TP3 – NOSQL MONGODB RESTAURANT INSPECTIONS

*Manon GARDIN*

*Matias OTTENSEN*

*Alexandre GARNIER*

*Tiphaine KACHKACHI*

Create the Database

After running our container for MongoDb we can connect ourselves to mongodbCompass (or in the mongodb CLI)

Une image contenant texte, Police, capture d’écran, logo

Description générée automatiquement

Une image contenant texte, capture d’écran, Police, logo

Description générée automatiquement

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement  
Une image contenant texte, capture d’écran, ligne, Police

Description générée automatiquement

If we were to do it using the Cli of mongodb we could do

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

To connect ourselves to the database

# Import the data

To import our data we can directly use the button :

# Une image contenant texte, capture d’écran, Police, nombre Description générée automatiquement

All objects were added to our database !

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Queries

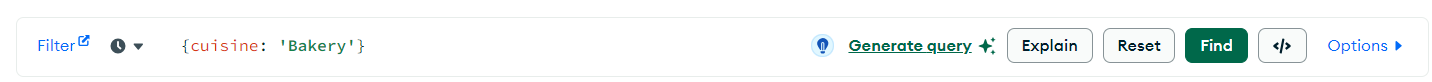
Since our dataset is a difficult one, we are going to do 8 easy queries, 1 complex and 1 hard.

Easy Queries

## Query 1 : To find all Bakery restaurants

In the filter section :

{cuisine: 'Bakery'}



Results :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

We got 691 results : 

## Query 2 : Find all restaurants that are located on Park Avenue in Brooklyn

Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

{

"address.street": "Park Avenue",

"borough": "Brooklyn"

}

Results :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

We had few results :

Une image contenant Police, Graphique, texte, blanc

Description générée automatiquement

## Query 3 : To find all restaurants who have been graded at least 6 times

To do so we can do an aggregation :

[{

$match: {

$expr: { $gte: [{ $size: "$grades" }, 6] }

}

}]

The $match is trying to find to find items who will be having the following characteristics.

$size returns the number of elements in an array. Here we look for the numbers of grades in ‘grades’ and we want them to be equal of higher than 6. We do that with $gte which stands for ‘greater than or equal’ operator.

Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement

Results (sample displayed) :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

## Query 4 : List all types of Restaurants and how many of them are they, descending order (aggregation)

[

{

$group: {

\_id: "$cuisine",

count: { $sum: 1 }

}

},

{

$sort: { count: -1 }

}

]

Une image contenant texte, Police, capture d’écran, écriture manuscrite

Description générée automatiquement

$group to group the results

We count the numbers of id of cuisine.

$sort : is where we choose the order of display (descending -> -1)

Results (not everything is on the screen) :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

## Query 5 : Find all restaurants that were graded at least once in 2012

{

"grades": {

$elemMatch: {

"date": {

$gte: new Date("2012-01-01"),

$lt: new Date("2013-01-01")

}

}

}

}

Une image contenant texte, Police, ligne, nombre

Description générée automatiquement

$elemMatch : look for at least one element where …

$gte : greater than and equal to

$lt : less than

Results :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

We have a lot of restaurants with a grade from 2012 !

Une image contenant texte, Police, blanc, conception

Description générée automatiquement

## Query 6 : Count how many distinct streets there are within a borough. (aggregation)

[

{

$group: {

\_id: {

borough: "$borough",

street: "$address.street"

}

}

},

{

$group: {

\_id: "$\_id.borough",

count: { $sum: 1 }

}

},

{

$sort: { count: -1 }

}

]

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

Results :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

## Query 7 : The restaurant who has the oldest grade (aggregation)

[

{

$match: {

"grades.date": { $exists: true, $ne: null }

}

},

{

$addFields: {

oldestGrade: { $min: "$grades.date" }

}

},

{

$sort: {

oldestGrade: 1

}

},

{

$limit: 1

}

]

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

We only want the ‘older’ restaurant, so we limit our results to one ($limit)

Result :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

From 2010 !

## Query 8 : The street (and its borough) that holds the most restaurants. (aggregation)

[

{

$group: {

\_id: { borough: "$borough", street: "$address.street" },

boroughName: { $first: "$borough" },

streetName: { $first: "$address.street" },

count\_of\_restaurants: { $sum: 1 }

}

},

{

$sort: { count\_of\_restaurants: -1 }

},

{

$limit: 1

}

]

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Result :

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Complex Query

## Query : List of restaurants that had a good first review and a ‘bad’ last one

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

[

{

$unwind: "$grades"

},

{

$group: {

\_id: "$\_id",

name: { $first: "$name" },

initialGrade: { $first: "$grades.grade" },

initialDate: { $min: "$grades.date" },

latestGrade: { $last: "$grades.grade" },

latestDate: { $max: "$grades.date" }

}

},

{

$match: {

$expr: {

$and: [

{ $eq: ["$initialGrade", "A"] },

{ $in: ["$latestGrade", ["B", "C"]] }

]

}

}

}

]

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Hard Query

## Query : The name and number of grades attributed to restaurants, order by the worst graded restaurants ever (aggregation) !

[{

$unwind: "$grades"

},

{

$group: {

\_id: "$\_id",

name: { $first: "$name" },

numA: { $sum: { $cond: [{ $eq: ["$grades.grade", "A"] }, 1, 0] } },

numB: { $sum: { $cond: [{ $eq: ["$grades.grade", "B"] }, 1, 0] } },

numC: { $sum: { $cond: [{ $eq: ["$grades.grade", "C"] }, 1, 0] } }

}

},

{

$sort: {

numC: -1,

numB: -1,

numA: -1

}

}

]

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Results :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

How is Chopstick still in business ?