avgMarks: 60 } ],
    facultyName: 'Jessica Campbell'
  },
  {
    students: [ { avgMarks: 53 }, { avgMarks: 100 } ],
    facultyName: 'William Adams'
  },
  { students: [ { avgMarks: 50 } ], facultyName: 'Robin Johnson' },
  { students: [ { avgMarks: 51 } ], facultyName: 'Stephen Galvan' },
  {
    students: [ { avgMarks: 65 }, { avgMarks: 86 } ],
    facultyName: 'Sandra Decker'
  },
  { students: [ { avgMarks: 64 } ], facultyName: 'Kathryn Young' }
]
Type "it" for more
collegeDB> it
[
  {
    students: [ { avgMarks: 58 }, { avgMarks: 97 } ],
    facultyName: 'Kevin Booth'
  },
  {
    students: [ { avgMarks: 54 }, { avgMarks: 82 } ],
    facultyName: 'Joshua Reyes'
  },
  {
    students: [ { avgMarks: 71 }, { avgMarks: 53 }, { avgMarks: 68 } ],
    facultyName: 'Julia Cole'
  },
  { students: [ { avgMarks: 91 } ], facultyName: 'Julie Elliott' },
  {
    students: [ { avgMarks: 73 }, { avgMarks: 56 } ],
    facultyName: 'Meghan Watson'
  },
  {
    students: [ { avgMarks: 94 }, { avgMarks: 60 }, { avgMarks: 58 } ],
    facultyName: 'Daniel Allen'
  },
  {
    students: [ { avgMarks: 91 }, { avgMarks: 52 } ],
    facultyName: 'Michele Hines'
  },
```

```
  {
    students: [ { avgMarks: 67 }, { avgMarks: 93 } ],
    facultyName: 'Maxwell Harrison'
  },
  { students: [ { avgMarks: 51 } ], facultyName: 'Autumn White' },
  {
    students: [ { avgMarks: 74 }, { avgMarks: 95 } ],
    facultyName: 'Robin West'
  },
  { students: [ { avgMarks: 86 } ], facultyName: 'Jessica Wiggins' },
  {
    students: [ { avgMarks: 54 }, { avgMarks: 71 }, { avgMarks: 51 } ],
    facultyName: 'Michael Johnson'
  },
  {
    students: [
      { avgMarks: 85 },
      { avgMarks: 94 },
      { avgMarks: 73 },
      { avgMarks: 61 }
    ],
    facultyName: 'Kelly Huang'
  },
  {
    students: [
      { avgMarks: 73 },
      { avgMarks: 91 },
      { avgMarks: 78 },
      { avgMarks: 98 }
    ],
    facultyName: 'John Duran'
  },
  {
    students: [ { avgMarks: 92 }, { avgMarks: 97 } ],
    facultyName: 'James Kirby'
  },
  {
    students: [ { avgMarks: 84 }, { avgMarks: 80 } ],
    facultyName: 'Cassandra Diaz'
  },
  {
    students: [ { avgMarks: 91 }, { avgMarks: 82 }, { avgMarks: 81 } ],
    facultyName: 'Jacqueline Miller'
  },
  {
    students: [ { avgMarks: 91 }, { avgMarks: 72 }, { avgMarks: 82 } ],
    facultyName: 'Michael Poole'
  }
]
Type "it" for more
```

- 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:

```
collegeDB> db.students.aggregate([
...   { $unwind: "$activities" },
...   { $group: { _id: "$activities", participantCount: { $sum: 1 } } },
...   { $sort: { participantCount: -1 } },
...   { $limit: 1 }
... ])
... // Name: Neetika Srivastava,Registration No: 1240258289
```

- 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.