MongoDB order with Sort() & Limit() Query

What is Query Modifications?

- Mongo DB provides query modifiers such as the 'limit' and 'Orders' clause to provide more flexibility when executing queries. We will take a look at the following query modifiers

MongoDB Limit Query Results

- This modifier is used to limit the number of documents which are returned in the result set for a query. The following example shows how this can be done.

```
db.Employee.find().limit(2).forEach(printjson);
```

MongoDB Sort by Descending Order

- One can specify the order of documents to be returned based on ascending or descending order of any key in the collection. The following example shows how this can be done.

```
db.Employee.find().sort({Employeeid:-1}).forEach(printjson)
```

MongoDB Count() & Remove() Functions with Examples

- The concept of aggregation is to carry out a computation on the results which are returned in a query. For example, suppose you wanted to know what is the count of documents in a collection as per the query fired, then MongoDB provides the count() function.
- Let's look at an example of this.

```
db.Employee.count()
```

Performing Modifications

The other two classes of operations in MongoDB are the update and remove statements. The update operations allow one to modify existing data, and the remove operations allow the deletion of data from a collection.

Deleting Documents

- In MongoDB, the **db.collection.remove ()** method is used to remove documents from a collection. Either all of the documents can be removed from a collection or only those which matches a specific condition.
- If you just issue the remove command, all of the documents will be removed from the collection.
- The following code example demonstrates how to remove a specific document from the collection.

```
db.Employee.remove({Employeeid:22})
```

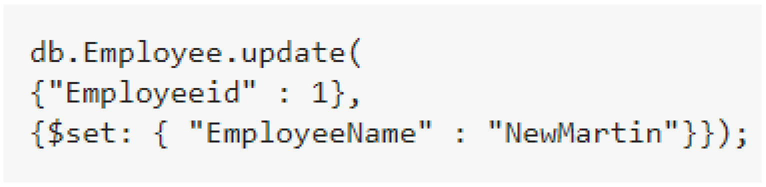
MongoDB Update() Document with Example

Basic document updates

- MongoDB provides the update () command to update the documents of a collection. To update only the documents you want to update, you can add a criterion to the update statement so that only selected documents are updated.

- The basic parameters in the command are a condition for which document needs to be updated, and the next is the modification which needs to be performed.
- The following example shows how this can be done.

1. Issue the update command
2. Choose the condition which you want to use to decide which document needs to be updated. In our example, we want to update the document which has the Employee id 22
3. Use the set command to modify the Field Name
4. Choose which Field Name you want to modify and enter the new value accordingly.

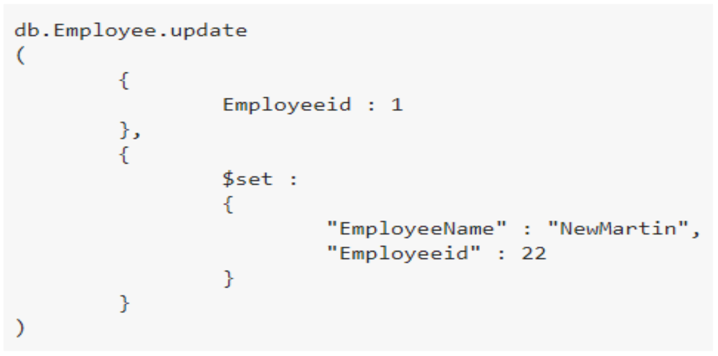


```
db.Employee.update(
{"Employeeid" : 1},
{$set: { "EmployeeName" : "NewMartin"}});
```

Updating Multiple Values

To ensure that multiple/bulk documents are updated at the same time in MongoDB you need to use the multi option because otherwise by default only one document is modified at a time. The following example shows how to update many documents.In this example, we are going to first find the document which has the Employee id as "1" and change the Employee name from "Martin" to "NewMartin"

1. Issue the update command & Choose the condition which you want to use to decide which document needs to be updated. In our example, we want the document which has the Employee id of "1" to be updated.
2. Choose which Field Name's you want to modify and enter their new value accordingly.



```
db.Employee.update
(
        {
                Employeeid : 1
        },
        {
                $set :
                {
                        "EmployeeName" : "NewMartin",
                        "Employeeid" : 22
                }
        }
)
```