Class 6

Aggregation Operators:

Aggregation operations process multiple documents and return computed results.

What is aggregation?

Aggregation in MongoDB is a powerful framework for data aggregation operations, used to process data records and return computed results. It can be compared to the SQL GROUP BY clause but offers much more flexibility and capability. The aggregation framework operates through a pipeline approach, where multiple stages are used to transform and filter documents.

Grouping : Aggregation enables you to group documents based on shared characteristics. You can group by one or more fields within your documents.

This grouping helps you analyze data for specific categories or segments within your collection.

Benefits of Aggregation:

- Aggregation offers several advantages, including:
- Efficient data exploration and analysis without needing to write complex queries.
- Ability to perform calculations and transformations on the server-side, reducing the workload on your application.
- Creation of new derived data fields for further analysis.

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" }

Average GPA of All Students

The provided MongoDB aggregation query calculates the average GPA of all students in the students collection.

- **db.students.aggregate(...)**: This line initiates the aggregation pipeline on the "students" collection within the database. The aggregation framework allows you to perform various operations to transform and group data.

- **[{$group:{...}}]**: This defines an array containing a single aggregation stage. Each stage in the pipeline manipulates the data passed from the previous stage.

- **_id: null**: This specifies that we don't want to group documents based on any specific field. Setting _id to null creates a single group containing all documents from the "students" collection.

- **averageGPA: {$avg: "$gpa"}**: This is the key part where the average GPA is calculated.

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

Output:

Explanation:

- $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.

Finding the Minimum and Maximum Age:

Here we use this to find out the minimum and maximum age in the given collection.

Output:

Explanation:

- Similar to the previous example, it uses $group to group all documents.
- 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.

Now we will see How to get a Average GPA for all home cities that are in the collection.

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

We use the above code to find out the average GPA of all home cities.

Output:

```
[
  { _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: 2.8969444444444448 },
  { _id: null, averageGPA: 2.9784313725490197 },
  { _id: 'City 4', averageGPA: 2.8251851851851852 },
  { _id: 'City 1', averageGPA: 3.003823529411765 },
  { _id: 'City 5', averageGPA: 3.0607499999999996 }
]
```

The above output shows us that the average of all the cities that are present in the collections.

This provides the aggregation pipeline successfully identified unique courses across all candidate documents in your "candidates" collection.

- **_id: null**: This confirms that the aggregation didn't group the data based on any specific field because _id is set to null. It calculated unique courses for the entire collection, considering all courses mentioned by all candidates.

- **uniqueCourses**: This field stores an array containing the unique course names found in the "courses" field of each candidate document. The list you might see in the output would include all the unique course names, not just the three examples ("Math", "Physics", "Chemistry").

Pushing All Courses into a Single Array:

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

- Aggregation Framework:
  - db.students.aggregate() is the entry point for the aggregation framework on the students collection.

- **$project Stage**:
  - { $project: { _id: 0, allCourses: { $push: "$courses" } } }is a projection stage in the aggregation pipeline.
  - _id: 0:
    - This excludes the _idfield from the output documents.
  - allCourses: { $push: "$courses" }:
    - The $pushoperator is used to append values to an array in the resulting documents.
    - "$courses"specifies the field whose values will be pushed into the allCourses array.

Explanation:

- $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.

BUT:

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

This is because our Array is incorrect :)

Collect Unique Courses Offered (Using $addToSet);

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

Aggregation Framework:

db.students.aggregate() is the entry point for the aggregation framework on the students collection.

$group Stage:{ $group: { _id: null, minAge: { $min: "$age" }, maxAge: { $max: "$age" } } }is a grouping stage in the aggregation pipeline.

Result:

The result of this aggregation query will be a single document containing the minimum and maximum ages of all students:

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',
```

- _id: null: This confirms that the aggregation didn't group the data based on any field (_id is set to null). It calculated unique courses across the entire collection.

- uniqueCourses: This field stores an array containing all the unique course names found in the "courses" field of each candidate document. The document snippet you provided truncates the list after a few elements ([...]).