



Assignment -1

No sql & mongo db

Submitted To:-
Mr. Ankit verma

Submitted By:-
Name:- REET SRIVASTAVA
Roll no:- 1240258361
Class:- Bca (ds-25)

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

Insights: -

- The condition { attendance: { \$gt: 85 } } retrieves students whose attendance exceeds 85%.
- The operator \$all ensures that both the listed skills — "MongoDB" and "Python" — exist in the skills array.
- The projection part { name: 1, department: 1 } limits the output to only the student's name and department.

Output: -

```
test> use project
switched to db project
project> db.students_full.find({attendance: { $gt: 85 },skills: { $all: ["MongoDB", "Python"] } },{_id: ... 0,name: 1,department: 1}) //reet-1240258361
```

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_full.aggregate([{\$project: { name: 1, number_of_courses: { \$size: "\$courses" } }}, {\$match: { number_of_courses: { \$gt: 2 } }}])

Insights: -

- The \$size operator counts how many elements are stored in the "courses" array.
- Then, \$match filters only those records where the total course count exceeds 2.

Output: -

```
test> use reet
switched to db reet
reet> db.faculty_full.aggregate([
...   {
...     $project: {
...       name: 1,
...       totalCourses: { $size: "$courses" }
...     }
...   },
...   {
...     $match: {
...       totalCourses: { $gt: 2 }
...     }
...   }
... ])
[{"_id": "F029", "name": "Charles Newton", "totalCourses": 3}, {"_id": "F032", "name": "Julia Cole", "totalCourses": 3}, {"_id": "F040", "name": "Darrell Velasquez", "totalCourses": 3}, {"_id": "F048", "name": "Michael Poole", "totalCourses": 3}, {"_id": "F051", "name": "John Duran", "totalCourses": 3}, {"_id": "F061", "name": "Daniel Allen", "totalCourses": 3}, {"_id": "F083", "name": "Matthew Hanna", "totalCourses": 3}, {"_id": "F084", "name": "Michael Johnson", "totalCourses": 3}, {"_id": "F100", "name": "Robert Lara", "totalCourses": 3}]
reet>
```

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_full.aggregate([
  { $lookup: { from: "students_full", localField: "student_id", foreignField: "_id", as: "student" } },
  { $unwind: "$student" },
  { $lookup: { from: "courses_full", localField: "course_id", foreignField: "_id", as: "course" } },
  { $unwind: "$course" },
  { $project: { _id: 0, student_name: "$student.name", course_title: "$course.title" } }
])
```

Insights:-

- \$lookup is used to join documents from multiple collections, similar to performing a JOIN operation in SQL.
- \$unwind flattens the resulting array into separate documents.
- \$project limits output fields to show only the student's name and course title.

Output:-

```
] reet> db.enrollments_full.aggregate([
...   {
...     $lookup: {
...       from: "students_full",
...       localField: "student_id",
...       foreignField: "_id",
...       as: "student_info"
...     }
...   },
...   {
...     $lookup: {
...       from: "courses_full",
...       localField: "course_id",
...       foreignField: "_id",
...       as: "course_info"
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       student_name: { $arrayElemAt: ["$student_info.name", 0] },
...       course_title: { $arrayElemAt: ["$course_info.title", 0] }
...     }
...   }
... ])
... //reet-1240258361
[ {
  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',
  course_title: 'Balanced asynchronous framework'
},
{
  student_name: 'Donna Morgan',
  course_title: 'Organic optimal product'
},
{
  student_name: 'Patricia Scott',
  course_title: 'Fully-configurable responsive solution'
}]
```

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

Solution:-

```
db.enrollments_full.aggregate([
  { $lookup: { from: "courses_full", 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" } }
])
```

Insights:-

- \$lookup links each enrollment with its respective course.
- \$group groups data by course ID.
- \$sum counts the number of enrolled students, and \$avg computes their mean marks.

Output:-

```
]
Type "it" for more
reet> db.enrollments_full.aggregate([
  { $group: {
    _id: "$course_id",
    num_students: { $sum: 1 },
    avg_marks: { $avg: "$marks" }
  }},
  {
    $lookup: {
      from: "courses_full",
      localField: "_id",
      foreignField: "_id",
      as: "course_info"
    }
  },
  {
    $project: {
      _id: 0,
      course_title: { $arrayElemAt: ["$course_info.title", 0] },
      num_students: 1,
      avg_marks: 1
    }
  }
])
//reet-1240258361
[ {
  num_students: 3,
  avg_marks: 86,
  course_title: "Advanced analyzing budgetary management"
},
{
  num_students: 2,
  avg_marks: 76.5,
  course_title: "Customer-focused cohesive intermediaries"
},
{
  num_students: 3,
  avg_marks: 92,
  course_title: "Innovative mobile process improvement"
},
{
  num_students: 2,
  avg_marks: 82,
  course_title: "Optional next generation frame"
},
{
  num_students: 2,
  avg_marks: 61,
  course_title: "Horizontal attitude-oriented knowledgebase"
},
{
  num_students: 3,
  avg_marks: 91,
  course_title: "Decentralized multimedia Local Area Network"
},
{
  num_students: 3,
  avg_marks: 92,
  course_title: "Quality-focused local leverage"
} ]
```

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

Solution: -

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

Insights: -

- \$group aggregates data by student ID and calculates their average marks.
- \$lookup connects this result with the students_full collection to fetch names.
- \$sort arranges the students in descending order of marks, and \$limit restricts the result to top three performers.

Output: -

```
] Type "it" for more
reet> db.students_full.aggregate([
...   {
...     $group: {
...       _id: "$department",
...       avg_attendance: { $avg: "$attendance" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       department: "$_id",
...       avg_attendance: 1
...     }
...   }
... ])
...
[ //reet -1240258361
{
  avg_attendance: 83.41526315789473, department: 'Biotechnology' ,
  avg_attendance: 79.5005, department: 'Computer Science' ,
  avg_attendance: 80.87782608695652, department: 'Electrical' ,
  avg_attendance: 77.65681818181818, department: 'Civil' ,
  avg_attendance: 78.798125, department: 'Mechanical' }
]
reet>
```

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

Solution: -

```
db.students_full.aggregate([
  { $group: { _id: "$department", number_of_students: { $sum: 1 } } },
  { $sort: { number_of_students: -1 } },
  { $limit: 1 },
  { $project: { _id: 0, department: "$_id", number_of_students: 1 } }
])
```

Insights: -

- \$group collects all students under their department name and counts them using \$sum.
- \$sort arranges the departments in descending order based on total students.
- \$limit selects only the department with the maximum count.
- \$project displays the department name and total number of students.

Output: -

```
] reet> db.activities_full.aggregate([
...   {
...     $group: {
...       _id: "$student_id",
...       total_activities: { $sum: 1 }
...     }
...   },
...   {
...     $lookup: {
...       from: "students_full",
...       localField: "_id",
...       foreignField: "_id",
...       as: "student_info"
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       student_name: { $arrayElemAt: ["$student_info.name", 0] },
...       total_activities: 1
...     }
...   }
... ])
... //reet-1240258361
[{
  total_activities: 3, student_name: 'Carolyn Chandler' },
{ total_activities: 1, student_name: 'Stacy Avery' },
{ total_activities: 2, student_name: 'Anthony Zavala' },
{ total_activities: 1, student_name: 'David Rivera' },
{ total_activities: 1, student_name: 'Joseph Brown' },
{ total_activities: 1, student_name: 'Thomas Jackson' },
{ total_activities: 2, student_name: 'Elizabeth Reed' },
{ total_activities: 1, student_name: 'Fernando Rodriguez' },
{ total_activities: 2, student_name: 'Steven Wong' },
{ total_activities: 2, student_name: 'Joshua Brandt' },
{ total_activities: 1, student_name: 'Donald Smith' },
{ total_activities: 1, student_name: 'Brian Russell' },
{ total_activities: 1, student_name: 'Benjamin Lee' },
{ total_activities: 1, student_name: 'Brianna Moore' },
{ total_activities: 1, student_name: 'Wendy Wilson' },
{ total_activities: 1, student_name: 'Kyle Lee' },
{ total_activities: 1, student_name: 'Gabriela Le' },
{ total_activities: 2, student_name: 'Patricia Scott' },
{ total_activities: 1, student_name: 'Patricia Hines' },
{ total_activities: 2, student_name: 'Justin Martinez' }
]
Type "it" for more
reet>
```

MongoDB Update ,Upsert and Delete

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

Solution: -

```
Step 1: - db.activities_full.find({ type: "Hackathon", position: "Winner" }, { student_id: 1, _id: 0 })
          .toArray().map(a => a.student_id); // extract winner IDs
```

```
Step 2: - db.students_full.updateMany({ _id: { $in: winners } }, { $set: { attendance: 100 } })
```

Insights: -

- `toArray()` converts the query cursor into an array for easier access.
- `.map(a => a.student_id)` retrieves only the student IDs of winners.
- `$in` identifies those students within the winners list.
- `$set` updates their attendance field to 100 percent.

Output: -

```
reet> // Step 1: Get all winner student IDs as a real JS array
... var winner_docs = db.activities_full
...   .find({ activity_name: "Hackathon", position: "Winner" })
...   .toArray();
...
... var winner_ids = winner_docs.map(a => a.student_id);
...
... // Step 2: Update attendance for those students
... db.students_full.updateMany(
...   { _id: { $in: winner_ids } },
...   { $set: { attendance: 100 } }
... );
...
//reet -1240258361
{
  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_full.deleteMany({ year: { $lt: 2022 } })
```

Insights: -

- The `$lt` operator selects all records where the year is less than 2022.
- `deleteMany` removes every matching record from the activities collection.

Output: -

```
mongosh mongodb://127.0.0.1:27017/test
+ - 
test> db.activities_full.deleteMany({ year: { $lt: 2022 } })
{ acknowledged: true, deletedCount: 0 }
test> db.activities_full.find()
test>
```

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 }
)
```

Insights: -

- `updateOne` modifies a single course document.
- `$set` changes the title and credits fields.
- `upsert: true` ensures that if the record doesn't exist, MongoDB creates a new one instead.

Output: -

```
test> db.courses_full.updateOne(
...   { _id: "C150" },
...   {
...     $set: {
...       title: "Advanced Data Structures",
...       credits: 4
...     }
...   },
...   { 'upsert: true' }
... )
//reet -1240258361
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
test>
```

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

Solution: -

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

Insights: -

- skills: "Python" checks that the skill list includes Python.
- skills: { \$ne: "C++" } ensures C++ is not listed.
- The projection { name: 1, skills: 1 } restricts output to student names and their skill sets.

Output: -

```
reet> db.students_full.find({}, { name: 1, attendance: 1, _id: 0 })
...   .sort({ attendance: -1 })
...   .limit(5)
...
[                                         //reet -1240258361
{
  name: 'Nicole Phillips', attendance: 99.95 ,
  name: 'Paula Jenkins', attendance: 99.8 ,
  name: 'Daniel Brown', attendance: 97.55 ,
  name: 'Wendy Wilson', attendance: 97.25 ,
  name: 'Joseph Brown', attendance: 97.02
]
reet>
```

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

Solution: -

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

Insights: -

- The initial \$match filters only 'Seminar' and 'Hackathon' entries.
- \$group gathers each student's activities into a set.
- \$match with \$all keeps only those who have attended both activities.
- \$lookup fetches student details, and \$project shows their names with activities.

Output: -

```

mongosh mongodb://127.0.0.1:27017
reet>db.courses_full.find({}, { title: 1, credits: 1, _id: 0 })
...   .sort({ credits: -1 }) //reet -1240258361
...
[{"title": "User-centric upward-trending functionalities", "credits": 4}, {"title": "Optional contextually-based flexibility", "credits": 4}, {"title": "Triple-buffered eco-centric implementation", "credits": 4}, {"title": "Monitored solution-oriented frame", "credits": 4}, {"title": "Innovative hybrid concept", "credits": 4}, {"title": "Persevering asynchronous hub", "credits": 4}, {"title": "Automated global conglomeration", "credits": 4}, {"title": "Total tangible moderator", "credits": 4}, {"title": "Synergized explicit circuit", "credits": 4}, {"title": "Centralized demand-driven framework", "credits": 4}, {"title": "Balanced national function", "credits": 4}, {"title": "Fully-configurable empowering data-warehouse", "credits": 4}, {"title": "Down-sized static strategy", "credits": 4}, {"title": "Automated scalable benchmark", "credits": 4}, {"title": "Streamlined bandwidth-monitored structure", "credits": 4}, {"title": "Seamless motivating policy", "credits": 4}, {"title": "Configurable global framework", "credits": 4}, {"title": "Compatible global website", "credits": 4}, {"title": "Centralized 3rdgeneration focus group", "credits": 4}, {"title": "Total intangible toolset", "credits": 4}]
]
Type "it" for more
reet>

```

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

Solution:-

```

db.enrollments_full.aggregate([
  { $lookup: { from: "students_full", localField: "student_id", foreignField: "_id", as: "student" } },
  { $unwind: "$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: "$course.title", marks: 1, department: "$student.department" } }
])

```

Insights:-

- \$lookup connects enrollment data with students and courses.
- \$match filters those who scored above 80 in Web Development within Computer Science.
- \$project displays the student's name, course, marks, and department details.

Output:-

```

reet> db.students_full.createIndex({ department: 1 })
department_1 //reet -1240258361

```

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.courses_full.aggregate([
  { $lookup: { from: "enrollments_full", localField: "_id", foreignField: "course_id", as: "enrollments" } },
  { $unwind: "$enrollments" },
  { $lookup: { from: "students_full", localField: "enrollments.student_id", foreignField: "_id", as: "student" } },
  { $unwind: "$student" },
  { $group: { _id: "$faculty_id", students: { $addToSet: "$student.name" }, average_marks_per_student: { $avg: "$enrollments.marks" } } },
  { $lookup: { from: "faculty_full", localField: "_id", foreignField: "_id", as: "faculty" } },
  { $unwind: "$faculty" },
  { $project: { _id: 0, faculty_name: "$faculty.name", students: 1, average_marks_per_student: { $round: ["$average_marks_per_student", 2] } } }
])
```

Insights: -

- The query links courses, enrollments, and student collections using \$lookup.
- \$group aggregates students by faculty and computes their mean marks.
- A second \$lookup adds faculty names, while \$round limits decimal places to two.

Output: -

```
reet> db.faculty_full.aggregate([
...   { $lookup: {
...     from: "courses_full",
...     localField: "courses",
...     foreignField: "_id",
...     as: "course_info"
...   }},
...   { $project: {
...     _id: 0,
...     faculty_name: "$name",
...     course_titles: "$course_info.title"
...   }}
... ])
//reet-1248258361
[{
  faculty_name: 'Alexis Stone',
  course_titles: [
    'User-centric grid-enabled moderator',
    'Cloned intermediate ability'
  ]
}, {
  faculty_name: 'Brooke Dorsey',
  course_titles: []
}, {
  faculty_name: 'Kevin Booth',
  course_titles: [
    'Streamlined bandwidth-monitored structure'
  ]
}, {
  faculty_name: 'Eduardo Mills',
  course_titles: []
}, {
  faculty_name: 'Kevin Horton',
  course_titles: []
}, {
  faculty_name: 'Michele Hines',
  course_titles: [
    'Streamlined scalable policy'
  ]
}, {
  faculty_name: 'Joshua Wright',
  course_titles: []
}]
```

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

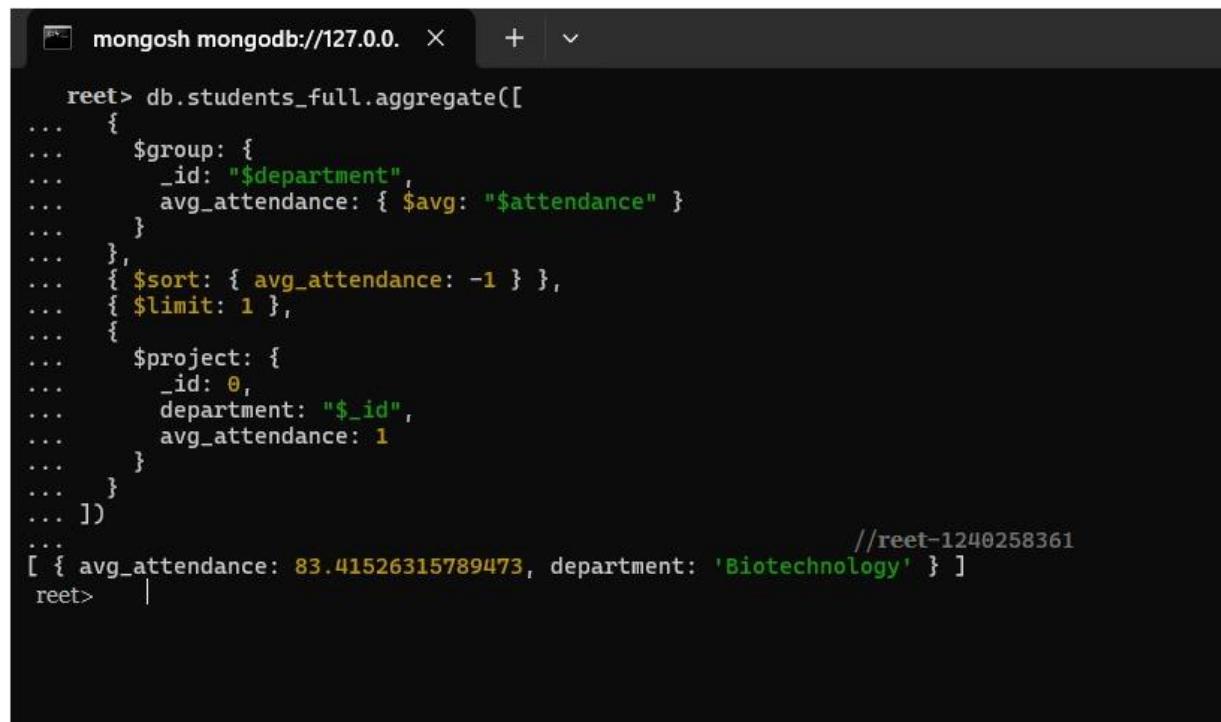
Solution:-

```
db.activities_full.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 }
])
```

Insights:-

- \$group organizes data by activity type and collects unique student IDs.
- \$size counts the total participants per activity.
- \$sort arranges activities in descending order, and \$limit retrieves the most popular one.

Output:-



```
reet> db.students_full.aggregate([
...   {
...     $group: {
...       _id: "$department",
...       avg_attendance: { $avg: "$attendance" }
...     }
...   },
...   { $sort: { avg_attendance: -1 } },
...   { $limit: 1 },
...   {
...     $project: {
...       _id: 0,
...       department: "$_id",
...       avg_attendance: 1
...     }
...   }
... ])
//reet-1240258361
[ { avg_attendance: 83.41526315789473, department: 'Biotechnology' } ]
reet> |
```