CLASS 6

AGGREGATION OPERATORS

AGGREGATION:

Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. In SQL count(*) and with group by is an equivalent of MongoDB aggregation.

The aggregate() Method

For the aggregation in MongoDB, you should use **aggregate()** method.

Syntax:

Basic syntax of aggregate() method is as follows –

```
>db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

Types:

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" }

Average GPA of all Students:

```
test> use db
switched to db db
db> db.students.aggregate([
... {$group:{_id:null,averageGPA:{$avg:"$gpa"}}}
... ]);
[ { _id: null, averageGPA: 3.2268699186991867 } ]
db>
```

Explanation:

- db.students.aggregate: This line initates the aggregation framework operation on the "students" collections.
- $group: This stage is responsible for grouping documents and performing calculations on the groups.
- _id null: This specifies that we don't need documents grouped by any particular field. We want the average age for all students combined. Setting _id: null creates a single group containing all documents.
- averageAge: { $avg: "$gpa" }: This calculates the average age of all the students.
- $avg: This is the accumulator that calculates the average value of the "age" field for all documents in the group (since we set _id: null).

Minimum and Maximum Age:

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

OUTPUT:

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

- db.students.aggregate(...): This initiates an aggregation operation on the students collection.
- [...]: The aggregation pipeline is defined within these square brackets. In this case, it consists of a single stage.
- { $group: { ... } }: This is the $group stage in the aggregation pipeline. It groups documents by a specified identifier and can perform various operations, such as calculating averages, sums, etc.

- _id: null: This specifies that all documents should be grouped into a single group. Setting _id to null means that there is no grouping by a particular field, so all documents are treated as a single group.
- averageGPA: { $avg: "$gpa" }: This calculates the average of the gpa field across all documents in the collection. The result will be stored in the averageGPA 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: 2.896944444444448 },
  { _id: null, averageGPA: 2.9784313725490197 },
  { _id: 'City 4', averageGPA: 2.8251851851851852 },
  { _id: 'City 1', averageGPA: 3.003823529411765 },
  { _id: 'City 5', averageGPA: 3.0607499999999996 }
]
```

- The average GPA for all home cities in MongoDB can be calculated using an aggregation query that groups documents by the `homeCity` field and computes the mean of the `gpa` field for each group.
- This is done by using the `$group` stage with `_id` set to `$homeCity` and the `$avg` operator applied to the `gpa` field.

Collect Unique Courses Offered (Using $addToSet):

The $addToSet operator in MongoDB's Aggregation Framework can be used to collect unique values in an array field. To collect unique courses offered using $addToSet, you would need to have a collection that contains documents with a field representing the courses offered.

Here's an example using the persons collection from the "Practical MongoDB Aggregations" documentation:

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

This aggregation pipeline groups all documents together (_id: null) and uses $addToSet to collect unique values from the course field into the coursesOffered array.