AGGREGATION OPERATORS

Aggregation operators are used in the MongoDB aggregation pipeline to process data in a series of stages. Each stage performs a specific operation on the data, and the output of one stage becomes the input for the next stage. The aggregation pipeline allows you to chain multiple operators together to perform complex data transformations.

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>)

Checking the average GPA of all students:

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

We have the explaination of the operations used below:

2. ({$group: {_id:null, averageGPA: {$avg:"$gpa"}}})

- This is the core of the aggregation query, specifying a single grouping stage.
- $group: This operator groups documents based on a specified _id field.
- _id: null: This means that all documents will be grouped together as a single group since the _id is set to null.
- averageGPA: {$avg:"$gpa"}:
- This defines a new field named "averageGPA" within the output documents.
- $avg:"$gpa": This uses the $avg accumulator to calculate the average value of the "gpa" field for all documents within the group. "$gpa" refers to the "gpa" field in each document of the "students" collection.

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 } ]
db>
```

Explanation:

1. db.students.aggregate([...]): This initiates an aggregation pipeline on the "students" collection within the MongoDB database.

2. {$group: { ... }}: This stage groups the documents based on the _id field, which is set to null in this case. This essentially means grouping all documents together.

3. _id: null: This specifies the grouping criteria. Since it's set to null, all documents will be grouped into a single group.

4. minAge: {$min: "$age"}: This defines a field called minAge within the grouped document. The $min operator finds the minimum value of the "age" field across all documents in the group.

5. maxAge: {$max: "$age"}: Similar to minAge, this defines a field called maxAge and uses the $max operator to find the maximum value of the "age" field within the group.

6. {$group: { ... }}]);: This concludes the aggregation pipeline and displays the result.

This output indicates that the minimum age among the students in the "students" collection is 18, and the maximum age is 25.

To get Average GPA for all home cities:

```
db> db.students.aggregate([{$group:{_id:"$home_city",averageGPA:{$avg:"$gpa"}}}]);
[
  { _id: 'City 6', averageGPA: 2.9076923076923076 },
  { _id: 'City 1', averageGPA: 3.0515789473684211 },
  { _id: 'City 8', averageGPA: 3.1194117647058826 },
  { _id: 'City 10', averageGPA: 3.029583333333333 },
  { _id: 'City 9', averageGPA: 3.1685 },
  { _id: 'City 4', averageGPA: 3.0992307692307692 },
  { _id: 'City 2', averageGPA: 2.943076923076923 },
  { _id: null, averageGPA: 2.987375 },
  { _id: 'City 7', averageGPA: 2.7763157894736844 },
  { _id: 'USA', averageGPA: null },
  { _id: 'City 3', averageGPA: 3.072 },
db>
```

Explanation:

1. **db.students.aggregate(...):** The"students" collection in the database.
2. **{$group: ...}:** This stage groups the documents in the collection based on the specified field.
    - **_id:** "$home_city": This specifies the field to group by - the "home_city" field in each document. This means that all documents with the same "home_city" value will be grouped together.

- **averageGPA:** {$avg:"$gpa"}: This specifies an aggregation operation to perform on the documents within each group. The $avg operator calculates the average value of the "gpa" field within each group. The calculated average is stored in the "averageGPA" field for each group.

3. **[]:** This denotes the end of the aggregation pipeline.

Output:

The output of this query will be a list of documents, each representing a group of students from a particular city. Each document will have the following fields:

- _id: The city name (from the "home_city" field).
- averageGPA: The average GPA of all students from that city.

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).

```
db> db.students.aggregate([
...   { $project: { _id: 0, allCourses: { $push: "$courses" } } }
... ]);
MongoServerError[Location31325]: Invalid $project :: caused by :: Unknown expression $push
db>
```

This is because the above given array is incorrect.

Collect Unique Courses Offered (Using $addToSet):

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

```
db.students.aggregate([
  {
    $group: {
      _id: null,
      uniqueCourses: { $addToSet: "$course" }
    }
  }
])
```

Explanation:

- db.students.aggregate([...]): Initiates the aggregation pipeline on the "students" collection.
- {$group: {...}}: Groups the documents in the collection.
- _id: null: Groups all documents together into a single group.
- uniqueCourses: { $addToSet: "$course" }: Creates a new field called "uniqueCourses" and uses the $addToSet operator to add each unique "course" value to an array. The $addToSet operator ensures that only unique values are added to the array.

Output:

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

This above output is the list of all unique courses offered by the students in the collection.