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.

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 analyse 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 performi

ng 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.

MongoDB provides several aggregation operators that can be used to perform various operations on data. Here are some of the most commonly used aggregation operators:

1.$sum:

The **$sum** operator calculates the total value of a field or an expression.

Example:

db.students.aggregate([$group:{_id:"$home_city",averagesum:"$gpa" }}}]);

```
db> db.students.aggregate([{$group:{_id:"$home_city",averagesum:{$sum:"$gpa"}}}]);
[
  { _id: 'City 4', averagesum: 76.28 },
  { _id: 'City 8', averagesum: 96.64 },
  { _id: 'City 1', averagesum: 102.13 },
  { _id: 'City 9', averagesum: 121.58 },
  { _id: 'City 2', averagesum: 99.65 },
  { _id: null, averagesum: 455.7 },
  { _id: 'City 6', averagesum: 104.29 },
  { _id: 'City 3', averagesum: 102.34 },
  { _id: 'City 7', averagesum: 82.59 },
  { _id: 'City 5', averagesum: 122.42999999999999 },
  { _id: 'City 10', averagesum: 129.15 }
]
db>
```

Here we used,

_ **id:home_city**:-which sets the identifier the homecity to document together.

Averagesum:-calculates the averagesum value of students who scored particular gpa field in home cities using **$sum operator .**

2.$avg:

Here to find averageGPA of all the students we need to use a command

db.students.aggregate([{$group:{_id:null,averageGPA:{$avg:"$gpa"}} }]);

```
db> db.students.aggregate([{$group:{_id:null,averageGPA:{$avg:"$gpa"}}}]);
[ { _id: null, averageGPA: 2.98556 } ]
```

Here we used,

$group:-Groups all documents together.

_ id:null:-sets the group identifier to null.

averageGPA:-calculates the average value of the "gpa"field using **$avg operator.** One more example using **$avg operator,**Here we are finding average gpa for all home cities use a command is **db.students.aggregate([{$group:{_id:"$home_city",averagGPA:{$avg:"$gpa"}}}]);**

```
db> db.students.aggregate([{$group:{_id:"$home_city",avergeGPA:{$avg:"$gpa"}}}]);
[
  { _id: 'City 6', avergeGPA: 2.8969444444444448 },
  { _id: 'City 10', avergeGPA: 2.935227272727273 },
  { _id: 'City 2', avergeGPA: 3.01969696969697 },
  { _id: 'City 9', avergeGPA: 3.1174358974358976 },
  { _id: 'City 5', avergeGPA: 3.0607499999999996 },
  { _id: 'City 1', avergeGPA: 3.003823529411765 },
  { _id: 'City 7', avergeGPA: 2.847931034482759 },
  { _id: null, avergeGPA: 2.9784313725490197 },
  { _id: 'City 8', avergeGPA: 3.11741935483871 },
  { _id: 'City 3', avergeGPA: 3.0100000000000002 },
  { _id: 'City 4', avergeGPA: 2.8251851851851852 }
]
```

3.$min and $max:

To find Minimum and Maximum age we need to use a command called

db.students.aggreagte([{$group:{_id:null,minAge:{$min:"$age"},max Age:{$max:"$age"}}}]);

```
db> db.students.aggregate([ {$group:{_id:null,minAge:{$min:"$age"},maxAge:{$max:"$age"}}}]);
[ { _id: null, minAge: 18, maxAge: 25 } ]
```

Here we used ,

$group:-Groups all documents together

_ **id:null**:- sets the group identifier to null.

using **$min and $max operator** we found a minimum value and max value of age field.

4.$push:

Here pushing all the courses into a single array using **$push operator** to receive an array in order. For this we use a command

db.students.aggregate([{$project:{_id:0,allCourses:{$push:"$courses" }}}]);

```
db> db.students.aggregate([{$project:{_id:0,allCourses:{$push:"$courses"}}}]);
MongoServerError[Location31325]: Invalid $project :: caused by :: Unknown expression $push
db>
```

Here we used

$project:- Transforms the input documents.

_ **id: 0:**-Excludes the _id field from the output documents.

allCourses:- Uses the **$push operator** to create an array. It pushes all elements from the "courses" field of each student document into the allCourses array.

5.$addToSet:

To collect unique courses offered we use a command called

db.candidates.aggregate([{ $unwind: "$courses" }, { $group: { _id: null, uniqueCourses: { $addToSet:"$courses" } } }]);

```
db> db.candidates.aggregate([{$unwind:"$courses"},{$group:{_id:null,uniqueCourses:{$addToSet:"$courses"}}}]);
[
  {
    _id: null,
    uniqueCourses: [
      'Statistics',
      'Psychology',
      'Engineering',
      'Robotics',
      'Sociology',
      'Marine Science',
      'Physics',
      'Mathematics',
      'Biology',
      'Environmental Science',
      'Creative Writing',
      'Film Studies',
      'Computer Science',
      'Artificial Intelligence',
      'Cybersecurity',
      'Art History',
      'Literature',
      'English',
      'Political Science',
      'Philosophy',
      'History',
      'Chemistry',
      'Ecology',
      'Music History'
    ]
  }
]
```

In output we got all the Unique courses which were offered to students.