
Collection: `students`

Sample Document:

```
{
  "_id": ObjectId("..."),
  "name": "Ayaan Khan",
  "age": 21,
  "gender": "Male",
  "department": "Computer Science",
  "courses": [
    { "name": "MongoDB", "score": 85 },
    { "name": "Python", "score": 90 }
  ],
  "address": {
    "city": "Hyderabad",
    "state": "Telangana",
    "pincode": 500032
  },
  "enrollmentDate": ISODate("2024-08-01T00:00:00Z"),
  "isActive": true
}
```

CRUD Operations

1. **Insert a new student record** with embedded `courses` and `address` data.
2. **Update score** for a course (`Python`) inside the `courses` array.
3. **Delete a student** whose `name` is `"John Doe"` .
4. **Find all students** in the `"Computer Science"` department.

Query Operators

5. Find students where `age` is **greater than 20**.
6. Find students enrolled **between two dates**.
7. Find students who are either in **"Computer Science"** or **"Mathematics"**.
8. Find students **not in the "Mechanical" department**.
9. Find students whose **`courses.score` is greater than 80**.

Aggregation Framework

10. **Group by department** and count students.
11. **Calculate average age** of students per department.
12. **Sort students** by total course score (computed using `$sum` inside `$project`).
13. **Filter only active students** before aggregation.
14. **Group and list unique cities** from the `address` field.

Projections

15. Find students with only `name` , `department` , and `city` fields shown.
16. Exclude the `_id` field from output.
17. Show each student's `name` and total score using `$project` .

Embedded Documents

18. Query students where `address.city = "Hyderabad"`.
19. Update `address.pincode` for a student.
20. Add a new field `landmark` to all `address` objects.

Array Operations

21. Add a new course `"Node.js"` to a student's `courses` array.
22. Remove a course by name `"MongoDB"` from the array.
23. Find students who have enrolled in **both Python and MongoDB**.
24. Use `$elemMatch` to query students where **score in MongoDB > 80**.
