# Aggregation Operators

**Aggregation:**

Aggregation operators in MongoDB are powerful tools for transforming and summarizing data. They allow you to process multiple documents and return calculated results.

## Syntax:

```
Db.collection.aggregate(<AGGREGATE OPERATION>)
```

**Benefits:**

- **Efficient data processing:** Aggregation pipelines efficiently process large datasets without fetching entire documents into memory.
- **Data summarization:** They provide a concise way to summarize and analyze data for insights.
- **Flexibility:** You can chain multiple stages to achieve complex data transformations.

## Types:

| Expression Type | | 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" } |

## 5.Execute Aggregation operations ($avg, $min,$max, $push, $addToSet etc.). students encourage to execute

## several queries to demonstrate various aggregation operators)

### Accumulators:

- In MongoDB, accumulators are special operators used within the aggregation framework, specifically in the $group stage.
- They allow you to perform calculations and maintain state as you process documents belonging to the same group.

### $avr:

- Calculates the average of a field for all documents in a group.

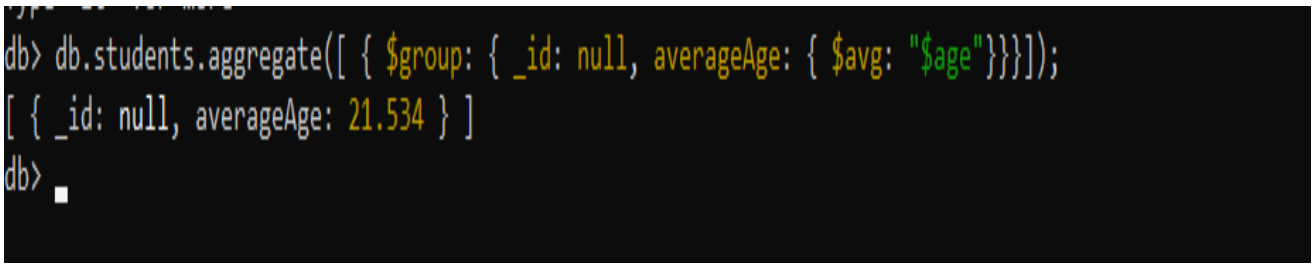
//Average GPA of ALL the students

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

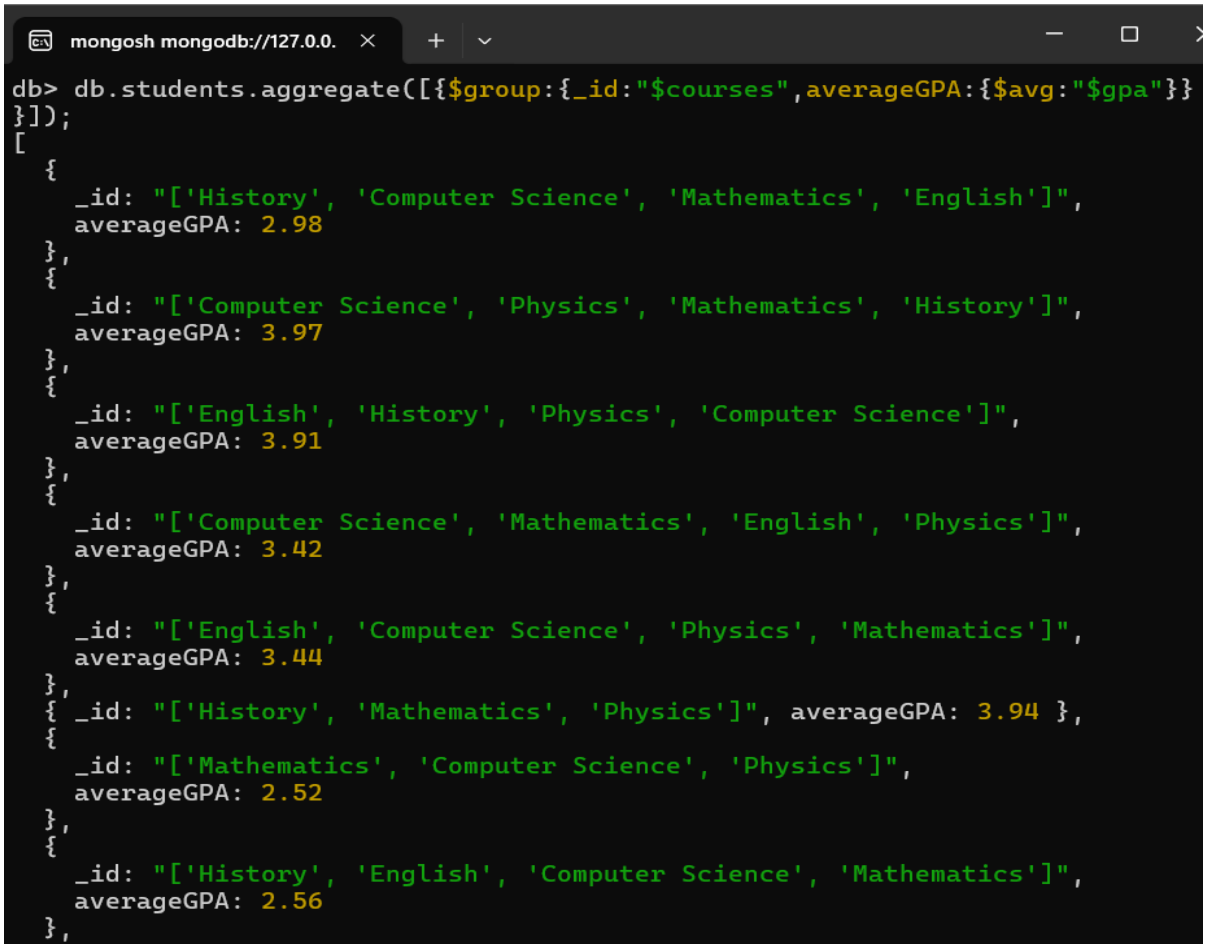
- db.students.aggregate: This line initiates the aggregation framework operation on the "students" collection.
- [{ $group: ... }]: This defines the aggregation pipeline as an array containing a single stage ($group).
- $group: This stage is responsible for grouping documents and performing calculations 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).

//Average AGE of All the students




// calculates the average GPA for different combinations of courses



- ✓ db.students.aggregate: This line initiates the aggregation framework operation on the "students" collection.
- ✓ [{ $group: ... }]: This defines the aggregation pipeline as an array containing a single stage ($group).
- ✓ $group: This stage is responsible for grouping documents and performing calculations on the groups.
- ✓ _id: "$courses": This groups documents based on the value of the "courses" field. This field likely contains an array of strings representing the courses a student is enrolled in. Since it's an array, grouping by this field will create separate groups for documents with different course combinations (e.g., ["History", "Math"] vs. ["Physics", "English"]).
- ✓ averageGPA: { $avg: "$gpa" }: Within each group (combination of courses), this calculates the average value of the "gpa" field using the $avg accumulator.

The output shows an array of documents, each representing a unique combination of courses (identified by _id as a string representation of the course array) and the corresponding average GPA for students taking those courses together.

//home_city and GPA

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

## $min and $max:

- ➢ **$min:** Returns the minimum value of a field for all documents in a group.
- ➢ **$max:** Returns the maximum value of a field for all documents in a group.

//Finding Minimum Age

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

- ✓ **db.students.aggregate**: This line initiates the aggregation framework operation on the "students" collection.
- ✓ **[{ $group: ... }]**: This defines the aggregation pipeline as an array containing a single stage ($group).
- ✓ **$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 specific field. We want the minimum age for all students combined. Setting _id: null creates a single group containing all documents.
- ✓ **minAge: { $min: "$age" }**: This calculates the minimum age of all students.
    - $min: This accumulator finds the lowest non-null value of the "age" field for all documents in the group (since we set _id: null).

## //Finding Maximum Age

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

- ✓ **$group**: Groups documents and performs calculations.
- ✓ **_id: null**: Groups all documents together.
- ✓ **maxAge: { $max: "$age" }**: This calculates the maximum age using the $max accumulator. It finds the highest non-null value of the "age" field within the group (all documents in this case)

## //calculates both the minimum and maximum age of students

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

**$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 specific field. We want the minimum and maximum age for all students combined. Setting _id: null creates a single group containing all documents.
- ✓ **minAge:{$min:"$age"}**: This calculates the minimum age using the $min accumulator. It finds the lowest non-null value of the "age" field for all documents in the group (since we set _id: null).
- ✓ **maxAge: { $max: "$age" }**: This calculates the maximum age using the $max accumulator within the same group. It finds the highest non-null value of the "age" field.

## $sum:

- ➢ Calculates the sum of all numeric values for a specified field across documents within a group.

//calculates the sum of all students GPA

```
db> db.students.aggregate([{ $group:{ _id:0,sumGPA:{$sum:"$gpa"}}}]);
[ { _id: 0, sumGPA: 145.34 } ]
db>
```

**$group**: This stage is responsible for grouping documents and performing calculations on the groups.

- ✓ **_id: 0**: This specifies that we don't need documents grouped by any specific field. We want the total sum of GPAs for all students combined. Setting _id: 0 creates a single group containing all documents.
- ✓ **sumGPA: { $sum: "$gpa" }**: This calculates the sum of all "gpa" field values.
  - $sum: This accumulator iterates through documents in the group and adds the values of the "gpa" field for each student, returning the total sum.

**//$sum,$min,$max**

```
db> db.students.aggregate([{ $group:{ _id:0,sumGPA:{$sum:"$gpa"},minAge:{$min:"$age"}}}]);
[ { _id: 0, sumGPA: 145.34, minAge: 18 } ]
db> db.students.aggregate([{ $group:{ _id:0,sumGPA:{$sum:"$gpa"},maxAge:{$max:"$age"}}}]);
[ { _id: 0, sumGPA: 145.34, maxAge: 25 } ]
db> db.students.aggregate([{ $group:{ _id:"$name",sumGPA:{$sum:"$gpa"}}}]);
[
  { _id: 'Student 347', sumGPA: 2.17 },
  { _id: 'Student 404', sumGPA: 2.59 },
  { _id: 'Student 127', sumGPA: 2.56 },
  { _id: 'Student 901', sumGPA: 2.19 },
  { _id: 'Student 647', sumGPA: 3.43 },
  { _id: 'Student 305', sumGPA: 3.4 },
  { _id: 'Student 652', sumGPA: 3.93 },
  { _id: 'Student 201', sumGPA: 2.66 },
  { _id: 'Student 504', sumGPA: 4.82 },
  { _id: 'Student 157', sumGPA: 2.27 },
  { _id: 'Student 690', sumGPA: 4.96 },
  { _id: 'Student 533', sumGPA: 3.26 },
  { _id: 'Student 232', sumGPA: 2.54 },
  { _id: 'Student 239', sumGPA: 3.75 },
  { _id: 'Student 771', sumGPA: 3.59 },
  { _id: 'Student 372', sumGPA: 3.06 },
  { _id: 'Student 111', sumGPA: 2.99 },
  { _id: 'Student 346', sumGPA: 3.31 },
  { _id: 'Student 873', sumGPA: 3.94 },
  { _id: 'Student 563', sumGPA: 2.25 }
]
```

## $push:

- Used within the $group stage to create an array containing all values for a specific field from documents within a group.

```
mongosh mongodb://127.0.0.
  }
]
db> db.students.aggregate([ { $group: { _id:"$name",courses:{$push:"$courses"}}}]);
[
  {
    _id: 'Student 384',
    courses: [ "['Mathematics', 'Computer Science']" ]
  },
  {
    _id: 'Student 127',
    courses: [ "['History', 'English', 'Computer Science', 'Mathematics']" ]
  },
  {
    _id: 'Student 901',
    courses: [ "['History', 'Computer Science']" ]
  },
  {
    _id: 'Student 676',
    courses: [ "['English', 'Physics', 'Mathematics', 'Computer Science']" ]
  },
  {
    _id: 'Student 347',
    courses: [ "['English', 'Computer Science', 'Physics']" ]
  },
  {
    _id: 'Student 404',
    courses: [ "['Mathematics', 'Physics', 'Computer Science', 'English']" ]
  },
  { _id: 'Student 647', courses: [ "['English', 'Physics']" ] },
```

**$group**: This stage is responsible for grouping documents and performing calculations on the groups.

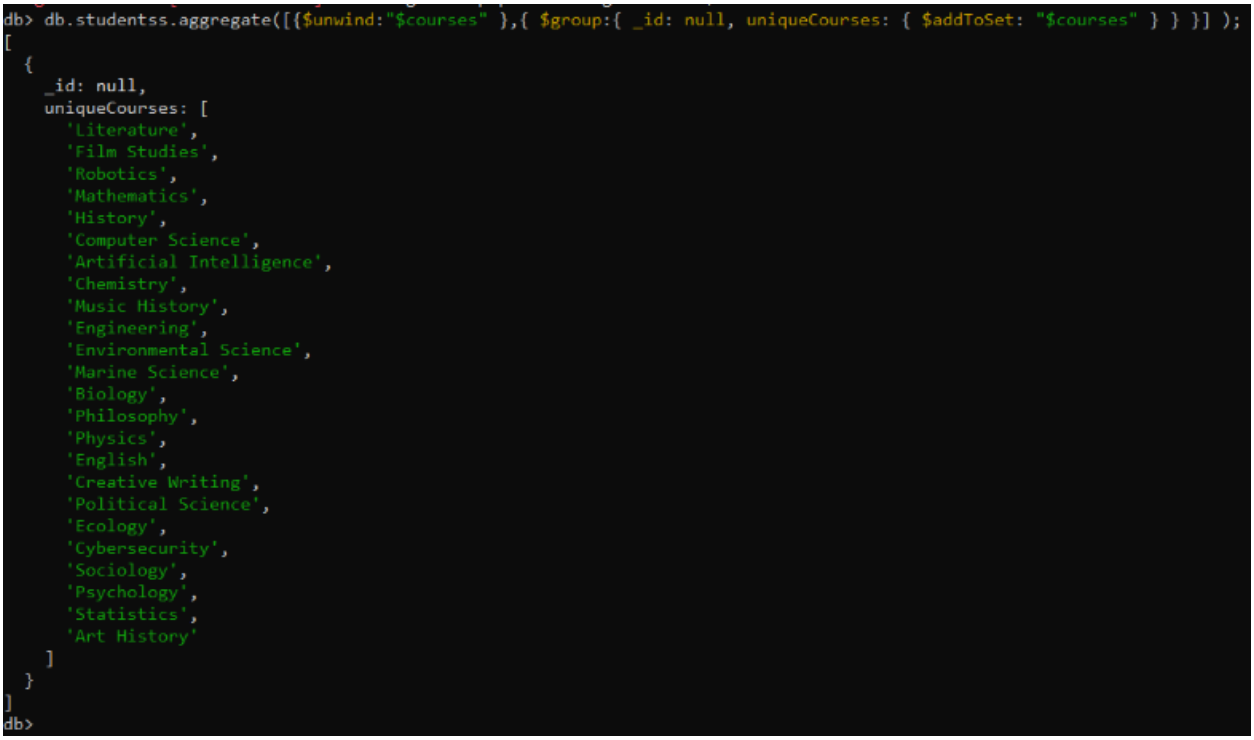
- ✓ **_id: "$name"**: This groups documents based on the value of the "name" field. All documents with the same name will be placed in the same group.
- ✓ **courses: { $push: "$courses" }**: This accumulates an array of courses for each group (identified by student name). It iterates through documents within each group and adds the value of the "courses" field (assumed to be an array of course names) to a new array named "courses" within the group output.

**//$push**

```
]
db> db.studentss.aggregate([ { $group:{ _id: "$name", sumGPA: { $push: "$age" } } }] );
[
  { _id: 'Emily Jones', sumGPA: [ 21 ] },
  { _id: 'Fatima Brown', sumGPA: [ 18 ] },
  { _id: 'Alice Smith', sumGPA: [ 20 ] },
  { _id: 'Gabriel Miller', sumGPA: [ 24 ] },
  { _id: 'Isaac Clark', sumGPA: [ 22 ] },
  { _id: 'Lily Robinson', sumGPA: [ 23 ] },
  { _id: 'Kevin Lewis', sumGPA: [ 21 ] },
  { _id: 'Hannah Garcia', sumGPA: [ 20 ] },
  { _id: 'David Williams', sumGPA: [ 23 ] },
  { _id: 'Charlie Lee', sumGPA: [ 19 ] },
  { _id: 'Jessica Moore', sumGPA: [ 19 ] },
  { _id: 'Bob Johnson', sumGPA: [ 22 ] }
```

## $addtoSet:

- In MongoDB aggregation, the $addToSet operator is used within the $group stage to create an array containing unique values for a specified field across documents within a group.




**[{ $unwind: "$courses" }, { $group: ... }]**: This defines the pipeline as an array containing two stages:

- ✓ **$unwind: "$courses"**:
  - This stage iterates through each document in the "students" collection.
  - If a document has an array field named "courses" containing multiple course names, $unwind creates a separate document for each course entry within that array.
  - For documents without the "courses" field or with an empty array, it leaves them unchanged.

- ✓ **$group: { _id: null, uniqueCourses: { $addToSet: "$courses" } }**: This stage groups the documents (potentially many after unwinding) based on the specified criteria.

- **_id: null**: This instructs the group to combine all documents into a single group, as we're interested in the overall unique courses across all students.
- **uniqueCourses: { $addToSet: "$courses" }**: This calculates the unique courses using $addToSet. It iterates through the "courses" field (after unwinding) and adds each unique course name (considering non-null values) to a new array named "uniqueCourses".