

Experiment:5

Execute Aggregation operations (\$avg, \$min,\$max, \$push, \$addToSet etc.). students encourage to execute several queries to demonstrate various aggregation operators)

AGGREGATE OPERATORS:

Aggregate functions in MongoDB are operations that process multiple documents and return computed results. They are typically used in the aggregation pipeline framework, which provides a powerful way to transform and analyze data. Here are some common aggregate functions in MongoDB .

For example: **sum, avg, min, max**

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.

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

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

Average GPA for all home cities:

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 `\$avg` operator applied to the `gpa` field.

```
db> db.students.aggregate([
...  {$group: {_id: "$home_city", averageGPA: {$avg: "$gpa"}}}
...  ]);
[
  { _id: 'City 9', averageGPA: 3.326842105263158 },
  { _id: 'City 10', averageGPA: 3.147 },
  { _id: 'City 8', averageGPA: 3.329285714285714 },
  { _id: 'City 1', averageGPA: 3.21 },
  { _id: 'City 4', averageGPA: 2.9992857142857146 },
  { _id: 'City 2', averageGPA: 3.2945 },
  { _id: null, averageGPA: 3.2426027397260273 },
  { _id: 'City 3', averageGPA: 3.2242105263157894 },
  { _id: 'City 6', averageGPA: 3.173478260869565 },
  { _id: 'City 5', averageGPA: 3.274736842105263 },
  { _id: 'City 7', averageGPA: 3.184 }
]
```

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 `\$group` stage that groups documents by the `home_city` field (`_id: "\$home_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.

Collect Unique Courses Offered (Using \$addToSet):

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