Student Database Management with MongoDB

Dataset

This project uses a simulated dataset to represent a real-world student database. The dataset was created in MongoDB with a database named **StudentDB** and a collection named **students**. Each record holds the following fields:

1. Roll – Unique integer identifier for each student.
2. Name – Full name of the student.
3. Course – Academic program (e.g., Math, Physics, Chemistry, Biology).
4. Marks – Numerical score obtained by the student.
5. City – The city where the student resides.
6. EmailID – Contact email of the student.

A total of 15 student records were inserted, covering multiple courses and cities. Marks range from very low (below passing) to excellent, making the dataset suitable for meaningful filtering, sorting, updates, and analysis.

Queries Executed with Code Snippets

Below are the key operations performed on the dataset along with the Python code snippets using the PyMongo library.

■■ Display all students

Fetches and displays all student records stored in the database.

```
for s in students.find({}):
print(s)
```

■■ Find students enrolled in a specific course

Retrieves all students who are studying in the 'Math' course.

```
course = "Math" for s in
students.find({"Course": course}):
print(s)
```

■■ Find students with marks greater than 75

Filters and displays students who scored more than 75 marks.

```
for s in students.find({"Marks": {"$gt": 75}}):
print(s)
```

■■ Retrieve students from a specific city

Fetches all students who are residents of Kochi.

```
city = "Kochi" for s in
students.find({"City": city}):
print(s)
```

■■ Update a student's email

Updates the email ID of the student with Roll Number 11.

```
students.update_one({"Roll": 11}, {"$set": {"EmailID":
"updated.email@gmail.com"}})
```

■■ Increase marks by 10 for Physics students

Increases the marks of all students enrolled in the Physics course by 10.

```
students.update_many({"Course": "Physics"}, {"$inc": {"Marks": 10}})
```

■■ Delete students with marks < 40

Removes students who failed (marks less than 40) from the collection.

```
students.delete_many({"Marks": {"$lt": 40}})
```

■■ Count students by course

Counts the number of students in each course.

```
students.aggregate([{"$group": {"_id": "$Course", "count": {"$sum": 1}}}])
```

■■ Calculate average marks per course

Computes the average marks of students for each course.

```
students.aggregate([{"$group": {"_id": "$Course", "avgMarks": {"$avg":
"$Marks"}}}])
```

■■ Find top 3 students by marks

Retrieves the top 3 highest scoring students.

```
students.find({}).sort("Marks", -1).limit(3)
```

■■ Sort students by marks in descending order

Displays all students sorted from highest to lowest marks.

```
students.find({}).sort("Marks", -1)
```

■■ Export data to JSON

Exports all student records to a JSON file named students.json.

```
import json with open("students.json", "w", encoding="utf-8")
as f:    json.dump(list(students.find({}, {"_id": 0})), f,
indent=2)
```

Insights

- Physics students benefited from a +10 mark increment.
- Top 3 students were identified quickly using sorting queries.
- Students with marks below 40 were deleted to maintain meaningful data.
- Average marks per course provided useful academic insights.
- MongoDB aggregation simplified student performance analysis.

How to Run the Project

1. Install MongoDB and start the MongoDB service.
2. Install Python and the required dependency: pip install pymongo.
3. Run the script using: python student_mongo.py.
4. Check MongoDB Compass or the terminal output to see results.
5. Verify that a file named students.json is created in your project folder.