***Babu Banarsi Das University***

******

***Name-Prakhar Pandey Submitted to-***

***Batch-BCADS-25 Ankit Verma***

***Roll No.-1240258318 (22)***

***CREATE DATABASE***

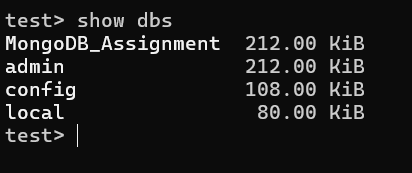
**-Open MongoDB shell**

Write :-mongosh



**-Check the Databases Present**

Show dbs

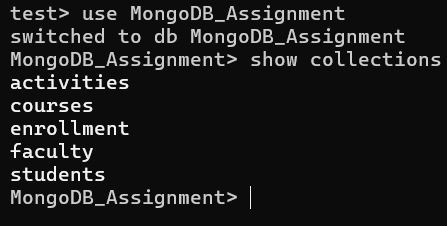


**-use your Database**

Use MongoDB\_Assignment

**-Check collections**

Show collections



**-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},

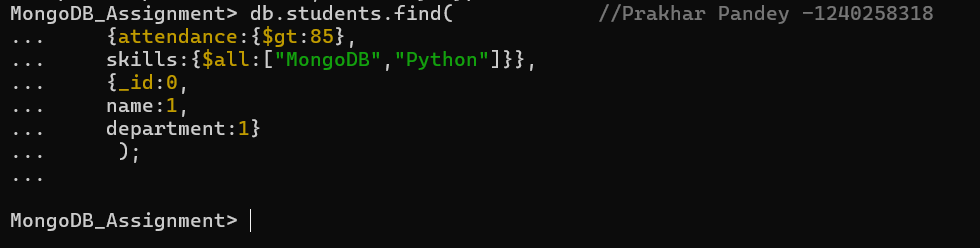
skills:{$all:["MongoDB","Python"]}},

{\_id:0,

name:1,

department:1}

);

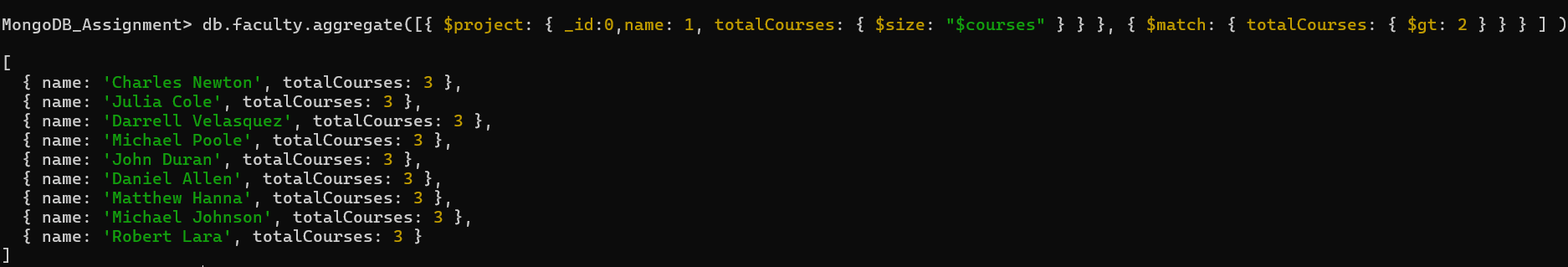


**\***It is not showing any output that means no documents match the condition.

**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 } } } ]



**-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"},

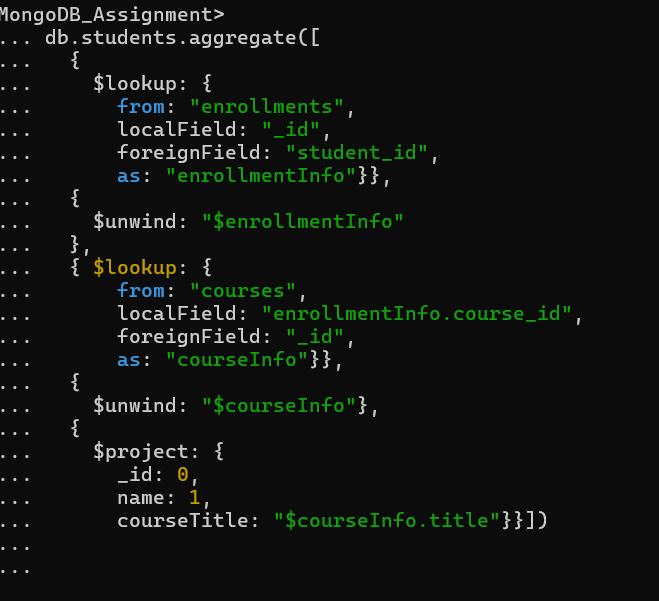
{

$project: {

\_id: 0,

name: 1,

courseTitle: "$courseInfo.title"}}])

****

**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,

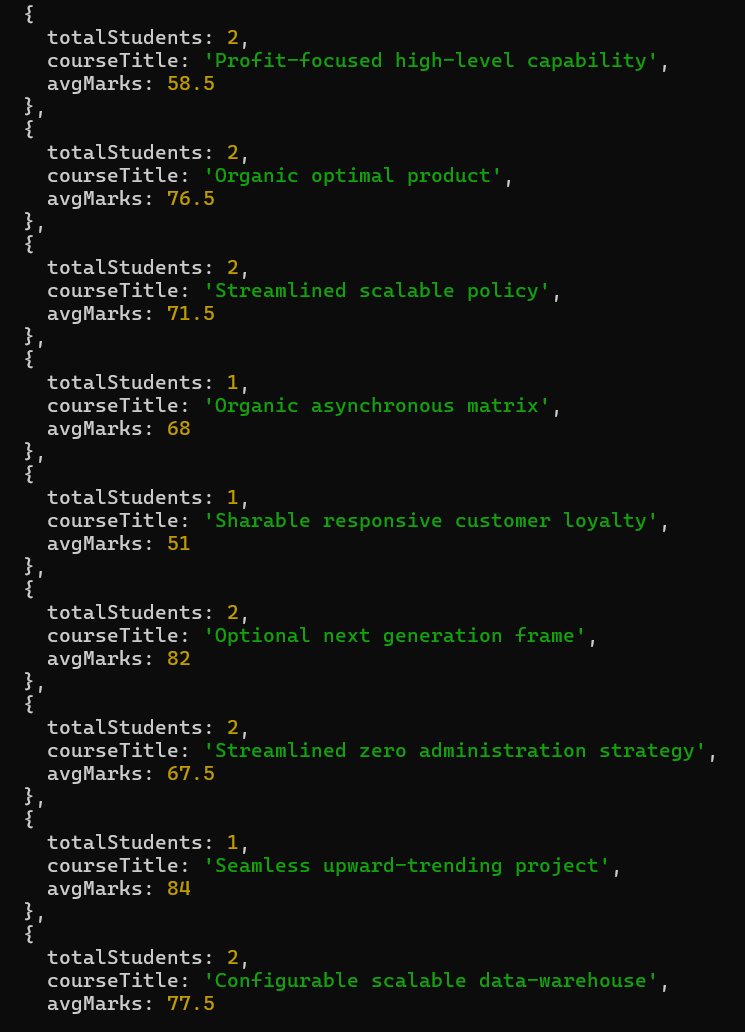
courseTitle: "$courseInfo.title",

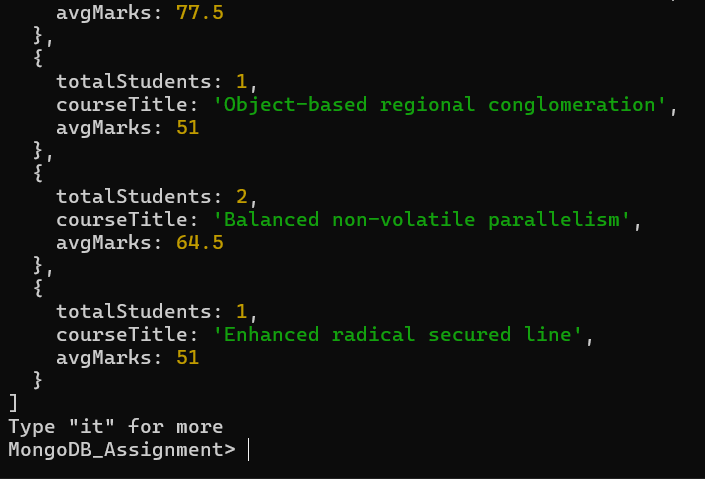
totalStudents: 1,

avgMarks: { $round: ["$avgMarks", 2] } // round to 2 decimals

}}])







**-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] }

}

}

])



**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,

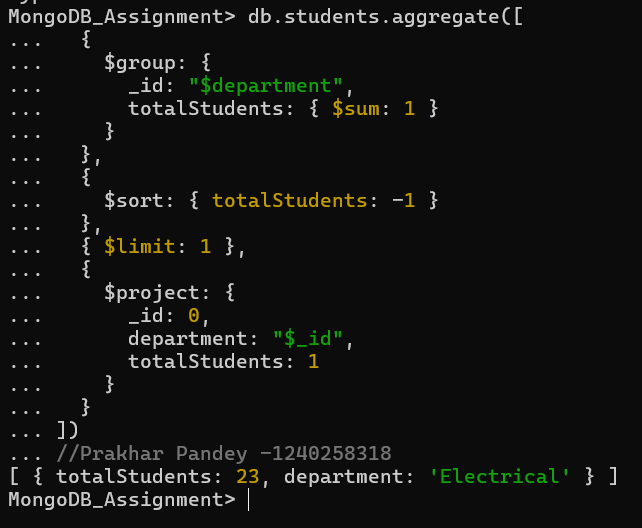
department: "$\_id",

totalStudents: 1

}

}

])



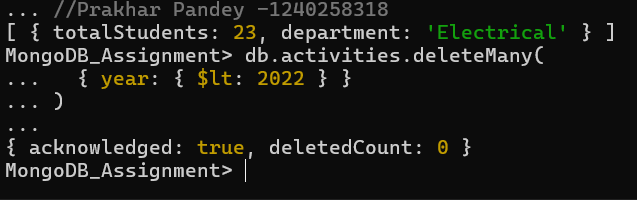
**-Update, Upsert, and Delete**

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

db.activities.deleteMany(

{ year: { $lt: 2022 } }

)



**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",

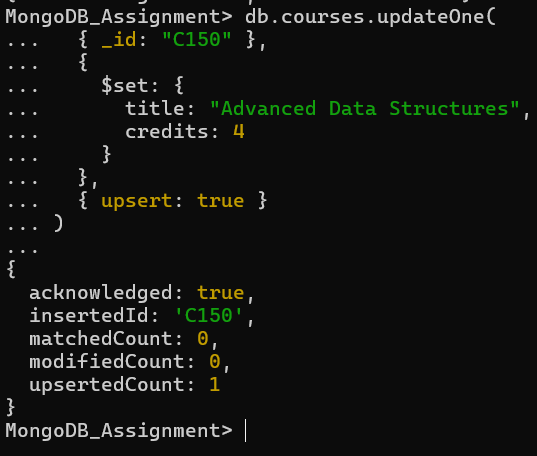
credits: 4

}

},

{ upsert: true }

)



**-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,

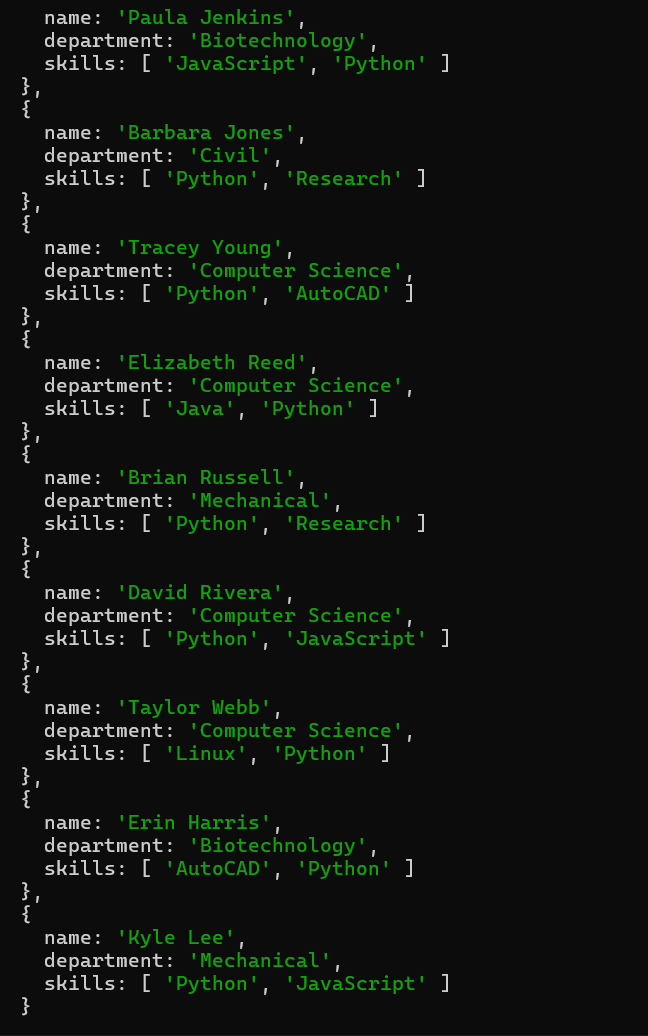
skills: 1,

\_id: 0

}

)





**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***

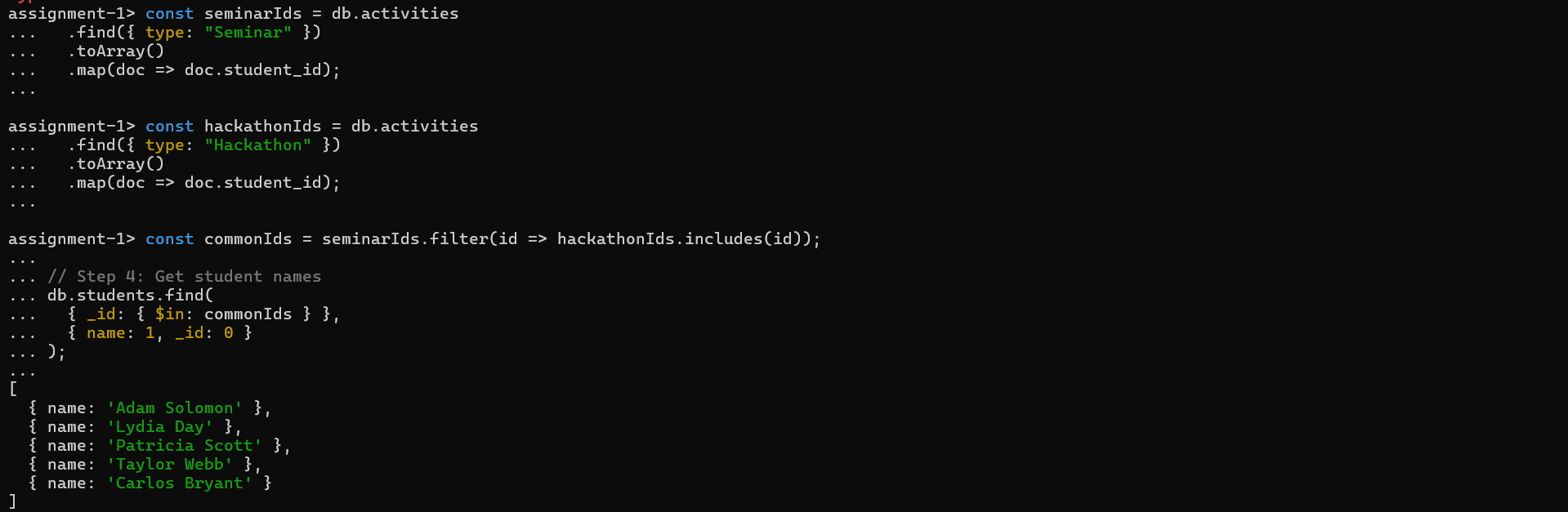
const commonIds = seminarIds.filter(id => hackathonIds.includes(id));

db.students.find(

{ \_id: { $in: commonIds } },

{ name: 1, \_id: 0 }

);

****

**-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

}

}

])



**\***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 } }

])



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,

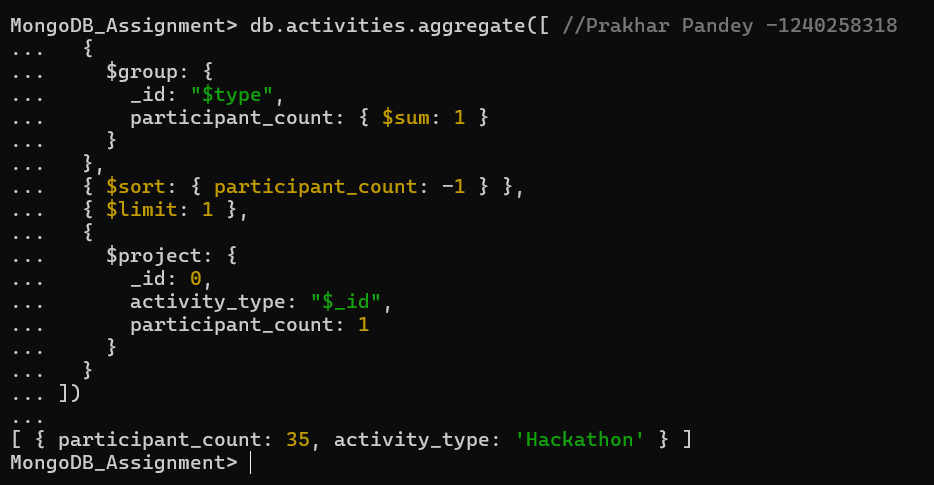
activity\_type: "$\_id",

participant\_count: 1

}

}

])

****