24st of April 2025

DOCUMENT-BASED QUERYING USING MONGODB

**Introduction**

In this practical session on Document-Based Querying Using MongoDB, we focused on how to retrieve and manage data stored in a document format. MongoDB uses collections and documents instead of traditional tables and rows, making it easier to handle flexible and unstructured data. We learned how to write queries to find specific documents, filter results using conditions, and display only the required fields. This practical helped us understand the basics of querying in MongoDB and how document-based databases can simplify data handling in modern applications.

**Objectives**

* To understand the concept of NoSQL and document-based databases.
* To perform basic data insertion using **insertOne() and insertMany().**
* To practice filtering documents using query operators.
* To apply projection and sorting in queries.
* To get familiar with MongoDB's flexible schema and document structure.

**Manual Data Entry using MongoDB Compass**

As part of our practical on Document-Based Querying Using MongoDB, we began by manually inserting documents using MongoDB Compass, which is a graphical user interface for MongoDB.

**Adding the First Document**

We clicked on the “Add Data” button and selected “Insert Document”. In the document editor, we typed key names without quotation marks (like regno, name, age) to allow Compass to auto-format the structure into proper JSON. We inserted this data:

**Duplicating and Modifying the Document**

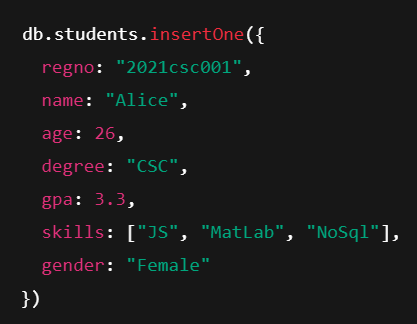
After adding the first document, we used the **“Clone” (Duplicate)** option to make a copy. In this duplicate, we added a new field named gender to include more details in our document.

The final version of the duplicated document looked like:

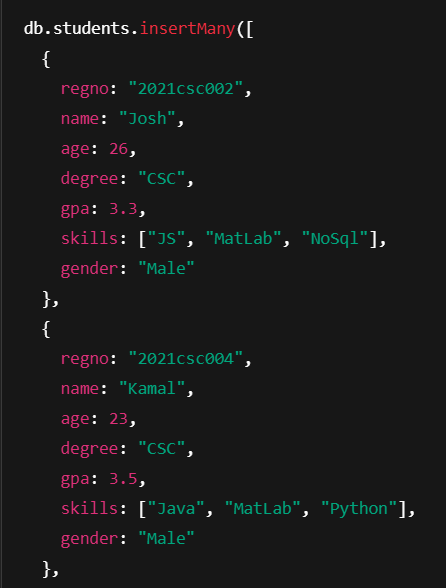


**Adding data using insertOne() Command**

insertOne() inserts a single document into the collection.



**Adding data using insertMany() Command**

insertMany() is used to **bulk insert** multiple documents at once, saving time compared to inserting them individually

**Queries Practiced**

We then performed various basic **find()**, **sort()**, and **projection** queries.

**Console Queries**

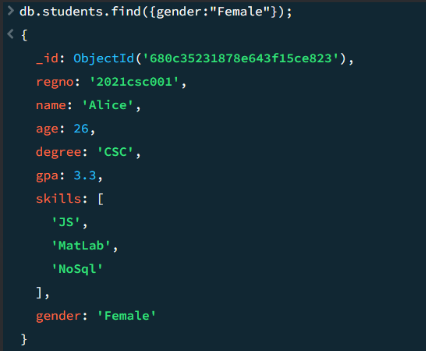
1. Find all documents where gender is "Female".

db.students.find({gender:"Female"})

2. Find the first document where gender is "Female".

|  |
| --- |
| db.students.findOne({gender:"Female"}) |

|  |
| --- |
|  |



|  |
| --- |
|  |

3. Find students whose age is greater than 25.

db.students.find({age:{$gt:25}})

4. Find IT students sorted by GPA in descending order.

db.students.find({'degree':"IT"}).sort({gpa:-1})

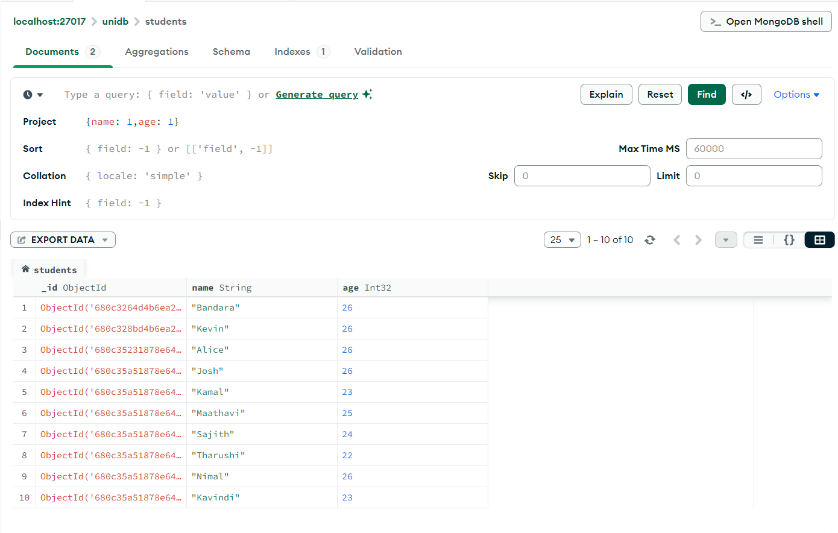
5. Find Male students sorted by age in descending order.

db.students.find({'gender':"Male"}).sort({age:-1})

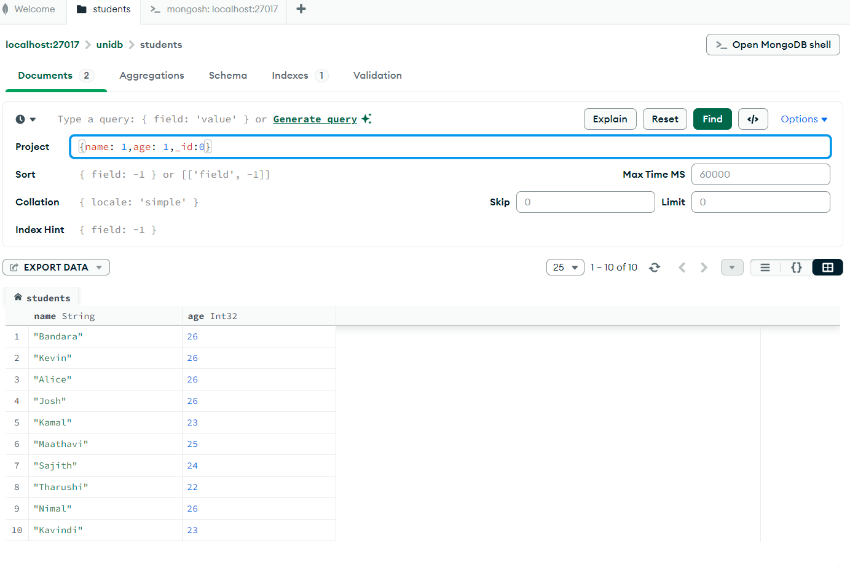
**Normal Queries**

1. Project (show) only name and age fields.

{project: {name: 1, age: 1}}

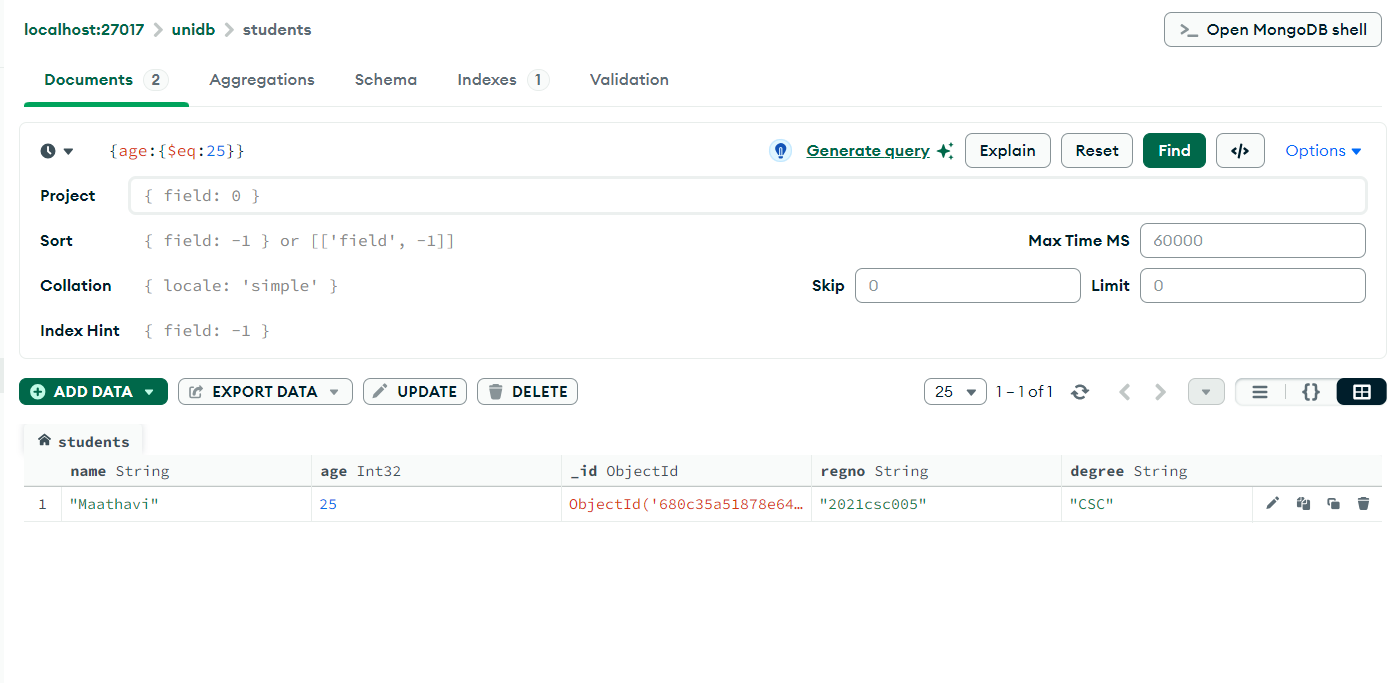


2. Project name and age only (hide \_id ).

{project: {name: 1, age: 1, \_id: 0}}

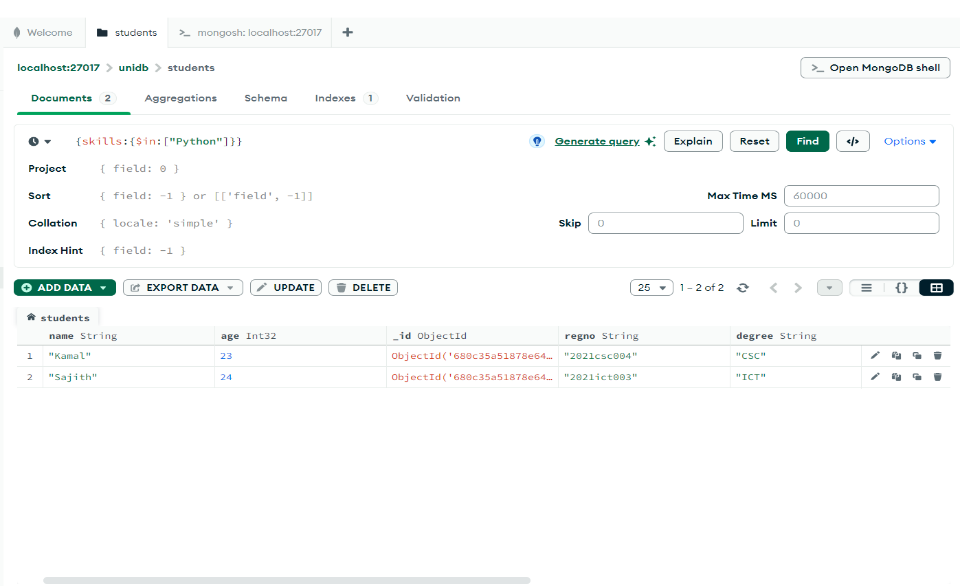
3. Find students aged exactly 25.

{age: {$eq:25}}



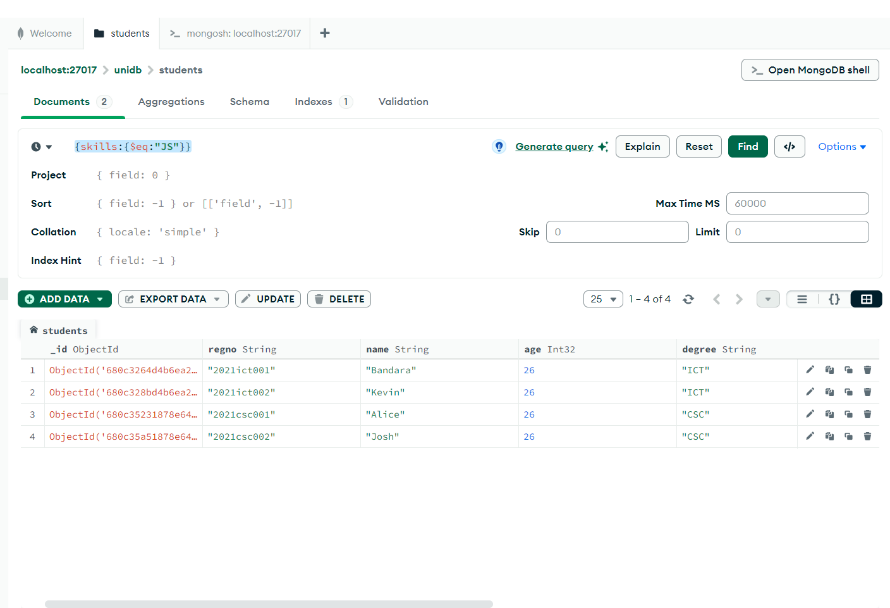
4. Find students who know Python.

{skills: {$in: ["Python"]}}



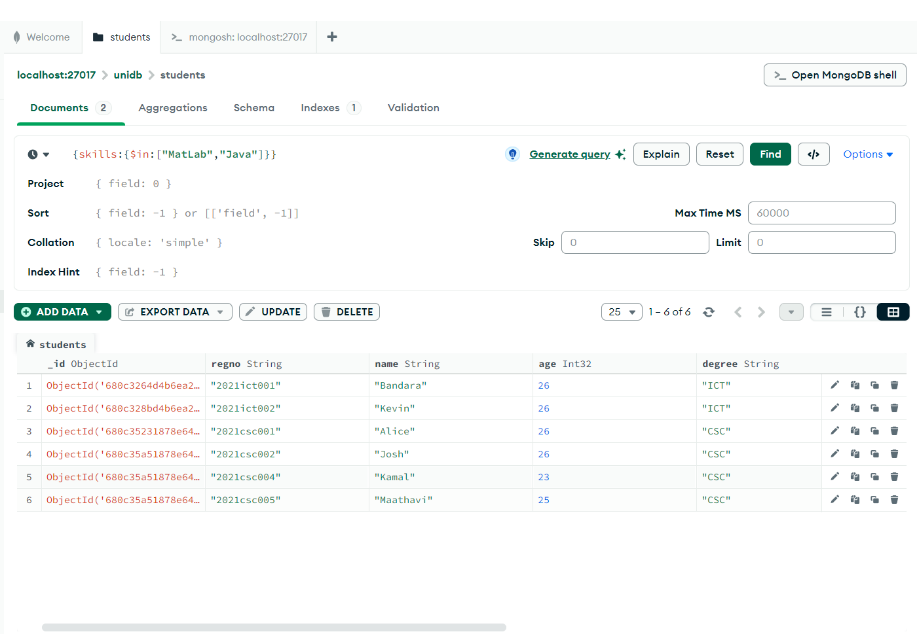
5. Find students who have JS as a skill.

{skills: {$eq: "JS"}}



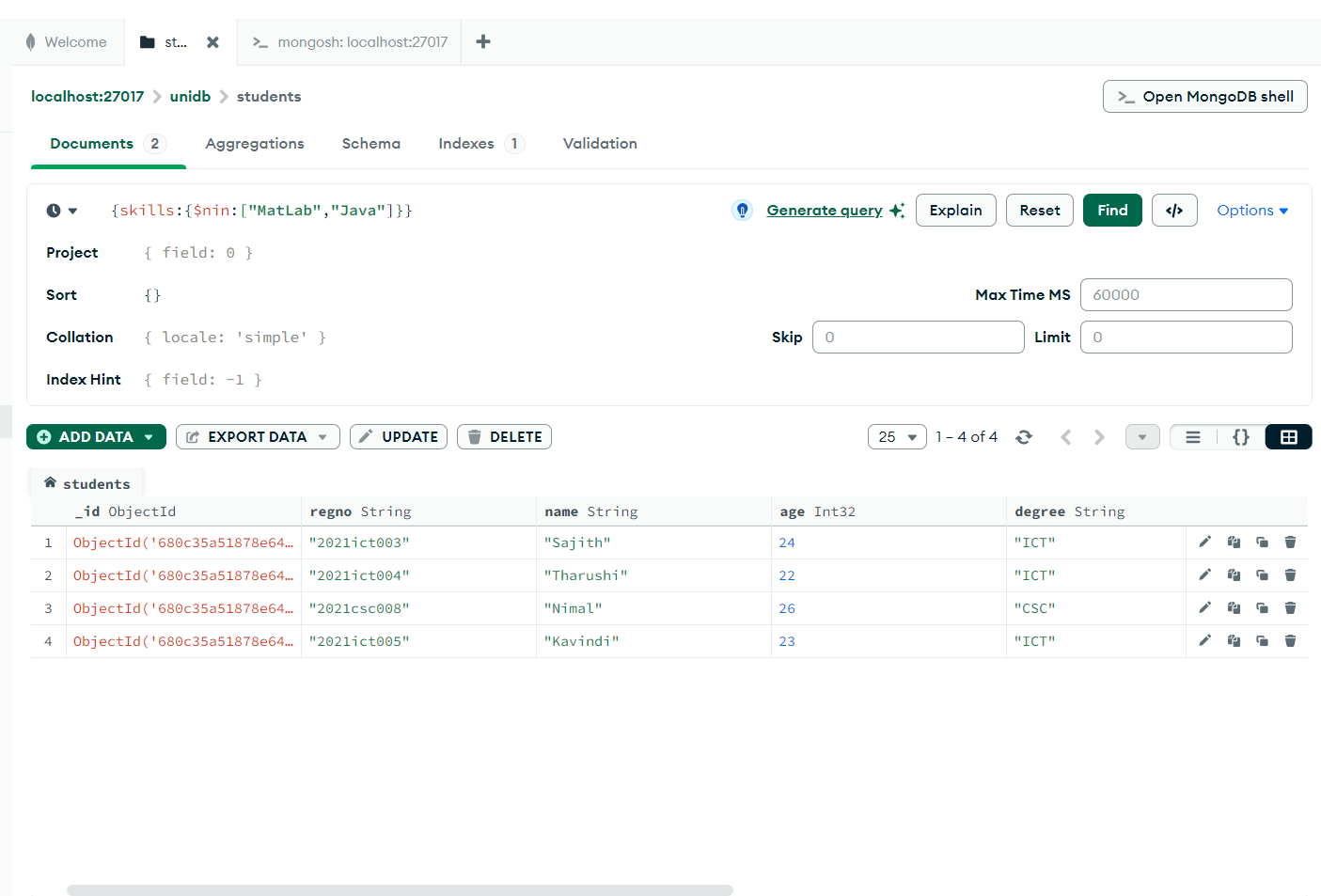
6. Find students who know either MatLab or Java.

{skills: {$in: ["MatLab", "Java"]}}



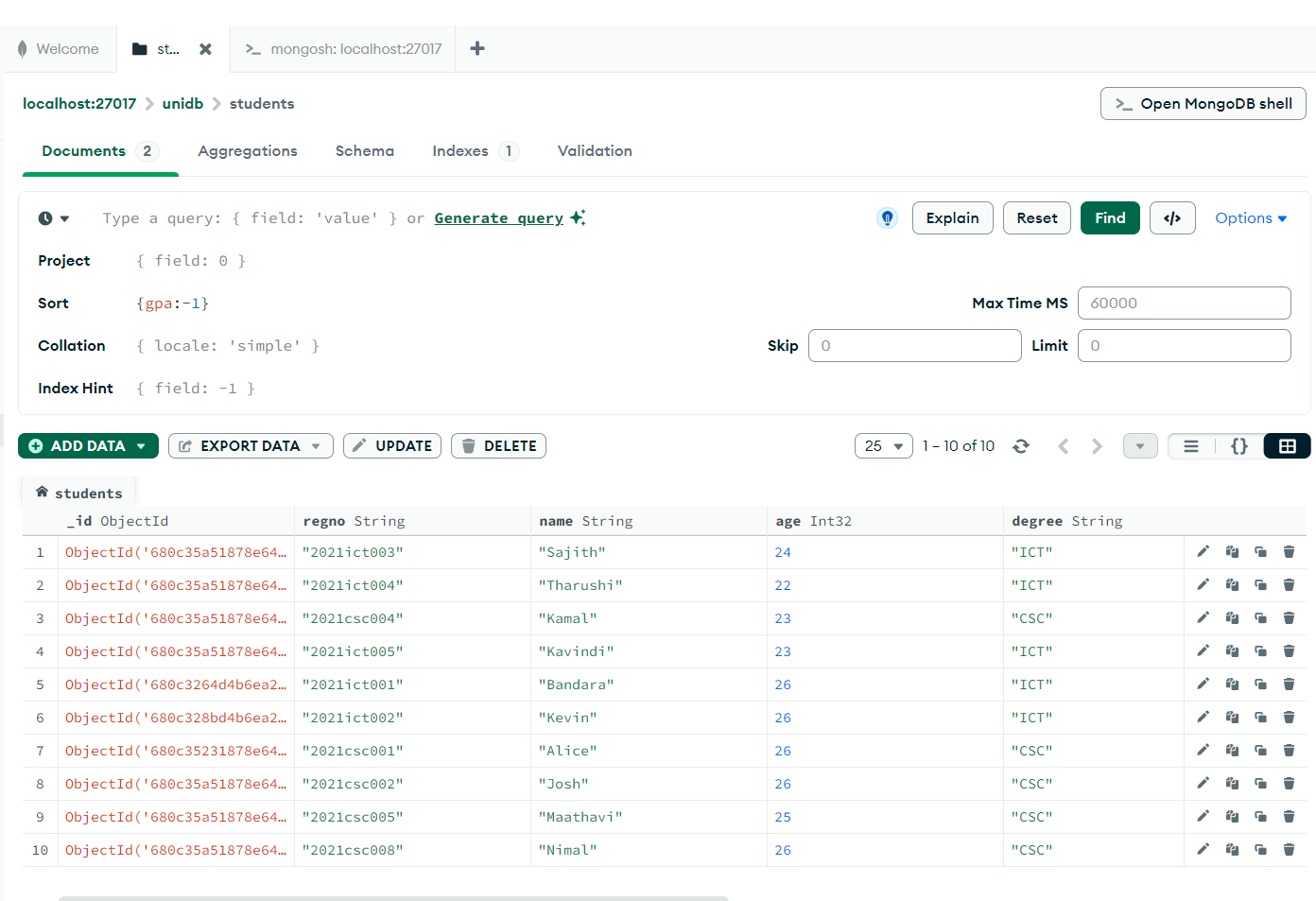
7. Find students who do not know MatLab or Java.

{skills: {$nin: ["MatLab", "Java"]}}



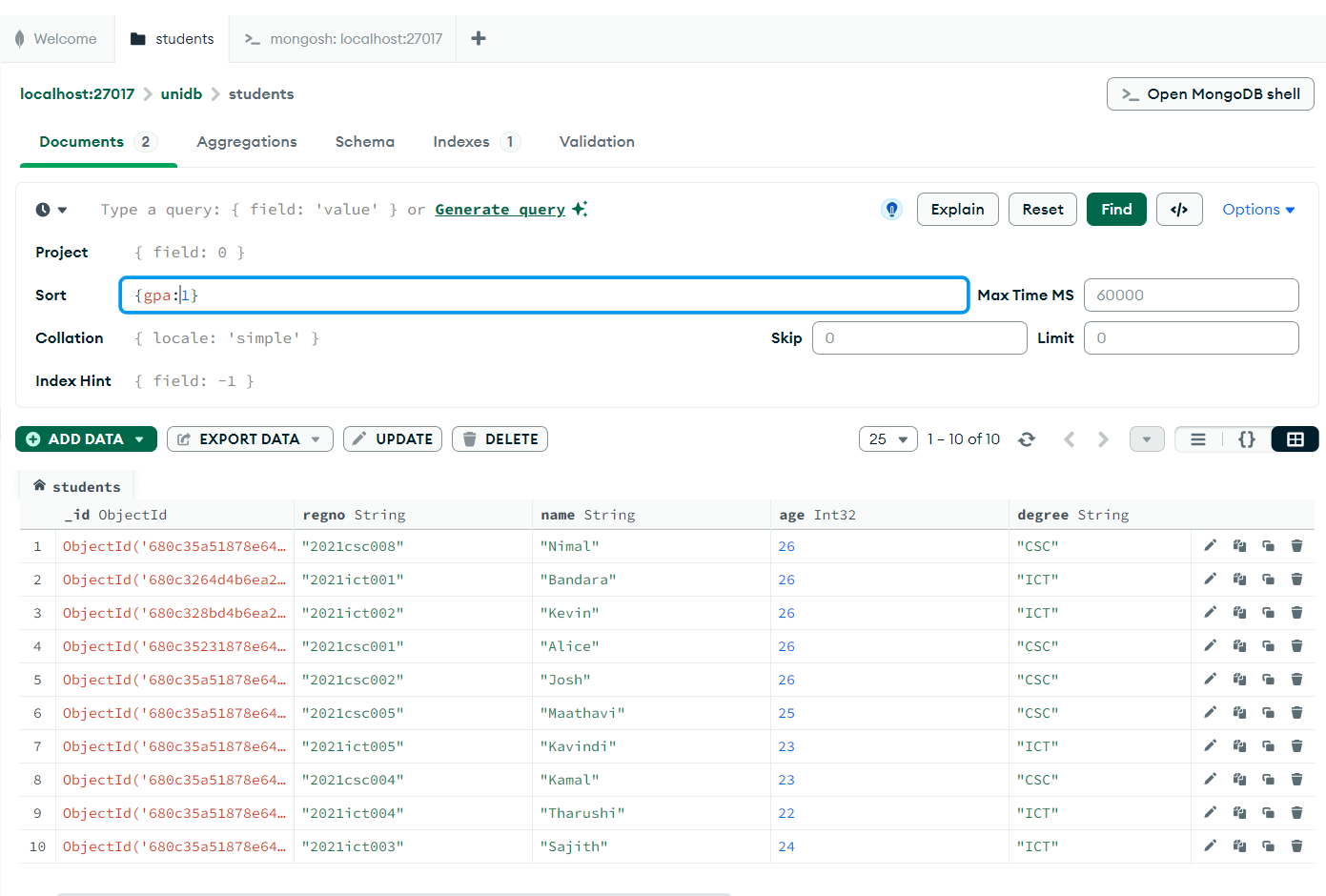
8. Sort by GPA in descending order.

{gpa:-1}



9. Sort by GPA in ascending order.

{gpa:1}



**Sample Data Entered**

 {"regno": "2021ict001", "name": "Bandara", "age": 26, "degree": "ICT", "gpa": 3.3, "skills": ["JS", "MatLab", "NoSql"]}

 {"regno": "2021ict002", "name": "Kevin", "age": 26, "degree": "ICT", "gpa": 3.3, "skills": ["JS", "MatLab", "NoSql"], "gender": "Male"}

 {"regno": "2021csc001", "name": "Alice", "age": 26, "degree": "CSC", "gpa": 3.3, "skills": ["JS", "MatLab", "NoSql"], "gender": "Female"}

 {"regno": "2021csc002", "name": "Josh", "age": 26, "degree": "CSC", "gpa": 3.3, "skills": ["JS", "MatLab", "NoSql"], "gender": "Male"}

 {"regno": "2021csc004", "name": "Kamal", "age": 23, "degree": "CSC", "gpa": 3.5, "skills": ["Java", "MatLab", "Python"], "gender": "Male"}

 {"regno": "2021csc005", "name": "Maathavi", "age": 25, "degree": "CSC", "gpa": 3.3, "skills": ["Java", "PHP", "MatLab", "MySql"], "gender": "Male"}

 {"regno": "2021ict003", "name": "Sajith", "age": 24, "degree": "ICT", "gpa": 3.9, "skills": ["Python", "MongoDB", "Django"], "gender": "Male"}

 {"regno": "2021ict004", "name": "Tharushi", "age": 22, "degree": "ICT", "gpa": 3.7, "skills": ["React", "NodeJS", "Express"], "gender": "Female"}

 {"regno": "2021csc008", "name": "Nimal", "age": 26, "degree": "CSC", "gpa": 2.9, "skills": ["C", "C++", "Assembly"], "gender": "Male"}

 {"regno": "2021ict005", "name": "Kavindi", "age": 23, "degree": "ICT", "gpa": 3.4, "skills": ["HTML", "CSS", "JavaScript"], "gender": "Female"}