MONGO DB

Aggregation pipeline

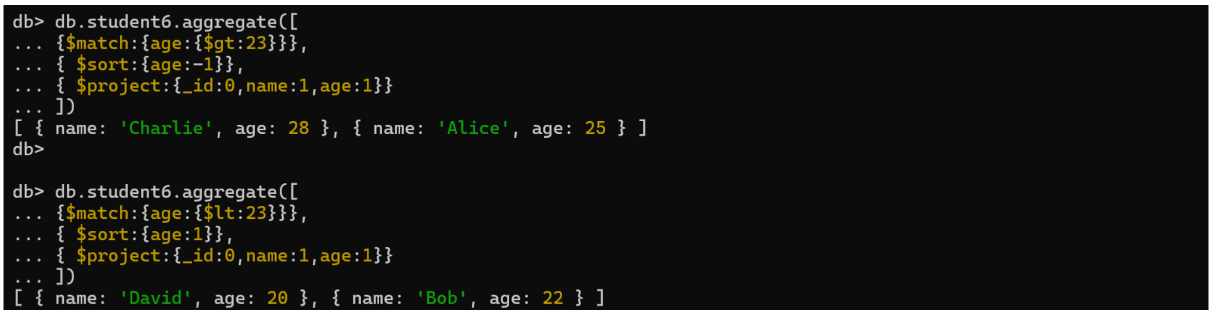
The aggregation pipeline is a series of data processing stages in MongoDB that allow you to transform, filter, and aggregate data from a collection. It's a powerful tool for data analysis and processing, and is often used to generate reports, perform data mining, and create data visualizations.

AGENDA

The agenda for this experiment is to execute aggregation pipelines and their operations. The pipeline must contain $match, $group, $sort, $project, $skip, etc. Students are encouraged to execute several queries to demonstrate various aggregation operators.

Keyword	**Description**	**Syntax**
$match	Filters documents in the pipeline	**{ $match: { <condition> } }**
$group	Groups documents in the pipeline	**{ $group: { _id: <expression>, <field>: { <accumulator> } } }**
$sort	Sorts documents in the pipeline(ascending or descending order)	**{ $sort: { <field>: <order> } }**
$project	Reshapes each document in the pipeline	**{ $project: { <field>: <expression>, ... } }**
$skip	Skips a specified number of documents in the pipeline	**{ $skip: <number> }**
$limit	Limits the number of documents in the pipeline	**{ $limit: <number> }**
$unwind	Deconstructs an array field from the input documents	**{ $unwind: <field> }**
$count	Returns a count of the number of documents in the pipeline	**{ $count: <field> }**

SORTING



```
db> db.student6.aggregate([
... {$match:{age:{$gt:23}}},
... { $sort:{age:-1}},
... { $project:{_id:0,name:1,age:1}}
... ])
[ { name: 'Charlie', age: 28 }, { name: 'Alice', age: 25 } ]
db>

db> db.student6.aggregate([
... {$match:{age:{$lt:23}}},
... { $sort:{age:1}},
... { $project:{_id:0,name:1,age:1}}
... ])
[ { name: 'David', age: 20 }, { name: 'Bob', age: 22 } ]
```

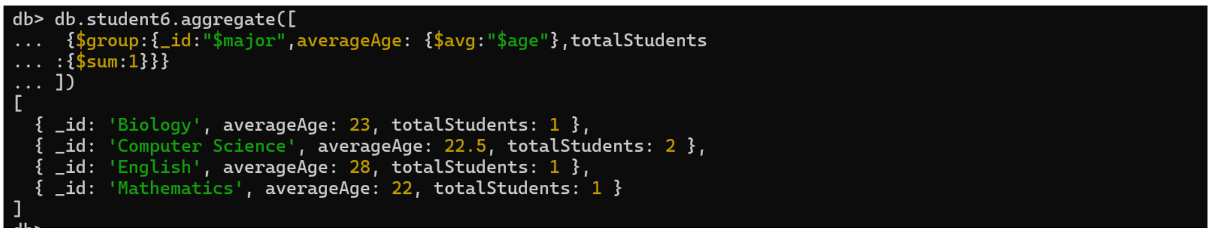
First Query

- **$match: {age:{$gt:23}}** This stage filters the documents in the "student6" collection, keeping only those where the "age" field is greater than 23.
- **$sort: {age:-1}**` This stage sorts the remaining documents in descending order based on the "age" field.
- **$project: {_id:0,name:1,age:1}** This stage transforms the documents by:
    - Removing the **_id** field (using **_id:0**).
    - Including the **name** and **age** fields (using **name:1, age:1**).

Second Query

- **$match: {age:{$lt:23}}** This stage filters documents, keeping only those where the "age" field is less than 23.
- **$sort: {age:1}**` This stage sorts the remaining documents in ascending order based on the "age" field.
- **$project: {_id:0,name:1,age:1}** Same as before, removes the **_id** and includes the **name** and **age** fields.

AVERAGE:

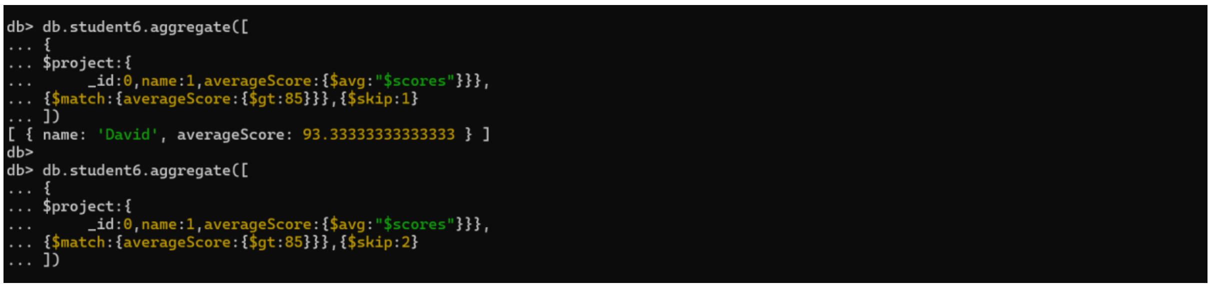


```
db> db.student6.aggregate([
...   {$group:{_id:"$major",averageAge: {$avg:"$age"},totalStudents
... :{$sum:1}}}
... ])
[
  { _id: 'Biology', averageAge: 23, totalStudents: 1 },
  { _id: 'Computer Science', averageAge: 22.5, totalStudents: 2 },
  { _id: 'English', averageAge: 28, totalStudents: 1 },
  { _id: 'Mathematics', averageAge: 22, totalStudents: 1 }
]
```

This is the core of the aggregation. It groups documents by their major (**_id: "$major"**). Then, it performs the following operations:

- **averageAge: {$avg: "$age"}**: Calculates the average age of students in each group by averaging the **age** field.
- **totalStudents: {$sum: 1}**: Counts the total number of students in each group by adding 1 for each document in the group.

SKIP:



```
db> db.student6.aggregate([
... {
... $project:{
...       _id:0,name:1,averageScore:{$avg:"$scores"}}},
... {$match:{averageScore:{$gt:85}}},{$skip:1}
... ])
[ { name: 'David', averageScore: 93.33333333333333 } ]
db>
db> db.student6.aggregate([
... {
... $project:{
...       _id:0,name:1,averageScore:{$avg:"$scores"}}},
... {$match:{averageScore:{$gt:85}}},{$skip:2}
... ])
```

- **$project**: This stage is used to reshape the documents in the collection. It defines which fields to keep, remove, or rename, and how to calculate new fields.
- **_id: 0**: This line removes the default "_id" field from the output documents.
- **name: 1**: This keeps the "name" field in the output documents.
- **averageScore: { $avg: "$scores" }**: This creates a new field called "averageScore" and calculates its value by averaging the values within the "scores" field of each document.
- **{ $match: { averageScore: { $gt: 85 } } }**
- **$match**: This stage filters documents based on a given criteria.
- **averageScore: { $gt: 85 }**: This filter only allows documents to pass through the pipeline if their "averageScore" is greater than 85.
- **{ $skip: 1 }**

  $skip: This stage skips a specified number of documents. Here, it skips the first document that matches the previous filters.
- **{ $skip: 2 }**

  $skip: This stage skips a specified number of documents. Here, it skips the first two document that matches the previous filters.