BABU BANARASI DAS UNIVERSITY

LUCKNOW

Session : 2025-2026

SCHOOL OF COMPUTER APPLICATIONS

# NO SQL AND DBAAS 101(NO SQL)

# (BCADSN13202)

NoSQL

**Submitted To: -**

**Mr. Ankit Verma**

**Submitted By: -**

**Name:**

**Anuj Srivastava**

**Section:BCADS22**

**Roll Num:**

**1240258108**

# PROJECT DESCRIPTION PAGE

## Project Title:

Student Performance Analysis using NoSQL Database.

### ❖ Objective:

- The main objective of this project is to design, store, and analyze academic data using a **NoSQL database**.
  It aims to demonstrate how NoSQL databases such as **MongoDB** can efficiently manage **large, unstructured, and hierarchical data** while supporting **advanced queries, aggregation, and analytics**.

## Project Description:

This project focuses on implementing a **NoSQL-based data model** for managing student academic records.

Unlike traditional relational databases, NoSQL databases offer **schema flexibility**, **faster scalability**, and **better handling of complex data** such as arrays and subdocuments.

## Tools & Technologies Used:

- **Database:** MongoDB (NoSQL)
- **Query Interface:** Mongo Shell / MongoDB Compass
- **Language (optional):** Python / Node.js (for connection and queries)
- **Data Source:** Student Academic Dataset (JSON / CSV)

## Problem Statements Covered:

- Complex Filters & Projections.
- Joins ($lookup) and Aggregations
- Grouping, Sorting, and Limiting.
- Update, UPSER, and Delete
- Array & Operator Usage.
- Subdocuments and Nested Conditions.
- Advanced Aggregation (Challenge Level).

# PROJECT

## Problem Statements-1: Complex Filters & Projections.

- ❖ Filters are used to select documents that match specific criteria in a NoSQL collection.
- ❖ Projections specify which fields to include (or exclude) in the output.

## Question:-1

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

**Query :-** db.students_full.find({ attendance: { $gt: 85 }, skills: { $in: ["MongoDB", "Python"] }})

### ➢ Output:-

```
Atlas atlas-idr26g-shard-0 [primary] project> db.students_full.find(       //Name:Anuj Srivastava , Reg No: 1240258108
... { attendence: { $gt: 85 }, skills: { $in: ["MongoDB", "Python"] } })

Atlas atlas-idr26g-shard-0 [primary] project>
```

- Nothing will show up because there aren't any students who have both 'MongoDB' and 'Python' skills and more than 85% attendance.
- Use comparison operators like $gt (greater than).
- Apply array matching with $all to ensure multiple elements exist.
- Use projection to show only required fields.
- Build compound filters using multiple conditions.

## Question:-2

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

### Query : -

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

### ➢ Output:-


- Use $project to create computed fields
- Use $size to count array elements.
- Combine $match after projection for conditional filtering.
- Understand aggregation pipelines

# Problem Statements-2:
# Joins ($lookup) and Aggregations.

- MongoDB is a **document-oriented NoSQL database**, so it doesn't have traditional SQL joins. But you can **join collections** using `$lookup` in an **aggregation pipeline**.
- MongoDB uses the **Aggregation Framework** to process data **like SQL GROUP BY, SUM, COUNT, AVG**.

## Question:-3

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).

### ➢ Query & Output:-

```
Atlas atlas-idr26g-shard-0 [primary] project> db.enrollments_full.aggregate( //Name:Anuj Srivastava , Reg No: 1240258108
... [{ $lookup: { from: "students_full", localField: "student_id", foreignField: "_id", as: "student_info" }},
... { $unwind: "$student_info" }, { $lookup: { from: "courses_full", localField: "course_id", foreignField: "_id", as: "course_info" }},
... { $unwind: "$course_info" }, { $project: { _id: 0, student_name: "$student_info.name", course_title: "$course_info.title" }}])
[
  {
    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'
  },
  {
    student_name: 'Carolyn Chandler',
    course_title: 'Horizontal attitude-oriented knowledgebase'
  },
```

### ➢ Insights:

- ✓ Use $lookup for joins between collections.
- ✓ Use $lookup for joins between collections.
- ✓ Use $arrayElemAt to extract single values from arrays.
- ✓ Understand MongoDB's relational-like linking.

## Question:- 4

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

## Query & Output:-

```
collage> db.enrollments.aggregate([
...   { // Name:Anuj  Shrivastava,Regestation No:1240258108
...     $lookup: {
...       from: "courses",
...       localField: "course_id",
...       foreignField: "_id",
...       as: "courseInfo"
...     }
...   },
...   {
...     $unwind: "$courseInfo"
...   },
...   {
...     $group: {
...       _id: {
...         id: "$course_id",
...         title: "$courseInfo.title"
...       },
...       numberOfStudents: { $sum: 1 },
...       averageMarks: { $avg: "$marks" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       courseTitle: "$_id.title",
...       numberOfStudents: 1,
...       averageMarks: { $round: ["$averageMarks", 2] }
...     }
...   },
...   {
...     $sort: { courseTitle: 1 }
...   }
... ])
[
  {
    numberOfStudents: 1,
    courseTitle: 'Advanced Data Structures',
    averageMarks: 83
  },
  {
    numberOfStudents: 2,
    courseTitle: 'Automated exuding matrix',
    averageMarks: 88
  },
  {
    numberOfStudents: 1,
```

- **Insights**:
  - ✓ Use $group for summarizing data.
  - ✓ Use $avg and $sum to calculate aggregates.
  - ✓ $unwind helps to deconstruct arrays.
  - ✓ $project to rename and structure output.

## Problem Statements-3:
## Grouping, Sorting, and Limiting.

- **Grouping** - Used to **aggregate** data — i.e., group records that have the same values in specific fields and apply functions like `COUNT`, `SUM`, `AVG`, etc.
- **Sorting**- Used to **order** data based on one or more fields.
- **Limiting-** Used to **restrict the number of results** returned by a query.

### Question:-5

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

### ➢ Query & Output:-

```
collage> db.enrollments.aggregate([
...     {     // Name:Anuj Srivastava,Regestation No:1240258108
...       $group: {
...         _id: "$student_id",
...         averageMarks: { $avg: "$marks" }
...       }
...     },
...     {
...       $sort: { averageMarks: -1 }
...     },
...     {
...       $limit: 3
...     },
...     {
...       $lookup: {
...         from: "students",
...         localField: "_id",
...         foreignField: "_id",
...         as: "studentInfo"
...       }
...     },
...     {
...       $unwind: "$studentInfo"
...     },
...     {
...       $project: {
...         _id: 0,
...         studentName: "$studentInfo.name",
...         averageMarks: { $round: ["$averageMarks", 2] }
...       }
...     }
... ])
...
[
  { studentName: 'Adam Solomon', averageMarks: 99 },
  { studentName: 'David Jones', averageMarks: 97 },
  { studentName: 'Colleen Todd', averageMarks: 93 }
]
```

### ➢ Insights:

- ✓ $sort sorts data in ascending/descending order.
- ✓ $limit restricts results to top records.
- ✓ $group for calculating averages.
- ✓ Combining joins with grouping.

## Question:- 6

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

## Query & Output:-

```
collage> db.students.aggregate([
...   {    // Name:Anuj Srivastava,Regestation No:1240258108
...     $group: {
...       _id: "$department",
...       studentCount: { $sum: 1 }
...     }
...   },
...   {
...     $sort: { studentCount: -1 }
...   },
...   {
...     $limit: 1
...   },
...   {
...     $project: {
...       _id: 0,
...       department: "$_id",
...       studentCount: 1
...     }
...   }
... ])
[ { studentCount: 23, department: 'Electrical' } ]
```

- **Insights:**
  - ✓ Count items per category with $sum: 1.
  - ✓ se $sort to rank results
  - ✓ Identify top-performing or most populated groups.
  - ✓ Apply $limit to get top results.

## Problem Statements-4: Update, UPSERT, and Delete.

- **UPDATE**- Used to **modify existing data** in a collection (NoSQL) or table (SQL).
- **UPSERT- Upsert = Update + Insert**
  If a document **exists**, it updates it.
  If it **does not exist**, it **inserts** a new one.
- **DELETE-** Removes one or more documents/records from a collection/table.

### Question:-7

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

### ➢ Query & Output:-

```
collage> var hackathonWinners = db.activities.distinct("student_id", {
... // Name:Anuj Shrivastava,Regestation No:1240258108
...     type: "Hackathon",
...     position: "Winner"
... });
...
... db.students.updateMany(
...   {
...     _id: { $in: hackathonWinners }
...   },
...   {
...     $set: { attendance: 100 }
...   }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 13,
  modifiedCount: 0,
  upsertedCount: 0
}
```

### ➢ Insights:

- ✓ Use updateMany() for bulk updates.
- ✓ $set modifies specific fields.
- ✓ Target documents via nested fields.
- ✓ Understand bulk updates with filters.

## Question:- 8

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

## Query & Output:-

```
collage> db.activities.deleteMany(
...   {   // Name:Anuj Shrivastava,Regestation No:1240258108
...     year: { $lt: 2022 }
...   }
... )
{ acknowledged: true, deletedCount: 0 }
```

- **Insights:**
  - ✓ $lt filters by less than a value.
  - ✓ Delete records conditionally using deleteMany().
  - ✓ Manage dataset cleanup.
  - ✓ Apply conditional data management.

## Question:- 9

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".

## Query & Output:-

```
collage> db.courses.updateOne(
...   {   // Name:Anuj Shrivastava,Regestation No:1240258108
...     _id: "C150"
...   },
...   {
...     $set: {
...       title: "Advanced Data Structures",
...       credits: 4
...     }
...   },
...   {
...     upsert: true
...   }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
```

- **Insights:**
  - ✓ upsert: true inserts if no match is found.
  - ✓ $setOnInsert applies only when inserting new data.
  - ✓ $set updates fields if record exists.
  - ✓ Handle both insert and update in one commands.

## Problem Statements:-5
## Array & Operator Usage.

- **Arrays in NoSQL-** In MongoDB (a document-based NoSQL database), a field can hold an **array** of values.
- **Array Operators -** MongoDB provides special **operators** to query and modify array fields.

### Question:-10

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

### ➢ Query & Output:-

```
collage> db.students.find(
...   {   // Name:Anuj Shrivastava,Regestation No:1240258108
...     skills: "Python",
...     skills: { $nin: ["C++"] }
...   },
...   {
...     name: 1,
...     skills: 1,
...     _id: 0
...   }
... )
[
  { 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
```

### ➢ Insights:

- ✓ $in checks for presence in arrays.
- ✓ $nin checks for absence in arrays.
- ✓ Combine both for exclusive conditions.
- ✓ Operate effectively on array fields**.**

## Question:- 11

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

## Query & Output:-

```
collegeDB>
... db.activities.aggregate([
...   {   // Name: Ankit Patel, Registration No: 1240258088
...     $group: {
...       _id: "$student_id",
...       "activity_types": { $addToSet: "$type" }
...     }
...   },
...   {
...     $match: {
...       "activity_types": { $all: ["Seminar", "Hackathon"] }
...     }
...   },
...   {
...     $lookup: {
...       from: "students",
...       localField: "_id",
...       foreignField: "_id",
...       as: "student_info"
...     }
...   },
...   {
...     $unwind: "$student_info"
...   },
...   {
...     $project: {
...       _id: 0,
...       "Student Name": "$student_info.name"
...     }
...   }
... ])
[
  { 'Student Name': 'Taylor Webb' },
  { 'Student Name': 'Patricia Scott' },
  { 'Student Name': 'Carlos Bryant' },
  { 'Student Name': 'Adam Solomon' },
  { 'Student Name': 'Lydia Day' }
]
```

- **Insights:**
  - ✓ $all ensures all specified elements exist in an array.
  - ✓ Simple array querying in MongoDB.
  - ✓ Combine multiple filters in a single query.
  - ✓ Efficient participation tracking.

## Problem Statements:- 6
## Subdocuments and Nested Conditions.

- **Subdocuments-** A **subdocument** (or embedded document) is a **document inside another document**.
- **Nested Conditions-** Nested conditions allow you to **filter based on multiple subdocument fields** simultaneously.

### Question:-12

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

### ➢ Query & Output:-

```
collage> db.enrollments.aggregate([
...     {   // Name:Anuj Shrivastava,Regestation No:1240258108
...       $lookup: {
...         from: "courses",
...         localField: "course_id",
...         foreignField: "_id",
...         as: "courseInfo"
...       }
...     },
...     {
...       $unwind: "$courseInfo"
...     },
...     {
...       $match: {
...         "courseInfo.title": "Web Development"
...       }
...     },
...     {
...       $lookup: {
...         from: "students",
...         localField: "student_id",
...         foreignField: "_id",
...         as: "studentInfo"
...       }
...     },
...     {
...       $unwind: "$studentInfo"
...     },
...     {
...       $match: {
...         "studentInfo.department": "Computer Science",
...         marks: { $gt: 80 }
...       }
...     },
...     {
...       $project: {
...         _id: 0,
...         studentName: "$studentInfo.name",
...         department: "$studentInfo.department",
...         courseTitle: "$courseInfo.title",
...         marks: 1
...       }
...     }
... ])
```

### ➢ Insights:

- ✓ **Nothing will show up because there are no students in the** Computer Science department who scored more than 80 in 'Web Development'. Simple array querying in MongoDB.
- ✓ **Access nested fields using dot notation.**
- ✓ **Combine multiple field conditions.**
- ✓ **Query subdocuments efficiently.**
- ✓ **Focused filtering by department and performance.**

## Problem Statements:-7
## Advanced Aggregation (Challenge Level).

- **Aggregation** means **processing data records** and **returning computed results** — like totals, averages, or filtered summaries.
- In MongoDB, aggregation is done using the **Aggregation Pipeline**, where data passes through **stages** that transform it.

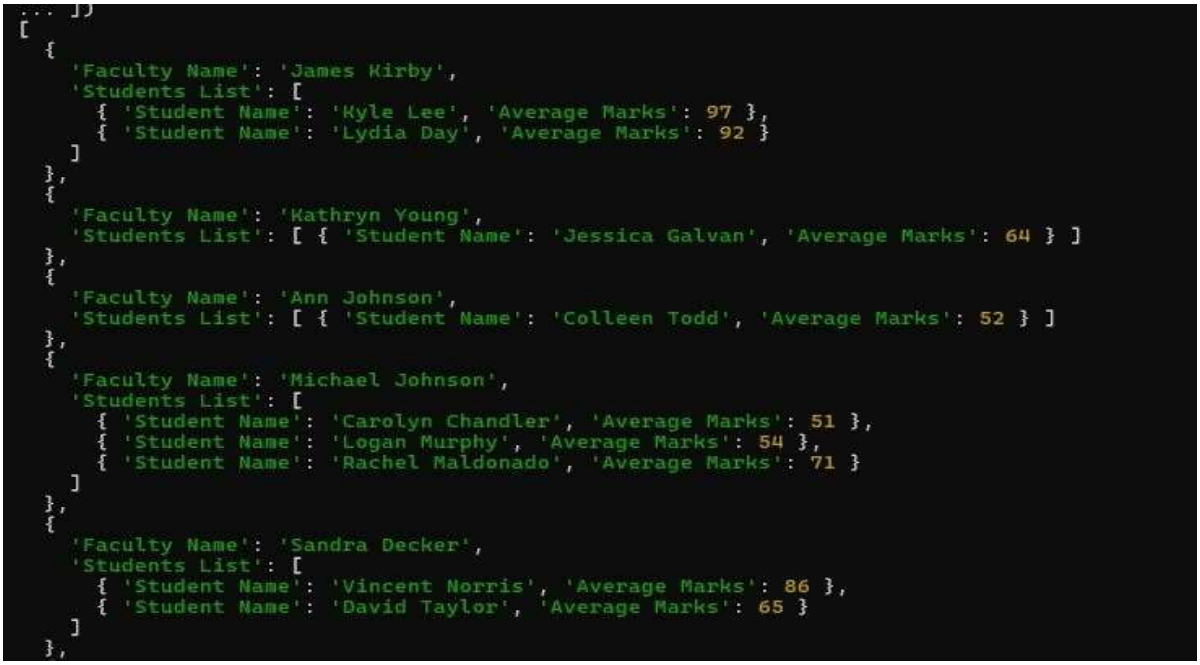
## Question:-13

For each faculty member, list the names of all students enrolled in their courses along with average marks per student per faculty.

## Query & Output:-

```
collage> db.faculty.aggregate([
...   {  // Name:Anuj Shrivastava,Regestation No:1240258108
...     $lookup: {
...       from: "courses",
...       localField: "courses",
...       foreignField: "_id",
...       as: "taughtCourses"
...     }
...   },
...   {
...     $unwind: "$taughtCourses"
...   },
...   {
...     $lookup: {
...       from: "enrollments",
...       localField: "taughtCourses._id",
...       foreignField: "course_id",
...       as: "enrolledStudents"
...     }
...   },
...   {
...     $unwind: "$enrolledStudents"
...   },
...   {
...     $lookup: {
...       from: "students",
...       localField: "enrolledStudents.student_id",
...       foreignField: "_id",
...       as: "studentInfo"
...     }
...   },
...   {
...     $unwind: "$studentInfo"
...   },
...   {
...     $group: {
...       _id: {
...         facultyName: "$name",
...         studentName: "$studentInfo.name"
...       },
...       averageMarks: {
...         $avg: "$enrolledStudents.marks"
...       }
...     }
...   },
...   {
...     $group: {
...       _id: "$_id.facultyName",
...       studentEnrollments: {
...         $push: {
...           studentName: "$_id.studentName",
...           averageMarks: {
...             $round: ["$averageMarks", 2]
...           }
...         }
...       }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       facultyName: "$_id",
...       studentEnrollments: 1
...     }
...   },
...   {
...     $sort: {
...       facultyName: 1
...     }
...   }
... ])
```

## Output:-



```
... ])
[
  {
    'Faculty Name': 'James Kirby',
    'Students List': [
      { 'Student Name': 'Kyle Lee', 'Average Marks': 97 },
      { 'Student Name': 'Lydia Day', 'Average Marks': 92 }
    ]
  },
  {
    'Faculty Name': 'Kathryn Young',
    'Students List': [ { 'Student Name': 'Jessica Galvan', 'Average Marks': 64 } ]
  },
  {
    'Faculty Name': 'Ann Johnson',
    'Students List': [ { 'Student Name': 'Colleen Todd', 'Average Marks': 52 } ]
  },
  {
    'Faculty Name': 'Michael Johnson',
    'Students List': [
      { 'Student Name': 'Carolyn Chandler', 'Average Marks': 51 },
      { 'Student Name': 'Logan Murphy', 'Average Marks': 54 },
      { 'Student Name': 'Rachel Maldonado', 'Average Marks': 71 }
    ]
  },
  {
    'Faculty Name': 'Sandra Decker',
    'Students List': [
      { 'Student Name': 'Vincent Norris', 'Average Marks': 86 },
      { 'Student Name': 'David Taylor', 'Average Marks': 65 }
    ]
  },
```

- **Insights:**
  - ✓ **$** Multi-level joins using $lookup.
  - ✓ $addToSet to avoid duplicate student names.
  - ✓ $avg to compute average marks per faculty.
  - ✓ Real-world aggregation chaining.

## Question:-14

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

## Query & Output:-

```
collage> db.activities.aggregate([
...   {    // Name:Anuj Shrivastava,Regestation No:1240258108
...     $group: {
...       _id: "$type",
...       uniqueParticipants: {
...         $addToSet: "$student_id"
...       }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       activityType: "$_id",
...       participantCount: {
...         $size: "$uniqueParticipants"
...       }
...     }
...   },
...   {
...     $sort: {
...       participantCount: -1
...     }
...   },
...   {
...     $limit: 1
...   }
... ])
[ { activityType: 'Hackathon', participantCount: 35 } ]
```

- **Insights:**
  - ✓ $unwind to count array elements.
  - ✓ $group and $sum for totals.
  - ✓ $sort to rank results.
  - ✓ Identify “most popular” entities.