Experiment 05

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

Introduction to MongoDB aggregation

Aggregation means the gathering of things together. In Computer Science, data aggregation means the grouping of data to prepare combined data sets helpful in generating better information. Aggregation in MongoDB groups the data from various collections and then performs various operations to generate one combined result.

The aggregation method in MongoDB groups the data from various collections, then, in turn, performs some operations, like the total number (sum), average, minimum, maximum, etc. out of the selected groups.

Aggregation method syntax and use

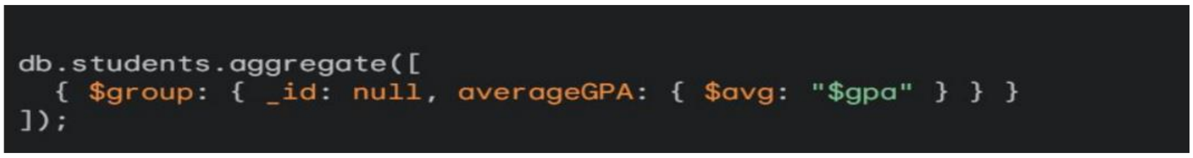
```
db.collection.aggregate (<AGGREGATE OPERATION>)
```

We can use the MongoDB aggregate method to perform the aggregate operation on the documents in MongoDB. We will understand this method better with the help of some practical examples. First, let us take a look at various expressions used in the aggregate operation. Following are the expression types used in the MongoDB aggregrate operation.

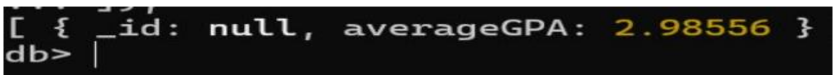
Expression type	Description	Expression type use and syntax
$sum	Sums up the defined value from all the documents in a collection.	db.collection. aggregate ([{$group: {_id: $<Field Name>, <Field Label> : {$sum: <Field Name, Number or Operation>}}}])
$avg	Calculates the average values from all the documents in a collection.	db.collection. aggregate ([{$group: {_id : $<Field Name>, <Field Label> (Savg: <Field Name, Number or Operation>}}}])
$min	Gives the minimum of all the values of documents in a collection.	db.collection. aggregate([{$group: {_id: $<Field Name>, <Field Label> ($min:

		<Field Name, Number or Operation>}}}])
$max	Gives the maximum of all the values of documents in a collection.	db.collection. aggregate ([{$group: {_id: $<Field Name>, <Field Label> {$max <Field Name, Number or Operation>}}}])
$push	Inserts values to an array of the resulting document.	db.collection. aggregate([{$group: (id: $<Field Name>, <Field Label> : {$push: <Field Name>}}}])
$addToSet	Inserts values to an array of the resulting document, but does not create duplicates in the resulting document.	db.collection. aggregate([{$group: {_id $<Field Name>, <Field Label> {$addToSet: <Field Name>}}}])
$first	Gives the first document from the source document.	db.collection. aggregate([{$group: {_id : $<Field Name>, <Field Label> : {$first: <Field Name>}}}])
$last	Gives the last document from the source document.	db.collection. aggregate([{$group: {_id : $<Field Name>, <Field Label> : ($last: <Field Name>}}}]

Average GPA of all Students



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



```
[ { _id: null, averageGPA: 2.98556 }
db> |
```

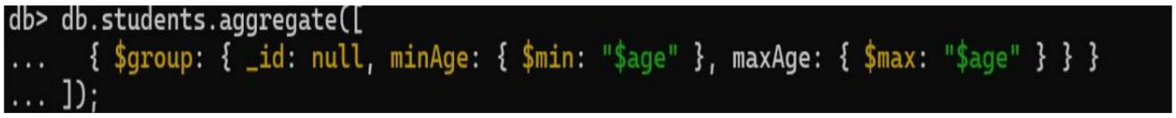
Explanation:

---> $group: Groups all documents together.

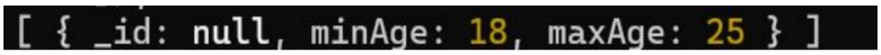
---> _id: null: Sets the group identifier to null (optional, as there's only one group in this case).

---> averageGPA: Calculates the average value of the "gpa" field using the $avg operator

Minimum and Maximum Age:



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



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

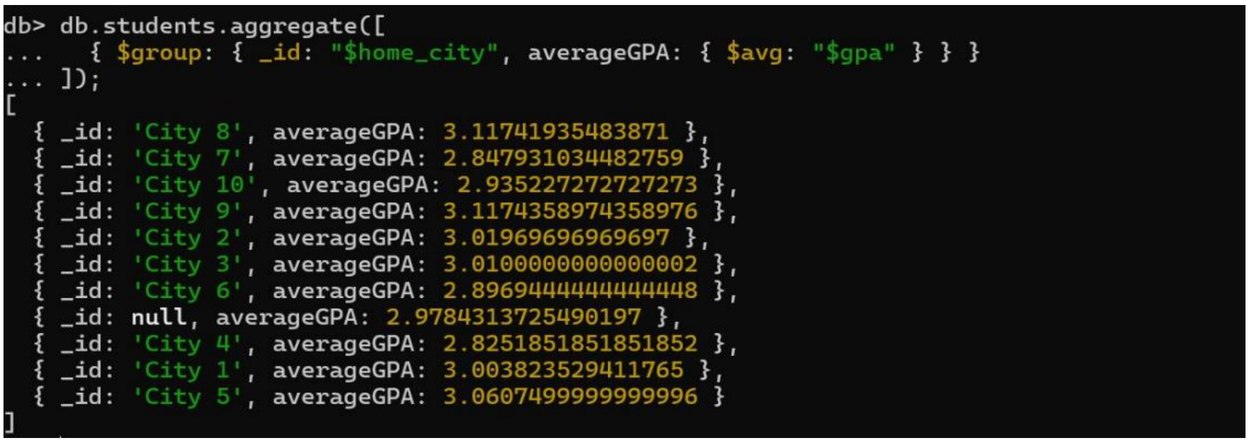
Explanation:

● 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.

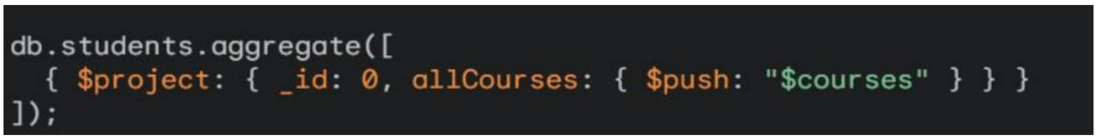
● maxAge: Uses the $max operator to find the maximum value in the "age" field

How to get Average GPA for all home cities



```
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 }
]
```

Pushing All Courses into a Single Array



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

Explanation:

→ $project: Transforms the input documents.

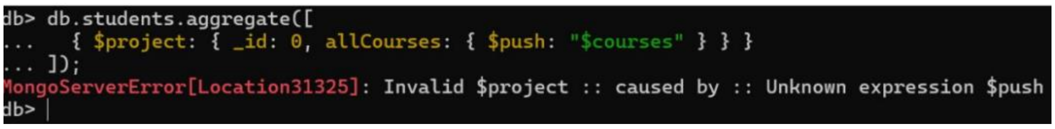
→ _id: 0: Excludes the _id field from the output documents.

→ 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)

BUT



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

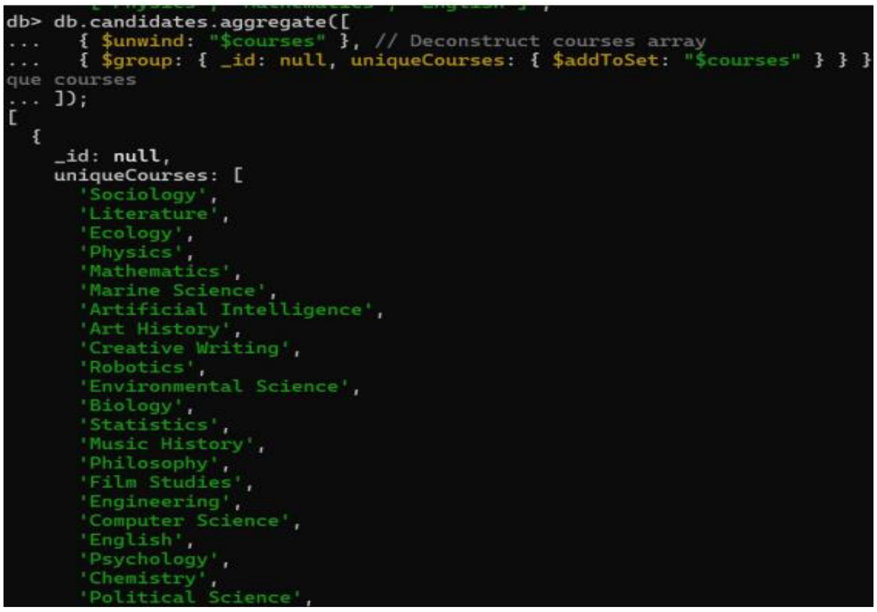
This is because our Array is incorrect :)

Collect Unique Courses Offered (Using $addToSet):

db.candidates.aggregate([{ $unwind: "$courses" },

{ $group: { _id: null, uniqueCourses: { $addToSet: "$courses" } }

}]);



```
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',
```