MongoDB Aggregation Framework – Complete Guide

MongoDB Aggregation is a powerful feature that processes documents in a pipeline and returns computed results.

■ Why Aggregation?

Aggregation is used for filtering, grouping, sorting, joining, projecting, calculating and reshaping data.

■ Pipeline Syntax

```
db.collection.aggregate([
  { stage1: {...} },
  { stage2: {...} }
])
```

■ Common Aggregation Stages

1 $match → Filter documents (like WHERE)
2 $group → Group documents and calculate values
3 $project → Select specific fields
4 $sort → Sort documents
5 $limit → Limit number of docs
6 $skip → Skip docs
7 $lookup → Join collections
8 $count → Count results

■ Example 1: $match – Filter

```
db.students.aggregate([
  { $match: { age: { $gt: 18 } } }
])
```

■ Example 2: $group – Group & Count

```
db.students.aggregate([
  {
    $group: {
      _id: "$class",
      totalStudents: { $sum: 1 },
      avgAge: { $avg: "$age" }
    }
  }
])
```

■ Example 3: $sort – Sorting

```
db.students.aggregate([
  { $sort: { age: -1 } }
])
```

■ Example 4: $project – Select Fields

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

■ Example 5: $lookup – Join Collections

```
db.orders.aggregate([
  {
    $lookup: {
      from: "customers",
      localField: "customerId",
      foreignField: "_id",
      as: "customerInfo"
    }
  }
])
```

■ Pagination using $skip & $limit

```
db.students.aggregate([
  { $skip: 5 },
  { $limit: 10 }
])
```

■ Summary Table

Use $match early for better performance. Aggregation is faster than loops. Use $project to hide sensitive fields.