CLASS 6

AGGREGATION OPERATORS

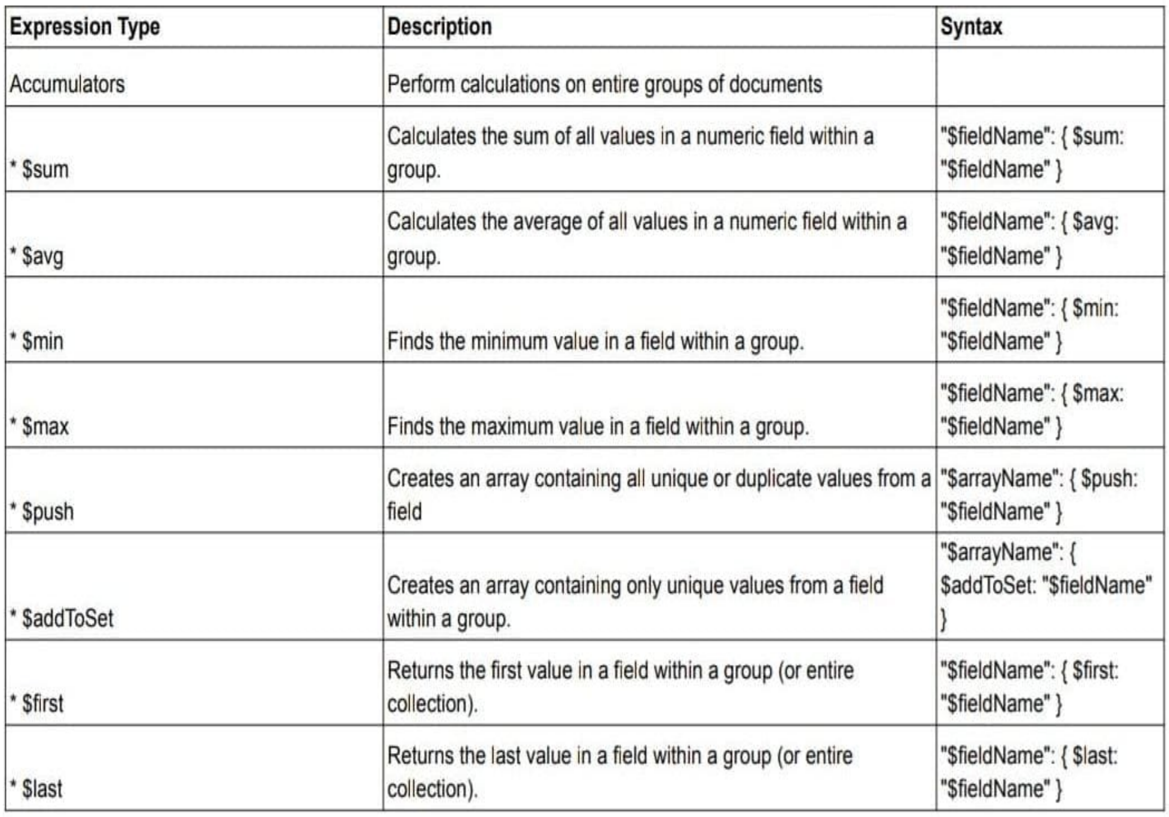
AGGREGATION in MongoDB is a powerful tool for processing data and returning computed results. It involves transforming documents into aggregated results. Think of it as a pipeline where documents flow through stages, each stage performing specific operations.

Syntax:

Basic syntax of aggregate() method is as follows –

```
>db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

Types:



Expression Type	Description	Syntax
Accumulators	Perform calculations on entire groups of documents	
* $sum	Calculates the sum of all values in a numeric field within a group.	"$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" }
* $push	Creates an array containing all unique or duplicate values from a field	"$arrayName": { $push: "$fieldName" }
* $addToSet	Creates an array containing only unique values from a field within a group.	"$arrayName": { $addToSet: "$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" }

Here's a breakdown of the syntax:

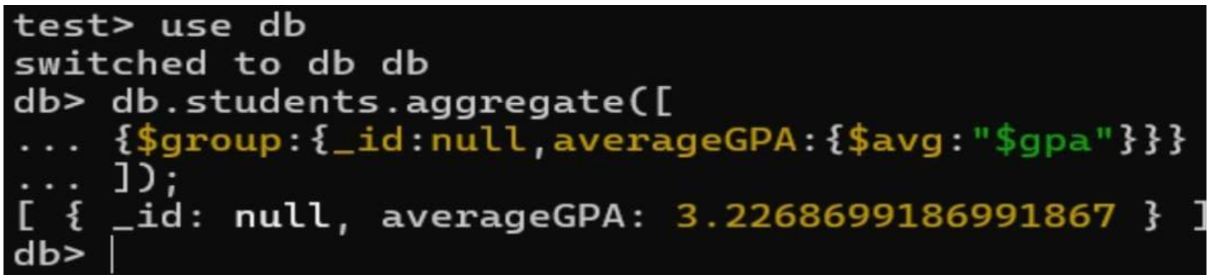
• db.collection_name: This specifies the collection on which you want to perform the aggregation.

• aggregate : This method initiates the aggregation process.

• []: This defines the aggregation pipeline, which is an array of stages that your data goes through.

• // Stage definition: Each element within the square brackets represents a stage in the pipeline. Each stage definition specifies an aggregation operator and its arguments.

Average GPA of all Students:

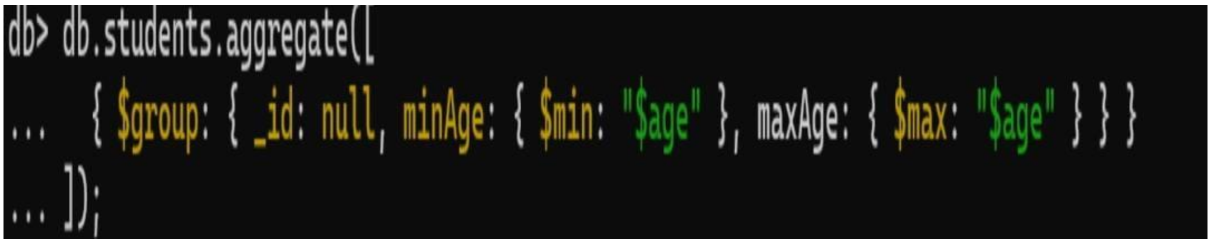


```
test> use db
switched to db db
db> db.students.aggregate([
... {$group:{_id:null,averageGPA:{$avg:"$gpa"}}}
... ]);
[ { _id: null, averageGPA: 3.2268699186991867 } ]
db> |
```

Explanation:

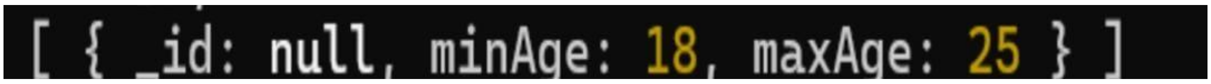
- db.students.aggregate: This line initates the aggregation framework operation on the "students" collections.
- $group: This stage is responsible for grouping documents and performing calculations on the groups.
- _id null: This specifies that we don't need documents grouped by any particular field. We want the average age for all students combined. Setting _id: null creates a single group containing all documents.
- averageAge: { $avg: "$gpa" }: This calculates the average age of all the students.
- $avg: This is the accumulator that calculates the average value of the "age" field for all documents in the group (since we set _id: null).

Minimum and Maximum Age:



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

OUTPUT:



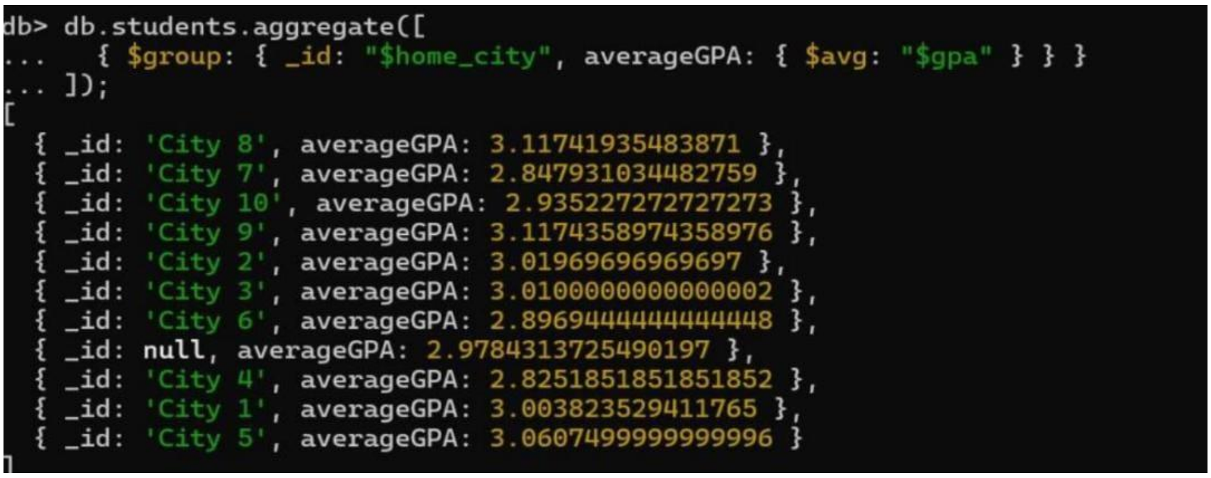
```
[ { _id: null, minAge: 18, maxAge: 25 } ]
```

Explanation:

- The minimum and maximum age in MongoDB can be calculated using an aggregation query that groups all documents in a collection and computes the minimum and maximum values of the `age` field.
- This is achieved by using the `$min` and `$max` operators within a `$group` stage, setting `_id` to `null` to consider the entire collection.

- The MongoDB aggregation query calculates the minimum and maximum age of all students in the `students` collection. Using the `db.students.aggregate` method, it runs an aggregation pipeline with a single `$group` stage where `_id` is set to `null`, grouping all documents together.
- Within this group, `minAge` is calculated using the `$min` operator to find the lowest `age` value, and `maxAge` is calculated using the `$max` operator to find the highest `age` value. The output is a single document `{ _id: null, minAge: 18, maxAge: 25 }`, showing the minimum age as 18 and the maximum age as 25 among all students.

How to get Average GPA for all Home Cities:



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

Explanation:

- db.students.aggregate(...): This initiates an aggregation operation on the students collection.

- [...]: The aggregation pipeline is defined within these square brackets. In this case, it consists of a single stage.

- { $group: { ... } }: This is the $group stage in the aggregation pipeline. It groups documents by a specified identifier and can perform various operations, such as calculating averages, sums, etc.

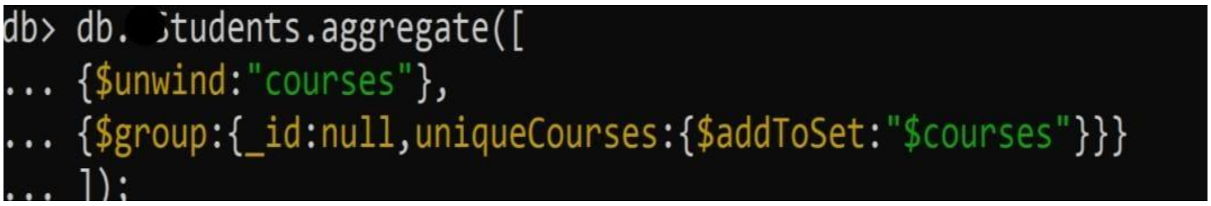
- _id: "$home_city": This specifies that documents should be grouped by the home_city field. Each unique value of home_city will form a group.

- averageGPA: { $avg: "$gpa" }: This calculates the average of the gpa field for each group (each unique home_city). The result will be stored in the averageGPA field.

Collect Unique Courses Offered (Using $addToSet):

The $addToSet operator in MongoDB's Aggregation Framework can be used to collect unique values in an array field. To collect unique courses offered using $addToSet, you would need to have a collection that contains documents with a field representing the courses offered.

Here's an example using the persons collection from the "Practical MongoDB Aggregations" documentation:



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

This aggregation pipeline groups all documents together (_id: null) and uses $addToSet to collect unique values from the course field into the coursesOffered array.