

1. AND, OR, LIMIT, SORT in MongoDB

◆ A. AND Operator

Returns documents that match all conditions.

javascript

CopyEdit

```
db.students.find({ age: { $gt: 20 }, course: "MCA" })
```

Alternative using \$and:

javascript

CopyEdit

```
db.students.find({
```

```
  $and: [
```

```
    { age: { $gt: 20 } },
```

```
    { course: "MCA" }
```

```
]
```

```
})
```

◆ B. OR Operator

Returns documents that match any condition.

javascript

CopyEdit

```
db.students.find({
```

```
  $or: [
```

```
    { age: { $lt: 21 } },
```

```
    { course: "BCA" }
```

```
]
```

```
})
```

◆ C. LIMIT Records

Limits the number of documents returned.

javascript

CopyEdit

```
db.students.find().limit(2)
```

- ◆ D. SORT Records

Sort documents in ascending (1) or descending (-1) order.

javascript

CopyEdit

```
// Sort by age in ascending order
```

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

```
// Sort by marks in descending order
```

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

 2. Indexing, Advanced Indexing, Aggregation, Map-Reduce

- ◆ A. Indexing

Create an index on the name field:

javascript

CopyEdit

```
db.students.createIndex({ name: 1 })
```

- ◆ B. Advanced Indexing (Compound Index)

Create an index on multiple fields:

javascript

CopyEdit

```
db.students.createIndex({ course: 1, age: -1 })
```

View all indexes:

javascript

CopyEdit

```
db.students.getIndexes()
```

Drop an index:

javascript

CopyEdit

```
db.students.dropIndex({ name: 1 })
```

◆ **C. Aggregation Framework**

Example: Group students by course and calculate average marks.

javascript

CopyEdit

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

◆ **D. Map-Reduce Example**

Count number of students in each course:

javascript

CopyEdit

```
var mapFunction = function() {
  emit(this.course, 1);
};

var reduceFunction = function(key, values) {
  return Array.sum(values);
};
```

```
db.students.mapReduce(mapFunction, reduceFunction, { out: "course_counts" })
```

```
// View results
```

```
db.course_counts.find()
```

 **3. Arithmetic Operations Program in MongoDB**

You can use aggregation operators like `$add`, `$subtract`, `$multiply`, `$divide`, and `$mod`.

Example:

```
javascript
CopyEdit
db.students.insertOne({
  name: "Sumit",
  marks1: 75,
  marks2: 85
})

// Perform all arithmetic operations
db.students.aggregate([
{
  $project: {
    name: 1,
    total: { $add: ["$marks1", "$marks2"] },
    difference: { $subtract: ["$marks2", "$marks1"] },
    product: { $multiply: ["$marks1", "$marks2"] },
    average: { $divide: [ { $add: ["$marks1", "$marks2"] }, 2 ] },
    mod: { $mod: ["$marks2", "$marks1"] }
  }
}
])
```