1. Sum of Marks for Each Student

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

- **Groups by `roll` and calculates total marks.**

---

2. Average Marks for Each Student

```
db.students.aggregate([
  {
    $group: {
      _id: "$roll",
      name: { $first: "$name" },
      averageMarks: { $avg: "$marks" }
    }
  }
])
```

- **Groups by `roll` and calculates the average marks.**

---

3. Minimum Marks for Each Student

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

- **Groups by `roll` and finds the minimum marks.**

---

4. Maximum Marks for Each Student

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

- **Groups by `roll` and finds the maximum marks.**

---

5. Sum of Marks for a Particular Student (e.g., Alice)

```
db.students.aggregate([
  { $match: { name: "Alice" } },
  {
    $group: {
      _id: "$roll",
      name: { $first: "$name" },
      marks10: {$first:"$marks"},
      marks11: {$first:"$marks"},
      totalMarks: { $sum: "$marks" }
    }
  }
])
```

- **Show the sum of marks 10 + 11**

- ### `$group` and `$project` in MongoDB Aggregation

Both `$group` and `$project` are essential **aggregation operators** in MongoDB, but they serve different purposes:

Operator	Purpose
`$group`	Groups documents and performs aggregate calculations (like sum, avg, min, max, etc.)
`$project`	Selects, renames, and modifies fields (like including/excluding specific fields, calculations, and transformations)

1. `$add` (Addition)

👉 **Example: Add 10 to each student's marks**

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

☑ **Effect:** Adds 10 to every student's marks.

---

2. `$subtract` (Subtraction)

👉 **Example: Subtract 5 from each student's marks**

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      roll: 1,
      class: 1,
      reducedMarks: { $subtract: ["$marks", 5] }
    }  }])
```

☑ **Effect:** Reduces every student's marks by 5.

---

3. `$multiply` (Multiplication)

👉 **Example: Multiply marks by 2**

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      roll: 1,
      class: 1,
      doubleMarks: { $multiply: ["$marks", 2] }
    }}])
```

☑ **Effect:** Each student's marks are doubled.

---

4. `$divide` (Division)

👉 **Example: Divide marks by 2**

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      roll: 1,
      class: 1,
      halfMarks: { $divide: ["$marks", 2] }
    }
  }
])
```

☑ **Effect:** Each student's marks are halved.

---

5. `$mod` (Modulus)

👉 **Example: Find remainder when dividing marks by 3**

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      roll: 1,
      class: 1,
      remainder: { $mod: ["$marks", 3] }
    }
  }
])
```

☑ **Effect:** Calculates `marks % 3`.

---

1. `$cond` (If-Else Condition)

👉 **Example: If marks >= 90, label as "Excellent", else "Good"**

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      roll: 1,
      class: 1,
      performance: {
        $cond: { if: { $gte: ["$marks", 90] }, then: "Excellent", else:
"Good" }
      }
    }  }])
```

☑ **Effect:** If `marks` is 90 or more → `"Excellent"`, else `"Good"`.

---

2. `$ifNull` (Set Default Value if Null)

👉 **Example: If marks are null, set default to 50**

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      roll: 1,
      class: 1,
      finalMarks: { $ifNull: ["$marks", 50] }
    }  }])
```

☑ **Effect:** If `marks` is missing or `null`, it will be set to **50**.

`$gte`, `$lte`, `$eq` (Comparison in Conditional Logic)

👉 **Example: Check if marks are passing (>= 40)**

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      roll: 1,
      class: 1,
      isPass: { $gte: ["$marks", 40] }
    }
  }
])
```

☑ **Effect:** Returns **true** if marks are `>= 40`, else **false**.

---

3. `$switch` (Multiple Conditions - Like `if-else if-else`)

👉 **Example: Categorize students based on marks**

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      roll: 1,
      class: 1,
      grade: {
        $switch: {
          branches: [
            { case: { $gte: ["$marks", 90] }, then: "A+" },
            { case: { $gte: ["$marks", 80] }, then: "A" },
            { case: { $gte: ["$marks", 70] }, then: "B" }
          ],
          default: "C"
        }
      }
    }
  }
])
```

☑ **Effect:**

- `marks >= 90` → `"A+"`
- `marks >= 80` → `"A"`
- `marks >= 70` → `"B"`
- Else → `"C"`
