Rupesh Dharme
31124

Brice
Page :
Date : / /

Assignment 10

TIHe: mongoDB aggregation and indexing

Problem statement:
Design and develop mongoDB queries using aggrigation and indexing

Objectives:
Understand and implement aggregate and indexing in mongodb

Outcomes:
Implement aggregation and indexing.

Theory:
* mongoDB is an open source document db and a leading noSQL db.
* mongodb is written in c++

Aggregation:

(a) These equations process data records and returns computed results.
(b) aggregation operation group values from multiple documents together, can perform a variety of equation on grouped data.
(c) In SQL count(*) and with group by is an equivalent of mongodb aggregator.
(d) The aggregator() method for the aggregation in mongodb you should use-
aggregate() method

31124

Brice
Page :
Date : / /

Indexes

(a) support efficient resolutions and queries
(b) without it, mongoDB will scan every document of collection to select those documents match query statement.
(c) This scan is highly inefficiend and requires large volume of data processing
(d) indexes are a special data structures that store a small portion of dataset in a easy to traverse form.
(e) The index stores the value of a specific field a set of fields ordered by the value of field as specified in index.

db.collection-name.ensureIndex({key:value})

Conclusion:
We have successfully implemented and learnt aggregation, indexing in mongoDB.

Two collections: universities, courses

Select Command Prompt - mongo

```
> db.universities.find().pretty()
{
        "_id" : ObjectId("61920f32471f48c7ca9f58cf"),
        "country" : "Spain",
        "city" : "Salamanca",
        "name" : "USAL",
        "location" : {
                "type" : "Point",
                "coordinates" : [
                        -5.6722512,
                        17,
                        40.9607792
                ]
        },
        "students" : [
                {
                        "year" : 2014,
                        "number" : 24774
                },
                {
                        "year" : 2015,
                        "number" : 23166
                },
                {
                        "year" : 2016,
                        "number" : 21913
                },
                {
                        "year" : 2017,
                        "number" : 21715
                }
        ]
}
{
        "_id" : ObjectId("61920f32471f48c7ca9f58d0"),
        "country" : "Spain",
        "city" : "Salamanca",
        "name" : "UPSA",
        "location" : {
                "type" : "Point",
                "coordinates" : [
                        -5.6691191,
                        17,
                        40.9631732
                ]
        },
        "students" : [
                {
                        "year" : 2014,
                        "number" : 4788
```

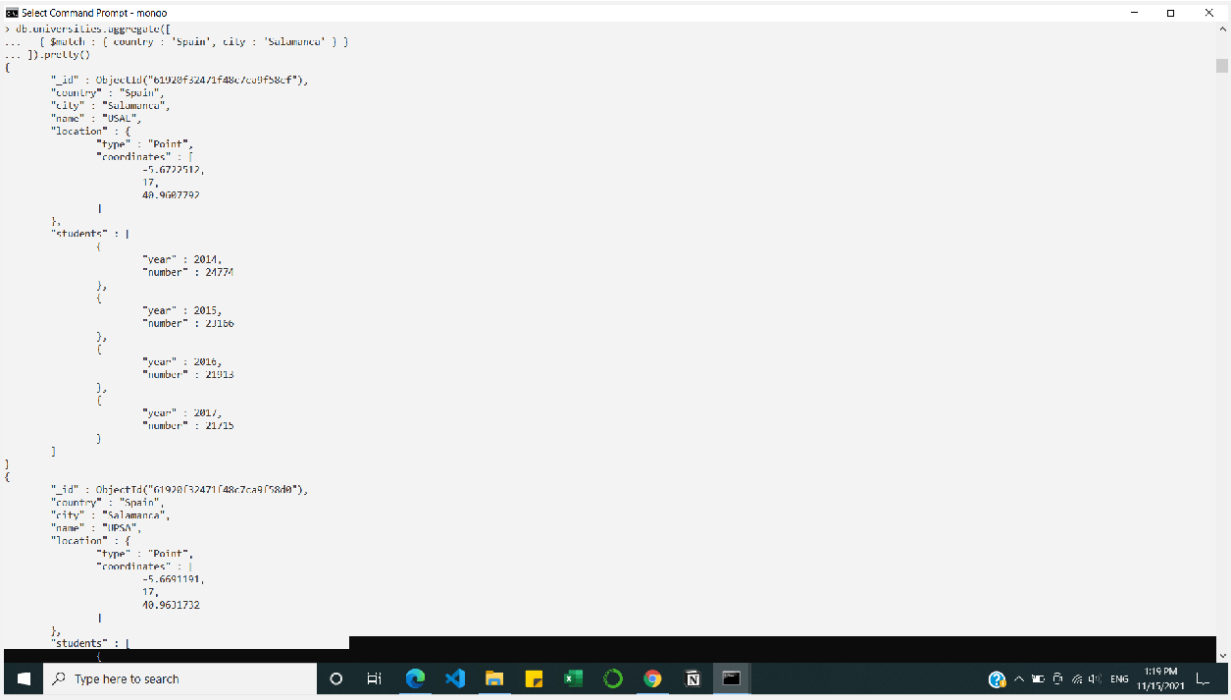
Select Command Prompt - mongo

```
> db.courses.find().pretty()
{
        "_id" : ObjectId("61920fe9471f48c7ca9f58d1"),
        "university" : "USAL",
        "name" : "Computer Science",
        "level" : "Excellent"
}
{
        "_id" : ObjectId("61920fe9471f48c7ca9f58d2"),
        "university" : "USAL",
        "name" : "Electronics",
        "level" : "Intermediate"
}
{
        "_id" : ObjectId("61920fe9471f48c7ca9f58d3"),
        "university" : "USAL",
        "name" : "Communication",
        "level" : "Excellent"
}
>
```

Work with those documents which specify that Spain as the value of the field country, and Salamanca is the value of the field city.

The `$match` stage allows us to choose just those documents from a collection that we want to work with. It does this by filtering out those that do not follow our requirements.

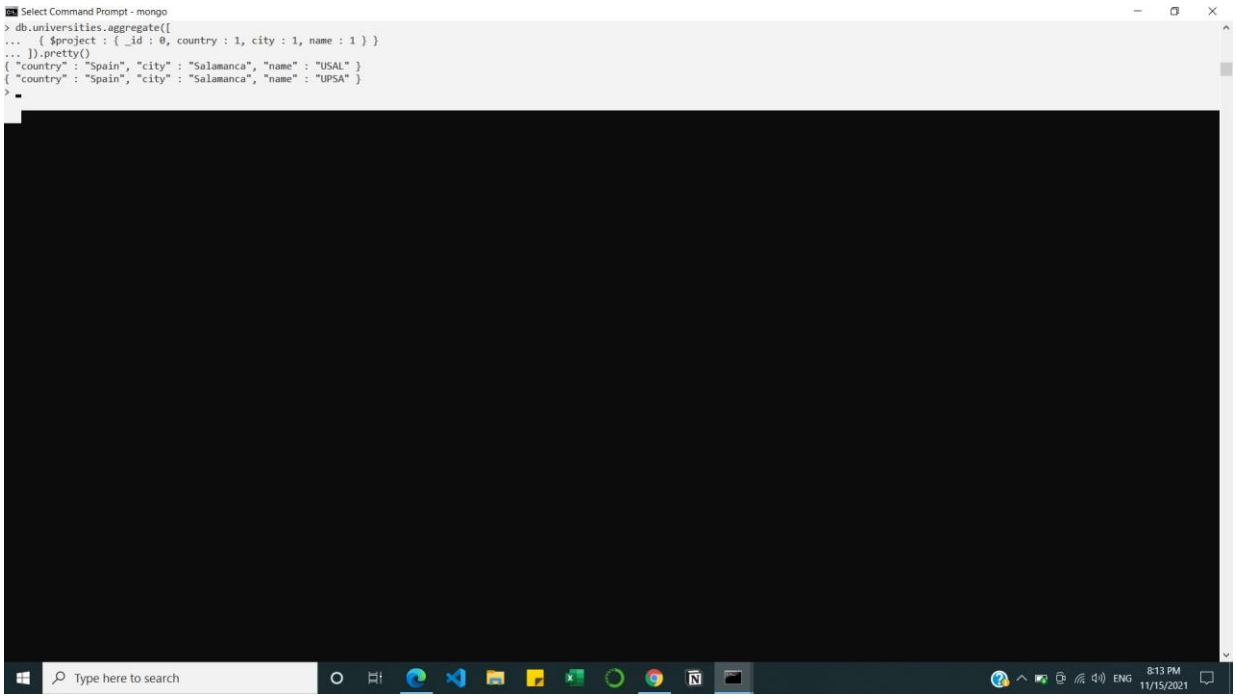
```
 db.universities.aggregate([
...  { $match : { country : 'Spain', city : 'Salamanca' } }
... ]).pretty()
```



The `$project()` stage is used to do this and to add any calculated fields that you need.

- We must explicitly write `_id : 0` when this field is not required
- Apart from the `_id` field, it is sufficient to specify only those fields we need to obtain as a result of the query

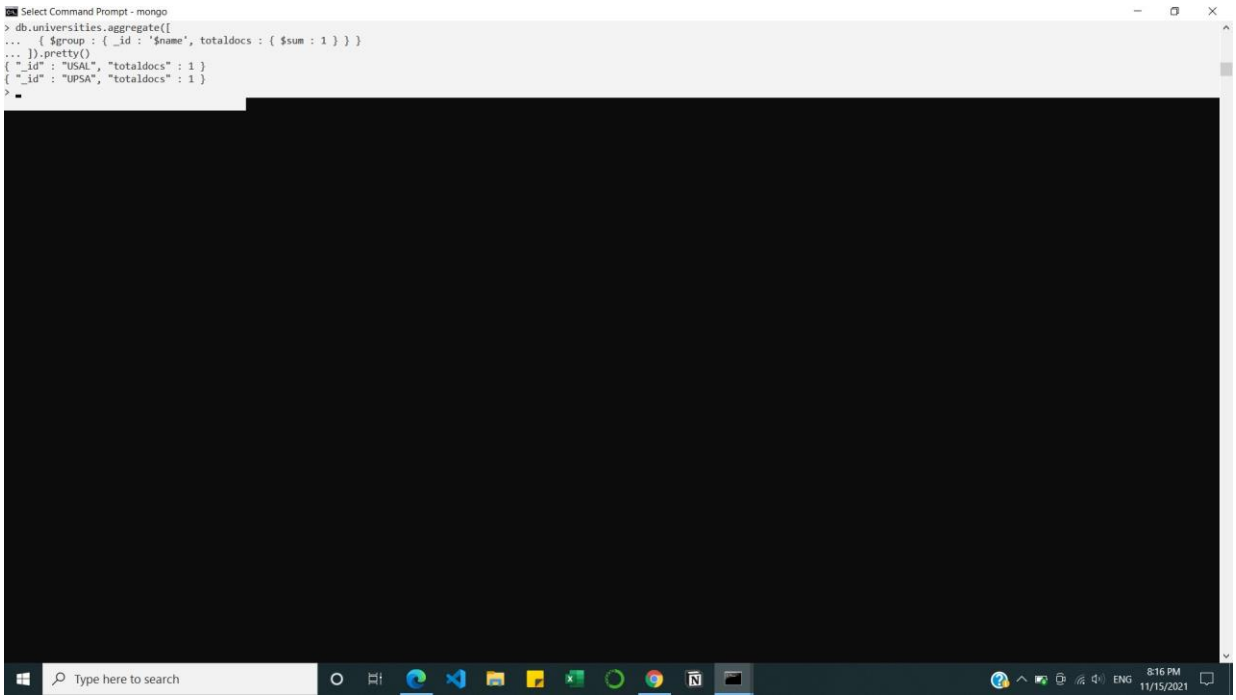
```
> db.universities.aggregate([
...  { $project : { _id : 0, country : 1, city : 1, name : 1 } }
... ]).pretty()
```



With the `$group()` stage, we can perform all the aggregation or summary queries that we need, such as finding counts, totals, averages or maximums.

We want to know the number of documents per university in our '`universities`' collection:

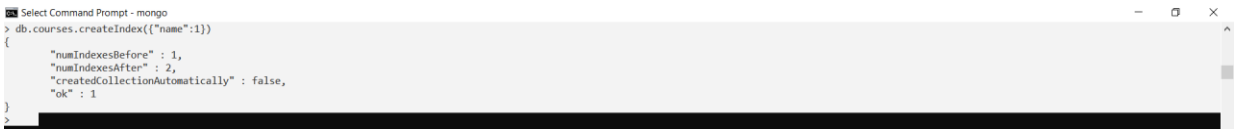
```
 db.universities.aggregate([
...  { $group : { _id : '$name', totaldocs : { $sum : 1 } } }
... ]).pretty()
```



Indexes

Create indexes

```
 db.courses.createIndex({"name":1}) #1 for ascending , -1 for descending
```



Get Indexes

Select Command Prompt - mongo

```
}
> db.courses.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_"
        },
        {
                "v" : 2,
                "key" : {
                        "name" : 1
                },
                "name" : "name_1"
        }
]
>
```

Drop index

Select Command Prompt - mongo

```
> db.courses.dropIndexes({"name":1})
{ "nIndexesWas" : 2, "ok" : 1 }
>
```