MONGO DB

CLASS 6: AGGREGATION OPERATION.

Aggregation in MongoDB is a powerful feature that allows for complex data transformations and computations on collections of documents. It enables users to group, filter, and manipulate data to produce summarized results.

What is aggregation?

● MongoDB Aggregation is a database process that allows us to perform complex data transformations and computations on collections of documents or rows.

● It enables us to group, filter, and manipulate data to produce summarized results. MongoDB Aggregation is typically carried out using the aggregation pipeline, which is a framework for data aggregation modeled on the concept of data processing pipelines. As they pass through it and allowing for operations like filtering, grouping, sorting, reshaping and performing calculations on the data.

Use Of Aggregation:

1. **Grouping Data:** You can group values from multiple documents together based on specific fields. For example, you might group sales data by product category or customer.
2. **Calculating Aggregates:** Aggregation pipelines allow you to perform operations on grouped data to return a single result. You can calculate totals, averages, maximums, minimums, and other aggregate values.
3. **Analyzing Data Changes Over Time:** Aggregations help you analyze data trends and changes over time. For instance, you can track monthly sales growth or user engagement patterns.

AGGREGATE OPERATORS:

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

Syntax:

db.collection.aggregate(<AGGREGATE OPERATION>)

TYPES OF AGGREGATE OPERATORS WITH THEIR SYNTAX:

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.$sum:

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.

Average sum:-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.

What is $group?

The $group stage is used to group documents based on one or more fields and perform aggregation operations on the grouped data. It allows you to:

• Group documents by one or more fields

• Perform aggregation operations on the grouped data, such as sum, average, count, etc.

• Create new fields that represent the aggregated values

The $group stage takes an object as its argument, where each key is the name of a field and the value is an expression that defines the aggregation operation.

_ 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:"$age"},max Age:{$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.

What is $project?

The $project stage is used to transform and reshape the data in the pipeline. It allows you to:

- Add new fields to the documents
- Rename existing fields
- Remove fields
- Perform calculations and transformations on fields
- Create new arrays or objects

The $project stage takes an object as its argument, where each key is the name of a field and the value is an expression that defines the transformation.

_ 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.