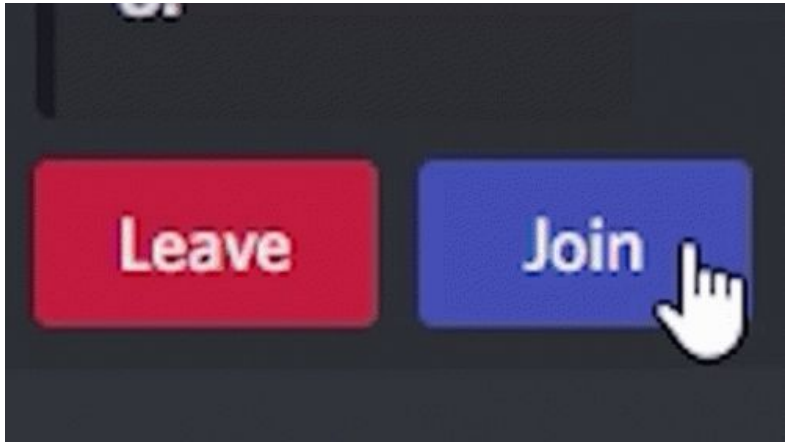
Newton School of Technology

22

MongoDB Querying & Aggregation

Join the Lecture on NS Portal :)

Newton School of Technology



Dataset Overview

Newton School of Technology

Collection Name: students

Each document includes:

- name, age, major
- subjects (array)
- scores (array of embedded documents)
- Optional graduation_Date

Ideal for demonstrating:

- Array queries
- Embedded document queries
- Aggregation analytics

Dataset

Querying Related Documents

Newton School of Technology

MongoDB allows querying:

- Simple fields
- Array values
- Embedded documents
- Nested conditions
- Existence of fields

Designed to retrieve related information without joins.

Querying Simple Fields

Newton School of Technology

Example: Students majoring in Computer Science

```
db.students.find({ major: "Computer Science" })
```

Fetches all documents where `major` matches exactly.

Querying Array Fields

Newton School of Technology

Example: Students studying “Math”

```
db.students.find({ subjects: "Math" })
```

Matches array elements automatically.

Newton School of Technology

Simple Field Query vs Array Field Query

Short Summary Difference

Newton School of Technology

Feature	Simple Field Query	Array Field Query
Data type	Single value	Multiple values (array)
Example field	`major: "CS"`	`subjects: ["Math", "AI"]`
Query	`{ major: "CS" }`	`{ subjects: "Math" }`
Matching	Exact equality	Matches any element inside array
Behavior	Straight comparison	MongoDB searches inside array automatically

Querying Embedded Documents

Newton School of Technology

Example: Students scoring 95 in Math

```
db.students.find({
  scores: { $elemMatch: { subject: "Math", score: 95 } }
})
```

`$elemMatch` ensures both conditions match the same embedded document.

Comparison & Logical Queries

Newton School of Technology

Comparison:

```
db.students.find({ age: { $gt: 22 } })
```

Logical:

```
db.students.find({
  $or: [
    { major: "Computer Science" },
    { major: "Information Technology" }
  ]
})
```

Aggregation Overview

Newton School of Technology

Aggregation performs:

- Summaries
- Analytics
- Grouping
- Calculations
- Data transformation

Uses multiple stages in a pipeline.

Key Aggregation Operators

Newton School of Technology

- `$match` – Filter
- `$group` – Aggregate
- `$project` – Select/reshape fields
- `$sort` – Sorting
- `$unwind` – Flatten arrays
- `$count` – Counts
- `$avg`, `$sum`, `$min`, `$max` – Computations

Counting Documents

Newton School of Technology

Find total number of documents in students collection.

```
db.studentdetails.aggregate([
  { $count: "Total Students" }
])
```

Result: Number of documents in students collection.

Grouping by Major

Newton School of Technology

Find how many students are there in each Department

```
db.studentdetails.aggregate(
[
  { $group:
        { _id: "$department",
        total: { $sum: 1 } } }
]
)
```

This pipeline groups students by their Department and uses $sum: 1

to count how many students belong to each Department.

We can also use $count:{} to count all students in each department.

Average Score in each subject

Newton School of Technology

Find the average score for in each subject

```
db.studentdetails.aggregate([
  { $unwind: "$subjects" },
  {
    $group: {
      _id: "$subjects.name",
      avgMarks: { $avg: "$subjects.marks" }
    }
  }
])
```

Average Score in DBMS

Newton School of Technology

Find the average score for all students in DBMS subject and round it off to two decimal places.

```
db.studentdetails.aggregate([
  { $unwind: "$subjects" },
  { $match: { "subjects.name": "DBMS" } },
  { $group: { _id: "DBMS", avgScore: { $avg: "$subjects.marks" } } },
  {$project:{_id:1,avgScore:{$round:["$avgScore",2]}}}
])
```

This pipeline calculates the average “DBMS” score by unwinding(separating) all scores, filtering only DBMS entries, and then grouping them to compute the average using $avg & further rounding off using $round operator in $project.

Highest Scorer in DBMS

Newton School of Technology

```
db.studentdetails.aggregate([
  { $unwind: "$subjects" },
  { $match: { "subjects.name": "DBMS" } },
  { $sort: { "subjects.marks": -1 } },
  { $limit: 1 }
])
```

This pipeline finds the highest DBMS score by unwinding all subjects, filtering only DBMS scores, sorting them from highest to lowest, and returning just the top result.

Aggregation Pipeline

Newton School of Technology

A MongoDB aggregation pipeline processes data step-by-step, where each stage performs a specific transformation and then passes the modified documents to the next stage.
Together, these stages form a pipeline that transforms raw data into meaningful analytical output.

Pipeline = series of stages

- Stage 1: `$match`
- Stage 2: `$unwind`
- Stage 3: `$group`
- Stage 4: `$sort`

Each stage transforms documents and passes them forward.

Total Score per Student

Newton School of Technology

```
db.studentdetails.aggregate([
  { $unwind: "$subjects" },
  { $group: {
      _id: "$name",
      totalScore: { $sum: "$subjects.marks" }
  }},
  { $sort: { totalScore: -1 } }
])
```

This pipeline expands each student's scores with **$unwind**, then **$group** sums all the score values to compute each student's **totalScore**, and finally **$sort** arranges them in descending order so the highest-scoring students appear first.

Average Score per Student

Newton School of Technology

```
db.studentdetails.aggregate([
  { $unwind: "$subjects" },
  { $group: {
      _id: { name: "$name" },
      avgScore: { $avg: "$subjects.marks" }
  }},
  { $sort: { avgScore: -1 } }
])
```

This pipeline uses **$unwind** to split each student's scores into separate documents, then **$group** (by name) to calculate each student's **average score** with **$avg**, and finally **$sort** orders them in descending order to show the highest-performing students first.

Highest Score per Student

Newton School of Technology

```
db.studentdetails.aggregate([
  { $unwind: "$subjects" },
  { $group: {
      _id: "$name",
      maxScore: { $max: "$subjects.marks" }
  }},
  { $sort: { maxScore: -1 } }
])
```

This pipeline uses **$unwind** to split each student's scores into individual entries, then **$group** (by name) to find each student's **highest score** using **$max**, and finally **$sort** arranges the students in descending order of their maxScore to show who achieved the highest single score.

$group

Count students per major

```
db.studentdetails.aggregate([
  {
    $group: {
      _id: "$major",
      totalStudents: { $sum: 1 }
    }
  }
]);
```

Counts how many students belong to each **major**.

How it works:

- **$group** groups all documents based on the value of **major**.
- **_id: "$major"** → Each unique major becomes one group.
- **$sum: 1** → Adds 1 for every student in that group → gives student count.

Result:

You get a list of majors with the **number of students** in each major.

$group — When to Use

Newton School of Technology

To group documents and perform aggregations like:

- sum
- count
- max
- min
- average

Use $group when:

- You want summary/aggregation results.
- You need student-wise, subject-wise, major-wise reports.
- You want counts, totals, highest score, lowest score etc.

$lookup — When to Use

Newton School of Technology

Purpose:

To join data from **another collection** into the current pipeline.

Use $lookup when:

- You have related data stored in **two different collections**.
- You need to **combine** or **merge** data (similar to SQL JOIN).
- Example: students + departments

Assume you have another collection:

```
db.departments.insertMany([
  { major: "Computer Science", hod: "Dr. Sharma" },
  { major: "Physics", hod: "Dr. Rao" },
  { major: "Math", hod: "Dr. Bose" }
]);
```

$lookup — Example

Newton School of Technology

Assume you have another collection:

```
db.studentdetails.aggregate([
  {
    $lookup: {
      from: "courses",
      localField: "course_ids",
      foreignField: "_id",
      as: "course_details"
    }
  }
]);
```

Performs a **LEFT JOIN** between `students` and `departments` based on the **major** field.

How it works:

- **localField: "major"** → Read the student's major
- **foreignField: "major"** → Match it with the department's major
- **from: "departments"** → Join with this collection
- **as: "major_details"** → Put matched department info into this new field

Result:

Each student gets an extra field **major_details** containing department info that matches their major

$lookup —

Newton School of Technology

Q. Count how many students are enrolled in each course.Display course name & count of students

```
db.studentdetails.aggregate([
  {
    $lookup: {
      from: "courses",
      localField: "course_ids",
      foreignField: "_id",
      as: "courseDetails" }
  },
  { $unwind: "$courseDetails" },
  { $group: {
      _id: "$courseDetails.name",
      totalStudents: { $sum: 1 }}}
])
```

$sort — When to Use

Newton School of Technology

Purpose:

To sort documents in **ascending** or **descending** order.

Use $sort when:

- You want **top scorers**, **youngest students**, **latest graduation year**, etc.
- You need ordering before or after grouping.

Sort students by age (descending)

```
db.studentdetails.find().sort({ age: -1 });
```

Sort students by highest score

```
db.studentdetails.aggregate([
  { $unwind: "$subjects" },
  {
    $group: {
      _id: "$name",
      maxScore: { $max: "$subjects.marks" }
    }
  },
  { $sort: { maxScore: -1 } }
]);
```

Summary – When to use

Newton School of Technology

Operator	Use When	Example Case (Your Dataset)
$lookup	Need to join 2 collections	Join `students` with `departments`
$group	Need aggregation/summaries	Find highest marks / student count
$sort	Need ordering	Top scorers, youngest students

Summary

Newton School of Technology

- MongoDB queries allow searching arrays, embedded docs, and conditions.
- Aggregation provides deep analytics using `$group`, `$unwind`, `$project`.
- Aggregation pipeline enables multi step data processing.
- Dataset helps demonstrate score analysis and graduation insights.
- **$group** → Groups documents by a field and performs operations like count, sum, avg, max.
- **$sort** → Arranges documents in ascending (1) or descending (-1) order.
- **$lookup** → Joins data from another collection based on matching fields.

Thank You !!

See You Guys in Next Session :)