Database Management Systems Laboratory

Assignment B3

Date of Submission: 5-11-2020

Title: Aggregation and Indexing

Roll number: 31301

Name: Aboli Marathe

Class: TE-3

Batch: K-3

PICT, Pune

ASSIGNMENT B3

TITLE	AGGREGATION AND INDEXING
DATE	
PROBLEM STATEMENT	Implement aggregation and indexing with suitable example using MongoDB.
LEARNING OBJECTIVES	• To understand indexing and aggregation concept in MongoBB
LEARNING OUTCOMES	Students will be able to implement aggregation and indexing in MongoDB.
H/W & S/W REQUIREMENTS	Computer System with Linux\ 64 bit OS, MongoDB server, mouse, keyboard

THEORY INDEXING

- Indexes support the effiient resolution of queries. Without indices, MongoDB must scan every document of a collection to select those documents that match the query statement. This scan is highly inefficient and requires the MongoDB to process a large volume of data.

- Indexes are special data structures, which store a small portion of the data set in an easy to traverse form The index stores the value of the specified field or set of fields, ordered by the value of the field as specified in index. Indexing can be achieved on any field in a document using ensure Index () method.

Syntax:

```
db . COLLECTION_NAME. ensureIndex
    ({ KEY: 1})
```

AGGREGATION

Aggregation operations process data records and return computed results. Aggregation operations graup values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. In sql count (*) and with groupby is an equivalent of MongoDB aggregation. The aggregate() method for the aggregation in MongoDB one should use aggregate () method

Syntax:

db. COLLECTION-NAME. aggregate (AGGREGATE OPERATION)

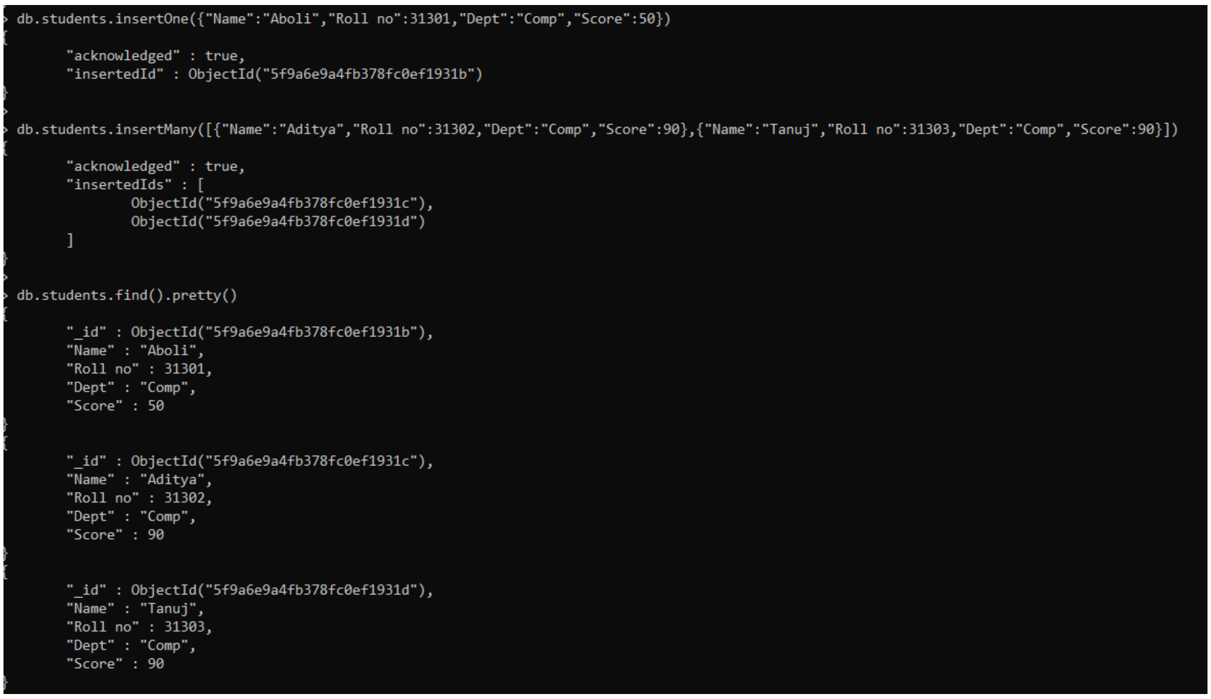
The various available aggregation expressions are:

$ sum - sums up the defined value from all document in the collection

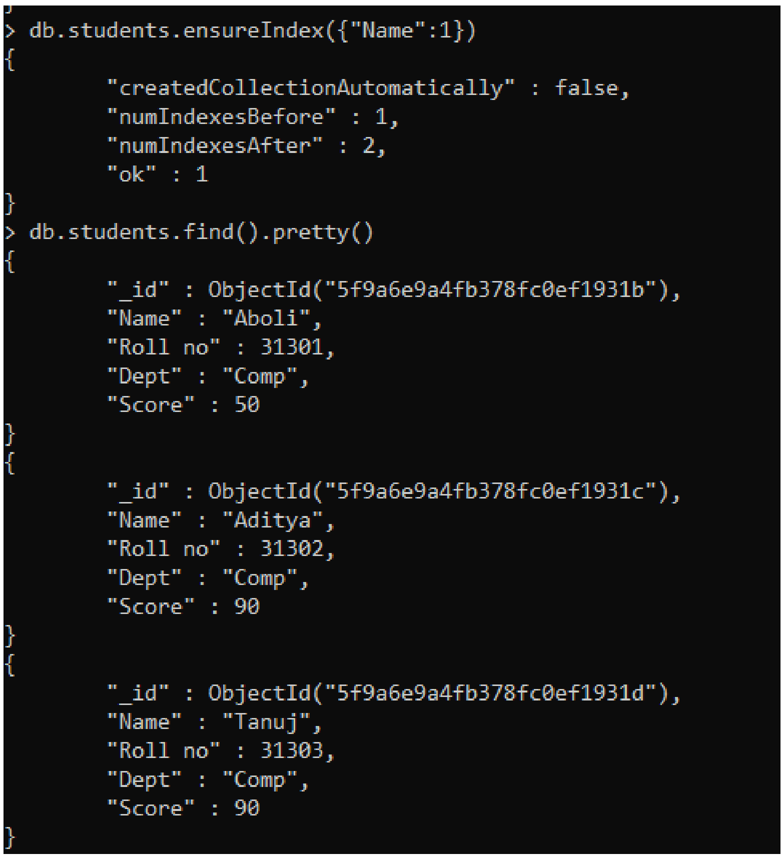
$ avg -	It computes average of defined value from all documents in the collection.
$ max -	Displays maximum of defined value from all documents in the collection.
$ min -	Displays minimum of defined value from all documents in collection.

Conclusion Thus we have successfully studied and implemented aggregating & indexing in MongoDB.

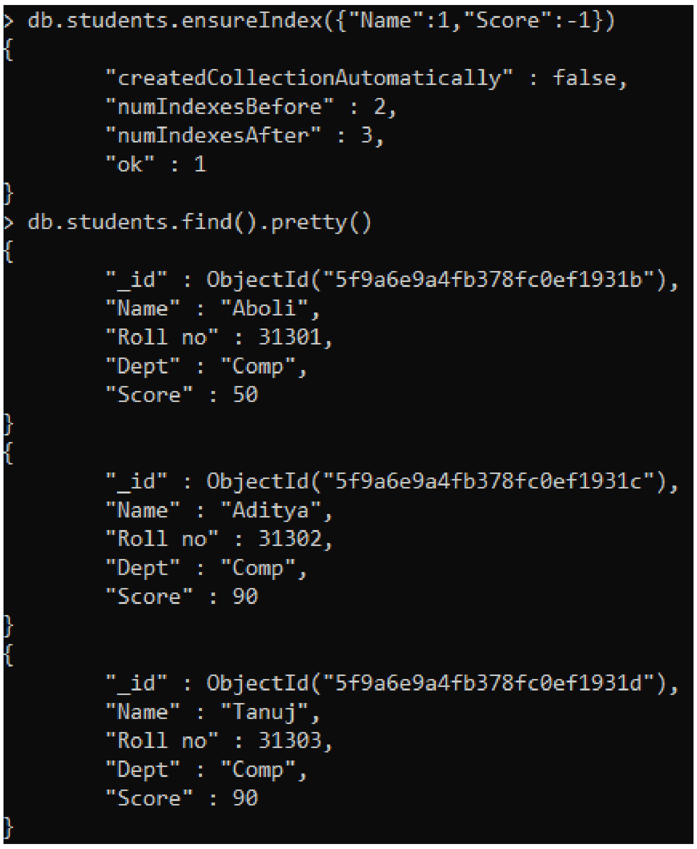
OUTPUT



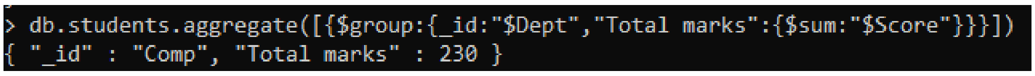
```
> db.students.insertOne({"Name":"Aboli","Roll no":31301,"Dept":"Comp","Score":50})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5f9a6e9a4fb378fc0ef1931b")
}
>
> db.students.insertMany([{"Name":"Aditya","Roll no":31302,"Dept":"Comp","Score":90},{"Name":"Tanuj","Roll no":31303,"Dept":"Comp","Score":90}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5f9a6e9a4fb378fc0ef1931c"),
                ObjectId("5f9a6e9a4fb378fc0ef1931d")
        ]
}
>
> db.students.find().pretty()
{
        "_id" : ObjectId("5f9a6e9a4fb378fc0ef1931b"),
        "Name" : "Aboli",
        "Roll no" : 31301,
        "Dept" : "Comp",
        "Score" : 50
}
{
        "_id" : ObjectId("5f9a6e9a4fb378fc0ef1931c"),
        "Name" : "Aditya",
        "Roll no" : 31302,
        "Dept" : "Comp",
        "Score" : 90
}
{
        "_id" : ObjectId("5f9a6e9a4fb378fc0ef1931d"),
        "Name" : "Tanuj",
        "Roll no" : 31303,
        "Dept" : "Comp",
        "Score" : 90
}
```



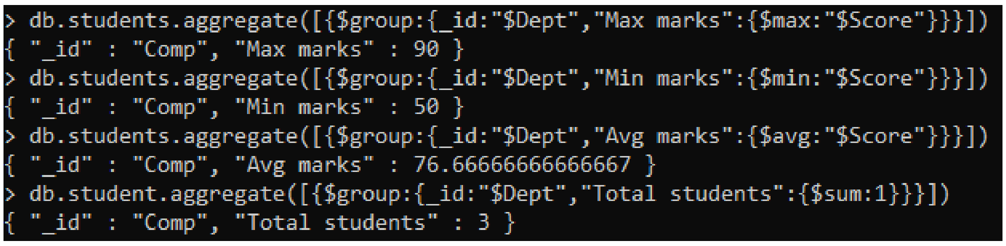
```
> db.students.ensureIndex({"Name":1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.students.find().pretty()
{
        "_id" : ObjectId("5f9a6e9a4fb378fc0ef1931b"),
        "Name" : "Aboli",
        "Roll no" : 31301,
        "Dept" : "Comp",
        "Score" : 50
}
{
        "_id" : ObjectId("5f9a6e9a4fb378fc0ef1931c"),
        "Name" : "Aditya",
        "Roll no" : 31302,
        "Dept" : "Comp",
        "Score" : 90
}
{
        "_id" : ObjectId("5f9a6e9a4fb378fc0ef1931d"),
        "Name" : "Tanuj",
        "Roll no" : 31303,
        "Dept" : "Comp",
        "Score" : 90
}
```



```
> db.students.ensureIndex({"Name":1,"Score":-1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
> db.students.find().pretty()
{
        "_id" : ObjectId("5f9a6e9a4fb378fc0ef1931b"),
        "Name" : "Aboli",
        "Roll no" : 31301,
        "Dept" : "Comp",
        "Score" : 50
}
{
        "_id" : ObjectId("5f9a6e9a4fb378fc0ef1931c"),
        "Name" : "Aditya",
        "Roll no" : 31302,
        "Dept" : "Comp",
        "Score" : 90
}
{
        "_id" : ObjectId("5f9a6e9a4fb378fc0ef1931d"),
        "Name" : "Tanuj",
        "Roll no" : 31303,
        "Dept" : "Comp",
        "Score" : 90
}
```



```
> db.students.aggregate([{$group:{_id:"$Dept","Total marks":{$sum:"$Score"}}}])
{ "_id" : "Comp", "Total marks" : 230 }
```



```
> db.students.aggregate([{$group:{_id:"$Dept","Max marks":{$max:"$Score"}}}])
{ "_id" : "Comp", "Max marks" : 90 }
> db.students.aggregate([{$group:{_id:"$Dept","Min marks":{$min:"$Score"}}}])
{ "_id" : "Comp", "Min marks" : 50 }
> db.students.aggregate([{$group:{_id:"$Dept","Avg marks":{$avg:"$Score"}}}])
{ "_id" : "Comp", "Avg marks" : 76.66666666666667 }
> db.student.aggregate([{$group:{_id:"$Dept","Total students":{$sum:1}}}])
{ "_id" : "Comp", "Total students" : 3 }
```