Aggregation Commands

Aggregation functions perform operations on groups of documents and return the computed result. Aggregation operation mainly groups the data/ values from the multiple documents and perform various operations on the grouped data and returns a single or multiple results.

MongoDB performs aggregate operations in one of the following three ways:

- Pipeline.
- Map Reduce
- Single-purpose aggregate methods and commands.

We have already studied about these. We will emphasis once again on the aggregation commands and methods.

Single-purpose aggregate commands

These are used for specific aggregation operations on sets of data. Single-purpose aggregate commands and methods are less complex but with limited scope compared to pipeline and map reduce operations. Single-purpose aggregate commands provide straightforward semantics for common data processing options. These are

- db.collection.count()
- db.collection.distinct()
- db.collection.estimatedDocumentCount()

All of these operations aggregate documents from a single collection. These operations provide simple access to common aggregation processes, but not as flexible and capable like aggregation pipeline and map-reduce.

a. db.collection.count() i.e. count

The Count operation returns the count of documents from the collection. It takes a number of

documents and depending on the match query returns the count of the documents. It is different from find() as find returns the documents while it returns the number of documents matching the query.

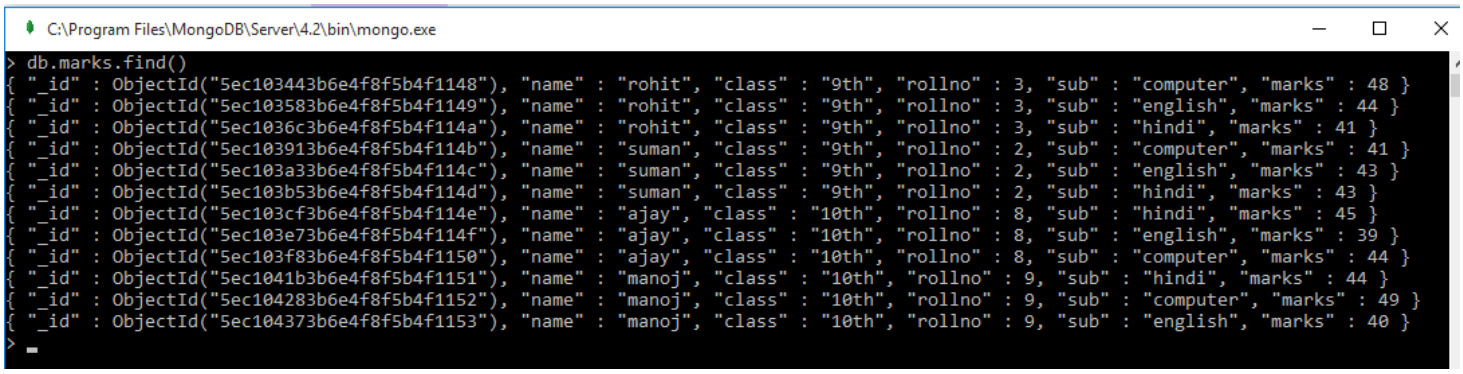
Syntax

db.collectionName.count(query)

where, **Query –** The selection Criteria and it is of document type

Example:

Lets Consider the marks collection having following documents:



1. To find no of documents where the **marks** is greater than 45. The command will be

db.marks.count({"marks":{$gt:45}})

Output

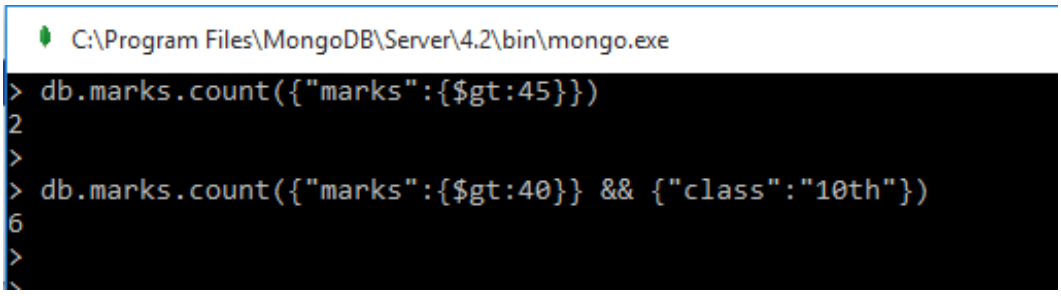
2

2. To find no of documents where the **marks** is greater than **40** and **class** is **10th**. The command will be

db.marks.count({"marks":{$gt:40}} && {"class":"10th"})

Output

6



b. db.collection.estimatedDocumentCount()

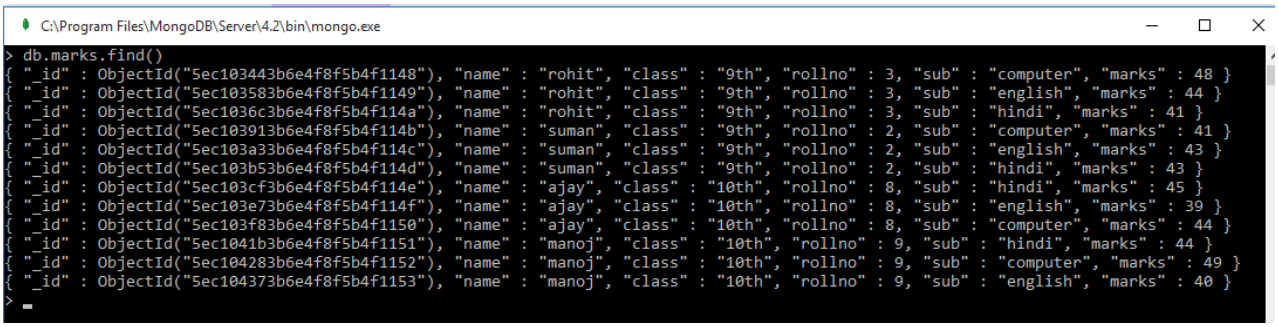
db.collection.estimatedDocumentCount() method returns the count of all documents in a collection or view.

Syntax

db.collectionName.estimatedDocumentCount()

It does not take a query filter and instead uses metadata to return the count for a collection.

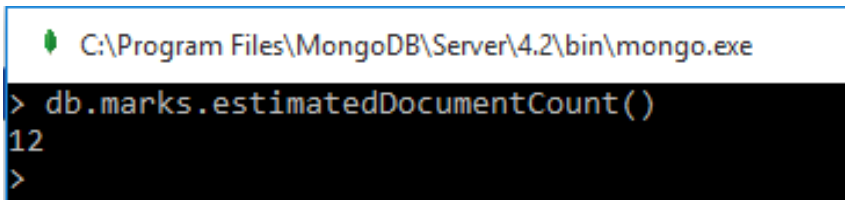
Example:
Lets Consider the marks collection having following documents:



To find no of documents in the marks collection, the command will be

db.marks.estimatedDocumentCount()

Output



c. db.collection.distinct()

The **db.collection.distinct() command** finds the distinct values for a specified field across a single collection or view and returns the results in an array. It takes a document and depending on the match query returns the unique values for a field.

Syntax

db.collectionName.distinct(field, query)

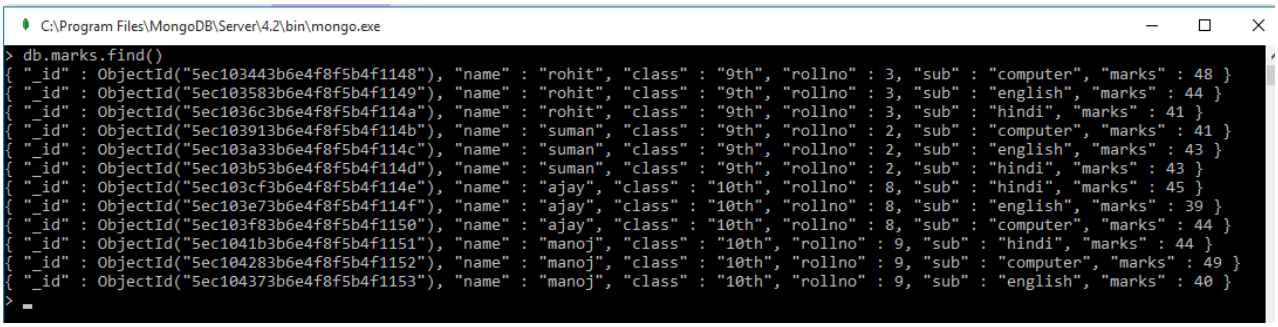
where,
Field - The field for which to return distinct values.

Query – specifies the documents from which to retrieve the distinct values. It is optional.

For Array Fields : If the value of the specified field is an array then the db.collectionName.distinct() considers each element of the array as a separate value.

For Embedded Documents: We may find distinct values from embedded documents also by specifying the specific field e.g. stock.color

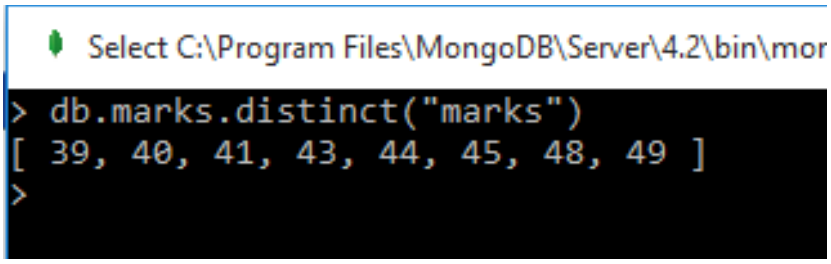
Example:
Lets consider the **marks** collection, having following documents



1. To find the distinct marks among the various documents from the collection, the command is

   db.marks.distinct("marks")

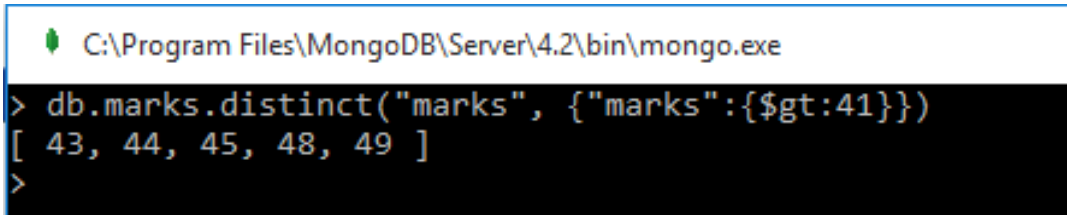
   Output



2. To find the distinct marks greater than 41 among the various documents from the collection, the command is

   db.marks.distinct("marks", {"marks":{$gt:41}})

Output



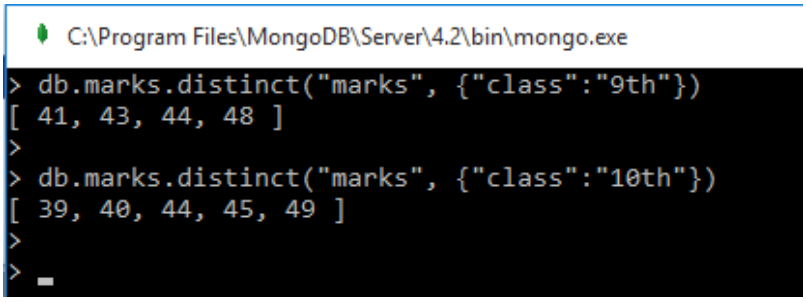
3. To find the distinct marks for the students of the **class 9th** among the various documents matching the query from the collection, the command is

   db.marks.distinct("marks", {"class":"9th"})

   and for class 10th, it will be

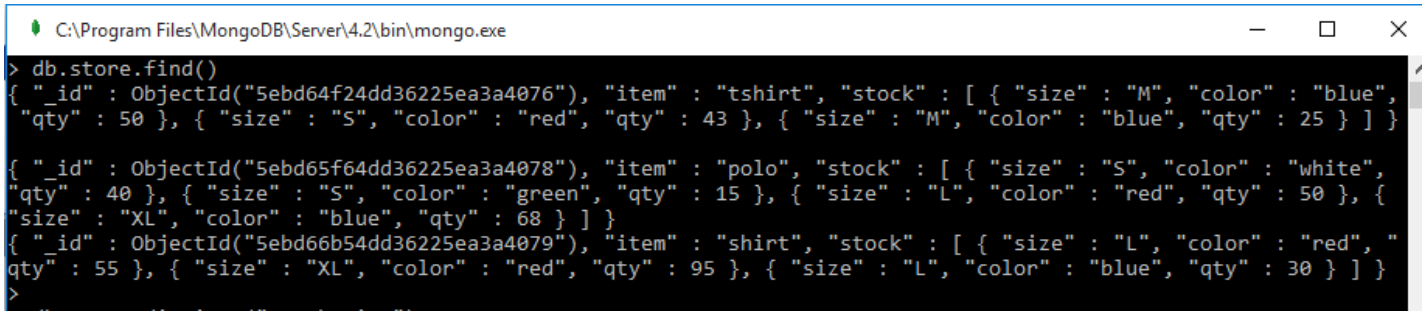
   db.marks.distinct("marks", {"class":"9th"})

Output



Example: To find distinct values in case of Embedded documents

Lets consider the store collection having stock as embedded documents.



1. To find the distinct values of sizes from the stock, the command will be **db.store.distinct("stock.size")**

   or
   to find the distinct color from the stock, the command will be

db.store.distinct("stock.color")

Output

```
>
> db.store.distinct("stock.size")
[ "L", "M", "S", "XL" ]
>
> db.store.distinct("stock.color")
[ "blue", "green", "red", "white" ]
>
```

2. To find the distinct color from the stock only if the quantity in stock is more than 70, the command will be

db.store.distinct("stock.color", {"stock.qty":{$gt:70}})

```
>
>
> db.store.distinct("stock.color", {"stock.qty":{$gt:70}})
[ "blue", "red" ]
>
>
```

Example: To find values in array fields

Lets consider, the **result** collection having marks as an array field.

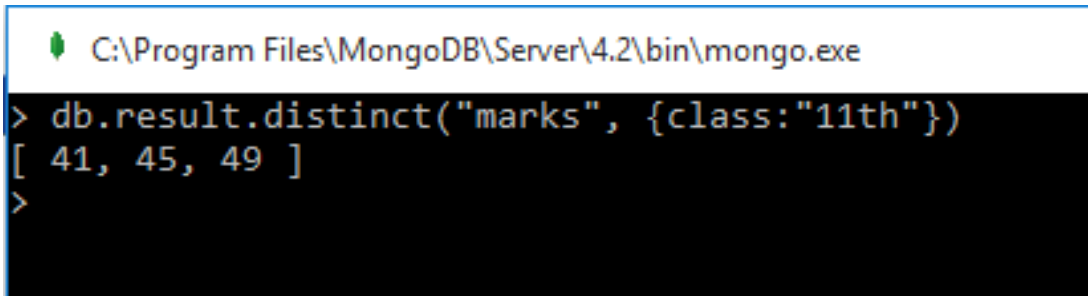
```
C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
> db.result.find()
{ "_id" : ObjectId("5ebced72c8f756b82a99743e"), "name" : "indu", "class" : "10th", "marks" : [ 45, 48, 43 ] }
{ "_id" : ObjectId("5ebced91c8f756b82a99743f"), "name" : "amit", "class" : "10th", "marks" : [ 44, 46, 42 ] }
{ "_id" : ObjectId("5ebceda6c8f756b82a997440"), "name" : "geetu", "class" : "10th", "marks" : [ 43, 46, 50 ] }
{ "_id" : ObjectId("5ebcf231c8f756b82a99744a"), "name" : "rohit", "class" : "11th", "marks" : [ 41, 45, 49 ] }
>
```

1. To find all distinct marks, the command will be
db.result.distinct("marks")

```
>
> db.result.distinct("marks")
[ 41, 42, 43, 44, 45, 46, 48, 49, 50 ]
>
```

2. To find all distinct marks of Class 11th, the command will be

   db.result.distinct("marks", {class:"11th"})



Assignment

1. What are aggregation commands?
2. How to use distinct command for Array field and embedded document?