CLASS-6

AGGREGATION OPERATORS

Aggregation operations process multiple documents and return computed results. You can use aggregation operations to:

- Group values from multiple documents together.
- Perform operations on the grouped data to return a single result.
- Analyse data changes over time.

Syntax:

db.collection.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" }

$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.

$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.

Example:

To find the average GPA of all students

```
db> db.students.aggregate([{$group:{_id: null, averageGPA:{$avg: "$gpa"}}}]);
[ { _id: null, averageGPA: 3.308273542600897 } ]
```

_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.

To find the Minimum and maximum age:

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

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.

To find the Minimum and maximum GPA:

```
db> db.students.aggregate([{$group:{_id: null, maxGpa:{$max: "$gpa"}}}]);
[ { _id: null, maxGpa: 3.99 } ]
db> db.students.aggregate([{$group:{_id: null, minGpa:{$min: "$gpa"}}}]);
[ { _id: null, minGpa: 2.51 } ]
```

minGpa: Uses the $min operator to find the minimum value in the "gpa" field.
maxGpa: Uses the $max operator to find the maximum value in the "gpa" field.

To get average GPA for all the home cities

```
db> db.students.aggregate([{$group:{_id: "$home_city", averageGPA: {$avg:"$gpa"}}}]);
[
  { _id: null, averageGPA: 3.3206474820143885 },
  { _id: 'City 7', averageGPA: 3.2042857142857137 },
  { _id: 'City 5', averageGPA: 3.366470588235294 },
  { _id: 'City 9', averageGPA: 3.381111111111111 },
  { _id: 'City 1', averageGPA: 3.3738709677419356 },
  { _id: 'City 6', averageGPA: 3.239375 },
  { _id: 'City 3', averageGPA: 3.3045161290322578 },
  { _id: 'City 2', averageGPA: 3.2856666666666663 },
  { _id: 'City 8', averageGPA: 3.3918518518518517 },
  { _id: 'City 4', averageGPA: 3.1856 },
  { _id: 'City 10', averageGPA: 3.24925 }
]
```

Collect Unique Courses Offered (Using $addToSet):

```
db> db.students.aggregate([{ $unwind: "$courses"},{ $group:{
_id:null, uniqueCourses:{ $addToSet: "$courses"}}}]);
[
  {
    _id: null,
    uniqueCourses: [
      "['Mathematics', 'History', 'Physics']",
      "['English', 'History', 'Physics', 'Computer Science']
",
      "['Physics', 'Mathematics', 'English', 'Computer Scien
ce']",
      "['Physics', 'Computer Science', 'History', 'Mathemati
cs']",
      "['Computer Science', 'Physics', 'History', 'Mathemati
cs']",
      "['Mathematics', 'English']",
      "['Computer Science', 'English', 'Physics', 'History']
",
      "['Physics', 'Computer Science', 'English']",
      "['History', 'Computer Science', 'Mathematics', 'Engli
sh']",
      "['Physics', 'History', 'English', 'Computer Science']
",
      "['Mathematics', 'Computer Science']",
      "['Mathematics', 'Computer Science', 'History', 'Physi
cs']",
      "['Physics', 'English', 'History', 'Computer Science']
",
      "['Computer Science', 'Mathematics', 'History', 'Engli
sh']",
      "['Physics', 'English']",
      "['Mathematics', 'English', 'Physics', 'History']",
      "['History', 'English', 'Physics', 'Mathematics']",
      "['History', 'English']",
```