CLASS 6: AGGREGATION OPERATORS

Aggregation in MongoDB is a powerful feature that allows you to process data records and return computed results. It's particularly useful for operations like grouping data, sorting it into a specific order, restructuring returned documents, or filtering data.

Aggregation Pipelines

The primary method for aggregation in MongoDB is the **aggregation pipeline**. An aggregation pipeline is a multi-stage process where documents enter the pipeline and pass through multiple stages that transform them in some way. Each stage in the pipeline processes the documents as they pass through and forwards the transformed documents to the next stage.

Syntax:

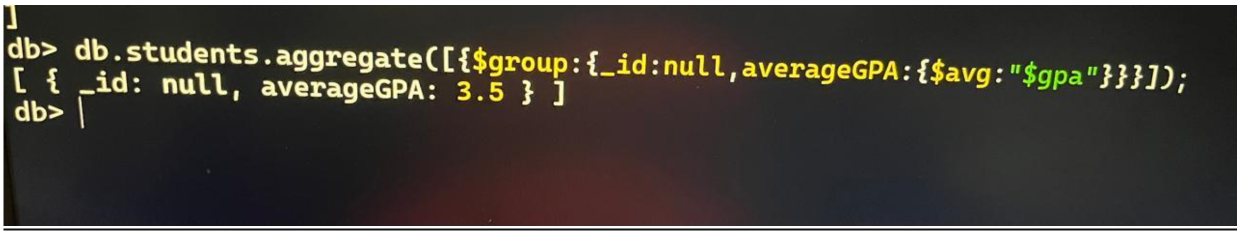
db.collection.aggregate(<AGGREGATE OPERATION>

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" }

Average GPA of all Students:

The average GPA of all students in MongoDB is calculated using an aggregation query that groups all documents in the 'students' collection and computes the mean of the ‘gpa’ field.This is achieved by using the '$avg’ operator within a '$group' stage, setting '_id' to 'null' to consider the entire collection.



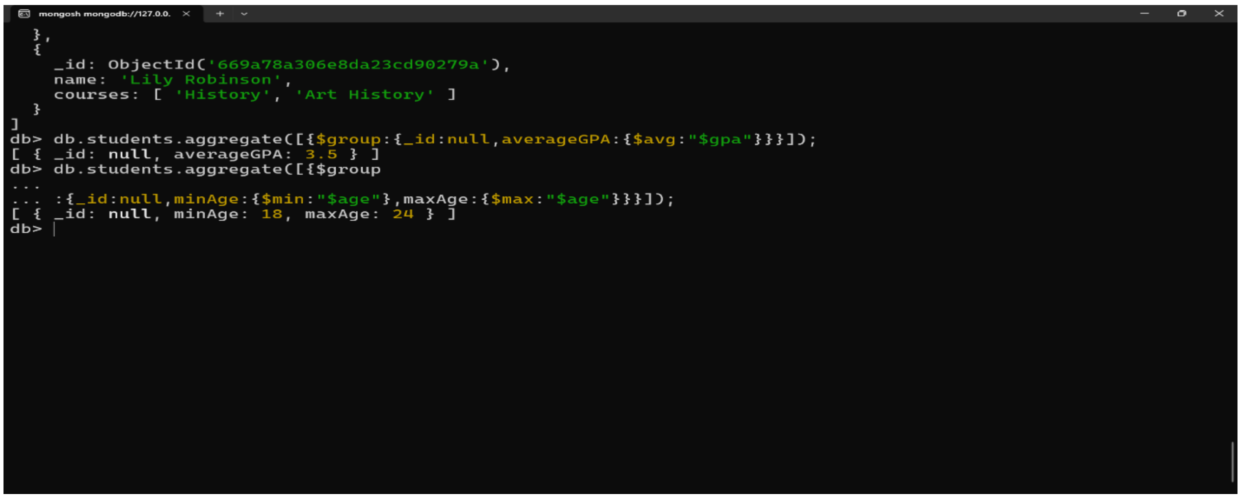
```
]
db> db.students.aggregate([{$group:{_id:null,averageGPA:{$avg:"$gpa"}}}]);
[ { _id: null, averageGPA: 3.5 } ]
db> |
```

The MongoDB aggregation query calculates the average GPA of all students in the'students' collection. By using the 'db.students.aggregate method, it executes an aggregation pipeline with a single '$group' stage. In this stage, setting ‘_id' to 'null' groups all documents together, ensuring the operation considers the entire collection. The 'averageGPA’ field computes the average value of the gpa' field across all documents using the '$avg' operator. The output is a single document with '_id: null' (indicating no specific grouping) and 'averageGPA:

3.2268699186991867 ‘, representing the average GPA of the students.

Minimum and Maximum Age:

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.



mongosh mongodb://127.0.0.

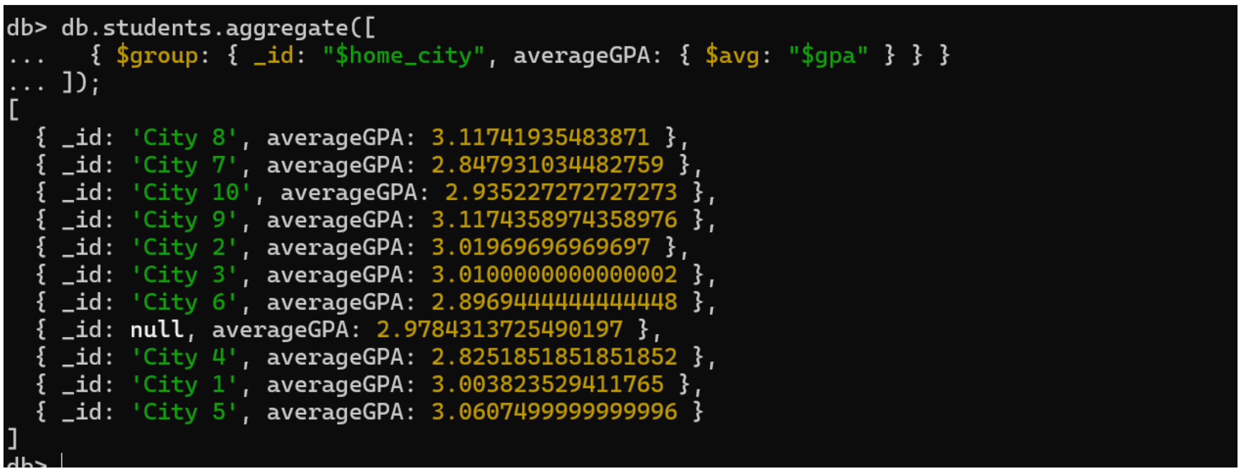
```
  },
  {
    _id: ObjectId('669a78a306e8da23cd90279a'),
    name: 'Lily Robinson',
    courses: [ 'History', 'Art History' ]
  }
]
db> db.students.aggregate([{$group:{_id:null,averageGPA:{$avg:"$gpa"}}}]);
[ { _id: null, averageGPA: 3.5 } ]
db> db.students.aggregate([{$group
...
... :{_id:null,minAge:{$min:"$age"},maxAge:{$max:"$age"}}}]);
[ { _id: null, minAge: 18, maxAge: 24 } ]
db> 
```

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.

<u>average GPA for all home cities:</u>

The average GPA for all home cities in MongoDB can be calculated using an aggregation query that groups documents by the 'homeCity' field and computes the mean of the ‘gpa’ field for each group. This is done by using the '$group' stage with id set to ‘$homeCity' and the

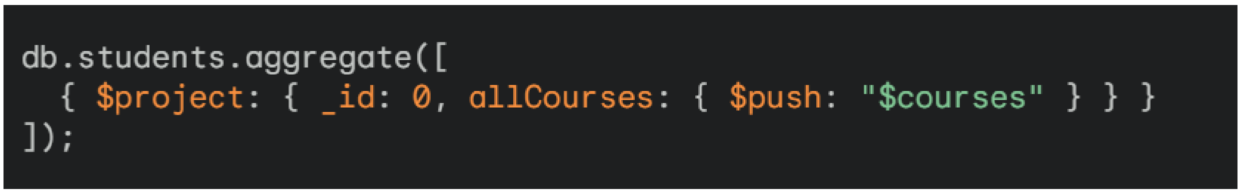
'Savg' operator applied to the 'gpa field.



```
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 }
]
db> 
```

The MongoDB aggregation query calculates the average GPA for students from each home city in the 'students' collection. Using the 'db.students.aggregate method, it executes an aggregation pipeline with a 'Sgroup' stage that groups documents by the home city field (id:"Shome _city'"). Within each group, the 'averageGPA' field is computed using the '$avg' operator applied to the 'gpa' field. The output is an array of documents where each document represents a home city (e.g., 'City 9', 'City 10') and its corresponding average GPA, along with a document where '_id' is 'null' representing students with no specified home city, showing the average GPA across all such groups.

Pushing all Courses into a Single Array:



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

Explanation:

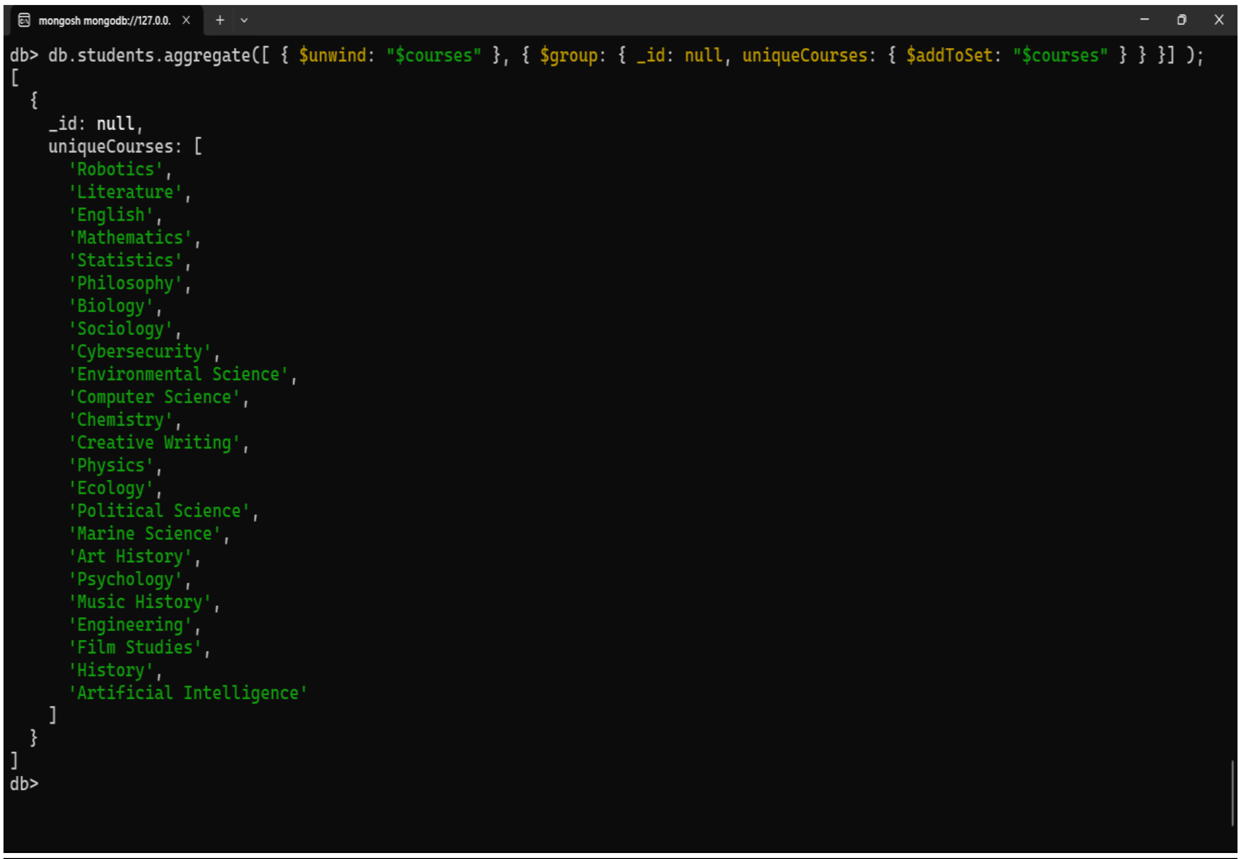
- `$project`: Transforms the input documents.
  - `_id: 0`: Excludes the `_id` field from the output documents.
  - `allCourses`: Uses the `$push` operator to create an array. It pushes all elements from the "courses" field of each student document into the `allCourses` array.

Result:

This will return a list of documents, each with an `allCourses` array containing all unique courses offered (assuming courses might be duplicated across students).

Collection unique courses offered (using $addtToSet):

Collecting unique courses offered using '$addToSet' in MongoDB involves aggregating documents and gathering unique values from the 'courses' field across all documents. This aggregation is achieved by using the '$group' stage with '$addToSet' to accumulate unique course names into a set within each group, ensuring no duplicates are included in the final aggregation result.



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

The MongoDB aggregation query unwinds the 'courses' array field in the 'candidates' collection, ensuring each document is replicated for every course it lists. Then, it groups all documents together (denoted by'_id: null') and applies the $addToSet operator to collect unique course names from all documents into the 'uniqueCourses' array. This results in a single document output where 'uniqueCourses' contains an array listing all distinct courses found across all candidates, ensuring each course appears only once regardless of how many candidates have it listed.