📌 20 MongoDB Tasks with Answers

◇ Basic CRUD Operations

1️⃣ Insert a new student record into the database.

```
db.students.insertOne({ name: "Lucas", roll: 21, age: 22, class: 11, marks: 90 })
```

2️⃣ Retrieve all student records from the collection.

```
db.students.find()
```

3️⃣ Find students who have scored more than 80 marks.

```
db.students.find({ marks: { $gt: 80 } })
```

4️⃣ Retrieve details of a specific student by name.

```
db.students.findOne({ name: "Alice" })
```

5️⃣ Update a student's marks by increasing them by 5.

```
db.students.updateOne({ name: "Bob" }, { $inc: { marks: 5 } })
```

6️⃣ Delete a student based on their roll number.

```
db.students.deleteOne({ roll: 16 })
```

◇ Aggregation Tasks

7️⃣ Calculate the total marks of all students.

```
db.students.aggregate([
  { $group: { _id: null, totalMarks: { $sum: "$marks" } } }
])
```

8️⃣ **Find the average marks of students in class 10.**

```
db.students.aggregate([
  { $match: { class: 10 } },
  { $group: { _id: "$class", avgMarks: { $avg: "$marks" } } }
])
```

9️⃣ **Identify the student(s) with the highest marks.**

```
db.students.aggregate([
  { $group: { _id: null, maxMarks: { $max: "$marks" } } }
])
```

🔟 **Find the student(s) with the lowest marks.**

```
db.students.aggregate([
  { $group: { _id: null, minMarks: { $min: "$marks" } } }
])
```

◇ Projection & Conditional Queries

1️⃣1️⃣ **Display only the name and marks of students, excluding other details.**

```
db.students.find({}, { _id: 0, name: 1, marks: 1 })
```

1️⃣2️⃣ **Rename the `roll` field to `studentID` in the output.**

```
db.students.aggregate([
  { $project: { _id: 0, name: 1, studentID: "$roll", class: 1 } }
])
```

1️⃣3️⃣ **Add 10 bonus marks to each student and display the updated marks.**

```
db.students.aggregate([
  { $project: { name: 1, roll: 1, class: 1, updatedMarks: { $add: ["$marks", 10] } } }
])
```

1 4 Categorize students as Pass or Fail (Pass if marks >= 40).

```
db.students.aggregate([
  { $project: { name: 1, marks: 1, status: { $cond: { if: { $gte: ["$marks", 40] }, then: "Pass", else: "Fail" } } } }
])
```

◇ Advanced Filtering & Sorting

1 5 Retrieve students who are 21 years or older.

```
db.students.find({ age: { $gte: 21 } })
```

1 6 Find students whose marks are between 80 and 90.

```
db.students.find({ marks: { $gte: 80, $lte: 90 } })
```

1 7 Sort students by their marks in descending order.

```
db.students.find().sort({ marks: -1 })
```

◇ Counting & Limit

1 8 Count the total number of students in class 10.

```
db.students.countDocuments({ class: 10 })
```

1 9 Retrieve the top 3 students with the highest marks.

```
db.students.find().sort({ marks: -1 }).limit(3)
```

2 0 Identify any duplicate roll numbers in the collection.

```
db.students.aggregate([
  { $group: { _id: "$roll", count: { $sum: 1 } } },
  { $match: { count: { $gt: 1 } } }
])
```
