AGGREGATION OPERATORS

In MongoDB, the **Aggregation framework** is a powerful tool for data processing and transformation. It uses a pipeline approach, where data is passed through a series of stages, each performing a **specific operation**.

The stages in pipeline can **filter, sort, group, reshape and modify** documents that pass through the pipeline.

The name itself says that **Aggregation** means **grouping together.**

For example: Sum,Avg,Min,Max.

Syntax for the Aggregation Operator is

db.collection.aggregate(<AGGREGATE OPERATION>)

→HERE'S AN OVERVIEW OF SOME AGGREGATION OPERATORS IN MONGODB WITH SYNTAX:

1.$sum: Sums numeric values for the documents in each group.

Syntax:"$fieldname":{$sum:"$fieldname"}

2.$avg: Calculates the average of numeric values.

Syntax;"$fieldname":{$avg:"$fieldname"}

3.$min: Finds the minimum value.

Syntax:"$fieldname":{$min:"$fieldname"}

4.$max: Finds the maximum value.

Syntax:"$fieldname":{$max:"$fieldname"}

5.$push: Appends a value to an array of values.

Syntax:" $fieldname":{$push:"$fieldname"}

6.$addtoset: Adds a value to an array, but only if the value is not already present in the array.

Syntax:"$fieldname":{$addtoset:"$fieldname"}

TYPES OF AGGREGATION OPERATORS

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" }

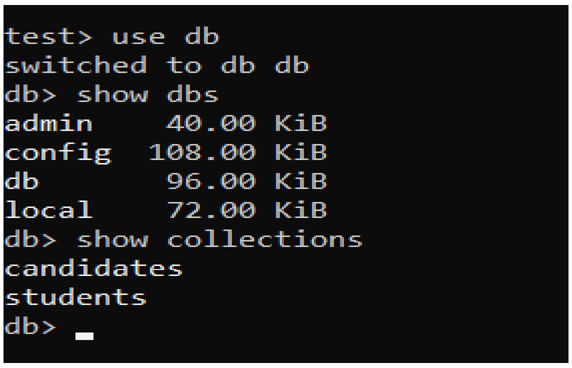
To perform aggregation operator lets import a collection called **”students”** through mongocompass.

To switch this database we must use a commands like

“use db”

“show dbs”

“show collections”

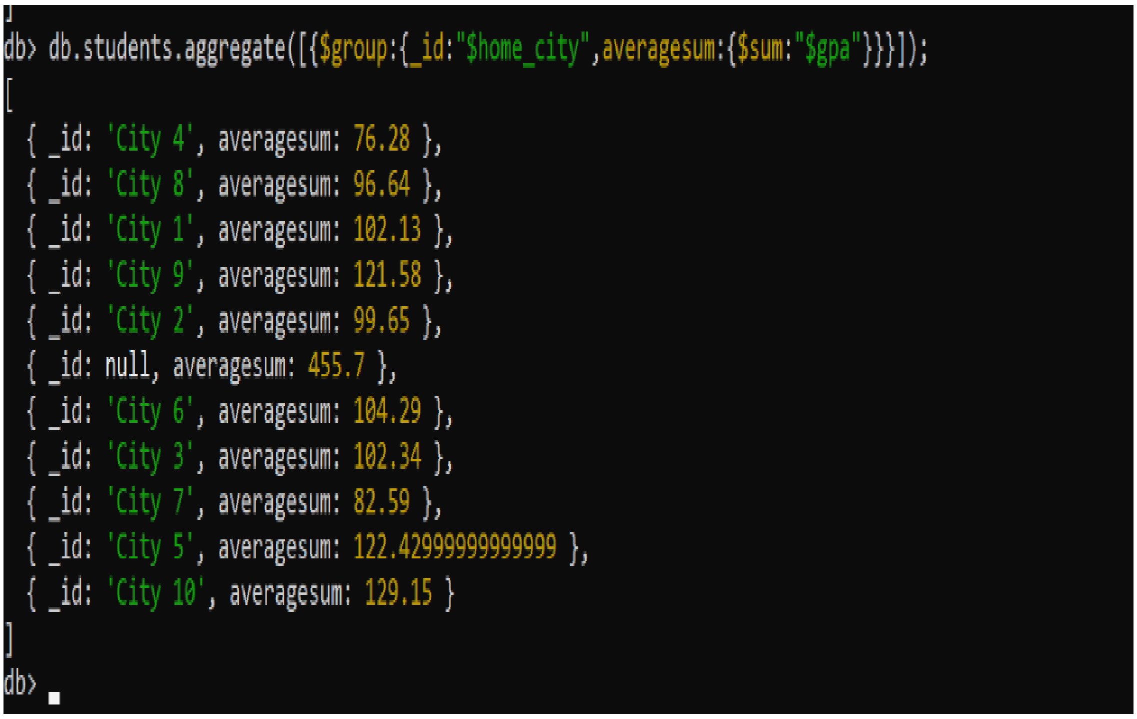


```
test> use db
switched to db db
db> show dbs
admin     40.00 KiB
config   108.00 KiB
db        96.00 KiB
local     72.00 KiB
db> show collections
candidates
students
db>
```

<u>1.$sum:</u>

Here is an example to find **averagesum** of gpa for all the home cities for this we have to use a command like

db.students.aggregate([$group:{_id:”$home_city”,averagesum:”$gpa” }}}]);



```
db> db.students.aggregate([{$group:{_id:"$home_city",averagesum:{$sum:"$gpa"}}}]);
[
  { _id: 'City 4', averagesum: 76.28 },
  { _id: 'City 8', averagesum: 96.64 },
  { _id: 'City 1', averagesum: 102.13 },
  { _id: 'City 9', averagesum: 121.58 },
  { _id: 'City 2', averagesum: 99.65 },
  { _id: null, averagesum: 455.7 },
  { _id: 'City 6', averagesum: 104.29 },
  { _id: 'City 3', averagesum: 102.34 },
  { _id: 'City 7', averagesum: 82.59 },
  { _id: 'City 5', averagesum: 122.42999999999999 },
  { _id: 'City 10', averagesum: 129.15 }
]
db>
```

Here we used,

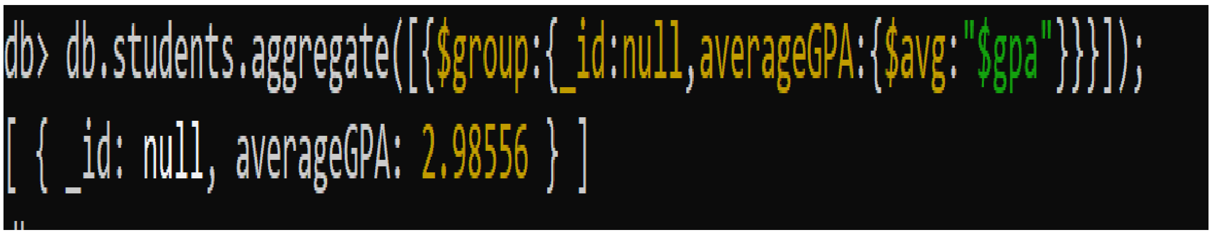
_ id:home_city:-which sets the identifier the homecity to document together.

Averagesum:-calculates the averagesum value of students who scored particular gpa field in home cities using **$sum operator .**

2.$avg:

Here to find averageGPA of all the students we need to use a command

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



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

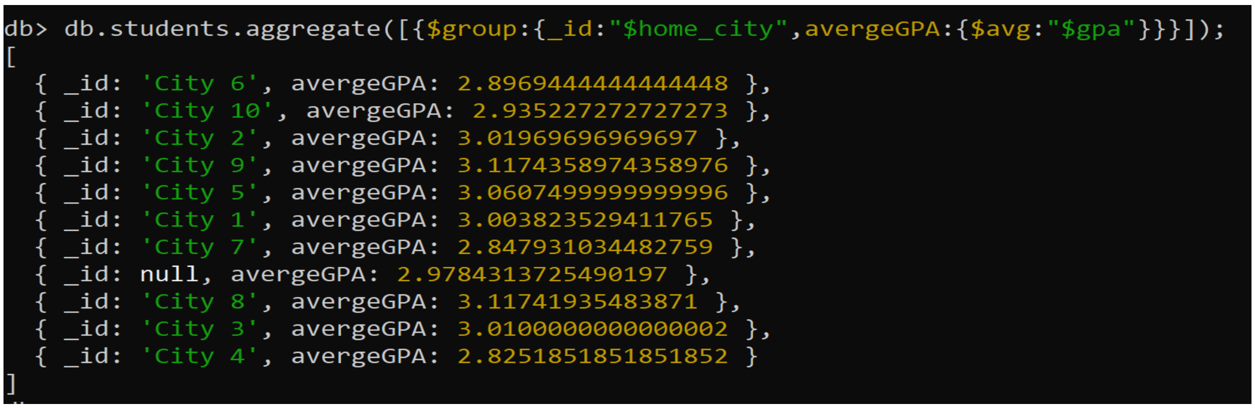
Here we used,

$group:-Groups all documents together

_ id:null:-sets the group identifier to null.

averageGPA:-calculates the average value of the “gpa”field using **$avg operator.**

One more example using **$avg operator**,Here we are finding average gpa for all home cities use a command is **db.students.aggregate([{$group:{_id:”$home_city”,averagGPA:{$avg:”$gpa”}}}]);**

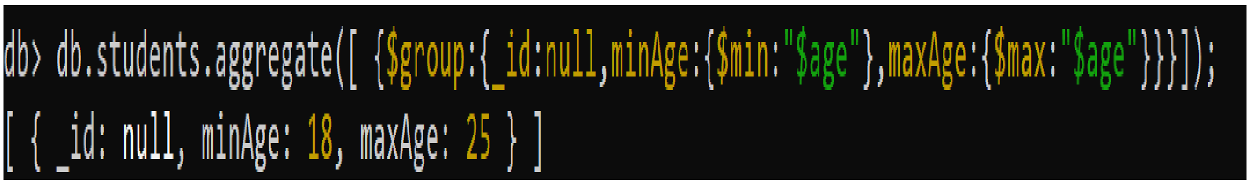


```
db> db.students.aggregate([{$group:{_id:"$home_city",avergeGPA:{$avg:"$gpa"}}}]);
[
  { _id: 'City 6', avergeGPA: 2.8969444444444448 },
  { _id: 'City 10', avergeGPA: 2.935227272727273 },
  { _id: 'City 2', avergeGPA: 3.01969696969697 },
  { _id: 'City 9', avergeGPA: 3.1174358974358976 },
  { _id: 'City 5', avergeGPA: 3.0607499999999996 },
  { _id: 'City 1', avergeGPA: 3.003823529411765 },
  { _id: 'City 7', avergeGPA: 2.847931034482759 },
  { _id: null, avergeGPA: 2.9784313725490197 },
  { _id: 'City 8', avergeGPA: 3.11741935483871 },
  { _id: 'City 3', avergeGPA: 3.0100000000000002 },
  { _id: 'City 4', avergeGPA: 2.8251851851851852 }
]
```

3.$min and $max:

To find Minimum and Maximum age we need to use a command called

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



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

Here we used ,

$group:-Groups all documents together

_ id:null:- sets the group identifier to null.

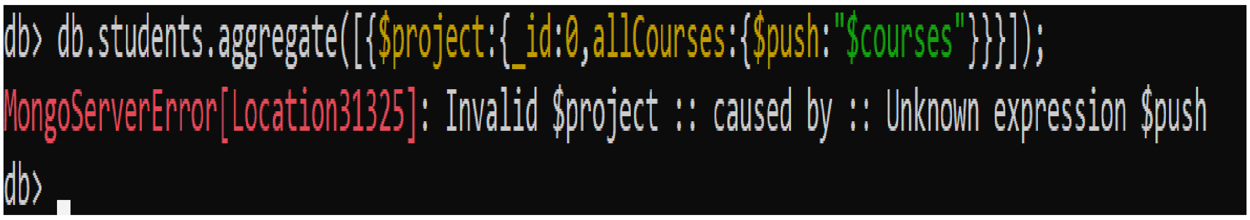
using **$min and $max operator** we found a minimum value and maximum value of age field.

4.$push:

Here pushing all the courses into a single array using **$push operator** to receive an array in order.

For this we use a command

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



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

Here we used

$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:

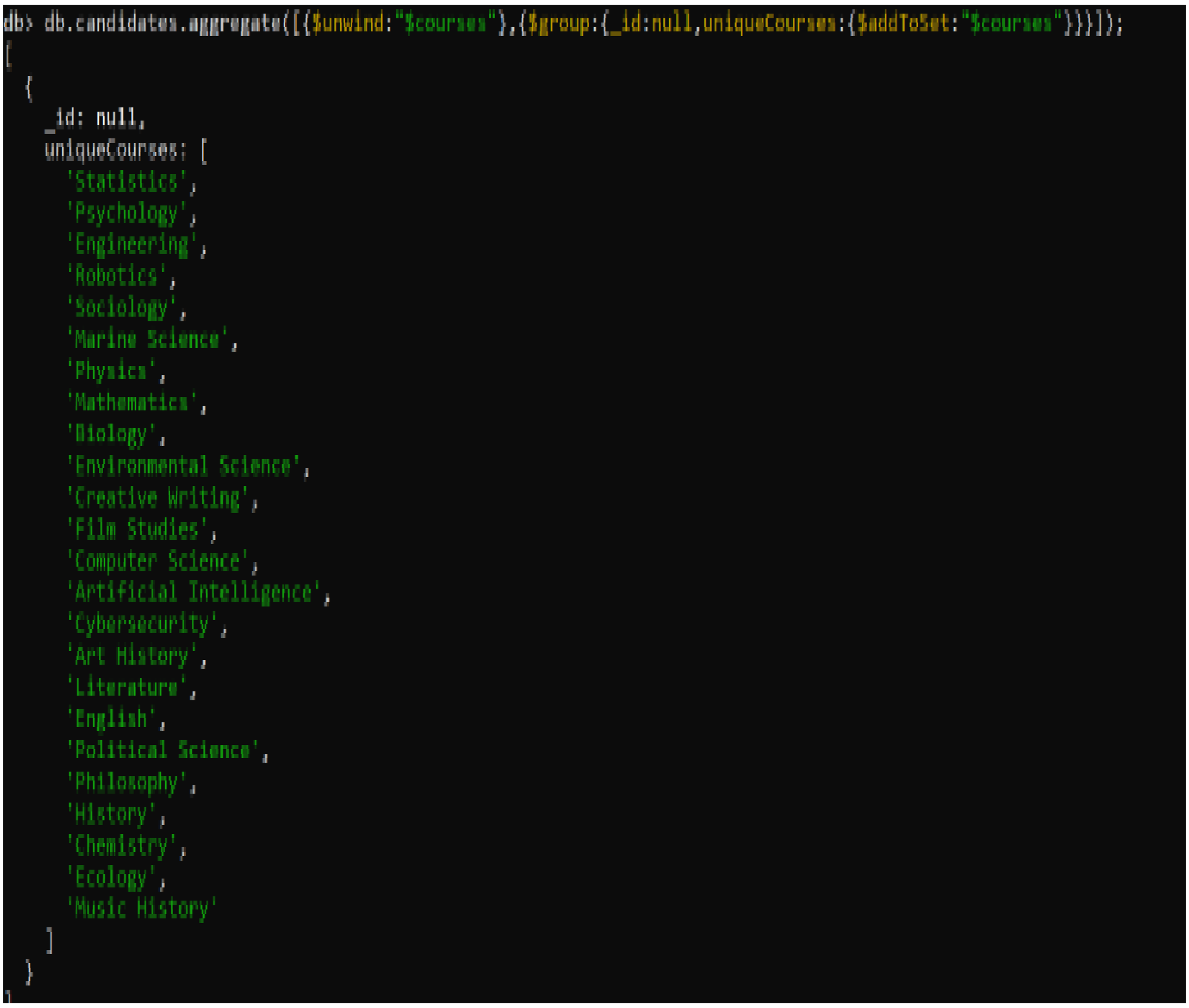
This will return a list of documents, each with an allCourses array containing all unique courses offered.

We received an output like **invalid $project** this is because our Array is incorrect.

5.$addToSet:

To collect unique courses offered we use a command called

db.candidates.aggregate([{ $unwind: "$courses" }, { $group: { _id: null, uniqueCourses: { $addToSet:"$courses" } } }]);



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

In output we got all the Unique courses which were offered to students.