Babu Banarsi Das University

Name-Prakhar Pandey

Batch-BCADS-25

Roll No.-1240258318 (22)

Submitted to-

Ankit Verma

CREATE DATABASE

-Open MongoDB shell

Write :-mongosh

```
PS C:\Users\conl7> mongosh
Current Mongosh Log ID: 68fe15394ec3cdcbcceec4a8
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.6
Using MongoDB:          8.0.12
Using Mongosh:          2.5.6
mongosh 2.5.8 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-10-21T16:58:48.427+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test>
```

-Check the Databases Present

Show dbs

```
test> show dbs
MongoDB_Assignment  212.00 KiB
admin               212.00 KiB
config              108.00 KiB
local                80.00 KiB
test>
```

-use your Database

Use MongoDB_Assignment

-Check collections

Show collections

```
test> use MongoDB_Assignment
switched to db MongoDB_Assignment
MongoDB_Assignment> show collections
activities
courses
enrollment
faculty
students
MongoDB_Assignment>
```

-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".

```
db.students.find(          //Prakhar Pandey -1240258318
   {attendance:{$gt:85},
   skills:{$all:["MongoDB","Python"]}},
   {_id:0,
   name:1,
   department:1}
    );
```

```
MongoDB_Assignment> db.students.find(            //Prakhar Pandey -1240258318
...      {attendance:{$gt:85},
...      skills:{$all:["MongoDB","Python"]}},
...      {_id:0,
...      name:1,
...      department:1}
...       );
...

MongoDB_Assignment>
```

*It is not showing any output that means no documents match the condition.

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

```
db.faculty.aggregate([{ $project: { _id:0,name: 1, totalCourses: { $size: "$courses" } } },
 { $match: { totalCourses: { $gt: 2 } } } ]
```

```
MongoDB_Assignment> db.faculty.aggregate([{ $project: { _id:0,name: 1, totalCourses: { $size: "$courses" } } }, { $match: { totalCourses: { $gt: 2 } } } ] )

[
  { 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 }
]
```

-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).

```
db.students.aggregate([
 {
  $lookup: {
   from: "enrollments",
   localField: "_id",
   foreignField: "student_id",
   as: "enrollmentInfo"}},
 {
  $unwind: "$enrollmentInfo"
 },
 { $lookup: {
   from: "courses",
   localField: "enrollmentInfo.course_id",
   foreignField: "_id",
   as: "courseInfo"}},
{
 $unwind: "$courseInfo"},
{
 $project: {
   _id: 0,
   name: 1,
   courseTitle: "$courseInfo.title"}}])
```

```
MongoDB_Assignment>
... db.students.aggregate([
...   {
...     $lookup: {
...       from: "enrollments",
...       localField: "_id",
...       foreignField: "student_id",
...       as: "enrollmentInfo"}},
...   {
...     $unwind: "$enrollmentInfo"
...   },
...   { $lookup: {
...       from: "courses",
...       localField: "enrollmentInfo.course_id",
...       foreignField: "_id",
...       as: "courseInfo"}},
...   {
...     $unwind: "$courseInfo"},
...   {
...     $project: {
...       _id: 0,
...       name: 1,
...       courseTitle: "$courseInfo.title"}}])
...
...
```

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

```
db.enrollments.aggregate([
  {
    $group: {
     _id: "$course_id",
     totalStudents: { $sum: 1 },
     avgMarks: { $avg: "$marks" }
   }
   },
   {
    $lookup: {
     from: "courses",
     localField: "_id",
     foreignField: "_id",
     as: "courseInfo"
   }
  },
  { $unwind: "$courseInfo" },
  {
  $project: {
 _id: 0,
 courseTitle: "$courseInfo.title",
  totalStudents: 1,
  avgMarks: { $round: ["$avgMarks", 2] } // round to 2 decimals
  }}])
```

```
[
  {
    totalStudents: 2,
    courseTitle: 'Persistent static migration',
    avgMarks: 64
  },
  {
    totalStudents: 1,
    courseTitle: 'Advanced analyzing budgetary management',
    avgMarks: 86
  },
  {
    totalStudents: 1,
    courseTitle: 'Configurable global framework',
    avgMarks: 67
  },
  {
    totalStudents: 1,
    courseTitle: 'Triple-buffered cohesive frame',
    avgMarks: 82
  },
  {
    totalStudents: 1,
    courseTitle: 'Automated global conglomeration',
    avgMarks: 52
  },
  {
    totalStudents: 1,
    courseTitle: 'Balanced asynchronous framework',
    avgMarks: 78
  },
  {
    totalStudents: 2,
    courseTitle: 'Enhanced full-range open architecture',
    avgMarks: 72
  },
  {
    totalStudents: 2,
    courseTitle: 'Total tangible moderator',
    avgMarks: 94.5
  },
  {
    totalStudents: 2,
    courseTitle: 'Profit-focused high-level capability',
    avgMarks: 58.5
  },
  {
    totalStudents: 2,
    courseTitle: 'Organic optimal product',
    avgMarks: 76.5
  },
  {
    totalStudents: 2,
    courseTitle: 'Streamlined scalable policy',
    avgMarks: 71.5
  },
  {
    totalStudents: 1,
    courseTitle: 'Organic asynchronous matrix',
    avgMarks: 68
  },
  {
    totalStudents: 1,
    courseTitle: 'Sharable responsive customer loyalty',
    avgMarks: 51
  },
  {
    totalStudents: 2,
    courseTitle: 'Optional next generation frame',
    avgMarks: 82
  },
  {
    totalStudents: 2,
    courseTitle: 'Streamlined zero administration strategy',
    avgMarks: 67.5
  },
  {
    totalStudents: 1,
    courseTitle: 'Seamless upward-trending project',
    avgMarks: 84
  },
  {
    totalStudents: 2,
    courseTitle: 'Configurable scalable data-warehouse',
    avgMarks: 77.5
```

```
    avgMarks: 77.5
  },
  {
    totalStudents: 1,
    courseTitle: 'Object-based regional conglomeration',
    avgMarks: 51
  },
  {
    totalStudents: 2,
    courseTitle: 'Balanced non-volatile parallelism',
    avgMarks: 64.5
  },
  {
    totalStudents: 1,
    courseTitle: 'Enhanced radical secured line',
    avgMarks: 51
  }
]
Type "it" for more
MongoDB_Assignment>
```

-Grouping, Sorting, and Limiting

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

```
db.enrollment.aggregate([
 {
  $group: {
   _id: "$student_id",
   avgMarks: { $avg: "$marks" }
  }
 },
 {
  $sort: { avgMarks: -1 }   // sort by average marks descending
 },
 { $limit: 3 },             // take top 3
 {
  $lookup: {
   from: "students",
   localField: "_id",
   foreignField: "_id",
   as: "studentInfo"
  }
 },
 { $unwind: "$studentInfo" },
 {
  $project: {
   _id: 0,
   name: "$studentInfo.name",
   department: "$studentInfo.department",
   avgMarks: { $round: ["$avgMarks", 2] }
  }
 }
```

])

```
MongoDB_Assignment> db.enrollment.aggregate([  {
...       $group: {
...         _id: "$student_id",
...         avgMarks: { $avg: "$marks" }
...       }
...     },
...     {
...       $sort: { avgMarks: -1 }   // sort by average marks descending
...     },
...     { $limit: 3 },              // take top 3
...     {
...       $lookup: {
...         from: "students",
...         localField: "_id",
...         foreignField: "_id",
...         as: "studentInfo"
...       }
...     },
...     { $unwind: "$studentInfo" },
...     {
...       $project: {
...         _id: 0,
...         name: "$studentInfo.name",
...         department: "$studentInfo.department",
...         avgMarks: { $round: ["$avgMarks", 2] }
...       }
...     }
... ])
...
[
  { name: 'Diane Phillips', department: 'Civil', avgMarks: 100 },
  { name: 'Brandon Rios', department: 'Biotechnology', avgMarks: 98 },
  { name: 'Larry Ramsey', department: 'Biotechnology', avgMarks: 94 }
]
```

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

```
db.students.aggregate([
 {
  $group: {
   _id: "$department",
   totalStudents: { $sum: 1 }
  }
 },
 {
  $sort: { totalStudents: -1 }
 },
 { $limit: 1 },
 {
  $project: {
   _id: 0,
   department: "$_id",
   totalStudents: 1
  }
 }
```

])

```
MongoDB_Assignment> db.students.aggregate([
...   {
...     $group: {
...       _id: "$department",
...       totalStudents: { $sum: 1 }
...     }
...   },
...   {
...     $sort: { totalStudents: -1 }
...   },
...   { $limit: 1 },
...   {
...     $project: {
...       _id: 0,
...       department: "$_id",
...       totalStudents: 1
...     }
...   }
... ])
... //Prakhar Pandey -1240258318
[ { totalStudents: 23, department: 'Electrical' } ]
MongoDB_Assignment>
```

-Update, Upsert, and Delete

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

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

```
... //Prakhar Pandey -1240258318
[ { totalStudents: 23, department: 'Electrical' } ]
MongoDB_Assignment> db.activities.deleteMany(
...   { year: { $lt: 2022 } }
... )
...
{ acknowledged: true, deletedCount: 0 }
MongoDB_Assignment>
```

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"

```
db.courses.updateOne(
  { _id: "C150" },
```

```
 {
  $set: {
   title: "Advanced Data Structures",
   credits: 4
  }
 },
 { upsert: true }
)
```

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

-Array & Operator Usage

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

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

```
MongoDB_Assignment> db.students.find(
...   {
...     skills: { $in: ['Python'], $nin: ["C++"] }
...   },
...   {
...     name: 1,
...     department: 1,
...     skills: 1,
...     _id: 0
...   }
... )
... //Prakhar Pandey -1248258318
[
  {
    name: 'Kyle Hale',
    department: 'Electrical',
    skills: [ 'Python', 'Java' ]
  },
  {
    name: 'Cody Whitehead',
    department: 'Biotechnology',
    skills: [ 'JavaScript', 'Python' ]
  },
  {
    name: 'Thomas Jackson',
    department: 'Electrical',
    skills: [ 'Python', 'AutoCAD' ]
  },
  {
    name: 'Steven Wong',
    department: 'Biotechnology',
    skills: [ 'MongoDB', 'Python' ]
  },
  {
    name: 'Cheryl Jackson',
    department: 'Civil',
    skills: [ 'Research', 'Python' ]
  },
  {
    name: 'Mr. Darius Newman',
    department: 'Mechanical',
    skills: [ 'Python', 'SQL' ]
  },
  {
    name: 'Paula Jenkins',
    department: 'Biotechnology',
    skills: [ 'JavaScript', 'Python' ]
  },
  {
    name: 'Barbara Jones',
    department: 'Civil',
    skills: [ 'Python', 'Research' ]
  },
  {
    name: 'Tracey Young',
    department: 'Computer Science',
    skills: [ 'Python', 'AutoCAD' ]
  },
  {
    name: 'Elizabeth Reed',
    department: 'Computer Science',
    skills: [ 'Java', 'Python' ]
  },
  {
    name: 'Brian Russell',
    department: 'Mechanical',
    skills: [ 'Python', 'Research' ]
  },
  {
    name: 'David Rivera',
    department: 'Computer Science',
    skills: [ 'Python', 'JavaScript' ]
  },
  {
    name: 'Taylor Webb',
    department: 'Computer Science',
    skills: [ 'Linux', 'Python' ]
  },
  {
    name: 'Erin Harris',
    department: 'Biotechnology',
    skills: [ 'AutoCAD', 'Python' ]
  },
  {
    name: 'Kyle Lee',
    department: 'Mechanical',
    skills: [ 'Python', 'JavaScript' ]
  }
```

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

```
//Step 1: Get student IDs for Seminar

const seminarIds = db.activities
  .find({ type: "Seminar" })
  .map(doc => doc.student_id);

//Step 2: Get student IDs for Hackathon

const hackathonIds = db.activities
  .find({ type: "Hackathon" })
  .map(doc => doc.student_id);

 //Step 3: Find intersection and get the students names

const commonIds = seminarIds.filter(id => hackathonIds.includes(id));
db.students.find(
  { _id: { $in: commonIds } },
  { name: 1, _id: 0 }
);
```

```
assignment-1> const seminarIds = db.activities
...   .find({ type: "Seminar" })
...   .toArray()
...   .map(doc => doc.student_id);
...

assignment-1> const hackathonIds = db.activities
...   .find({ type: "Hackathon" })
...   .toArray()
...   .map(doc => doc.student_id);
...

assignment-1> const commonIds = seminarIds.filter(id => hackathonIds.includes(id));
...
... // Step 4: Get student names
... db.students.find(
...   { _id: { $in: commonIds } },
...   { name: 1, _id: 0 }
... );
...
[
  { name: 'Adam Solomon' },
  { name: 'Lydia Day' },
  { name: 'Patricia Scott' },
  { name: 'Taylor Webb' },
  { name: 'Carlos Bryant' }
]
```

-Subdocuments and Nested Conditions

db.enrollment.aggregate([

```
{
  $lookup: {
    from: "courses",
    localField: "course_id",
    foreignField: "_id",
    as: "course_info"
  }
},
{ $unwind: "$course_info" },
{
  $match: {
    "course_info.title": "Web Development",
    marks: { $gt: 80 }
  }
},
{
  $lookup: {
    from: "students",
    localField: "student_id",
    foreignField: "_id",
    as: "student_info"
  }
},
{ $unwind: "$student_info" },
{
  $match: {
    "student_info.department": "Computer Science"
  }
},
{
  $project: {
    _id: 0,
    name: "$student_info.name",
    department: "$student_info.department",
    marks: 1
  }
}
])
```

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

MongoDB_Assignment>
```

*It is not showing any output that means no documents match the condition.

-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.

```
db.faculty.aggregate([
 {
  $lookup: {
   from: "courses",
   localField: "courses",
   foreignField: "_id",
   as: "courses_info"} },
 { $unwind: "$course_info" },
 {
  $lookup: {
   from: "enrollment",
   localField: "course_info._id",
   foreignField: "course_id",
   as: "enrollment_info"} },
 { $unwind: "$enrollment_info" },
 {
  $lookup: {
```

```
    from: "students",
    localField: "enrollment_info.student_id",
    foreignField: "_id",
    as: "student_info"
  }
},
{ $unwind: "$student_info" },
{
  $group: {
    _id: {
      faculty_name: "$name",
      student_name: "$student_info.name"
    },
    average_marks: { $avg: "$enrollment_info.marks" }
  }
},
{
  $project: {
    _id: 0,
    faculty: "$_id.faculty_name",
    student: "$_id.student_name",
    average_marks: 1
  }
 },
 { $sort: { faculty: 1, student: 1 } }
])
```

```
MongoDB_Assignment> db.faculty.aggregate([
...   {
...     $lookup: {
...       from: 'courses',
...       localField: 'courses',
...       foreignField: '_id',
...       as: 'courses_info'} },
...   { $unwind: "$course_info" },
...   {
...     $lookup: {
...       from: 'enrollment',
...       localField: 'course_info._id',
...       foreignField: 'course_id',
...       as: 'enrollment_info'} },
...   { $unwind: "$enrollment_info" },
...   {
...     $lookup: {
...       from: 'students',
...       localField: 'enrollment_info.student_id',
...       foreignField: '_id',
...       as: 'student_info'
...     }
...   },
...   { $unwind: "$student_info" },
...   {
...     $group: {
...       _id: {
...         faculty_name: "$name",
...         student_name: "$student_info.name"
...       },
...       average_marks: { $avg: "$enrollment_info.marks" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       faculty: "$_id.faculty_name",
...       student: "$_id.student_name",
...       average_marks: 1
...     }
...   },
...   { $sort: { faculty: 1, student: 1 } }
... ])//Prakhar Pandey  12408258318
...
```

Output

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

```
db.activities.aggregate([
 {
  $group: {
   _id: "$type",
   participant_count: { $sum: 1 }
  }
 },
 { $sort: { participant_count: -1 } },
 { $limit: 1 },
 {
  $project: {
   _id: 0,
   activity_type: "$_id",
   participant_count: 1
  }
 }
])
```

```
MongoDB_Assignment> db.activities.aggregate([ //Prakhar Pandey -1240258318
...   {
...     $group: {
...       _id: "$type",
...       participant_count: { $sum: 1 }
...     }
...   },
...   { $sort: { participant_count: -1 } },
...   { $limit: 1 },
...   {
...     $project: {
...       _id: 0,
...       activity_type: "$_id",
...       participant_count: 1
...     }
...   }
... ])
...
[ { participant_count: 35, activity_type: 'Hackathon' } ]
MongoDB_Assignment>
```