Db.persons.aggregate([{$match:{“gender”:”male”}}, {$group:{\_id:”$age”, count:{$sum:1}}}])

.aggregate([{},{},{}])

-Each one of these objects is going to represent a “stage”

* $match -> filters by a certain condition
* $group->groups the documents based on a certain criteria
* $project-> filters the fields in the document
* $sort->sorts in descending or ascending order, or possibly a custom sort
* $count->counts all the documents
* $limit->limits the documents returned by your query
* $skip->skips a certain amount or certain number of returned documents
* $out->Writes the output of the aggregation framework or the returned documents of the queries to another collection

-Going into each stage, an operation will be performed on the documents, and then it will create a new subset of documents based on the outputs.

-match: produces a subset of the passed documents based on a given criteria

{$match:{<query>}}

{$match:{“age”:{$gt:25}}}

-group: takes the subset of documents passed by the 1st stage, and then creates new documents, based on a certain format/syntax/condition

{$group:{\_id:<expression>, <field1>: {<accumulator1>:<expression1>},<field2>:{<accumulator2>:<expression2>},…}}

\_id= required

-Field=whatever you want to name the new field

-Accumulator=function on how to process or what to do with the data

-Expression=what to do inside the accumulator

{$group:{\_id:”$age”, count:{$sum:1}}}])

Unwind

{

Name:Patrick,

Age:24,

Interests:[ Playing, Listening to Music, Working],

}

Unwind(

{

Name:Patrick,

Age:24,

Interests:[ Playing, Listening to Music, Working],

}

)

{

Name:Patrick,

Age:24,

Interests:Playing

//1

}

{

Name:Patrick,

Age:24,

Interests:Listening to Music

//2

}

{

Name:Patrick,

Age:24,

Interests:Working

//3

}

Using a MongoDB expression as a value for the field $sum:

db.persons.aggregate([{$match:{"age":{$gt:25}}},{$group:{\_id:"$age", total:{$sum:{$add:["$age",5]}},count:{$sum:1}}}])

Using unwind to separate the array values of a document:

db.persons.aggregate([{$match:{"name":"Aurelia Gonzales"}},{$unwind:"$tags"}]).pretty()

Sorting the documents returned by the 2nd stage:

db.persons.aggregate([{$match:{"age":{$gt:25}}},{$group:{\_id:"$age",count:{$sum:1}}},{$sort:{\_id:1}}])

Reversing group and match:

db.persons.aggregate([{$group:{\_id:"$age",count:{$sum:1}}},{$match:{"count":{$gt:5}}}])

1. Query to find all of the documents where isActive:false;

db.persons.aggregate([{$match:{"isActive":false}}]).pretty()



1. Query to find all of the documents where isActive:true;

db.persons.aggregate([{$match:{"isActive":true}}]).pretty()



1. Query to find number of documents where isActive:true;

db.persons.aggregate([{$match:{"isActive":true}},{$group:{\_id:"",count:{$sum:1}}}])



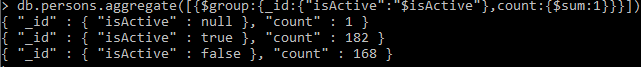
1. Query to find number of documents where isActive:false;

db.persons.aggregate([{$match:{"isActive":false}},{$group:{\_id:"",count:{$sum:1}}}])



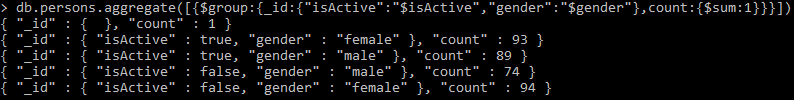
1. Query to create new documents showing the aggregate data of the isActive field

db.persons.aggregate([{$group:{\_id:{"isActive":"$isActive"},count:{$sum:1}}}])



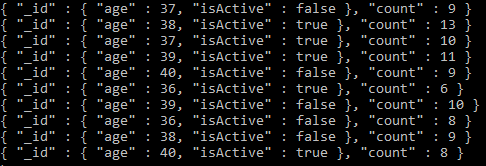
1. Query to create new documents showing the aggregate data of the isActive field and gender field.

db.persons.aggregate([{$group:{\_id:{"isActive":"$isActive","gender":"$gender"},count:{$sum:1}}}])



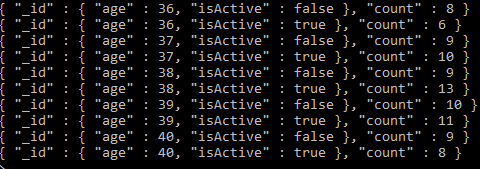
1. Query to create a new documents showing the aggregate data of the isActive field and the age field, with the exception of anyone below the age of 35.

db.persons.aggregate([{$match:{"age":{$gt:35}}},{$group:{\_id:{"age":"$age","isActive":"$isActive"},count:{$sum:1}}}])



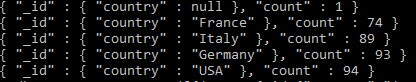
1. Sort the documents created in #7 in Ascending Order.

db.persons.aggregate([{$match:{"age":{$gt:35}}},{$group:{\_id:{"age":"$age","isActive":"$isActive"},count:{$sum:1}}},{$sort:{\_id:1}}])



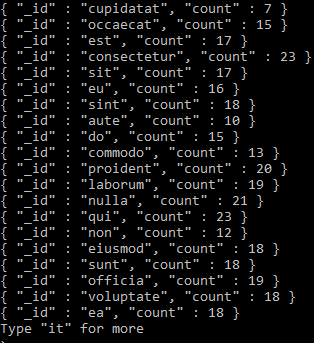
1. Query to create new documents showing the aggregate data of the number of companies found in a specific country. E.g USA: 5, Italy:10 , France: 15, etc.

db.persons.aggregate([{$group:{\_id:{"country":"$company.location.country"},count:{$sum:1}}}])



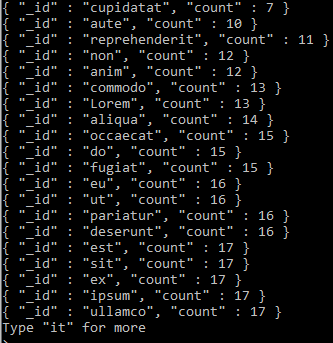
1. Query to create new documents showing the aggregate data of the number of documents associated to each specific tag. E.g ex:5, magna:10, enim: 10;

db.persons.aggregate([{$unwind:"$tags"},{$group:{\_id:"$tags",count:{$sum:1}}}])



1. Query to sort the output of #10 in Descending Order

db.persons.aggregate([{$unwind:"$tags"},{$group:{\_id:"$tags",count:{$sum:1}}},{$sort:{count:1}}])



1. Query to create 2 new documents showing the average age of each gender.

e.g

{\_id:female average:43};

{\_id:male average:44};

db.persons.aggregate([{$group:{\_id:"$gender", average:{$avg:"$age"}}}])

