📚 Student Management System (MongoDB Project)

🔹 Project Overview

This project is a simple **Student Management Database** built using **MongoDB**.
It stores student details like name, class, roll number, and subjects with marks.
The project demonstrates **CRUD operations** and **Aggregation Pipeline** concepts of MongoDB, including some advanced aggregation stages.

🔹 Features Implemented

- Added records of 10 students with fields:
    - Name
    - Class
    - Roll Number
    - Subjects (with marks)
- Performed **CRUD Operations**:
    - `insertOne`, `insertMany` → Insert records
    - `updateOne`, `updateMany` with `$set` → Update student details
    - `deleteOne`, `deleteMany` → Remove records
    - `find` with **projection** → Display selected fields
- Implemented **Aggregation Pipeline** with both basic and advanced stages:
    - Basic: `$match`, `$project`, `$group`, `$sort`, `$limit`, `$skip`, `$unwind`

- Advanced: `$addFields`, `$unset` (to remove fields from output), computed fields and reshaping documents

---

🔹 Sample Aggregation Queries (Advanced examples)

1) Average marks per class (using `$unwind` + `$group`)

```
db.student.aggregate([
  { $unwind: "$subjects" },
  { $group: { _id: "$class", avgMarks: { $avg: "$subjects.marks" } } },
  { $project: { class: "$_id", avgMarks: { $round: ["$avgMarks", 2] }, _id: 0 } }
])
```

2) Add a field `totalMarks` per student (using `$addFields` and `$sum` over the subjects array)

```
db.student.aggregate([
  { $addFields: { totalMarks: { $sum: "$subjects.marks" } } },
  { $project: { name: 1, class: 1, totalMarks: 1, _id: 0 } }
])
```

3) Get top scorer in each class (combine `$unwind`, `$group`, `$sort`, `$first`)

```
db.student.aggregate([
  { $unwind: "$subjects" },
  { $group: {
      _id: { studentId: "$_id", name: "$name", class: "$class" },
      total: { $sum: "$subjects.marks" }
    }
  },
  { $group: { _id: "$_id.class", topStudent: { $max: { name: "$_id.name", total: "$total" } } } },
  { $project: { class: "$_id", topStudent: 1, _id: 0 } }
])
```

> Note: Depending on structure you may prefer computing `totalMarks` first with `$addFields`, then `$group` by class to pick the max.

4) Remove sensitive or unnecessary fields from output (using `$unset`)

```
db.student.aggregate([
  { $project: { name: 1, class: 1, subjects: 1 } },
  { $unset: ["rollNumber"] }
])
```

5) Reshape documents: show subject-wise performance with average per subject (combine `$unwind`, `$group`)

```
db.student.aggregate([
  { $unwind: "$subjects" },
  { $group: { _id: "$subjects.sub", avgMarks: { $avg: "$subjects.marks" } } },
  { $project: { subject: "$_id", avgMarks: { $round: ["$avgMarks", 2] }, _id: 0 } },
  { $sort: { avgMarks: -1 } }
])
```

🔹 Technologies Used

- MongoDB
- Mongo Shell / MongoDB Compass

(Node.js removed as requested — this repo focuses on database and aggregation work only.)

🔹 How to Run / Reproduce

1. Clone the repository (if using GitHub).
2. Open Mongo Shell or MongoDB Compass and create a database (e.g., `studentDB`) and collection `student`.
3. Use `insertMany()` with the provided sample data (add your 10 student documents).
4. Copy and paste aggregation queries from this README into the shell or Compass aggregation builder to reproduce results.

🔹 Sample Output (example)

```
[
  { "class": "5th", "avgMarks": 78.00 },
  { "class": "6th", "avgMarks": 82.00 }
]
```

🔹 Learning Outcomes

- Practical experience building a small but real-world MongoDB dataset.
- Confidence using both basic CRUD operations and advanced aggregation stages like `$addFields` and `$unset`.
- Ability to analyze data (average calculations, top-performers, subject-wise analytics) using aggregation pipeline.

✍️ **Author**: [Shruti Sharma]

If you want, I can also add:

- Sample `insertMany()` data (10 students) to the repo, or
- Screenshots of MongoDB Compass showing query outputs, or
- A short `queries.md` file with labeled queries for interview reference.