BABU BANARSI DAS UNIVERSITY

NO SQL and Dbaas

(BCADSN13202)

PROJECT

SUBMITTED TO:

Mr Ankit Verma

SUBMITTED BY:

Name: Prachi Pathak

Section: BCADS-25

Roll no: 1240258316

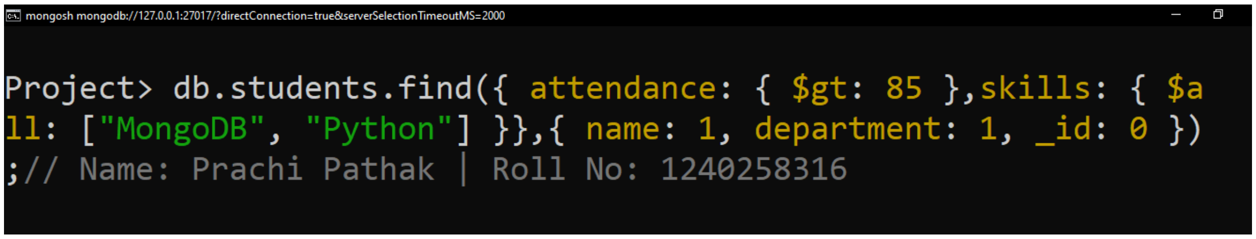
PROJECT – 1

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:

db.students_full.find({attendance: { $gt: 85 },skills: { $all: ["MongoDB", "Python"] }},{_id: 0,name: 1,department: 1})



mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
Project> db.students.find({ attendance: { $gt: 85 },skills: { $a
ll: ["MongoDB", "Python"] }},{ name: 1, department: 1, _id: 0 })
;// Name: Prachi Pathak | Roll No: 1240258316
```

Explanation:

This query retrieves all students who have **more than 85% attendance** and possess **both "MongoDB" and "Python"** as skills.

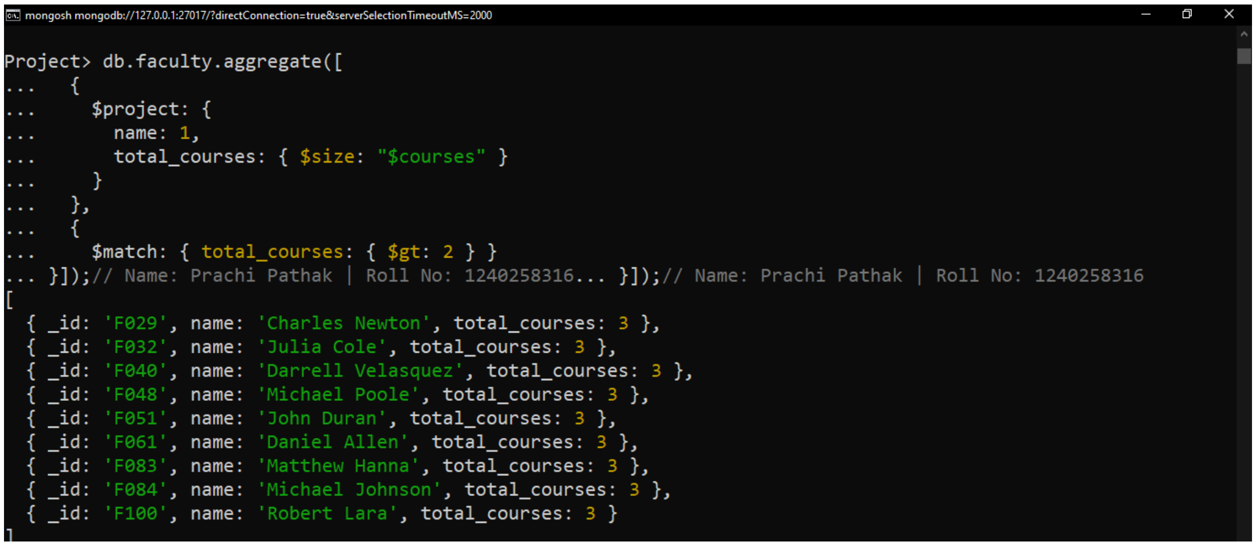
- The **`$gt`** operator filters students with attendance greater than 85.
- The **`$all`** operator ensures that both specified skills exist in the `skills` array.
- The projection `{ name: 1, department: 1, _id: 0 }` displays only the student's name and department while hiding the `_id` field.

Result:
Shows names and departments of students with attendance above 85% who have both skills.

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

Solution:

db.faculty.aggregate([{$project: {name: 1,total_courses: { $size: "$courses" }}},{$match: { total_courses: { $gt: 2 } }}]);



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Project> db.faculty.aggregate([
...   {
...     $project: {
...       name: 1,
...       total_courses: { $size: "$courses" }
...     }
...   },
...   {
...     $match: { total_courses: { $gt: 2 } }
... }]);// Name: Prachi Pathak | Roll No: 1240258316... }]);// Name: Prachi Pathak | Roll No: 1240258316
[
  { _id: 'F029', name: 'Charles Newton', total_courses: 3 },
  { _id: 'F032', name: 'Julia Cole', total_courses: 3 },
  { _id: 'F040', name: 'Darrell Velasquez', total_courses: 3 },
  { _id: 'F048', name: 'Michael Poole', total_courses: 3 },
  { _id: 'F051', name: 'John Duran', total_courses: 3 },
  { _id: 'F061', name: 'Daniel Allen', total_courses: 3 },
  { _id: 'F083', name: 'Matthew Hanna', total_courses: 3 },
  { _id: 'F084', name: 'Michael Johnson', total_courses: 3 },
  { _id: 'F100', name: 'Robert Lara', total_courses: 3 }
]
```

Explanation:

This aggregation identifies faculty members teaching **more than two courses**.

- The ***`$project`*** stage calculates the total number of courses for each faculty using ***`$size`*** on the **`courses`** array.
- The `$match` stage filters only those records where **`total_courses`** is greater than 2. The output includes the faculty name and their respective course count.

Result:

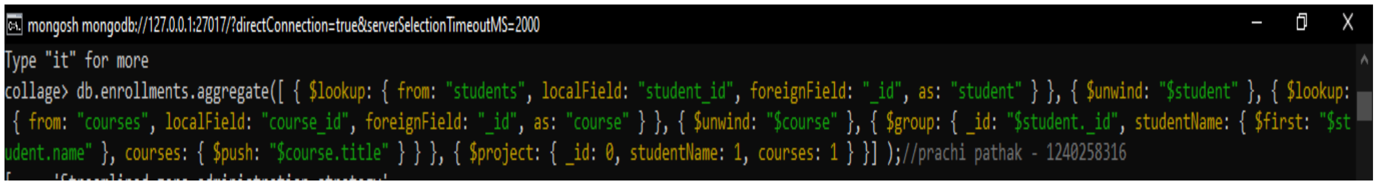
Displays faculty name and the number of courses they handle.

Joins ($lookup) and Aggregations:

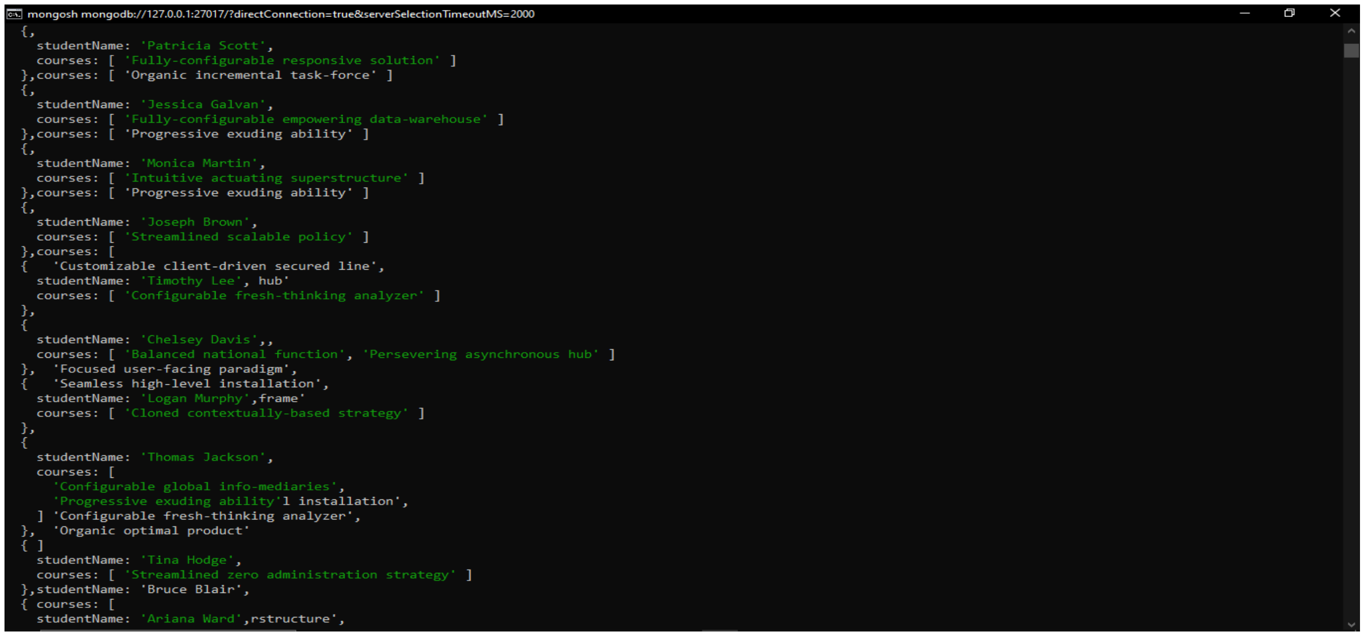
Q3. Write a query to show each student’s name along with the course titles they are enrolled in (use $lookup between enrollments, students, and courses)

Solution:

db.enrollments.aggregate([{$lookup: {from: "students" localField: "student_id",foreignField: "_id",as: "student"}},{ $unwind: "$student" },{$lookup: {from: "courses",localField: "course_id",foreignField: "_id",as: "course"}},{ $unwind: "$course"},{$group: {_id: "$student._id",studentName: { $first: "$student.name" },courses: { $push: "$course.title" }}},{ $project: { _id: 0, studentName: 1, courses: 1 } }]);



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Type "it" for more
collage> db.enrollments.aggregate([ { $lookup: { from: "students", localField: "student_id", foreignField: "_id", as: "student" } }, { $unwind: "$student" }, { $lookup:
 { from: "courses", localField: "course_id", foreignField: "_id", as: "course" } }, { $unwind: "$course" }, { $group: { _id: "$student._id", studentName: { $first: "$st
udent.name" }, courses: { $push: "$course.title" } } }, { $project: { _id: 0, studentName: 1, courses: 1 } }] );//prachi pathak - 1240258316
```



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
  {,
    studentName: 'Patricia Scott',
    courses: [ 'Fully-configurable responsive solution' ]
  },courses: [ 'Organic incremental task-force' ]
  {,
    studentName: 'Jessica Galvan',
    courses: [ 'Fully-configurable empowering data-warehouse' ]
  },courses: [ 'Progressive exuding ability' ]
  {,
    studentName: 'Monica Martin',
    courses: [ 'Intuitive actuating superstructure' ]
  },courses: [ 'Progressive exuding ability' ]
  {,
    studentName: 'Joseph Brown',
    courses: [ 'Streamlined scalable policy' ]
  },courses: [
  {   'Customizable client-driven secured line',
    studentName: 'Timothy Lee', hub'
    courses: [ 'Configurable fresh-thinking analyzer' ]
  },
  {
    studentName: 'Chelsey Davis',,
    courses: [ 'Balanced national function', 'Persevering asynchronous hub' ]
  },  'Focused user-facing paradigm',
  {   'Seamless high-level installation',
    studentName: 'Logan Murphy',frame'
    courses: [ 'Cloned contextually-based strategy' ]
  },
  {
    studentName: 'Thomas Jackson',
    courses: [
      'Configurable global info-mediaries',
      'Progressive exuding ability'l installation',
    ] 'Configurable fresh-thinking analyzer',
  },  'Organic optimal product'
  { ]
    studentName: 'Tina Hodge',
    courses: [ 'Streamlined zero administration strategy' ]
  },studentName: 'Bruce Blair',
  { courses: [
    studentName: 'Ariana Ward',rstructure',
```

Explanation:

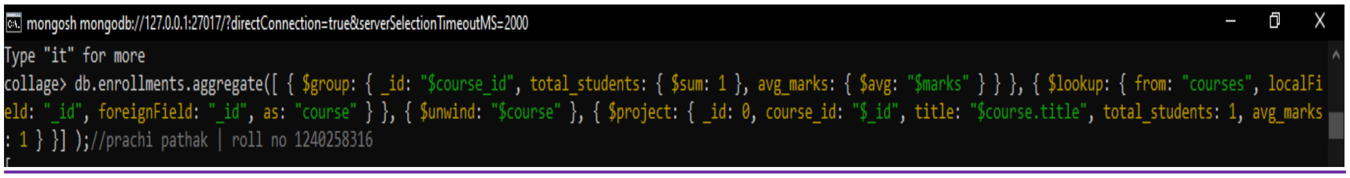
- Uses `$lookup` twice to join **students** and **courses** with **enrollments**.
- `$group` groups by each student, collecting all their enrolled course titles.
- `$project` shows each student name with their courses.

Result:
Displays each student’s name with all courses they are enrolled in.

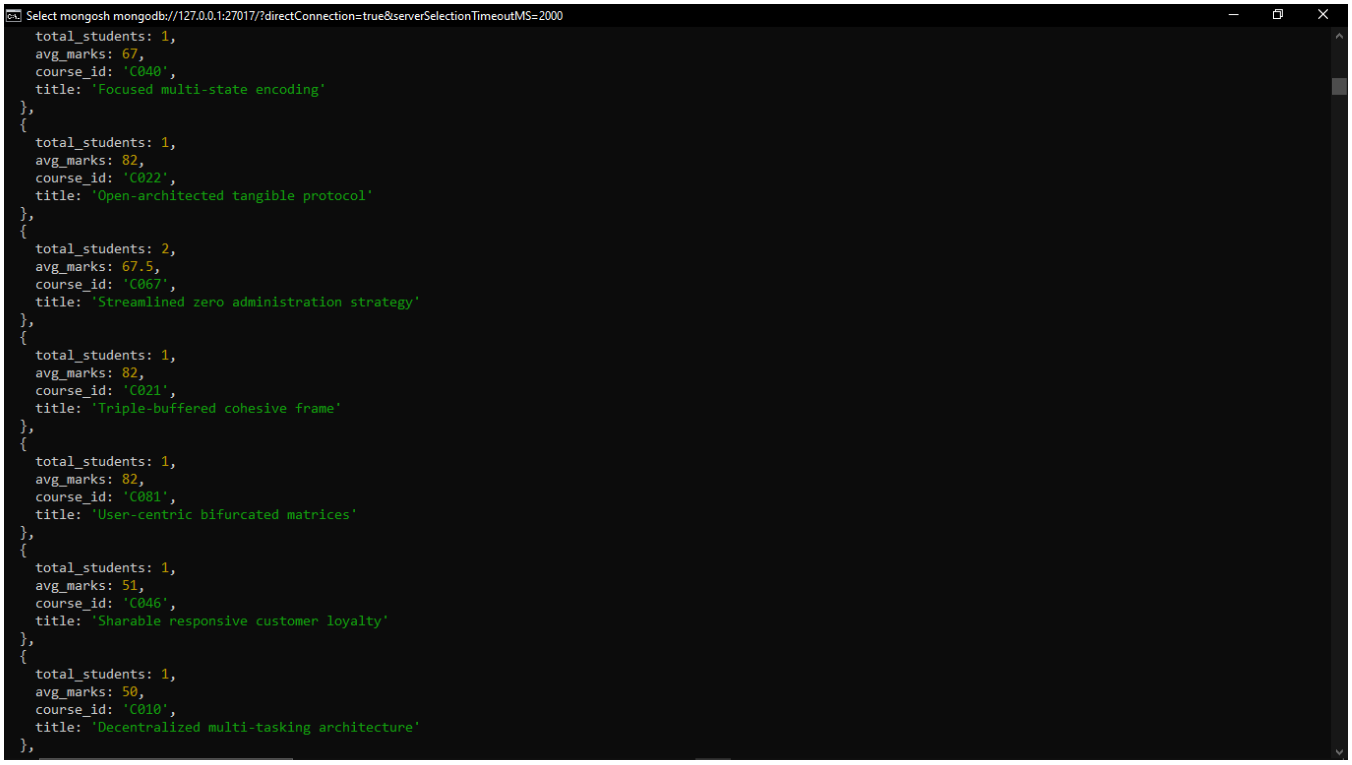
Q4. For each course, display the course title, number of students enrolled, and average marks (use $group).

Solution:

db.enrollments.aggregate([{ $group: { _id: "$course_id", total_students: { $sum: 1 }, avg_marks: { $avg: "$marks" } } }, { $lookup: { from: "courses", localField: "_id", foreignField: "_id", as: "course" } }, { $unwind: "$course" }, { $project: { _id: 0, course_id: "$_id", title: "$course.title", total_students: 1, avg_marks: 1 } }]);



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Type "it" for more
collage> db.enrollments.aggregate([ { $group: { _id: "$course_id", total_students: { $sum: 1 }, avg_marks: { $avg: "$marks" } } }, { $lookup: { from: "courses", localFi
eld: "_id", foreignField: "_id", as: "course" } }, { $unwind: "$course" }, { $project: { _id: 0, course_id: "$_id", title: "$course.title", total_students: 1, avg_marks
: 1 } }] );//prachi pathak | roll no 1240258316
```



```
Select mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
    total_students: 1,
    avg_marks: 67,
    course_id: 'C040',
    title: 'Focused multi-state encoding'
  },
  {
    total_students: 1,
    avg_marks: 82,
    course_id: 'C022',
    title: 'Open-architected tangible protocol'
  },
  {
    total_students: 2,
    avg_marks: 67.5,
    course_id: 'C067',
    title: 'Streamlined zero administration strategy'
  },
  {
    total_students: 1,
    avg_marks: 82,
    course_id: 'C021',
    title: 'Triple-buffered cohesive frame'
  },
  {
    total_students: 1,
    avg_marks: 82,
    course_id: 'C081',
    title: 'User-centric bifurcated matrices'
  },
  {
    total_students: 1,
    avg_marks: 51,
    course_id: 'C046',
    title: 'Sharable responsive customer loyalty'
  },
  {
    total_students: 1,
    avg_marks: 50,
    course_id: 'C010',
    title: 'Decentralized multi-tasking architecture'
  },
```

Explanation:

- `$group` calculates total students and average marks for each course.
- `$lookup` fetches the course title from the `courses` collection.
- `$project` formats the final output neatly.

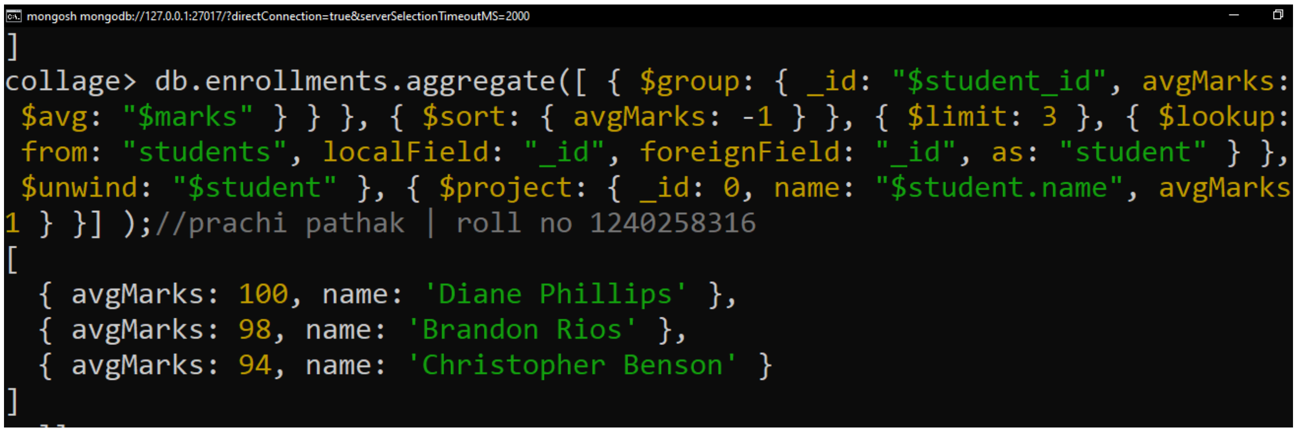
Result:
Shows course title, total enrolled students, and average marks

Grouping, Sorting, and Limiting

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

Solution:

([{ $group: { _id: "$student_id", avgMarks: { $avg: "$marks" } } }, { $sort: { avgMarks: -1 } }, { $limit: 3 }, { $lookup: { from: "students", localField: "_id", foreignField: "_id", as: "student" } }, { $unwind: "$student" }, { $project: { _id: 0, name: "$student.name", avgMarks: 1 } }]);


mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
]
collage> db.enrollments.aggregate([ { $group: { _id: "$student_id", avgMarks:
 $avg: "$marks" } } }, { $sort: { avgMarks: -1 } }, { $limit: 3 }, { $lookup:
 from: "students", localField: "_id", foreignField: "_id", as: "student" } },
 $unwind: "$student" }, { $project: { _id: 0, name: "$student.name", avgMarks
1 } }] );//prachi pathak | roll no 1240258316
[
  { avgMarks: 100, name: 'Diane Phillips' },
  { avgMarks: 98, name: 'Brandon Rios' },
  { avgMarks: 94, name: 'Christopher Benson' }
]
```

Explanation:

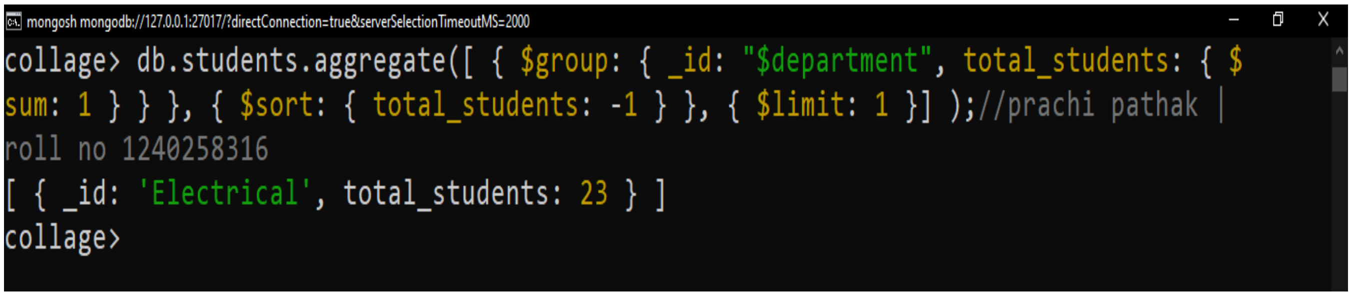
- `$group` computes average marks for each student.
- `$sort` arranges students in descending order.
- `$limit` keeps only top 3.
- `$lookup` retrieves student names.

Result:
Displays top 3 students with their average marks.

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

Solution:

db.students.aggregate([{ $group: { _id: "$department", total_students: { $sum: 1 } } }, { $sort: { total_students: -1 } }, { $limit: 1 }]);



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
collage> db.students.aggregate([ { $group: { _id: "$department", total_students: { $
sum: 1 } } }, { $sort: { total_students: -1 } }, { $limit: 1 }] );//prachi pathak |
roll no 1240258316
[ { _id: 'Electrical', total_students: 23 } ]
collage>
```

Explanation:

- `$group` counts students per department.
- `$sort` arranges departments by total students (descending).
- `$limit` returns the top one.

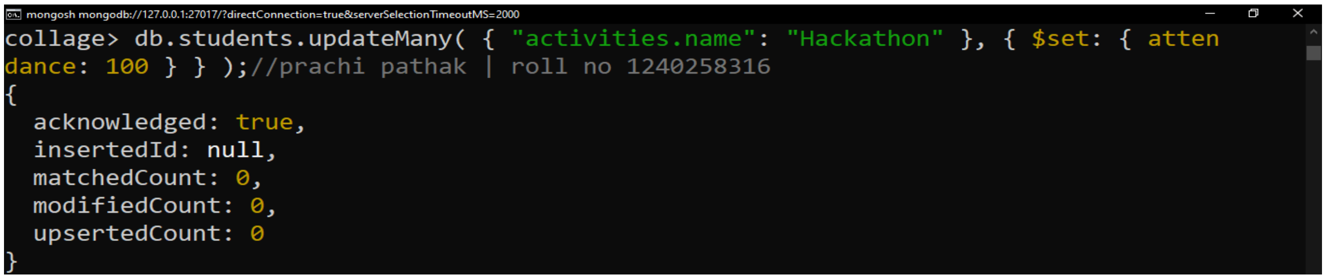
Result:
Shows the department having the most students.

Update, Upsert, and Delete

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

Solution:

db.students.updateMany({ "activities.name": "Hackathon" },{ $set: { attendance: 100 } });



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
collage> db.students.updateMany( { "activities.name": "Hackathon" }, { $set: { atten
dance: 100 } } );//prachi pathak | roll no 1240258316
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

Explanation:

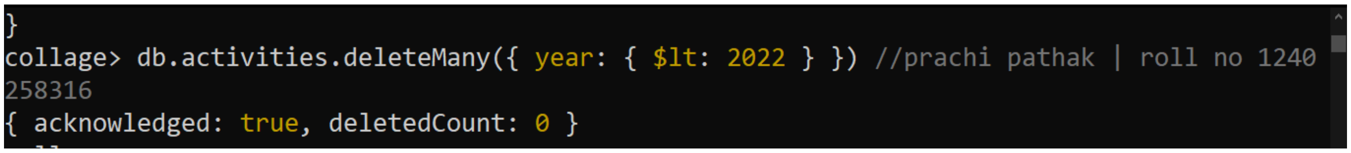
- Finds all students having an activity named “Hackathon”.
- `$set` updates their attendance to 100%.

Result:
All Hackathon winners’ attendance becomes 100%.

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

Solution:

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



```
}
collage> db.activities.deleteMany({ year: { $lt: 2022 } }) //prachi pathak | roll no 1240
258316
{ acknowledged: true, deletedCount: 0 }
```

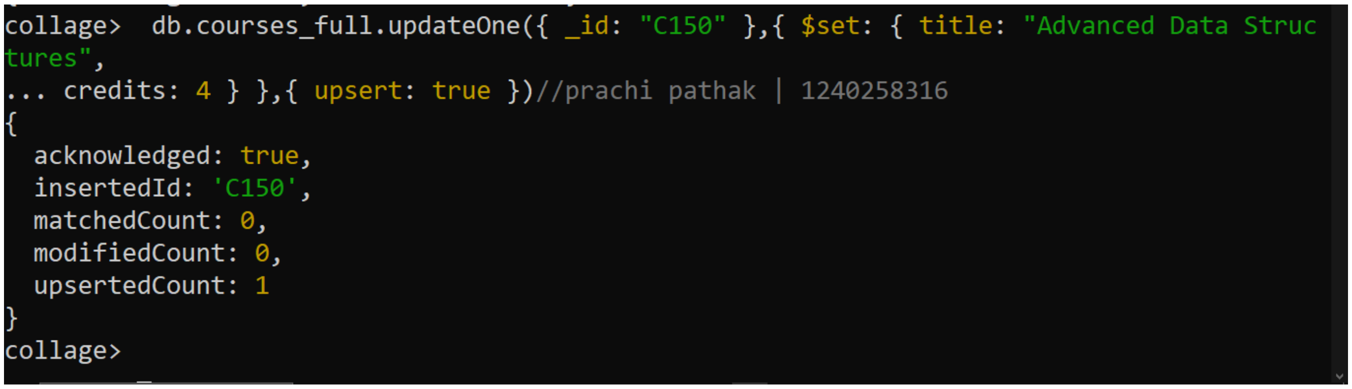
Explanation:

- $lt: 2022 → Finds activities before 2022(less than 2022)
- deleteMany → Removes all matching documents

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".

Solution:

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



```
collage>  db.courses_full.updateOne({ _id: "C150" },{ $set: { title: "Advanced Data Struc
tures",
... credits: 4 } },{ upsert: true })//prachi pathak | 1240258316
{
  acknowledged: true,
  insertedId: 'C150',
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
collage>
```

Explanation:

- Checks if a course with `_id: "C150"` exists.
- If yes → updates title and credits.
- If no → inserts a new course document.

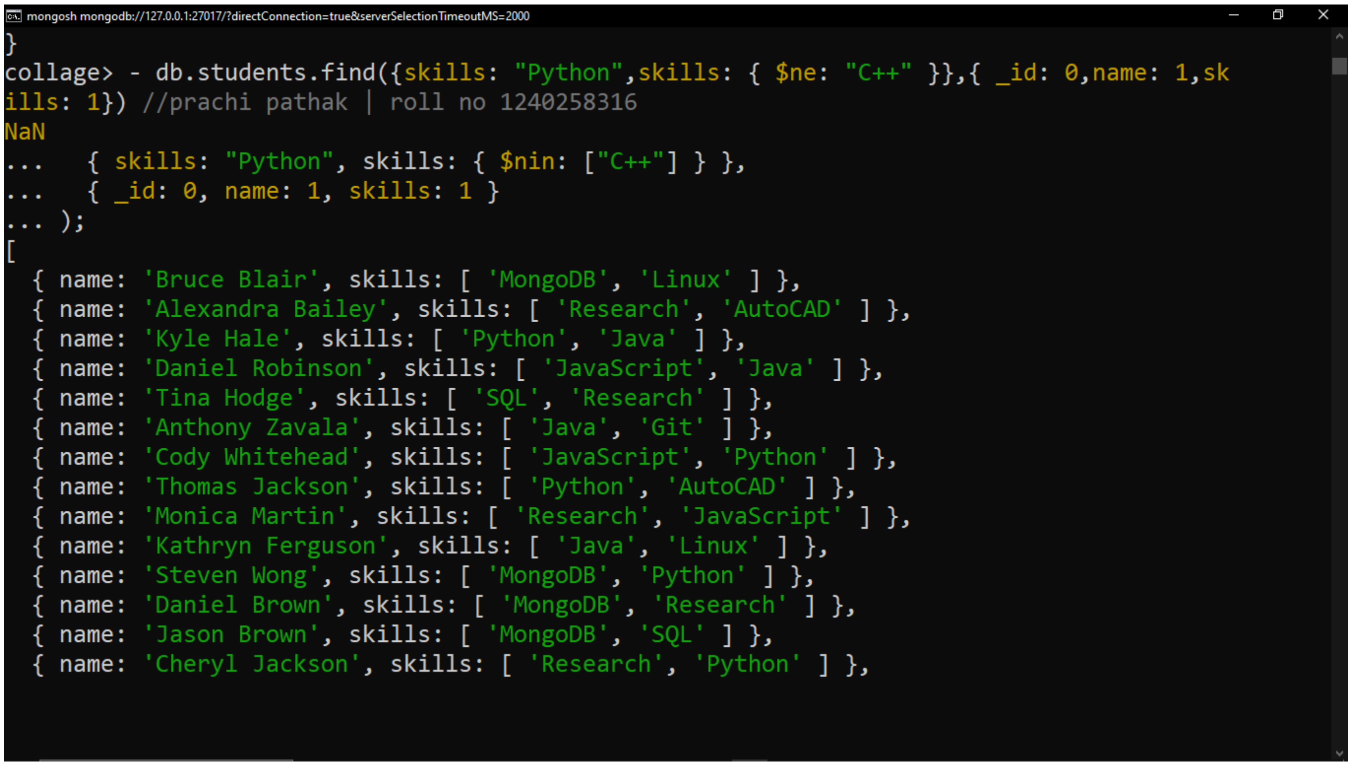
Result:
Ensures the course "Advanced Data Structures" exists in the collection.

Array & Operator Usage

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

Solution:

db.students.find({ skills: "Python", skills: { $nin: ["C++"] } },{ _id: 0, name: 1, skills: 1 });



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
}
collage> - db.students.find({skills: "Python",skills: { $ne: "C++" }},{ _id: 0,name: 1,sk
ills: 1}) //prachi pathak | roll no 1240258316
NaN
...   { skills: "Python", skills: { $nin: ["C++"] } },
...   { _id: 0, name: 1, skills: 1 }
... );
[
  { name: 'Bruce Blair', skills: [ 'MongoDB', 'Linux' ] },
  { name: 'Alexandra Bailey', skills: [ 'Research', 'AutoCAD' ] },
  { name: 'Kyle Hale', skills: [ 'Python', 'Java' ] },
  { name: 'Daniel Robinson', skills: [ 'JavaScript', 'Java' ] },
  { name: 'Tina Hodge', skills: [ 'SQL', 'Research' ] },
  { name: 'Anthony Zavala', skills: [ 'Java', 'Git' ] },
  { name: 'Cody Whitehead', skills: [ 'JavaScript', 'Python' ] },
  { name: 'Thomas Jackson', skills: [ 'Python', 'AutoCAD' ] },
  { name: 'Monica Martin', skills: [ 'Research', 'JavaScript' ] },
  { name: 'Kathryn Ferguson', skills: [ 'Java', 'Linux' ] },
  { name: 'Steven Wong', skills: [ 'MongoDB', 'Python' ] },
  { name: 'Daniel Brown', skills: [ 'MongoDB', 'Research' ] },
  { name: 'Jason Brown', skills: [ 'MongoDB', 'SQL' ] },
  { name: 'Cheryl Jackson', skills: [ 'Research', 'Python' ] },
```

Explanation:

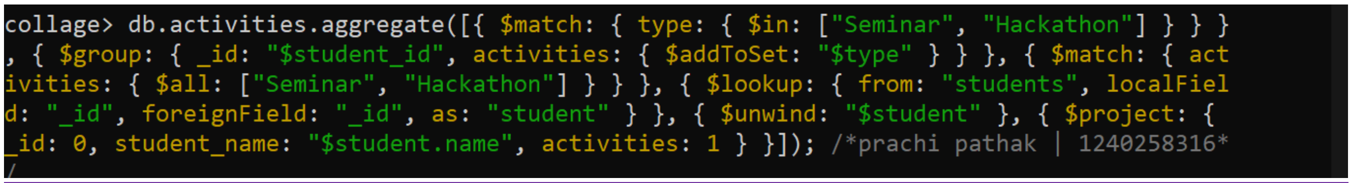
- `$nin` excludes students who have `"C++"` in their skills.
- Includes only those who have `"Python"`.

Result:
Lists students who know Python but not C++.

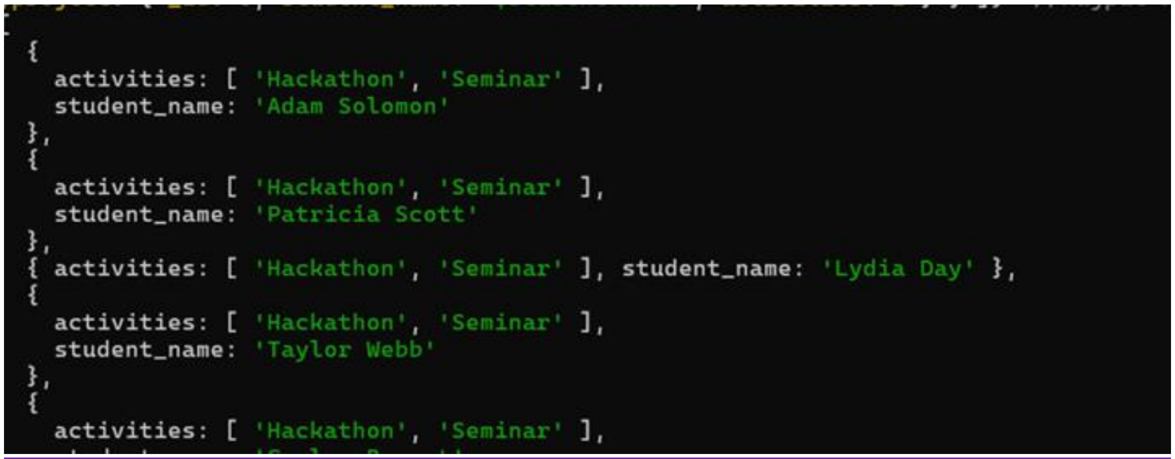
Q11. Return names of students who participated in "Seminar" and "Hackathon" both.

Solution:

db.activities.aggregate([{ $match: { type: { $in: ["Seminar", "Hackathon"] } } }, { $group: { _id: "$student_id", activities: { $addToSet: "$type" } } }, { $match: { activities: { $all: ["Seminar", "Hackathon"] } } }, { $lookup: { from: "students", localField: "_id", foreignField: "_id", as: "student" } }, { $unwind: "$student" }, { $project: { _id: 0, student_name: "$student.name", activities: 1 } }])



```
collage> db.activities.aggregate([{ $match: { type: { $in: ["Seminar", "Hackathon"] } } }
, { $group: { _id: "$student_id", activities: { $addToSet: "$type" } } }, { $match: { act
ivities: { $all: ["Seminar", "Hackathon"] } } }, { $lookup: { from: "students", localFiel
d: "_id", foreignField: "_id", as: "student" } }, { $unwind: "$student" }, { $project: {
_id: 0, student_name: "$student.name", activities: 1 } }]); /*prachi pathak | 1240258316*
/
```



```
  {
    activities: [ 'Hackathon', 'Seminar' ],
    student_name: 'Adam Solomon'
  },
  {
    activities: [ 'Hackathon', 'Seminar' ],
    student_name: 'Patricia Scott'
  },
  { activities: [ 'Hackathon', 'Seminar' ], student_name: 'Lydia Day' },
  {
    activities: [ 'Hackathon', 'Seminar' ],
    student_name: 'Taylor Webb'
  },
  {
    activities: [ 'Hackathon', 'Seminar' ],
```

Explanation:

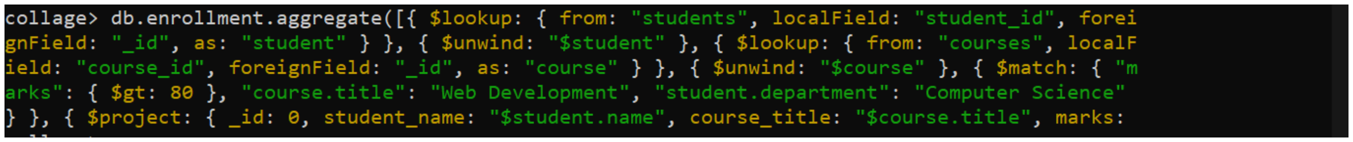
- $group → Groups by student and collects unique activity types.
- $match → Keeps only students who have both activities.
- $lookup → Gets student names from students.
- $project → Shows only student name and their activities.

Subdocuments and Nested Conditions

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

Solution:

db.enrollment.aggregate([{$lookup: {from: "students",localField:"student_id",foreignField: "_id",as: "student" }},{ $unwind: "$student" },{$lookup: {from: "courses",localField: "course_id",foreignField: "_id",as: "course"}},{ $unwind: "$course" },{$match: {"marks": { $gt: 80 },"course.title": "Web Development","student.department": "Computer Science"}},{$project: { _id: 0,student_name: "$student.name",course_title: "$course.title",marks: 1,department: "$student.department"}}])



```
collage> db.enrollment.aggregate([{ $lookup: { from: "students", localField: "student_id", forei
gnField: "_id", as: "student" } }, { $unwind: "$student" }, { $lookup: { from: "courses", localF
ield: "course_id", foreignField: "_id", as: "course" } }, { $unwind: "$course" }, { $match: { "m
arks": { $gt: 80 }, "course.title": "Web Development", "student.department": "Computer Science"
} }, { $project: { _id: 0, student_name: "$student.name", course_title: "$course.title", marks:
```

Explanation:

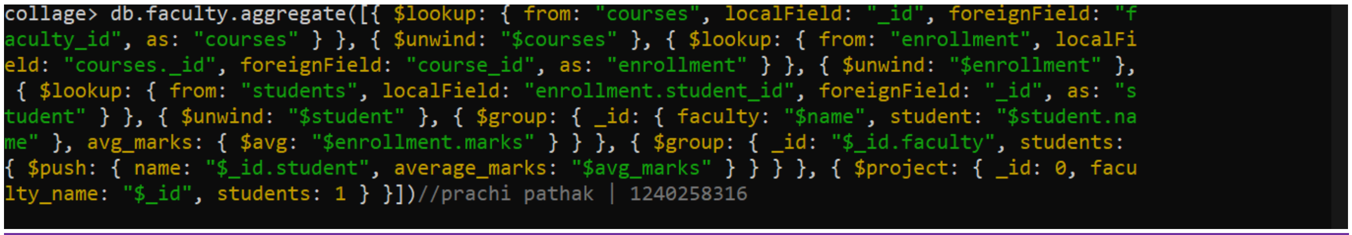
- $lookup → Joins enrollments with students_full and courses_full.
- $match → Filters students with marks >80 in "Web Development" and in "Computer Science".
- $project → Shows student name, course title, marks, and department.

Advanced Aggregation (Challenge Level)

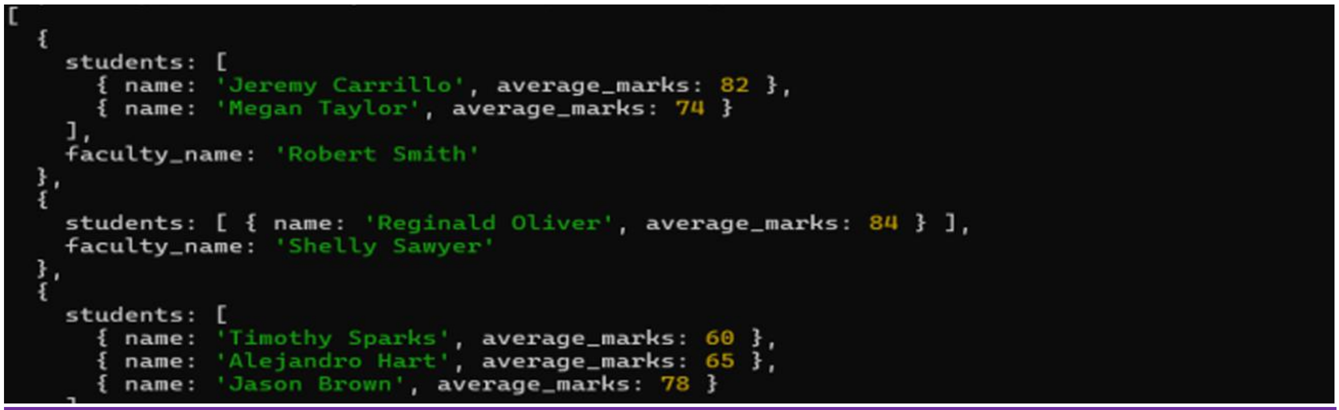
Q13. For each faculty member, list the names of all students enrolled in their courses along with average marks per student per faculty.

Solution:

db.faculty.aggregate([{$lookup:{from:"courses",localField:"_id",foreignField:"faculty_id",as: "courses"}},{$unwind:"$courses"},{$lookup:{from:"enrollment",localField:"courses._id",foreignField: "course_id",as: "enrollment" }},{ $unwind: "$enrollment" },{$lookup: {from: "students",localField: "enrollment.student_id", foreignField:"_id", as: "student"}},{ $unwind: "$student" },{$group: { _id: { faculty: "$name", student: "$student.name" }, avg_marks: { $avg: "$enrollment.marks" }}}, {$group: { _id:"$_id.faculty",students: { $push: { name: "$_id.student",average_marks: "$avg_marks" }}}},{$project: {_id: 0,faculty_name: "$_id", students: 1}}])



```
collage> db.faculty.aggregate([{ $lookup: { from: "courses", localField: "_id", foreignField: "f
aculty_id", as: "courses" } }, { $unwind: "$courses" }, { $lookup: { from: "enrollment", localFi
eld: "courses._id", foreignField: "course_id", as: "enrollment" } }, { $unwind: "$enrollment" },
 { $lookup: { from: "students", localField: "enrollment.student_id", foreignField: "_id", as: "s
tudent" } }, { $unwind: "$student" }, { $group: { _id: { faculty: "$name", student: "$student.na
me" }, avg_marks: { $avg: "$enrollment.marks" } } }, { $group: { _id: "$_id.faculty", students:
{ $push: { name: "$_id.student", average_marks: "$avg_marks" } } } }, { $project: { _id: 0, facu
lty_name: "$_id", students: 1 } }])//prachi pathak | 1240258316
```



```
[
  {
    students: [
      { name: 'Jeremy Carrillo', average_marks: 82 },
      { name: 'Megan Taylor', average_marks: 74 }
    ],
    faculty_name: 'Robert Smith'
  },
  {
    students: [ { name: 'Reginald Oliver', average_marks: 84 } ],
    faculty_name: 'Shelly Sawyer'
  },
  {
    students: [
      { name: 'Timothy Sparks', average_marks: 60 },
      { name: 'Alejandro Hart', average_marks: 65 },
      { name: 'Jason Brown', average_marks: 78 }
```

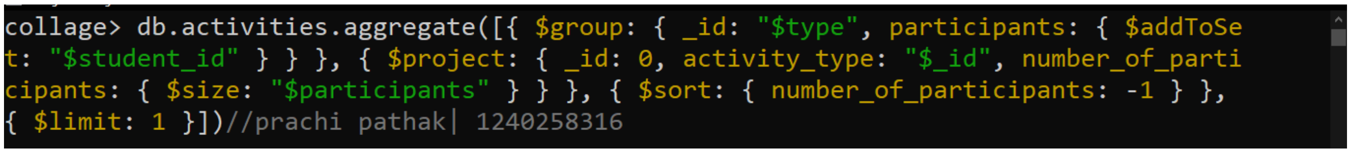
Explanation:

- Joins courses_full → enrollments_full → students_full.
- $group → Groups by faculty ID and collects all student names and their average marks.
- $lookup → Fetches faculty names.
- $round → Rounds average marks to 2 decimals.

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

Solution:

db.activities.aggregate([{ $group: { _id: "$type", participants: { $addToSet: "$student_id" } } }, { $project: { _id: 0, activity_type: "$_id", number_of_participants: { $size: "$participants" } } }, { $sort: { number_of_participants: -1 } }, { $limit: 1 }])



```
collage> db.activities.aggregate([{ $group: { _id: "$type", participants: { $addToSe
t: "$student_id" } } }, { $project: { _id: 0, activity_type: "$_id", number_of_parti
cipants: { $size: "$participants" } } }, { $sort: { number_of_participants: -1 } },
{ $limit: 1 }])//prachi pathak| 1240258316
```

Explanation:

- $group → Groups activities by type and collects unique student IDs.
- $size → Counts number of participants per activity type.
- $sort → Orders in descending order.
- $limit: 1 → Returns most popular activity.