# Aggregation Operators

## Types of Aggregation Operators:

### 1. Filter Operators:

Filter operators are used to select specific documents from a collection based on conditions. Examples of filter operators include:

- $match: Filters documents based on a condition.
- $filter: Filters an array of documents based on a condition.

### 2. Group Operators:

Group operators are used to group documents based on one or more fields.Examples of group operators include:

- $group: Groups documents based on one or more fields and performs aggregation operations.
- $bucket: Groups documents into buckets based on a specified expression.

### 3. Array Operators:

Array operators are used to perform operations on arrays. Examples of array operators include:

- $unwind: Deconstructs an array field into separate documents.
- $arrayElemAt: Returns a specific element from an array.

● $arrayToObject: Converts an array of key-value pairs into an object.

**4. String Operators:**

String operators are used to perform operations on string fields. Examples of string operators include:

● $substr: Returns a substring of a string field.
● $toLower: Converts a string field to lowercase.
● $toUpper: Converts a string field to uppercase.

**5. Math Operators:**

Math operators are used to perform mathematical operations on numeric fields. Examples of math operators include:

● $add: Adds two numeric fields.
● $subtract: Subtracts one numeric field from another.
● $multiply: Multiplies two numeric fields.
● $divide: Divides one numeric field by another.

**Syntax:**

**db.collection.aggregate(<AGGREGATE OPERATION>)**

**Average GPA of all Students:**

```
db> db.students.aggregate([ {$group:{_id:null,averageGPA:{$avg:"$gpa"}}}]);
[ { _id: null, averageGPA: 3.013661417322835 } ]
db>
```

**Explanation:**

- $group: Groups all documents together.
- _id: null: Sets the group identifier to null (optional, as there's only one group in this case).
- averageGPA: Calculates the average value of the "gpa" field using the $avg operator

**Maximum and Minimun age of the students:**

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

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

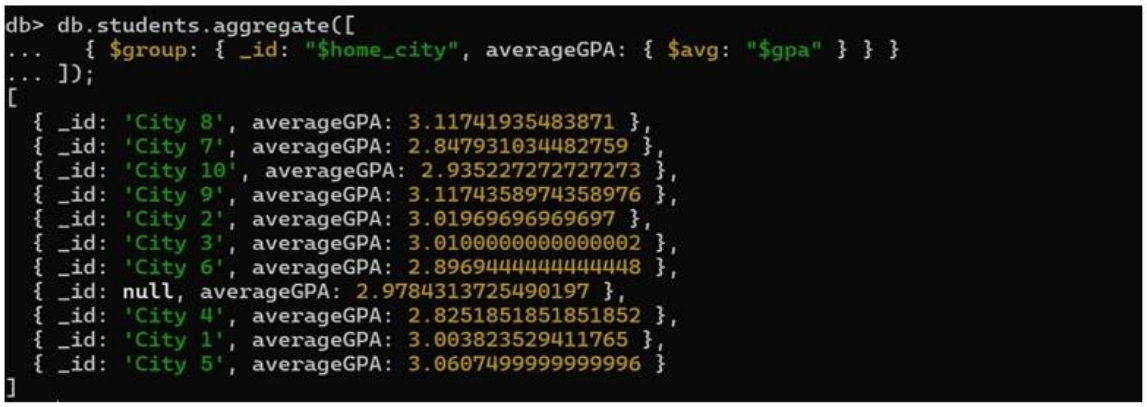
**Explanation:**

- Similar to the previous example, it uses $group to group all documents.
- minAge: Uses the $min operator to find the minimum value in the "age" field.
- maxAge: Uses the $max operator to find the maximum value in the "age" field

**Pushing All Courses into a Single Array :**

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

**How to get Average GPA for all home cities :**

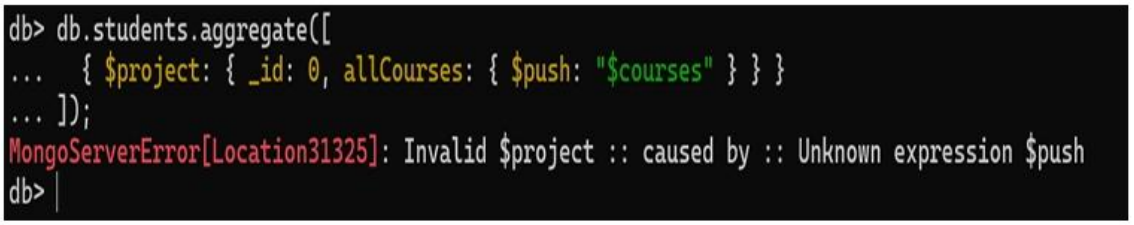



**Explanation:**

- $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)



## Collect Unique Courses Offered (Using $addToSet) :

To collect unique courses offered, you can use the $addToSet aggregation operator in MongoDB. Here's an example