
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)

The screenshot shows the Docker Desktop interface. At the top, a container named 'mongo' with ID 'e6119e33' is running on the 'mongo:late' image. Below this, a detailed view of the 'mongo' container shows its ID as 'e6119e3355ff' and its ports as '27017:27017'. The main part of the image is the 'New Connection' dialog in MongoDB Compass. It has a title 'New Connection' and a subtitle 'Connect to a MongoDB deployment'. On the right, there is a 'FAVORITE' toggle. The 'URI' field is set to 'mongodb://localhost:27017/'. Below the URI field is a link to 'Advanced Connection Options'. At the bottom, there are three buttons: 'Save', 'Save & Connect', and 'Connect'.

mongo
e6119e33 [mongo:late](#) Running

mongo
[mongo:latest](#)
e6119e3355ff
[27017:27017](#)

New Connection
Connect to a MongoDB deployment

URI ⓘ Edit Connection String ☒

mongodb://localhost:27017/

➤ Advanced Connection Options

Save Save & Connect Connect

Database Name

Restaurants

Collection Name

Restaurant_Inspections

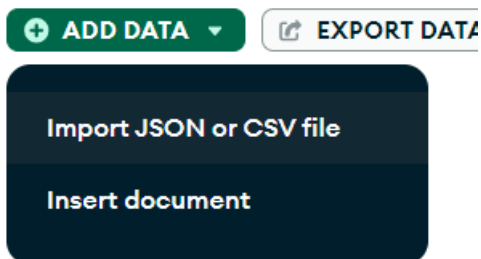
If we were to do it using the Cli of mongodb we could do

```
mongosh mongodb://127.0.0.1:27017
Please enter a MongoDB connection string (Default: mongodb://localhost/): 27017
27017
Current Mongosh Log ID: 65c81baafdd9ccf499c9e8be
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=30000
Using MongoDB:      7.0.5
Using Mongosh:      2.1.4
```

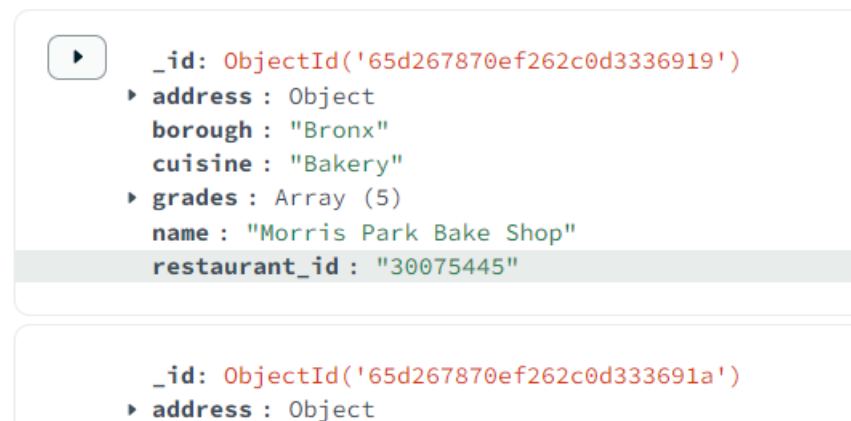
To connect ourselves to the database

Import the data

To import our data we can directly use the button :



All objects were added to our database !



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'}

[Filter](#)  {cuisine: 'Bakery'}

 [Generate query](#)  [Explain](#) [Reset](#) [Find](#)  [Options](#) 

Results :

```
_id: ObjectId('65d267870ef262c0d3336919')
  address: Object
    borough: "Bronx"
    cuisine: "Bakery"
  grades: Array (5)
  name: "Morris Park Bake Shop"
  restaurant_id: "30075445"
```




```
_id: ObjectId('65d267870ef262c0d333693b')
  address: Object
    borough: "Manhattan"
    cuisine: "Bakery"
  grades: Array (5)
  name: "Olive'S"
  restaurant_id: "40363151"
```

```
_id: ObjectId('65d267870ef262c0d3336a7a')
  address: Object
    borough: "Brooklyn"
    cuisine: "Bakery"
```

We got 691 results : 121 – 140 of 691

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

[Filter](#)  {

"address.street": "Park Avenue",
"borough": "Brooklyn"

}

[Filter](#)  {"address.street": "Park Avenue" }

```
{
  "address.street": "Park Avenue",
  "borough": "Brooklyn"
}
```

Results :

```
  _id: ObjectId('65d267870ef262c0d3336c52')
  address: Object
    building: "537"
    coord: Object
      street: "Park Avenue"
      zipcode: "11205"
    borough: "Brooklyn"
    cuisine: "Spanish"
  grades: Array (5)
  name: "Charle'S Corner Restaurant & Deli"
  restaurant_id: "40392285"
```

```
▶ _id: ObjectId('65d267880ef262c0d3338170')
  address: Object
    borough: "Brooklyn"
    cuisine: "American "
```

We had few results :

1 - 4 of 4 ↻

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.

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

Results (sample displayed) :

PIPELINE OUTPUT

Sample of 10 documents

```
_id: ObjectId('65d267870ef262c0d3336920')
  address: Object
    borough: "Brooklyn"
    cuisine: "Delicatessen"
  grades: Array (6)
    name: "Wilken'S Fine Food"
    restaurant_id: "40356483"
```

```
_id: ObjectId('65d267870ef262c0d3336930')
  address: Object
    borough: "Manhattan"
    cuisine: "Chicken"
  grades: Array (6)
    name: "Harriet'S Kitchen"
    restaurant_id: "40362098"
```

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 }
  }
]
```

```
1 ▼ [
2 ▼   {
3 ▼     $group: {
4       _id: "$cuisine",
5       count: { $sum: 1 }
6     }
7   },
8 ▼   {
9     $sort: { count: -1 }
10  }
11 ]
12 |
```

\$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) :

PIPELINE OUTPUT

Sample of 20 documents

```
_id: "American "  
count : 6181
```

```
_id: "Chinese"  
count : 2418
```

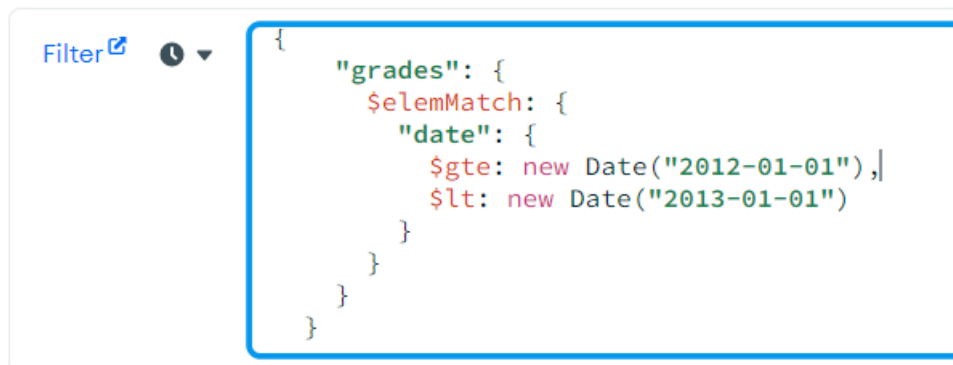
```
_id: "Café/Coffee/Tea"  
count : 1214
```

```
_id: "Pizza"  
count : 1163
```

```
_id: "Italian"  
count : 1069
```

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")  
      }  
    }  
  }  
}
```



\$elemMatch : look for at least one element where ...

\$gte : greater than and equal to

\$lt : less than

Results :

```
▼ 3: Object
  date : 2012-05-08T00:00:00.000+00:00
  grade : "A"
  score : 12
  name : "Wendy'S"
  restaurant_id : "30112340"

_id: ObjectId('65d267870ef262c0d333691b')
▶ address : Object
  borough : "Manhattan"
  cuisine : "Irish"
▼ grades : Array (4)
  ▶ 0: Object
  ▶ 1: Object
  ▼ 2: Object
    date : 2012-07-31T00:00:00.000+00:00
    grade : "A"
```

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

1 – 20 of 15825

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 }
  }
]
```

```

1  ▼ [
2  ▼  {
3  ▼    $group: {
4  ▼      _id: {
5      borough: "$borough",
6      street: "$address.street"
7      }
8    }
9  },
10 ▼  {
11 ▼    $group: {
12      _id: "$_id.borough",
13      count: { $sum: 1 }
14    }
15  },
16 ▼  {
17    $sort: { count: -1 }
18  }
19 ]
20

```

Results :

```

_id: "Manhattan"
count : 1035

```

```

_id: "Brooklyn"
count : 707

```

```

_id: "Queens"
count : 645

```

```

_id: "Bronx"
count : 455

```

```

_id: "Staten Island"
count : 176

```

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

```

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

```

8


```

    $addFields: {
      oldestGrade: { $min: "$grades.date" }
    }
  },
  {
    $sort: {
      oldestGrade: 1
    }
  },
  {
    $limit: 1
  }
]

```

```

1  ▾ [
2  ▾ {
3  ▾   $match: {
4     "grades.date": { $exists: true, $ne: null }
5     }
6  },
7  ▾ {
8  ▾   $addFields: {
9     oldestGrade: { $min: "$grades.date" }
10    }
11  },
12 ▾ {
13 ▾   $sort: {
14    oldestGrade: 1
15    }
16  },
17 ▾ {
18    $limit: 1
19  }
20 ]
21

```

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

Result :

```

_id: ObjectId('65d267880ef262c0d3338237')
address: Object
  borough: "Manhattan"
  cuisine: "Latin (Cuban, Dominican, Puerto Rican, South & Central American)"
grades: Array (4)
  name: "El Rancho Los Compadres"
  restaurant_id: "41190515"
  oldestGrade: 2010-08-26T00:00:00.000+00:00

```

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
  }
]
```

```
1  [
2  {
3    $group: {
4      _id: { borough: "$borough", street: "$address.street" },
5      boroughName: { $first: "$borough" },
6      streetName: { $first: "$address.street" },
7      count_of_restaurants: { $sum: 1 }
8    }
9  },
10 {
11   $sort: { count_of_restaurants: -1 }
12 },
13 {
14   $limit: 1
15 }
16 ]
```

Result :

```
▸ _id: Object
  boroughName : "Manhattan"
  streetName : "Broadway"
  count_of_restaurants : 615
```

Complex Query

Query : List of restaurants that had a good first review and a 'bad' last one

```

1  ▾ [
2  ▾   {
3      $unwind: "$grades"
4  },
5  ▾   {
6      ▾ $group: {
7          _id: "$_id",
8          name: { $first: "$name" },
9          initialGrade: { $first: "$grades.grade" },
10         initialDate: { $min: "$grades.date" },
11         latestGrade: { $last: "$grades.grade" },
12         latestDate: { $max: "$grades.date" }
13     }
14 },
15 ▾   {
16     ▾ $match: {
17         ▾ $expr: {
18             ▾ $and: [
19                 { $eq: ["$initialGrade", "A"] },
20                 { $in: ["$latestGrade", ["B", "C"]] }
21             ]
16         }
17     }
18 }
19 ]
20
21
22
23
24
25 ]
26

```

```

[
  {
    $unwind: "$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"]] }
        ]
      }
    }
  }
]

```

PIPELINE OUTPUT

Sample of 20 documents

```
_id: ObjectId('65d2678b0ef262c0d333b6c0')
name: "Port Authority Food Court"
initialGrade: "A"
initialDate: 2013-09-24T00:00:00.000+00:00
latestGrade: "B"
latestDate: 2014-08-15T00:00:00.000+00:00
```

```
_id: ObjectId('65d2678a0ef262c0d33394ba')
name: "Traif"
initialGrade: "A"
initialDate: 2011-09-22T00:00:00.000+00:00
latestGrade: "B"
latestDate: 2014-01-08T00:00:00.000+00:00
```

```
_id: ObjectId('65d2678a0ef262c0d333981b')
name: "Fried Dumpling Jie Jie Sheng"
initialGrade: "A"
initialDate: 2011-08-11T00:00:00.000+00:00
latestGrade: "A"
latestDate: 2014-01-08T00:00:00.000+00:00
```

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
  }
}
]
```

12

```

1  [ {
2    $unwind: "$grades"
3  },
4  {
5    $group: {
6      _id: "$_id",
7      name: { $first: "$name" },
8      numA: { $sum: { $cond: [{ $eq: ["$grades.grade", "A"] }, 1, 0] } },
9      numB: { $sum: { $cond: [{ $eq: ["$grades.grade", "B"] }, 1, 0] } },
10     numC: { $sum: { $cond: [{ $eq: ["$grades.grade", "C"] }, 1, 0] } }
11   }
12 },
13 {
14   $sort: {
15     numC: -1,
16     numB: -1,
17     numA: -1
18   }
19 }
20 ]
21
22
23

```

Results :

PIPELINE OUTPUT

OUTPUT O

Sample of 20 documents

```

_id: ObjectId('65d2678a0ef262c0d3339fe1')
name: "Red Chopstick"
numA: 0
numB: 0
numC: 6

```

```

_id: ObjectId('65d267890ef262c0d33391a2')
name: "Bella Vita"
numA: 0
numB: 3
numC: 4

```

```

_id: ObjectId('65d267890ef262c0d333939a')
name: "Amici 36"
numA: 2
numB: 1
numC: 4

```

How is Chopstick still in business ?