# **MongoDB Aggregation**

## ****1. Filtering –**** $match

Use $match early in the pipeline to reduce the data size and improve performance.

### Q1: Get all products with price above ₹10,000.

db.products.aggregate([

{ $match: { price: { $gt: 10000 } } }

])

### Q2: Find users from city "Chennai".

db.users.aggregate([

{ $match: { "address.city": "Chennai" } }

])

## ****2. Field Shaping –**** $project****,**** $addFields

Restructure documents, add or rename fields, or compute new fields.

### Q3: Show only product name and price.

db.products.aggregate([

{ $project: { name: 1, price: 1, \_id: 0 } }

])

### Q4: Add a priceWithTax field assuming 18% tax.

db.products.aggregate([

{

$addFields: {

priceWithTax: { $multiply: ["$price", 1.18] }

}

}

])

## ****3. Aggregation –**** $group

Use to group documents and calculate values like count, sum, avg, etc.

### Q5: Count number of products per category.

db.products.aggregate([

{

$group: {

\_id: "$categoryId",

productCount: { $sum: 1 }

}

}

])

### Q6: Get average rating per product.

db.reviews.aggregate([

{

$group: {

\_id: "$productId",

avgRating: { $avg: "$rating" }

}

}

])

## ****4. Array Handling –**** $unwind

Use $unwind to flatten arrays before further processing.

### Q7: List every product in every order (1 per line).

db.orders.aggregate([

{ $unwind: "$items" },

{

$project: {

productId: "$items.productId",

name: "$items.name",

quantity: "$items.qty"

}

}

])

### Q8: Show user and each product from wishlist.

db.users.aggregate([

{ $unwind: "$wishlist" },

{

$project: {

userId: "$\_id",

productId: "$wishlist"

}

}

])

## ****5. Pagination & Sorting –**** $sort****,**** $skip****,**** $limit

(Coming in Part 2)

### Q9: Show top 5 expensive products.

db.products.aggregate([

{ $sort: { price: -1 } },

{ $limit: 5 }

])

### Q10: Get page 2 with 5 products per page.

db.products.aggregate([

{ $skip: 5 },

{ $limit: 5 }

])

## ****6. Analytics –**** $count****,**** $bucket****,**** $facet

(Coming in Part 2)

### Q11: Count number of users.

db.users.aggregate([

{ $count: "totalUsers" }

])

### Q12: Bucket products into price ranges.

db.products.aggregate([

{

$bucket: {

groupBy: "$price",

boundaries: [0, 10000, 25000, 50000, 100000],

default: "Above 1L",

output: {

count: { $sum: 1 }

}

}

}

])

### Q13: Run multiple stages together (facet).

db.products.aggregate([

{

$facet: {

expensive: [{ $match: { price: { $gt: 50000 } } }],

cheap: [{ $match: { price: { $lt: 10000 } } }]

}

}

])

## ****7. Joins –**** $lookup

(Will be introduced after core aggregation)

### Q14: Join products with their category.

db.products.aggregate([

{

$lookup: {

from: "categories",

localField: "categoryId",

foreignField: "\_id",

as: "categoryInfo"

}

}

])

### Q15: Join orders with user info.

db.orders.aggregate([

{

$lookup: {

from: "users",

localField: "userId",

foreignField: "\_id",

as: "user"

}

}

])

## ****8. Advanced –**** $merge****,**** $replaceRoot****,**** $setWindowFields

(Bonus for advanced learners)

### Q16: Merge summary into a new collection.

db.orders.aggregate([

{

$group: {

\_id: "$userId",

totalSpent: { $sum: "$total" }

}

},

{ $merge: "user\_spending\_summary" }

])