BABU BANARSI DAS UNIVERSITY

NO SQL and Dbaas

(BCADSN13202)

PROJECT

SUBMITTED TO:

Mr. Ankit Verma

SUBMITTED BY:

Nistha Gupta
(1240258301)
BCADS25

PROJECT

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

Solution: -

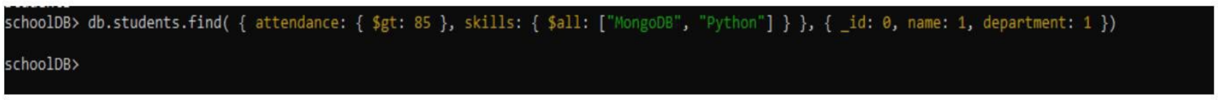
db.students.find({attendance: { $gt: 85 },skills: { $all: ["MongoDB", "Python"] }},{_id: 0,name: 1,department: 1})

Explanation: -

• attendance: { $gt: 85 } → means attendance greater than 85.

• $all → checks that both skills "MongoDB" and "Python" are present.

• Projection { name: 1, department: 1 } → shows only name and department. Output: No record found.



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

schoolDB>
```

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

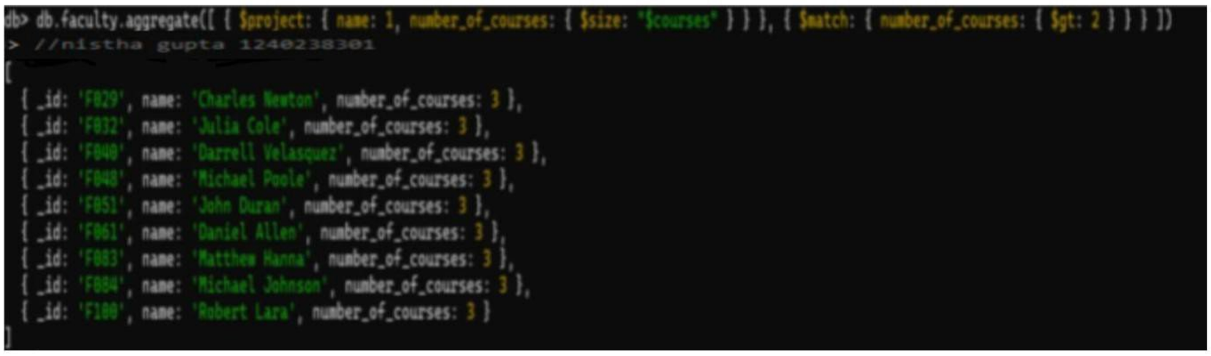
Solution: - db.faculty.aggregate([{ $project: { name: 1, number_of_courses: { $size: "$courses" } } }, { $match: { number_of_courses: { $gt: 2 } } }])

Explanation: -

• $size → counts how many items are in the "courses" array.

• $match → filters only those who teach more than 2 courses.

Output:



```
db> db.faculty.aggregate([ { $project: { name: 1, number_of_courses: { $size: "$courses" } } }, { $match: { number_of_courses: { $gt: 2 } } } ])
> //nistha gupta 1240238301
[
  { _id: 'F029', name: 'Charles Newton', number_of_courses: 3 },
  { _id: 'F032', name: 'Julia Cole', number_of_courses: 3 },
  { _id: 'F040', name: 'Darrell Velasquez', number_of_courses: 3 },
  { _id: 'F048', name: 'Michael Poole', number_of_courses: 3 },
  { _id: 'F051', name: 'John Duran', number_of_courses: 3 },
  { _id: 'F061', name: 'Daniel Allen', number_of_courses: 3 },
  { _id: 'F083', name: 'Matthew Hanna', number_of_courses: 3 },
  { _id: 'F084', name: 'Michael Johnson', number_of_courses: 3 },
  { _id: 'F100', name: 'Robert Lara', number_of_courses: 3 }
]
```

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: -

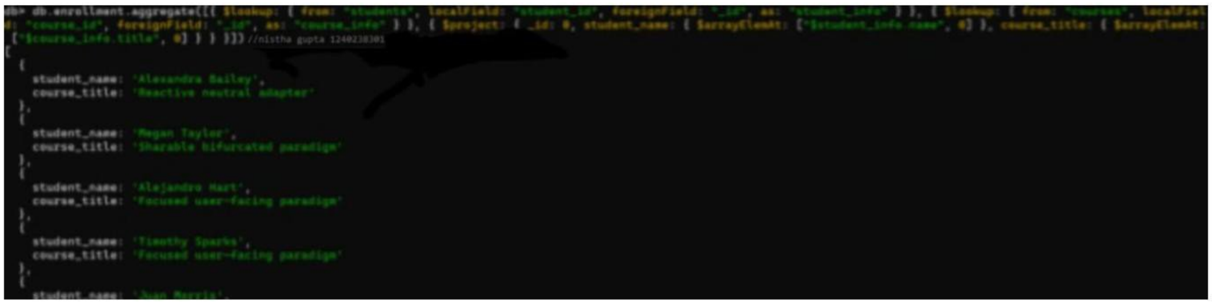
db.enrollment.aggregate([{$lookup: {from: "students",localField: "student_id",foreignField: "_id",as: "student"}},{ $unwind: "$student" },{$lookup: {from: "courses",localField: "course_id",foreignField: "_id",as: "course"}},{ $unwind: "$course" },{$project: {_id: 0,student_name: "$student.name",course_title: "$course.title"}}])

Explanation: -

• $lookup → joins collections (like SQL JOIN).

• $unwind → flattens joined results.

• $project → shows only student name and course title.



```
db> db.enrollment.aggregate([{ $lookup: { from: "students", localField: "student_id", foreignField: "_id", as: "student_info" } }, { $lookup: { from: "courses", localFiel
d: "course_id", foreignField: "_id", as: "course_info" } }, { $project: { _id: 0, student_name: { $arrayElemAt: ["$student_info.name", 0] }, course_title: { $arrayElemAt:
 ["$course_info.title", 0] } } }]) //nistha gupta 1240238301
[
  {
    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',
```

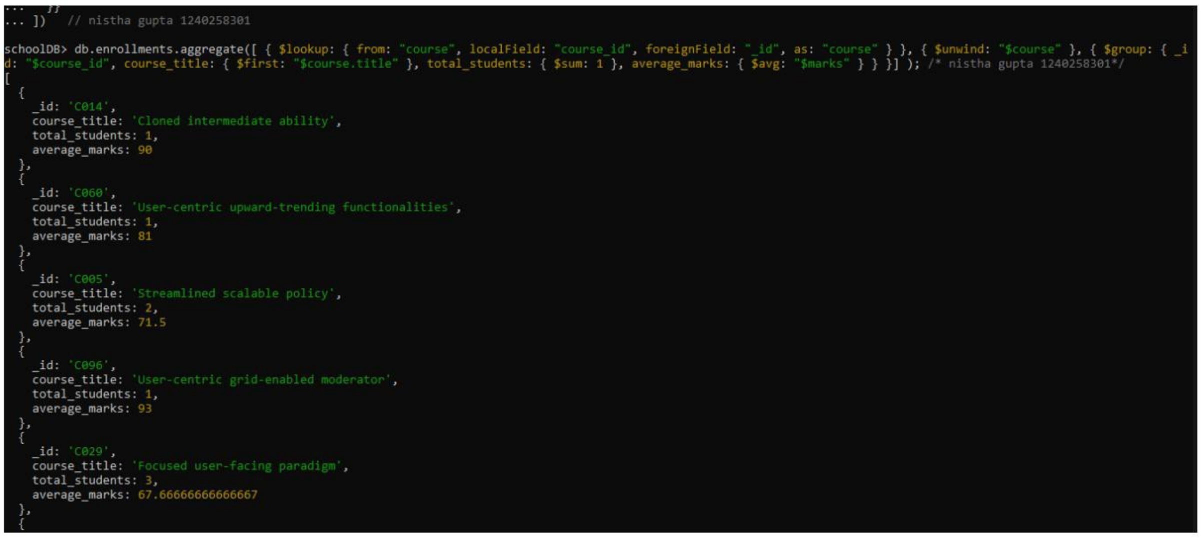
Q4. For each course, display the course title, number of students enrolled, and average marks (use $group).

Solution: -

db.enrollment.aggregate([{ $lookup: { from: "courses", localField: "course_id", foreignField: "_id", as: "course" } }, { $unwind: "$course" }, { $group: { _id: "$course_id", course_title: { $first: "$course.title" }, total_students: { $sum: 1 }, average_marks: { $avg: "$marks" } } }])

Explanation: -

- $lookup → connects enrollments with courses.
- $group → groups all students by course.
- $sum: 1 → counts students.
- $avg → calculates average marks.



```
... ])   // nistha gupta 1240258301

schoolDB> db.enrollments.aggregate([ { $lookup: { from: "course", localField: "course_id", foreignField: "_id", as: "course" } }, { $unwind: "$course" }, { $group: { _i
d: "$course_id", course_title: { $first: "$course.title" }, total_students: { $sum: 1 }, average_marks: { $avg: "$marks" } } }] ); /* nistha gupta 1240258301*/
[
  {
    _id: 'C014',
    course_title: 'Cloned intermediate ability',
    total_students: 1,
    average_marks: 90
  },
  {
    _id: 'C060',
    course_title: 'User-centric upward-trending functionalities',
    total_students: 1,
    average_marks: 81
  },
  {
    _id: 'C005',
    course_title: 'Streamlined scalable policy',
    total_students: 2,
    average_marks: 71.5
  },
  {
    _id: 'C096',
    course_title: 'User-centric grid-enabled moderator',
    total_students: 1,
    average_marks: 93
  },
  {
    _id: 'C029',
    course_title: 'Focused user-facing paradigm',
    total_students: 3,
    average_marks: 67.66666666666667
  },
  {
```

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

Solution:

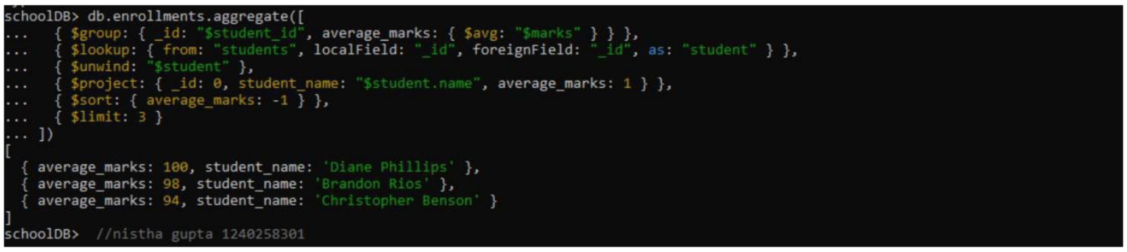
db.enrollment.aggregate([{ $group: { _id: "$student_id", average_marks: { $avg: "$marks" } } }, { $lookup: { from: "students", localField: "_id", foreignField: "_id", as: "student" } }, { $unwind: "$student" }, { $project: { _id: 0, student_name: "$student.name", average_marks: 1 } }, { $sort: { average_marks: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

Explanation: -

- $group → Groups by student_id and calculates the average marks.
- $lookup → Joins with students_full to get student names.
- $sort → Orders by average_marks in descending order.

• $limit → Shows only top 3 students.

Output: -



```
schoolDB> db.enrollments.aggregate([
...   { $group: { _id: "$student_id", average_marks: { $avg: "$marks" } } },
...   { $lookup: { from: "students", localField: "_id", foreignField: "_id", as: "student" } },
...   { $unwind: "$student" },
...   { $project: { _id: 0, student_name: "$student.name", average_marks: 1 } },
...   { $sort: { average_marks: -1 } },
...   { $limit: 3 }
... ])
[
  { average_marks: 100, student_name: 'Diane Phillips' },
  { average_marks: 98, student_name: 'Brandon Rios' },
  { average_marks: 94, student_name: 'Christopher Benson' }
]
schoolDB>  //nistha gupta 1240258301
```

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

Solution: -

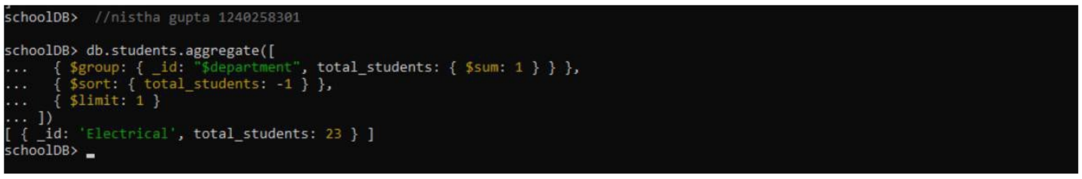
db.students.aggregate([{ $group: { _id: "$department", total_students: { $sum: 1 } } }, { $sort: { total_students: -1 } }, { $limit: 1 }])

Explanation: -

• $group → Groups students by department and counts them.

• $sort → Orders by student count in descending order.

• $limit: 1 → Shows only the department with the highest number of students. Output



```
schoolDB>  //nistha gupta 1240258301

schoolDB> db.students.aggregate([
...   { $group: { _id: "$department", total_students: { $sum: 1 } } },
...   { $sort: { total_students: -1 } },
...   { $limit: 1 }
... ])
[ { _id: 'Electrical', total_students: 23 } ]
schoolDB> 
```

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

Solution: -

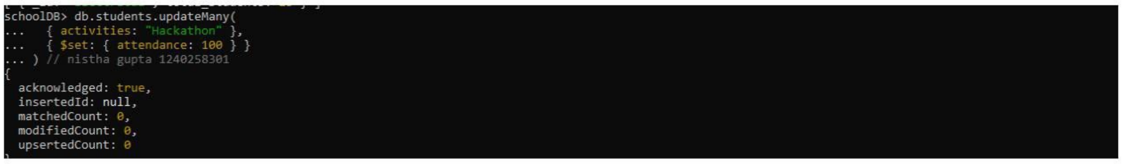
db.students.updateMany({activities : "Hackathon "}, { $set : { attendance : 100 }})

Explanation: -

• $set - is used to add a new field or update an existing field in documents.

• updateMany() -is used to update multiple documents in a collection that match a given condition.

Output:-



```
schoolDB> db.students.updateMany(
...   { activities: "Hackathon" },
...   { $set: { attendance: 100 } }
... ) // nistha gupta 1240258301
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
```

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

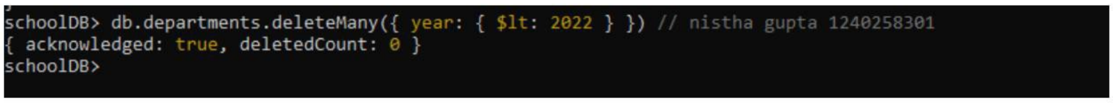
Solution: -

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

Explanation: -

• $lt: 2022 → Finds activities before 2022(less than 2022)

• deleteMany → Removes all matching documents



```
schoolDB> db.departments.deleteMany({ year: { $lt: 2022 } }) // nistha gupta 1240258301
{ acknowledged: true, deletedCount: 0 }
schoolDB>
```

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 })

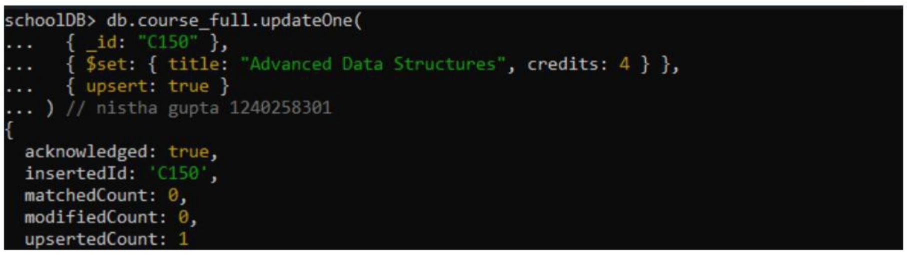
Explanation: -

• updateOne → Updates a single document.

• $set → Updates title and credits.

• upsert: true → If _id: "C150" doesn't exist, it will insert a new document.

Output:-



```
schoolDB> db.course_full.updateOne(
...    { _id: "C150" },
...    { $set: { title: "Advanced Data Structures", credits: 4 } },
...    { upsert: true }
... ) // nistha gupta 1240258301
{
  acknowledged: true,
  insertedId: 'C150',
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
```

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

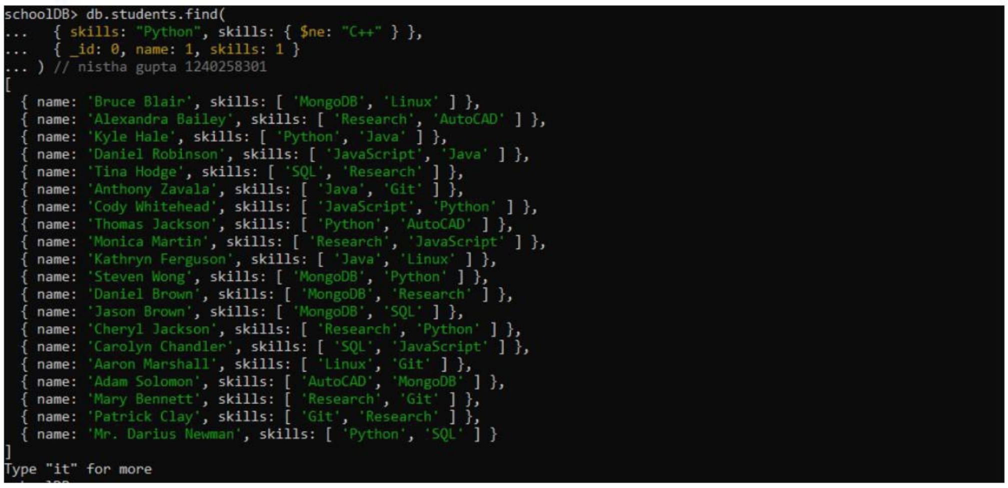
Solution: -

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

Explanation: -

- skills: "Python" → Checks that the array includes "Python".
- skills: { $ne: "C++" } → Ensures "C++" is not in the array.
- Projection { name: 1, skills: 1 } → Shows only names and skills.

Output:



```
schoolDB> db.students.find(
...   { skills: "Python", skills: { $ne: "C++" } },
...   { _id: 0, name: 1, skills: 1 }
... ) // nistha gupta 1240258301
[
  { 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' ] },
  { name: 'Carolyn Chandler', skills: [ 'SQL', 'JavaScript' ] },
  { name: 'Aaron Marshall', skills: [ 'Linux', 'Git' ] },
  { name: 'Adam Solomon', skills: [ 'AutoCAD', 'MongoDB' ] },
  { name: 'Mary Bennett', skills: [ 'Research', 'Git' ] },
  { name: 'Patrick Clay', skills: [ 'Git', 'Research' ] },
  { name: 'Mr. Darius Newman', skills: [ 'Python', 'SQL' ] }
]
Type "it" for more
```

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

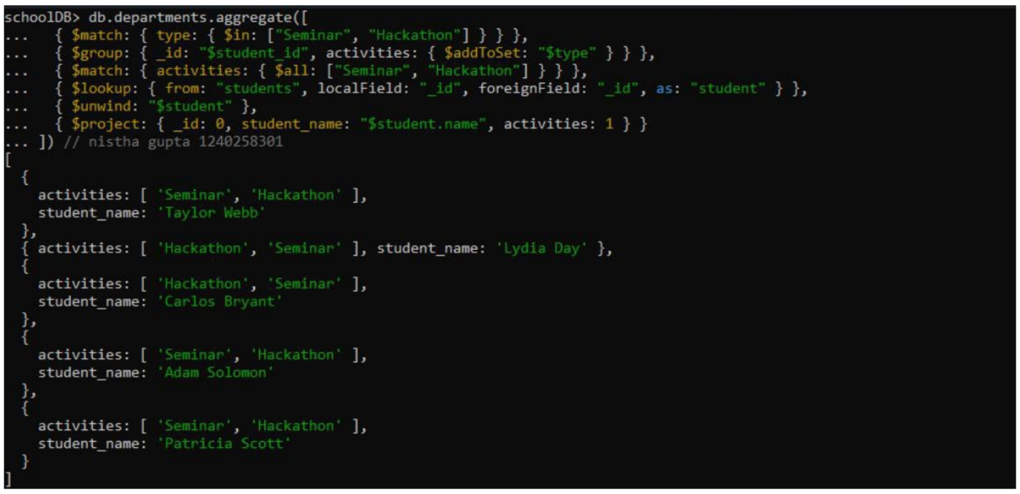
Solution: -

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 } }])

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.

Output:



```
schoolDB> db.departments.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 } }
... ]) // nistha gupta 1240258301
[
  {
    activities: [ 'Seminar', 'Hackathon' ],
    student_name: 'Taylor Webb'
  },
  { activities: [ 'Hackathon', 'Seminar' ], student_name: 'Lydia Day' },
  {
    activities: [ 'Hackathon', 'Seminar' ],
    student_name: 'Carlos Bryant'
  },
  {
    activities: [ 'Seminar', 'Hackathon' ],
    student_name: 'Adam Solomon'
  },
  {
    activities: [ 'Seminar', 'Hackathon' ],
    student_name: 'Patricia Scott'
  }
]
```

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"}}])

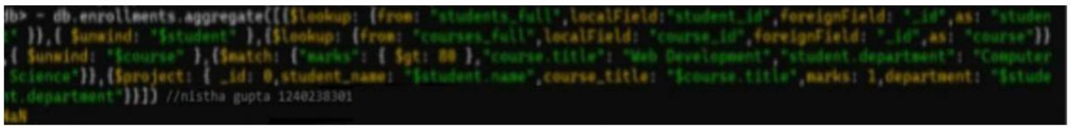
Explanation: -

• $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.

Output:



```
db> - db.enrollments.aggregate([{$lookup: {from: "students_full",localField:"student_id",foreignField: "_id",as: "studen
t" }},{ $unwind: "$student" },{$lookup: {from: "courses_full",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: "$stude
nt.department"}}]) //nistha gupta 1240238301
NaN
```

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}}])

Explanation: -

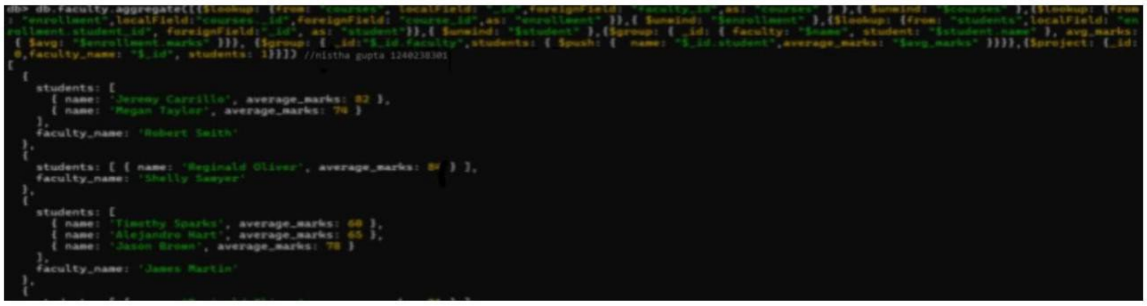
• Joins courses_full → enrollments_full → students_full.

• $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.

Output: -



```
db> 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: "en
rollment.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}}]) //nistha gupta 1240238301
[
  {
    students: [
      { name: 'Jeremy Carrillo', average_marks: 82 },
      { name: 'Megan Taylor', average_marks: 70 }
    ],
    faculty_name: 'Robert Smith'
  },
  {
    students: [ { name: 'Reginald Oliver', average_marks: 8[illegible] } ],
    faculty_name: 'Shelly Sawyer'
  },
  {
    students: [
      { name: 'Timothy Sparks', average_marks: 60 },
      { name: 'Alejandro Hart', average_marks: 65 },
      { name: 'Jason Brown', average_marks: 78 }
    ],
    faculty_name: 'James Martin'
  },
  {
```

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 }])

Explanation: -

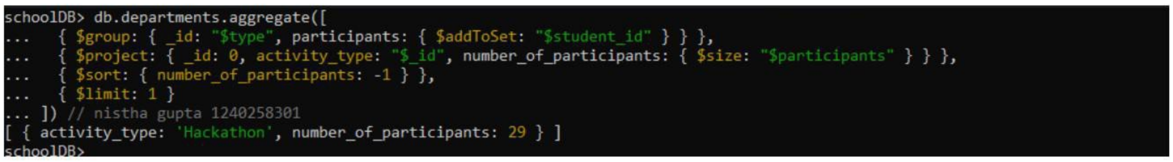
• $group → Groups activities by type and collects unique student IDs.

• $size → Counts number of participants per activity type.

• $sort → Orders in descending order.

• $limit: 1 → Returns most popular activity.

Output: -



```
schoolDB> db.departments.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 }
... ]) // nistha gupta 1240258301
[ { activity_type: 'Hackathon', number_of_participants: 29 } ]
schoolDB>
```