**Aggregation**

* **Aggregation** – este o analiza sau sumar a datelor.
* **Stage** – este o operatie concreta de agregare efectuata asupra datelor
* **Aggregation pipeline** – o colectie de stages(operatii) care proceseaza, transforma si returneaza rezultate
* **Aggregation pipeline este pur si simplu o colectie de queries rulate ca un tot intreg si gata, pe aceleasi date ce se trimit de la o operatie la alta si gata**
* Am putea rula si cate un stage rand pe rand, scriind comenzi de mana, dar mongod ne ofera posibilitatea de a le scrie impreuna, intr-un pipeline, si sa fie mereu rulate impreuna, in ordinea potrivita, ca un tot intreg.
* Datele pot fi :

- filtered

- sorted

- grouped

- transformed

* Documentele care sunt output de la un stage devin apoi input pentru urmatorul stage

**Stage**

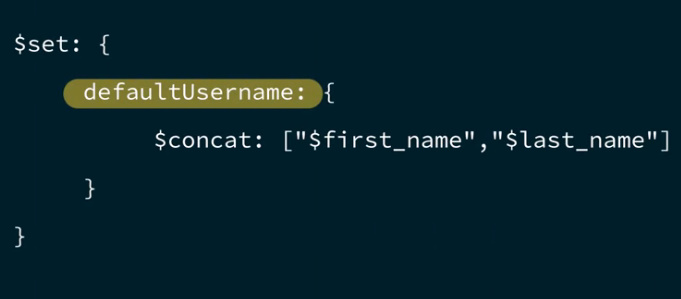
* **Stage** este o simpla operatie efectuata asupra la date si gata
* Stages:

- **$match** – filtru pentru date, si filtreaza datele ce corespund undei conditii

- **$group** – grupeaza documentele pe baza la un criteriu

- **$sort** – pune documentele intr-o anumita ordine

* Vedem ca in unele dintre ele, $ e folosit pentru a face referinta la un field din document:



Aici cream un nou field, defaultUsername, ce e creat prin concatenarea la “first\_name” si “last\_name”, si vedem ca facem referinta la ele prin $, ca de altfel “first\_name” ar fi simplu string, dar $ spune ca e field, nu string!

* Crearea la un aggreation pipeline se face prin **db.collection.aggregate([{},{},...])**

**$match**

* Este pentru a filtra datele. Este exact ca find()
* Odata de datele sunt filtrate, ele trec la umatorul stage(operatie)
* E bine sa punem $match ca primul argument, deoarece el reduce numarul de documente ce urmeaza sa fie procesate de urmatoarele operatii

db.restaurants.aggregate([{$match: {borough: "Brooklyn"}}])

**$group**

* Grupeaza documentele intr-un singur document dupa un group key
* Returneaza un document pentru fiecare valoare unica din group key

$group {

\_id : “$Field”,

newField : {<accumulator> : <expression>}

}

\_id – se refera la field dupa care se va face gruparea. Folosim $ pentru a face referinta la ace lfield

newField – orice nume putem sa-i dam. Va fi noul field ce va arata informatia ce o vrem sa se faca dupa grupare

accumulator – este o expresie ce spune cum sa agregam datele, cum e SUM() in SQL

expression – expresie pentru a arata gen asupra la ce coloana facem $sum, $min etc.

db.restaurants.aggregate([

{

$match : { }

},

{

$group : {\_

id : "$cuisine",

Total : {$count : {} }

}

}

])

* Group accumulator operations:

$count

$sum

$avg

$min

$max

$first – adauga o valoare din primul document pentru fiecare grup

$last

db.companies.aggregate([

{ $match: {} },

{ $group:

{ \_id: "$category\_code",

total: { $sum: "$number\_of\_employees" } }

}

] )

Vedem ca grupam dupa category code si functia de agregare e suma pe baza la field number\_of\_employees

**Punerea lui $match : {} pentru a returna toate documents e optionala!**

**$sort and $limit stages**

db.companies.aggregate([{$sort : {number\_of\_employees : -1}}, {$limit : 2}])

Mare grija la ordine!Daca punem intai limit si apoi sort, vom primi primele 2 documente luate random, dar sortate intre ele 2

**$project $count $set**

* $project – este pentru projection, adica pentru a selecta ce fields sa fie returnate sau care nu. E exact ca la find() cand punem al 2 argument. E bine sa fie ultimul



Putem si adauga vreun field nou daca vrem

Putem folosi si un field deja existent prin $ daca vrem pur si simplu sa-i modificam numele dar sa avem aceeasi valoare

db.trips.aggregate([{

$project : {

tripduration: 1,

usertype:1,

**bike** : "$bikeid"}

}

])

bike nu exista ca field, exista bikeid, dar il redenumim

* $set – permite sa modificam sau sa adugam fields. E folositor cand vrem sa modficam fields sau sa mai adaugam altele pentru urmatoarele stages

db.trips.aggregate([{$set : {

tripduration: {

$round : {

$multiply : ["$tripduration",60.0]}

}

}

}

])

asa, vom modifica field tripduration

* $round : – aproximeaza un numar
* $multiply : [] – inmulteste 2 numere
* $concat : [] – concateneaza
* $count : “NumeFiledPentruTotal” – este exact ca countDocuments(). Numele pus intre “” e doar numele la field returnat ce va arata total la count

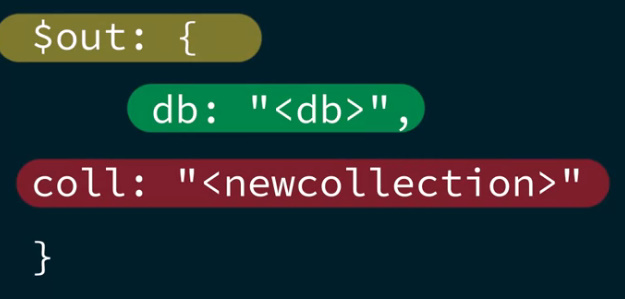
db.trips.aggregate([{$count : "Totals"}])

si primim

[ { Totals: 10000 } ]

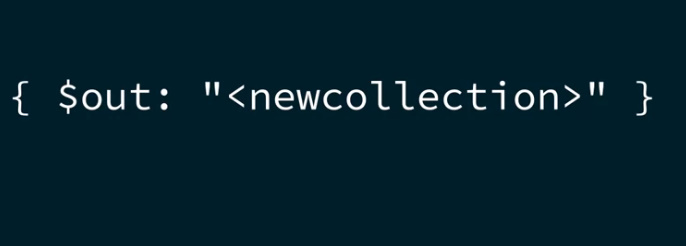
**$out**

* $out scrie documentele returnate de un aggregation pipeline intr-o noua collection
* **Trebuie sa fie mereu ultima stage in pipelin!!!**
* Creaza o colectie daca nu exista, **dar daca exista, atentie,va inlocui toate datele cu cele returnate!!!!**
* Creaza si o db daca cea scrisa nu exista



db – numele la baza de date unde sa se creeze colectia

coll – numele la colectie



Fara db field, colectia se va pune in db actuala

**Arithemtic operations**

| **Name** | **Description** |
| --- | --- |
| [$add](https://www.mongodb.com/docs/v2.4/reference/operator/aggregation/add/#exp._S_add) | Computes the sum of an array of numbers. |
| [$divide](https://www.mongodb.com/docs/v2.4/reference/operator/aggregation/divide/#exp._S_divide) | Takes two numbers and divides the first number by the second. |
| [$mod](https://www.mongodb.com/docs/v2.4/reference/operator/aggregation/mod/#exp._S_mod) | Takes two numbers and calculates the modulo of the first number divided by the second. |
| [$multiply](https://www.mongodb.com/docs/v2.4/reference/operator/aggregation/multiply/#exp._S_multiply) | Computes the product of an array of numbers. |
| [$subtract](https://www.mongodb.com/docs/v2.4/reference/operator/aggregation/subtract/#exp._S_subtract) | Takes two numbers and subtracts the second number from the first. |