MONGODB

Aggregation operators

Aggregation operators in MongoDB are used to process data records and return computed results. They are used in the aggregation pipeline to perform various operations such as grouping filtering, and transforming data.

The essential concepts and syntax of aggregation, equipping you with the knowledge to perform powerful data analysis using MongoDB.

We'll explore various operators like $avg, $min, $max, $push, and $addToSet, demonstrating how they can be used to group and summarize data in meaningful ways.

- **Agenda**

  Aggregation operations in MongoDB are a series of data processing stages that transform and manipulate data in a pipeline fashion.

  The agenda of aggregation operations refers to sequence of operations that are executed on the data as it flows through pipeline.

INTRODUCTION

Aggregation in MongoDB is a powerful tool that allows you to group and process data in your collections, generating meaningful insights and summaries. Think of it as a way to analyze and condense your data into more manageable and informative forms. The process involves grouping similar documents together and performing operations on specific fields, enabling you to understand trends, patterns, and overall characteristics of your data.

Syntax

The fundamental syntax for performing aggregation in MongoDB is as follows:

```
db.collection.aggregate(pipeline)
```

Where:

‘db’ is the database object.

‘collection’ is the name of the collection on which to perform the aggregation.

‘pipeline’is an array of aggregation stages

Each stage in the pipeline is an object that specifies an aggregation operation to be performed on the data. The stages are executed in the order they are specified in the pipeline.

Example:

```
db.orders.aggregate([
  { $match: { status: " shipped" } },
  { $group: { _id: "$cust_id", total: { $sum: "$amount" } } },
  { $sort: { total: -1 } }
])
```

In this example, the aggregation pipeline consists of three stages:

1. The $match stage filters the documents to only include those with a status field equal to "shipped".
2. The $group stage groups the documents by the cust_id field and calculates the total amount for each group using the $sum operator.
3. The $sort stage sorts the results by the total field in descending order.

➢ TYPES

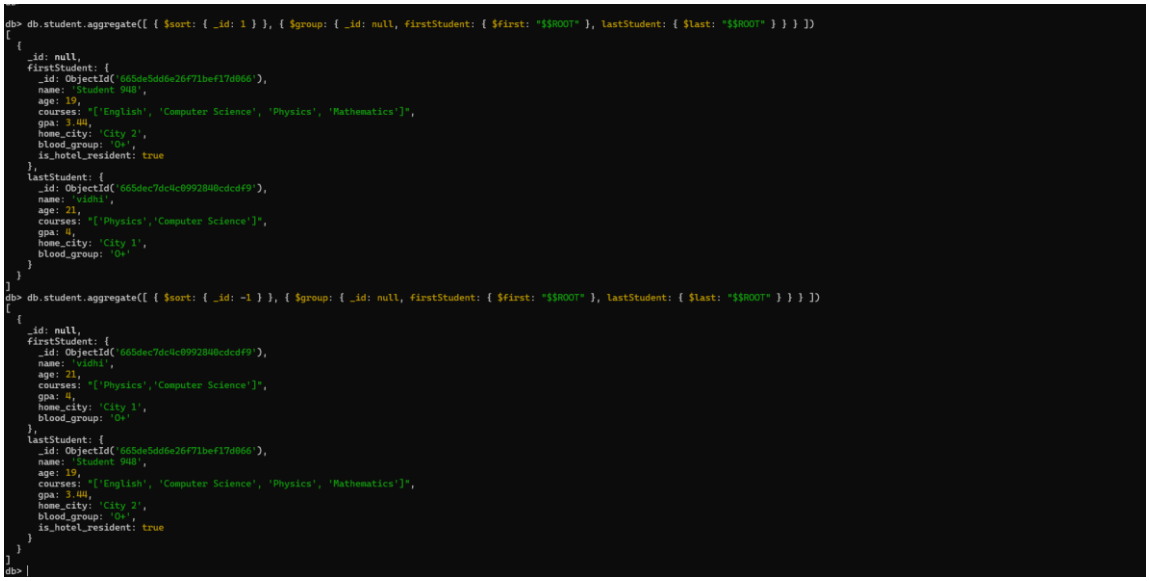
Expression type	Description	Syntax
$sum	Calculates the sum of all values in a numeric field within a group	{ fieldName: { $sum: "$fieldName"} }
$avg	Calculates the average of all values in a numeric field within a group.	{ fieldname: { $avg:"$fieldname" }}
$min	Finds the minimum value in a field within a group.	{ fieldname: { $min: "$fieldname" } }
$max	Finds the maximum value in a field within a group.	{ fieldname: { $max: "$fieldname" } }
$push	Creates an array containing all unique or duplicate values from a field	{ fieldname: { $push: "$fieldname" } }
$addToSet	Creates an array containing only unique values from a field within a group.	{ fieldname: { $addToSet:"$fieldname" } }
$first	Returns the first value in a field within a group (or entire collection).	{ fieldname: { $first: "$fieldname" } }
$last	Returns the last value in a field within a group (or entire collection).	{ fieldname: { $last: "$fieldname" } }

```
db> db.student.aggregate([
...   { $group: { _id: null, totalGPA: { $sum: "$gpa" } } }
... ])
[ { _id: null, totalGPA: 1504.38 } ]
db>

db> db.student.aggregate([
...   { $group: { _id: null, avgGPA: { $avg: "$gpa" } }  }
... ])
[ { _id: null, avgGPA: 2.9908151093439366 } ]
db>

db> db.student.aggregate([
...   { $group: { _id: null, minGPA: { $min: "$gpa" } ,maxGPA: { $max: "$gpa" } } }
... ])
[ { _id: null, minGPA: 2.01, maxGPA: 4 } ]
db>

db> db.student.aggregate([
... { $group: { _id: null, allCourses: { $push: "$courses" } } }
... ])
[
  {
    _id: null,
    allCourses: [
      "['English', 'Computer Science', 'Physics', 'Mathematics']",
```

The code is written in MongoDB aggregation framework. It uses the **$group** operator to group

Documents in a collection and then calculates the sum, average,minimum,and maximum of the GPA values, and creates an array of all courses.

1. Total GPA:
   This code groups all documents in the **student** collection and calculates the total GPA using the **$sum** operator. The **_id** is set to **null** to group all documents together.
2. Average GPA:
   This code calculates the average GPA of all students. It uses the $avg operator to calculate the average of all GPA values in the collection.
3. Minimum and Maximum GPA:
   This code finds the minimum and maximum GPA values in the collection using the **$min** and **$max** operators.
4. All courses:
   This code groups all documents and creates an array of all courses taken by the students in the collection using the $push operator. This will give you a list of all unique courses taken by any student .The $gpa and $courses are fields in the students collection that represent the GPA and the courses taken by each student.



- { $group: { _id: null, firstStudent: { $first: "$$ROOT" }, lastStudent: { $last: "$$ROOT" } } }: This is a stage in the pipeline that groups all the documents together and extracts the first and last documents.
- { $sort: { _id: 1 } } vs { $sort: { _id: -1 } }: This is a stage in the pipeline that sorts the documents by their _id field. The difference is in the direction of the sort:

  1 means sort in ascending order (smallest _id first)

  -1 means sort in descending order (largest _id first)

- Code 1:

  Sorts the documents in the student collection by their _id field in ascending order

  (smallest_id first).

  Groups all the sorted documents together and extracts the first document

  (with the smallest _id) and assigns it to firstStudent.

  Extracts the last document (with the largest _id) and assigns it to lastStudent.

- Code 2:

  Sorts the documents in the student collection by their _id field in descending order

  (largest _id first).

  Groups all the sorted documents together and extracts the first document

  (with the largest _id) and assigns it to firstStudent.

  Extracts the last document (with the smallest _id) and assigns it to lastStudent.

In summary, both codes extract the first and last documents from the student collection, but they differ in the sorting order. Code 1 sorts in ascending order, while Code 2 sorts in descending order.