Assessment Task 1

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".

Output:

a

```
collegeDB> db.students.find(
...   { attendance: { $gt: 85 }, skills: { $all: ["MongoDB", "Python"] } },
...   { name: 1, department: 1, _id: 0 }
... );
... //Name: MD. Arman, Rollno:1240258257
```

- find() → Retrieves documents from a collection based on conditions.
- $gt → “Greater than” comparison operator.
- $all → Matches arrays that contain all specified elements.

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

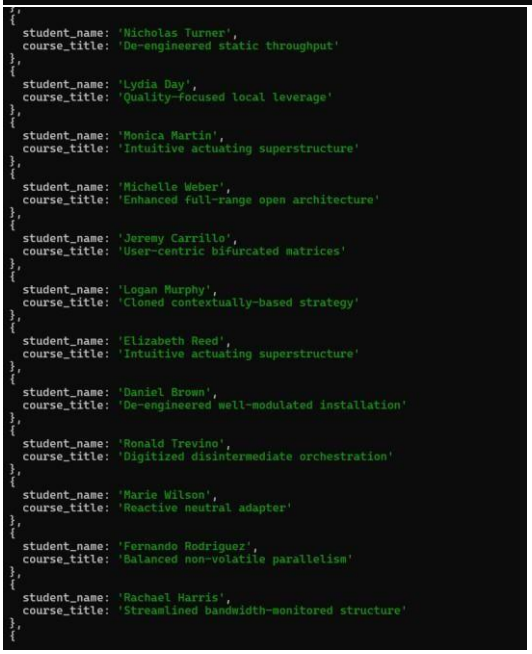
```
collegeDB> db.faculty.aggregate([
...   {
...     $project: {
...       name: 1,
...       totalCourses: { $size: "$courses" } // Count number of courses
...     }
...   },
...   {
...     $match: {
...       totalCourses: { $gt: 2 } // Only faculty teaching more than 2 courses
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       name: 1,
...       totalCourses: 1
...     }
...   }
... ]);
... //Name: MD. Arman, Rollno:1240258257
[
  { name: 'Charles Newton', totalCourses: 3 },
  { name: 'Julia Cole', totalCourses: 3 },
  { name: 'Darrell Velasquez', totalCourses: 3 },
  { name: 'Michael Poole', totalCourses: 3 },
  { name: 'John Duran', totalCourses: 3 },
  { name: 'Daniel Allen', totalCourses: 3 },
  { name: 'Matthew Hanna', totalCourses: 3 },
  { name: 'Michael Johnson', totalCourses: 3 },
  { name: 'Robert Lara', totalCourses: 3 }
]
```

- $size: "$courses" → counts how many courses each faculty teaches
- Use $match to filter faculty members who are teaching more than 2 courses, using total Courses: { $gt: 2 }.
- Use $project to display only the fields needed in output: name and totalCourses, hide _id.

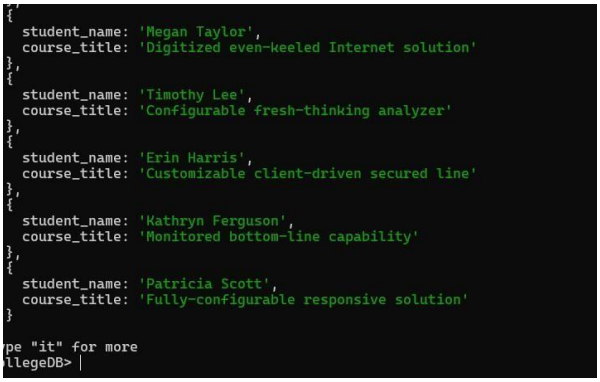
2. Joins ($lookup) and Aggregations

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).

```
collegeDB> db.enrollments.aggregate([
...   { $lookup: { from: "students", localField: "student_id", foreignField: "student_id", as: "student_info" } },
...   { $unwind: "$student_info" },
...   { $lookup: { from: "courses", localField: "course_id", foreignField: "course_id", as: "course_info" } },
...   { $unwind: "$course_info" },
...   { $project: { _id: 0, student_name: "$student_info.name", course_title: "$course_info.title" } }
... ]);
...   //Name: MD. Arman, Rollno:1240258257
```



```
[
  {
    student_name: 'Nicholas Turner',
    course_title: 'De-engineered static throughput'
  },
  {
    student_name: 'Lydia Day',
    course_title: 'Quality-focused local leverage'
  },
  {
    student_name: 'Monica Martin',
    course_title: 'Intuitive actuating superstructure'
  },
  {
    student_name: 'Michelle Weber',
    course_title: 'Enhanced full-range open architecture'
  },
  {
    student_name: 'Jeremy Carrillo',
    course_title: 'User-centric bifurcated matrices'
  },
  {
    student_name: 'Logan Murphy',
    course_title: 'Cloned contextually-based strategy'
  },
  {
    student_name: 'Elizabeth Reed',
    course_title: 'Intuitive actuating superstructure'
  },
  {
    student_name: 'Daniel Brown',
    course_title: 'De-engineered well-modulated installation'
  },
  {
    student_name: 'Ronald Trevino',
    course_title: 'Digitized disintermediate orchestration'
  },
  {
    student_name: 'Marie Wilson',
    course_title: 'Reactive neutral adapter'
  },
  {
    student_name: 'Fernando Rodriguez',
    course_title: 'Balanced non-volatile parallelism'
  },
  {
    student_name: 'Rachael Harris',
    course_title: 'Streamlined bandwidth-monitored structure'
  },
  {
```



```
{
  student_name: 'Megan Taylor',
  course_title: 'Digitized even-keeled Internet solution'
},
{
  student_name: 'Timothy Lee',
  course_title: 'Configurable fresh-thinking analyzer'
},
{
  student_name: 'Erin Harris',
  course_title: 'Customizable client-driven secured line'
},
{
  student_name: 'Kathryn Ferguson',
  course_title: 'Monitored bottom-line capability'
},
{
  student_name: 'Patricia Scott',
  course_title: 'Fully-configurable responsive solution'
}

pe "it" for more
llegeDB>
```

- Use $project to show only what we need:
- Student's name → from student_info.name •
- Course title → from course_info.title
- Hide _id so output looks clean.

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

```
collegeDB> db.enrollments.aggregate([
...   {
...     $group: {
...       _id: "$course_id",                    // Group enrollments by course_id
...       totalStudents: { $sum: 1 },           // Count how many students per course
...       avgMarks: { $avg: "$marks" }          // Calculate average marks
...     }
...   },
...   {
...     $lookup: {
...       from: "courses",                      // Join with courses collection
...       localField: "_id",                    // course_id from enrollments
...       foreignField: "_id",                  // _id from courses
...       as: "course_info"
...     }
...   },
...   { $unwind: "$course_info" },
...   {
...     $project: {
...       _id: 0,
...       course_title: "$course_info.title",
...       totalStudents: 1,
...       avgMarks: 1
...     }
...   }
... ]);
... //Name: MD. Arman, Rollno:1240258257
[
  {
    totalStudents: 1,
    avgMarks: 75,
    course_title: 'Organic incremental task-force'
  },
  {
    totalStudents: 3,
    avgMarks: 84,
    course_title: 'Progressive exuding ability'
  },
  {
    totalStudents: 2,
    avgMarks: 80.5,
    course_title: 'Configurable fresh-thinking analyzer'
  },
  {
    totalStudents: 2,
```

```
},
{
  totalStudents: 2,
  avgMarks: 80.5,
  course_title: 'Configurable fresh-thinking analyzer'
},
{
  totalStudents: 2,
  avgMarks: 76.5,
  course_title: 'Customer-focused cohesive info-mediaries'
},
{
  totalStudents: 1,
  avgMarks: 90,
  course_title: 'Cloned intermediate ability'
},
{
  totalStudents: 1,
  avgMarks: 67,
  course_title: 'Switchable regional open system'
},
{
  totalStudents: 2,
  avgMarks: 77.5,
  course_title: 'Streamlined bandwidth-monitored structure'
},
{
  totalStudents: 2,
  avgMarks: 67.5,
  course_title: 'Streamlined zero administration strategy'
},
{
  totalStudents: 1,
  avgMarks: 91,
  course_title: 'Decentralized multimedia Local Area Network'
},
{
  totalStudents: 1,
  avgMarks: 78,
  course_title: 'Enhanced intangible emulation'
},
{
  totalStudents: 1,
  avgMarks: 78,
  course_title: 'Balanced asynchronous framework'
},
```

```
  avgMarks: 87.6666666666667,
  course_title: 'Focused user-facing paradigm'
},
{
  totalStudents: 1,
  avgMarks: 68,
  course_title: 'Organic asynchronous matrix'
},
{
  totalStudents: 1,
  avgMarks: 97,
  course_title: 'Triple-buffered eco-centric implementatio
},
{
  totalStudents: 2,
  avgMarks: 58.5,
  course_title: 'Profit-focused high-level capability'
},
{
  totalStudents: 4,
  avgMarks: 82.5,
  course_title: 'Customizable client-driven secured line'
},
{
  totalStudents: 2,
  avgMarks: 68,
  course_title: 'Balanced national function'
},
{
  totalStudents: 1,
  avgMarks: 54,
  course_title: 'Cloned contextually-based strategy'
},
{
  totalStudents: 2,
  avgMarks: 72,
  course_title: 'Enhanced full-range open architecture'
},
{
  totalStudents: 1,
  avgMarks: 73,
  course_title: 'Innovative hybrid concept'
}

ype "it" for more
ollegeDB>
```

- totalStudents → counts how many students are enrolled using $sum: 1
- avgMarks → finds average marks using $avg: "$marks".
- Use $lookup to join this data with the courses collection.

3.Grouping, Sorting, and Limiting

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

```
collegeDB> db.enrollments.aggregate([
...   // Group by student_id to calculate average marks
...   {
...     $group: {
...       _id: "$student_id",
...       average_marks: { $avg: "$marks" }
...     }
...   },
...   // Join with students collection to get student names
...   {
...     $lookup: {
...       from: "students",
...       localField: "_id",   // student_id from enrollments
...       foreignField: "_id", // _id from students
...       as: "student_info"
...     }
...   },
...   { $unwind: "$student_info" },
...   // Select required fields
...   {
...     $project: {
...       _id: 0,
...       student_name: "$student_info.name",
...       average_marks: 1
...     }
...   },
...   // Sort descending by average marks
...   { $sort: { average_marks: -1 } },
...   // Limit to top 3 students
...   { $limit: 3 }
... ]);
... //Name: MD.Arman, Rollno: 1240258257
[
  { average_marks: 100, student_name: 'Diane Phillips' },
  { average_marks: 98, student_name: 'Brandon Rios' },
  { average_marks: 94, student_name: 'Christopher Benson' }
]
```

$group → Groups documents by student_id and calculates the average of marks.

$lookup → Joins each grouped record with the students collection to get student details.

$unwind → Flattens the joined array from $lookup.

$project → Selects only student name and average marks.

$sort → Orders by average_marks in descending order. $limit

→ Returns only the top 3 studen

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

```
collegeDB> db.students.aggregate([
...   { $group: { _id: "$department", totalStudents: { $sum: 1 } } },
...   { $sort: { totalStudents: -1 } },
...   { $limit: 1 }
... ]);
...  //Name: MD.Arman, Rollno: 1240258257
[ { _id: 'Electrical', totalStudents: 23 } ]
collegeDB>
```

- $group and $sum → Grouping and counting.
- $sort and $limit → Ranking results.

4.Update, Upsert, and Delete

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

```
collegeDB> db.students.updateMany(
...   { "activities.name": "Hackathon" },
...   { $set: { attendance: 100 } }
... );
...  //Name: MD.Arman, Rollno: 1240258257
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
collegeDB>
```

- { "activities.name": "Hackathon" } → condition
- { $set: { attendance: 100 } } → sets attendance to 100%.

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

```
collegeDB> db.activities.deleteMany({ year: { $lt: 2022 } });//Name: MD. Arman, Rollno: 1240258257
{ acknowledged: true, deletedCount: 0 }
collegeDB>
```

.

Deletes all activity records before 2022.

- deleteMany() → Removes multiple documents.
- $lt → “Less than” comparison operator.

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".

```
collegeDB> db.courses.updateOne(
...   { course_id: "C150" },
...   { $set: { title: "Advanced Data Structures", credits: 4 } },
...   { upsert: true }
... );
...  //Name: MD.Arman, Rollno: 1240258257
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
collegeDB>
```

- { _id: "C150" } → finds the course with ID "C150".
- Update Part: { $set: { title: "Advanced Data Structures", credits: 4 } }
- { upsert: true } : inserts if not found

5. Array & Operator Usage

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

```
collegeDB> db.students.find(
...   { skills: { $in: ["Python"], $nin: ["C++"] } },
...   { name: 1, _id: 0 }
... );
... //Name: MD.Arman, Rollno: 1240258257
[
  { name: 'Kyle Hale' },
  { name: 'Thomas Jackson' },
  { name: 'Steven Wong' },
  { name: 'Cheryl Jackson' },
  { name: 'Cody Whitehead' },
  { name: 'Mr. Darius Newman' },
  { name: 'Paula Jenkins' },
  { name: 'Derrick Humphrey' },
  { name: 'Barbara Jones' },
  { name: 'Tracey Young' },
  { name: 'Brian Russell' },
  { name: 'Elizabeth Reed' },
  { name: 'David Rivera' },
  { name: 'Taylor Webb' },
  { name: 'Erin Harris' },
  { name: 'Kyle Lee' }
]
collegeDB>
```

- $in → Matches documents containing Python.
- $nin → Matches documents that do **not** contain C++.

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

```
collegeDB> db.students.find(
...   { "activities.name": { $all: ["Seminar", "Hackathon"] } },
...   { name: 1, _id: 0 }
... );
...  //Name: MD.Arman, Rollno: 1240258257

collegeDB>
```

- Condition: activities.name: { $all: ["Seminar", "Hackathon"] }
- $all ensures the student has both "Seminar" and "Hackathon" in their activities array

6.Subdocuments and Nested Conditions

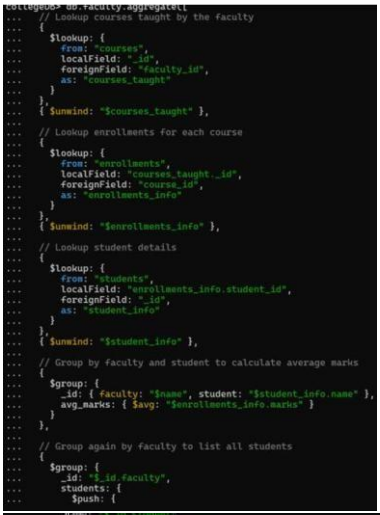
Q12. Find students who scored more than 80 in "Web Development" only if they belong to the "Computer Science" department.

```
collegeDB> db.students.find(
...   { department: "Computer Science", "marks.Web Development": { $gt: 80 } },
...   { name: 1, department: 1, "marks.Web Development": 1, _id: 0 }
... );
...  //Name: MD.Arman, Rollno: 1240258257
```

- Dot notation (marks.Web Development) → Accesses nested fields.
- $gt → Greater than condition.

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.



```
...   // Lookup courses taught by the faculty
...   {
...     $lookup: {
...       from: "courses",
...       localField: "_id",
...       foreignField: "faculty_id",
...       as: "courses_taught"
...     }
...   },
...   { $unwind: "$courses_taught" },
...
...   // Lookup enrollments for each course
...   {
...     $lookup: {
...       from: "enrollments",
...       localField: "courses_taught._id",
...       foreignField: "course_id",
...       as: "enrollments_info"
...     }
...   },
...   { $unwind: "$enrollments_info" },
...
...   // Lookup student details
...   {
...     $lookup: {
...       from: "students",
...       localField: "enrollments_info.student_id",
...       foreignField: "_id",
...       as: "student_info"
...     }
...   },
...   { $unwind: "$student_info" },
...
...   // Group by faculty and student to calculate average marks
...   {
...     $group: {
...       _id: { faculty: "$name", student: "$student_info.name" },
...       avg_marks: { $avg: "$enrollments_info.marks" }
...     }
...   },
...
...   // Group again by faculty to list all students
...   {
...     $group: {
...       _id: "$_id.faculty",
...       students: {
...         $push: {
...           name: "$_id.student",
...           average_marks: "$avg_marks"
...         }
...       }
...     }
...   },
...
...   // Project final output
...   {
...     $project: {
...       _id: 0,
...       faculty_name: "$_id",
...       students: 1
...     }
...   }
... ])
{
  students: [
    { name: 'Vincent Norris', average_marks: 86 },
    { name: 'David Taylor', average_marks: 65 }
  ],
  faculty_name: 'Sandra Decker'
}
{
  students: [
    { name: 'Kathryn Ferguson', average_marks: 84 },
    { name: 'Timothy Lee', average_marks: 85 },
    { name: 'Lydia Day', average_marks: 76 }
  ],
  faculty_name: 'Robert Lara'
}
{
  students: [ { name: 'Reginald Oliver', average_marks: 81 } ],
  faculty_name: 'Ashley Parker'
}
{
  students: [ { name: 'Carolyn Chandler', average_marks: 51 } ],
  faculty_name: 'Charles Cunningham'
}
{
  students: [
    { name: 'Rachael Harris', average_marks: 58 },
    { name: 'Ronald Trevino', average_marks: 97 }
  ],
  faculty_name: 'Kevin Booth'
}
{
  students: [
    { name: 'Lydia Day', average_marks: 53 },
    { name: 'Danielle Rich', average_marks: 68 },
    { name: 'Michelle Weber', average_marks: 71 }
  ],
  faculty_name: 'Julia Cole'
}
{
  students: [ { name: 'Ashley Myers', average_marks: 51 } ],
  faculty_name: 'Stephen Galvan'
}
{
  students: [ { name: 'Gabriela Le', average_marks: 55 } ],
  faculty_name: 'Matthew Hanna'
}
{
  students: [
    { name: 'Nicholas Turner', average_marks: 100 },
    { name: 'Juan Morris', average_marks: 88 }
  ],
  faculty_name: 'Laura Flores'
}
{
  students: [ { name: 'Jeremy Carrillo', average_marks: 50 } ],
  faculty_name: 'Robin Johnson'
}
{
  students: [
    { name: 'Benjamin White', average_marks: 91 },
    { name: 'Thomas Jackson', average_marks: 82 },
    { name: 'Donna Spencer', average_marks: 81 }
  ],
  faculty_name: 'Jacqueline Miller'
}
{
  students: [ { name: 'Nicholas Turner', average_marks: 51 } ],
  faculty_name: 'Autumn White'
}
{
  students: [
    { name: 'Jason Brown', average_marks: 78 },
    { name: 'Alejandro Hart', average_marks: 65 },
    { name: 'Timothy Sparks', average_marks: 60 }
  ],
  faculty_name: 'James Martin'
}
{
  students: [
    { name: 'Marie Wilson', average_marks: 61 },
    { name: 'Travis Johnson', average_marks: 73 },
    { name: 'Christopher Benson', average_marks: 94 },
    { name: 'Thomas Jackson', average_marks: 85 }
  ],
  faculty_name: 'Holly Huang'
}
{
  students: [
    { name: 'Barbara Jones', average_marks: 93 },
    { name: 'Anthony Zavala', average_marks: 90 }
  ],
  faculty_name: 'Alexis Stone'
}
{
  students: [
    { name: 'Lydia Day', average_marks: 92 },
    { name: 'Kyle Lee', average_marks: 97 }
  ],
  faculty_name: 'James Wirby'
}
{
  students: [
    { name: 'Timothy Sparks', average_marks: 80 },
    { name: 'Ryan Flores', average_marks: 84 }
  ],
  faculty_name: 'Cassandra Diaz'
}
{
  students: [
    { name: 'Timothy Sparks', average_marks: 95 },
    { name: 'Megan Taylor', average_marks: 76 }
  ],
  faculty_name: 'Robin West'
}
Type "it" for more
```

Q14. Show the most popular activity type (eg, Hackathon or Seminar) by number of student participants.

```
collegeDB> db.activities.aggregate([
...   { $group: { _id: "$name", participants: { $sum: 1 } } },
...   { $sort: { participants: -1 } },
...   { $limit: 1 },
...   { $project: { _id: 0, activity: "$_id", participants: 1 } }
... ]);
...  //Name: MD.Arman, Rollno: 1240258257
[ { participants: 1, activity: 'Syndicate Innovative Relationships' } ]
collegeDB>
```

- $group and $sum → Count total participants.
- $sort and $limit → Rank and display the top result.
- $project → Rename and format final output.