**What is Aggregation in MongoDB?**

Aggregation is a way to process data and return computed results, such as summaries, counts, averages, groupings, and more. It can transform data into meaningful insights and allow complex data processing.

MongoDB provides the **aggregation pipeline**, which consists of multiple stages. Each stage transforms the data in some way, and the result is passed to the next stage.

**Aggregation Pipeline Stages:**

1. **$match**: Filters documents to pass only those that match the specified condition(s).
2. **$group**: Groups documents together based on a field and performs aggregation operations like sum, count, average, etc.
3. **$sort**: Sorts the documents in ascending or descending order.
4. **$project**: Reshapes each document in the stream by adding, removing, or changing fields.
5. **$limit**: Limits the number of documents to pass through the pipeline.
6. **$skip**: Skips a specified number of documents.

Basic Syntax:

db.collection.aggregate([

{ $stage1: { /\* stage details \*/ } },

{ $stage2: { /\* stage details \*/ } },

// Add more stages if necessary

])

**Simple Example**

Let's say you have a collection called orders, with documents like:

{

"\_id": 1,

"customer": "Alice",

"totalAmount": 150

},

{

"\_id": 2,

"customer": "Bob",

"totalAmount": 200

},

{

"\_id": 3,

"customer": "Alice",

"totalAmount": 300

}

Now, you want to calculate the **total amount spent by each customer**. You can do this with the aggregation pipeline.

**Aggregation Query:**

db.orders.aggregate([

{

$group: {

\_id: "$customer", // Group by the customer field

totalSpent: { $sum: "$totalAmount" } // Sum the totalAmount for each customer

}

}

])

**Explanation of this Example:**

* **$group**: Groups the documents by the customer field.
* **$sum**: Calculates the sum of the totalAmount for each group (i.e., each customer).

Expected Output:

[

{ "\_id": "Alice", "totalSpent": 450 },

{ "\_id": "Bob", "totalSpent": 200 }

]

In this example:

* Alice spent a total of 450 (150 + 300).
* Bob spent a total of 200.

**Scenario:**

You have a collection called sales that tracks transactions. Each document in the sales collection represents a sale and contains the following fields:

* **\_id**: Unique identifier for each sale
* **product**: Name of the product
* **amount**: Amount for which the product was sold
* **date**: Date the sale was made
* **store**: Store where the sale occurred

Example data:

{

"\_id": 1,

"product": "Laptop",

"amount": 1200,

"date": ISODate("2025-04-01T10:00:00Z"),

"store": "Store A"

},

{

"\_id": 2,

"product": "Smartphone",

"amount": 700,

"date": ISODate("2025-04-02T11:00:00Z"),

"store": "Store B"

},

{

"\_id": 3,

"product": "Laptop",

"amount": 1300,

"date": ISODate("2025-04-02T12:00:00Z"),

"store": "Store A"

},

{

"\_id": 4,

"product": "Tablet",

"amount": 500,

"date": ISODate("2025-04-03T10:30:00Z"),

"store": "Store B"

}

**Question:**

**How can we calculate the total sales amount for each product, grouped by the product name, using MongoDB's aggregation functions?**

**Solution using Aggregation:**

We want to group the sales by the product field and sum up the amount for each product.

**Aggregation Query:**

db.sales.aggregate([

{

$group: {

\_id: "$product", // Group by the product name

totalSales: { $sum: "$amount" } // Sum the amount for each product

}

}

])

**Explanation:**

1. **$group**: This stage groups the documents based on the product field (represented as \_id).
2. **$sum**: We use the $sum operator to calculate the total sales (amount) for each product.

**Expected Output:**

[

{ "\_id": "Laptop", "totalSales": 2500 },

{ "\_id": "Smartphone", "totalSales": 700 },

{ "\_id": "Tablet", "totalSales": 500 }

]

**Output Breakdown:**

* **Laptop**: The total sales for "Laptop" are 1200 + 1300 = 2500.
* **Smartphone**: The total sales for "Smartphone" are 700.
* **Tablet**: The total sales for "Tablet" are 500.

**Scenario:**

You have a collection called employees that stores data about employees in different departments. Each document in the collection contains:

* **\_id**: Employee's unique identifier
* **name**: Name of the employee
* **department**: Department where the employee works (e.g., "Sales", "HR")
* **salary**: Monthly salary of the employee

Example data:

{

"\_id": 1,

"name": "Alice",

"department": "Sales",

"salary": 3000

},

{

"\_id": 2,

"name": "Bob",

"department": "HR",

"salary": 3500

},

{

"\_id": 3,

"name": "Charlie",

"department": "Sales",

"salary": 4000

},

{

"\_id": 4,

"name": "David",

"department": "Sales",

"salary": 5000

},

{

"\_id": 5,

"name": "Eve",

"department": "HR",

"salary": 4200

}

**Question:**

**How can we calculate the average salary in each department?**

**Solution using Aggregation:**

We want to group the employees by their department and calculate the average salary for each department.

Aggregation Query:

db.employees.aggregate([

{

$group: {

\_id: "$department", // Group by the department field

averageSalary: { $avg: "$salary" } // Calculate the average salary for each department

}

}

])

**Explanation:**

1. **$group**: This stage groups the documents by the department field (represented as \_id).
2. **$avg**: We use the $avg operator to calculate the average salary for each department.

**Expected Output**

[

{ "\_id": "Sales", "averageSalary": 4000 },

{ "\_id": "HR", "averageSalary": 3850 }

]

**Output Breakdown:**

* **Sales**: The average salary for the Sales department is calculated as (3000 + 4000 + 5000) / 3 = 4000.
* **HR**: The average salary for the HR department is calculated as (3500 + 4200) / 2 = 3850.