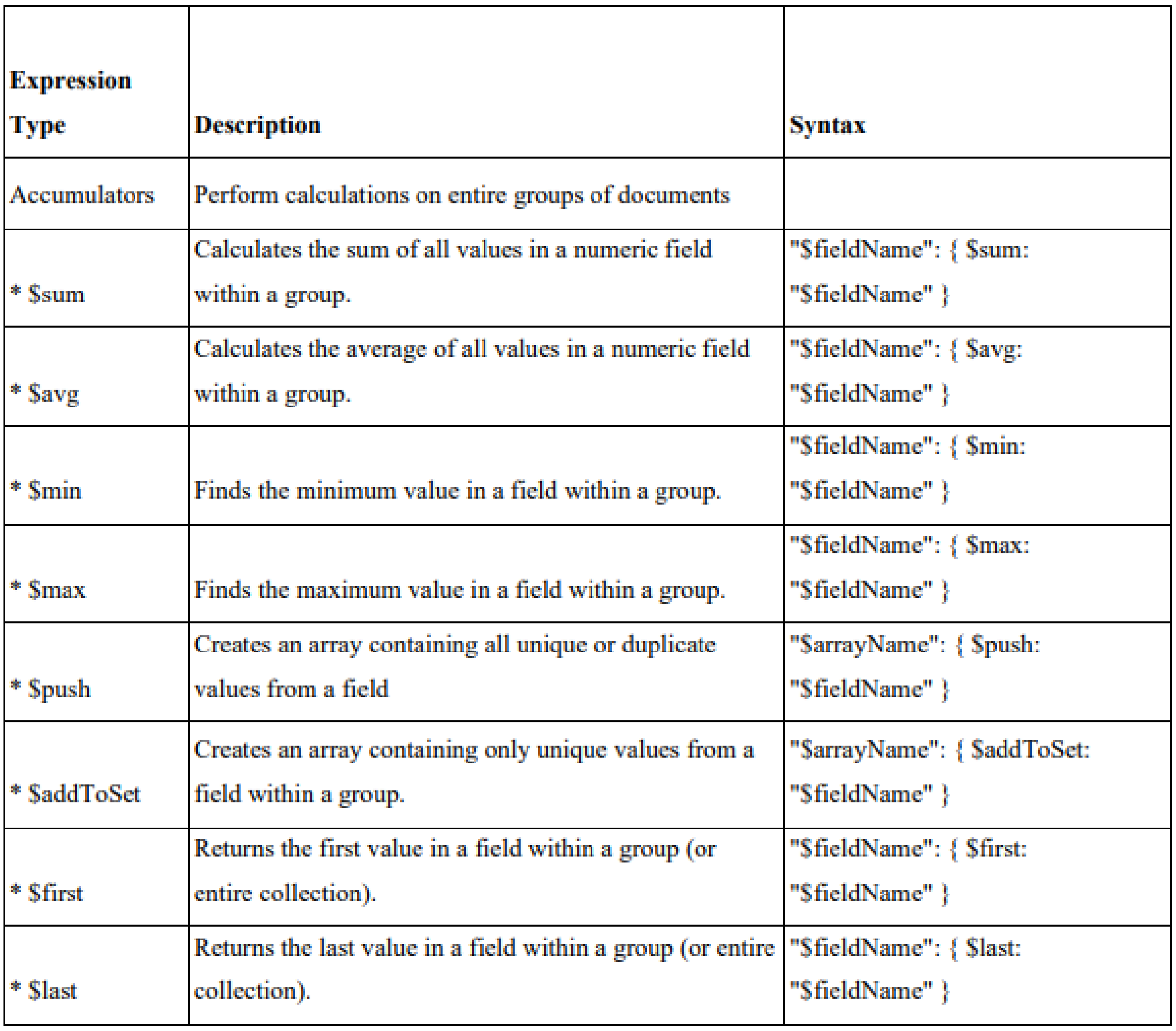
Class 6:
Aggregation Operators

$avg, $min,$max, $push, $addToSet
Aggregation operation where students encourage to execute several queries to demonstrate various aggregation operators.

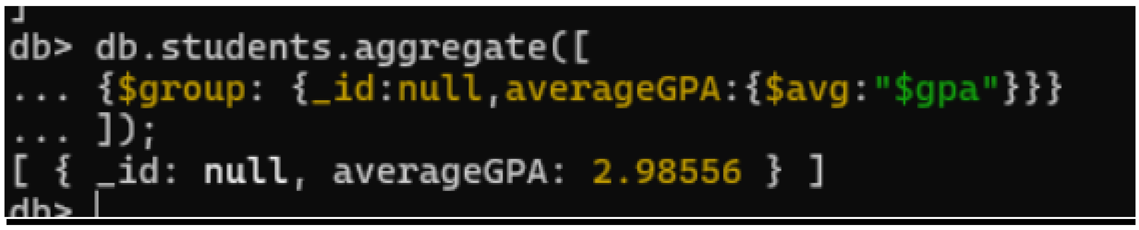
INTRODUCTION:
Aggregation operations summarize data by performing calculations on a group of values. They take multiple rows and return a single result. Common examples include COUNT, SUM, AVG, MIN, and MAX.

SYNTAX:
db.collection.aggregate(<AGGRIGATE OPRRATION>)



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" }

1. Average GPA of all students



```
db> db.students.aggregate([
... {$group: {_id:null,averageGPA:{$avg:"$gpa"}}}
... ]);
[ { _id: null, averageGPA: 2.98556 } ]
db> |
```

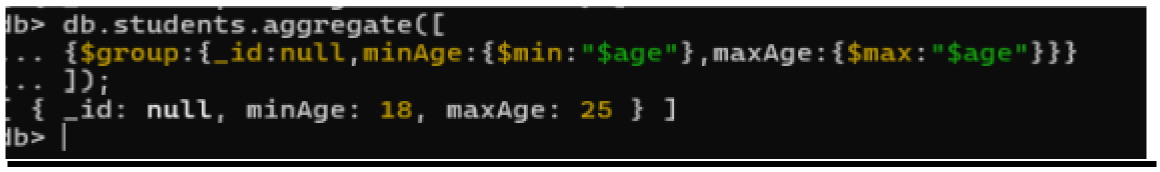
_id: null:
Sets the group identifier to null (optional, as there's only one group in this case).
averageGPA: Calculates the average value of the "gpa" field using the $avg operator.

EXPLANATION:
 In reference with the above code *$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.

2.Minimum and Maximum Age:

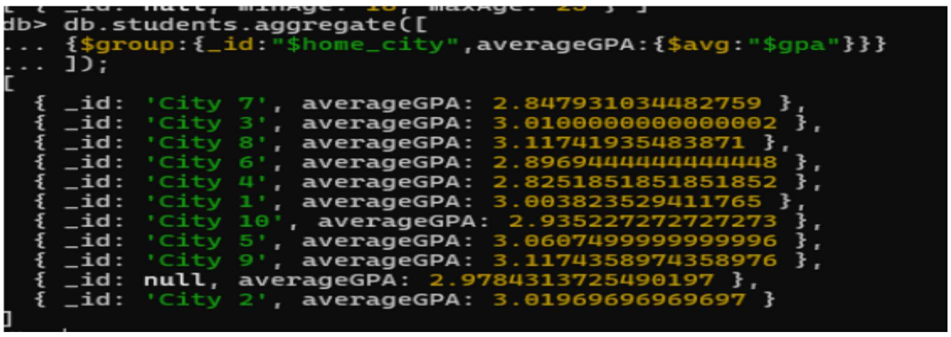


```
lb> db.students.aggregate([
.. {$group:{_id:null,minAge:{$min:"$age"},maxAge:{$max:"$age"}}}
.. ]);
 { _id: null, minAge: 18, maxAge: 25 } ]
lb> |
```

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. This shows only min age and max age.

3. Average GPA of all home cities:



```
db> db.students.aggregate([
... {$group:{_id:"$home_city",averageGPA:{$avg:"$gpa"}}}
... ]);
[
  { _id: 'City 7', averageGPA: 2.847931034482759 },
  { _id: 'City 3', averageGPA: 3.0100000000000002 },
  { _id: 'City 8', averageGPA: 3.11741935483871 },
  { _id: 'City 6', averageGPA: 2.8969444444444448 },
  { _id: 'City 4', averageGPA: 2.8251851851851852 },
  { _id: 'City 1', averageGPA: 3.003823529411765 },
  { _id: 'City 10', averageGPA: 2.9352272727272727 },
  { _id: 'City 5', averageGPA: 3.0607499999999996 },
  { _id: 'City 9', averageGPA: 3.1174358974358976 },
  { _id: null, averageGPA: 2.9784313725490197 },
  { _id: 'City 2', averageGPA: 3.0196969696969697 }
]
```

- Group the documents in the students collection by the home_city field.
- For each group, calculate the average GPA (gpa) and store it in a field called averageGPA.
- The result will be a set of documents where each document represents a unique home_city and the corresponding average GPA of students from that city.

PUSHING ALL COURSES INTO SINGLE ARRAY:

EXPLANATION:

- Aggregation Pipeline: Utilize the aggregate function to initiate the aggregation pipeline.

- Unwind Courses: Consider using the $unwind operator if your documents have an array field containing courses. This deconstructs the array into separate documents, one for each course. Skip this step if courses are already stored as separate documents.

- Empty Array Initialization: Introduce a stage with the $project operator. Within $project, define a new field (e.g., allCourses) to hold the combined courses and initialize it as an empty array ([]).

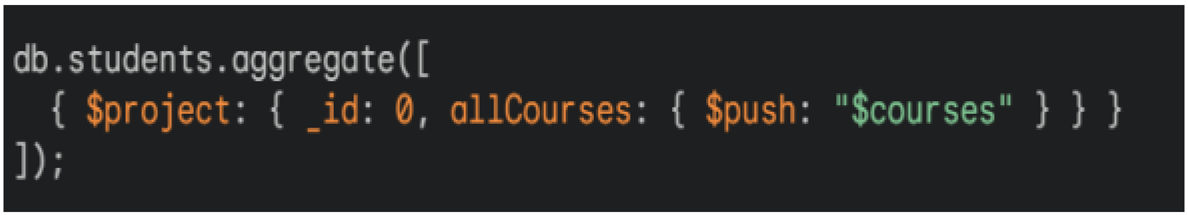
- Push Courses: Still within the $project stage, use the $push accumulator to append each course document (unwound or original) to the allCourses array.

Example (assuming courses are stored as separate documents):

1.db.students.aggregate([
{ $project: { allCourses: { $push: "$$ROOT" } } } // Push entire document])
2.db.students.aggregate([

{ $unwind: "$courses" }, // Deconstruct courses array

{ $project: { allCourses: { $push: "$courses" } } } // Push each course])



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

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 UINQUE COURSES OFFERED USING $ADD TO SET:

Here's how to collect unique forces offered using the $addToSet operator in MongoDB aggregation:

1. Aggregation Pipeline:
Utilize the aggregate function to initiate the aggregation pipeline.

2. $group Stage:
Include a stage with the $group operator to group documents potentially containing an array of forces offered.

3. Specify _id:
Within the $group stage, define _id: null (or another field for specific grouping if needed).

4. $addToSet for Forces:
Also within $group, use the $addToSet accumulator with the field name containing the forces offered (e.g., "forces").
This ensures only unique elements are added to the resulting array.

```
JavaScript
db.militaryUnits.aggregate([
{
    $group: {
    _id: null, // Group all documents
     uniqueForces: { $addToSet: "$forces" } // Collect unique forces
Offered
 }
 }
])
```

EXPLANATION:

- This pipeline processes all documents in the militaryUnits collection.
- The $group stage groups all documents into a single document.
- Inside $group, _id: null instructs it to consider all documents as a single group for force collection.
- The $addToSet operator, applied to the "forces" field, ensures only unique entries from the "forces" field across all documents are added to the newly created uniqueForces array in the resulting document.

RESULT:

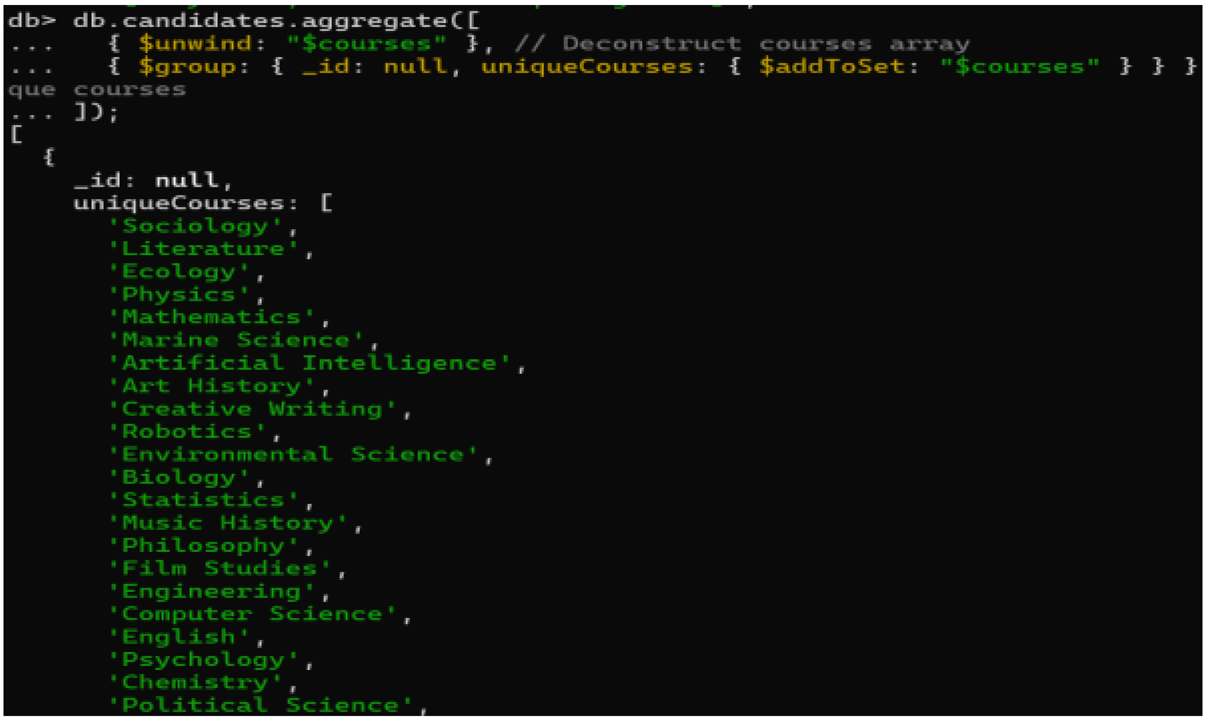
The resulting document will have:

I. _idThe value you specified (here, null).

II.

III. uniqueForces: An array containing all unique forces offered across all military units in the collection.

WHAT DOES IT DO:?



```
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',
```