AGGREGATION PIPELINE

Aggregation in MongoDB involves processing data records and returning computed results. The aggregation framework is powerful and can be used to perform a wide variety of operations. Here are the key operators used in MongoDB aggregation pipelines:

Aggregation Pipeline Stages

1. **$match**: Filters the documents to pass only those that match the specified condition(s).

```
{ $match: { status: "A" } }
```

2. **$group:** Groups input documents by a specified identifier expression and applies the accumulator expressions to each group.

```
{ $group: { _id: "$status", total: { $sum: 1 } } }
```

3. $project: Reshapes each document in the stream by adding, removing, or renaming fields.

```
{ $project: { item: 1, total: 1, status: 1 } }
```

4. **$sort:** Sorts all input documents and returns them in sorted order.

```
{ $sort: { total: -1 } }
```

5. **$limit**: Limits the number of documents passed to the next stage.

```
{ $limit: 5 }
```

6. $skip: Skips the first n documents and passes the remaining documents to the next stage.

```
{ $skip: 10 }
```

7. **$unwind:** Deconstructs an array field from the input documents to output a document for each element.

```
{ $unwind: "$items" }
```

8. **$lookup**: Performs a left outer join to another collection in the same database to filter in documents from the "joined" collection for processing.

```
{
  $lookup: {
    from: "otherCollection",
    localField: "fieldName",
    foreignField: "foreignFieldName",
    as: "alias"
  }
}
```

9. **$addFields**: Adds new fields to documents.

```
{ $addFields: { newField: "$existingField" } }
```

10. **$replaceRoot**: Replaces the input document with the specified document.

```
{ $replaceRoot: { newRoot: "$newDoc" } }
```

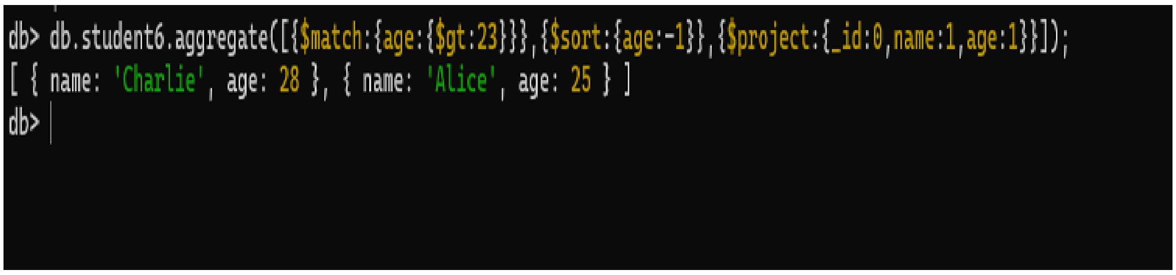
11. **$count**: Returns a count of the number of documents input to the stage.

```
{ $count: "total" }
```

12**. $facet**: Processes multiple aggregation pipelines within a single stage on the same set of input documents.

```
{
  $facet: {
    "groupedByStatus": [{ $group: { _id: "$status", count: { $sum: 1 } } }],
    "groupedByCategory": [{ $group: { _id: "$category", count: { $sum: 1 } } }]
  }
}
```

Find students with age greater than 23, sorted by age in descending order, and only return name and age :



```
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>
```

1. db.student6.aggregate([...]);

● This starts an aggregation pipeline on the student6 collection within your MongoDB database.

2. {$match:{age:{$gt:23}}};

● $match: This stage filters the documents.
● {age:{$gt:23}}: This filters documents where the "age" field is greater than 23.

3. {$sort:{age:-1}};

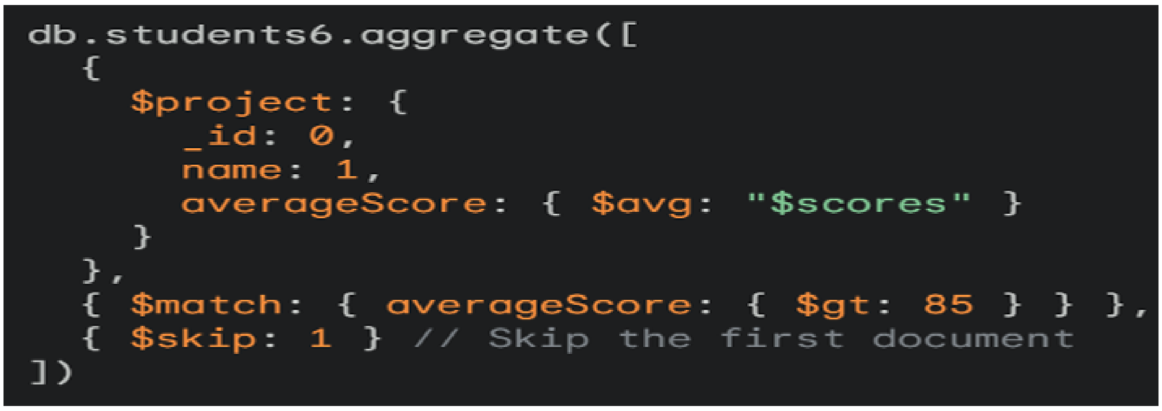
● $sort: This stage sorts the documents.
● {age:-1}: This sorts documents in descending order based on the "age" field.

4. {$project: {_id:0,name:1,age:1}};

● $project: This stage reshapes the documents, specifying which fields to keep and how to format them.
● {_id:0,name:1,age:1}:
● _id:0 removes the default _id field.
● name:1 keeps the "name" field as is.
● age:1 keeps the "age" field as is. The pipeline will produce an array of documents, each containing only the "name" and "age"

fields. The documents will be sorted in descending order by age, ensuring that the oldest student appears first

Find students with an average score (from scores array) above 85 and skip the first document:



```
db.students6.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      averageScore: { $avg: "$scores" }
    }
  },
  { $match: { averageScore: { $gt: 85 } } },
  { $skip: 1 } // Skip the first document
])
```

Explanation:

The code snippet is a MongoDB aggregation pipeline that performs the following actions:

1. $project: It selects only the name and averageScore fields from each document, and calculates the average of the scores field as averageScore.

2. $match: It filters the results to include only documents where averageScore is greater than 85.

3. $skip: It skips the first document in the filtered results

This indicates that David has the highest average score (greater than 85) among all the students, excluding the first student in the collection