

ETUDIANT : Pierre Hérissé

MongoDB - CRUD & Aggregations

I. Opérations CRUD

II.1 Importer le fichier piscines_paris.json dans une base paris dans une collection piscines.
(voir mongoimport)

mongoimport --db paris --collection piscines --file ./piscines_paris.json

```
pierre@Pierre:advanced_database/tp3 (master)$ mongoimport --db paris --collection piscines --file ./piscines_paris.json
2019-10-09T09:15:03.865+0200    connected to: mongodb://localhost/
2019-10-09T09:15:03.879+0200    31 document(s) imported successfully. 0 document(s) failed to import.
```

// Dans la collection "piscines" de la base "paris", trouver en utilisant les opérateurs de requête

use paris

db.piscines.find()

```
> use paris
switched to db paris
> db.piscines.find()
{ "_id" : ObjectId("5d9d88f775d6a5051079cb7f"), "id" : 2, "name" : "Piscine Saint-Merri", "address" : "16 rue du renard", "zipCode" : 75004, "lat" : 48.859406, "lon" : 2.352479 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb80"), "id" : 2920, "name" : "Piscine Georges Drigny", "address" : "18, rue Bochart de Saron ", "zipCode" : 75009, "lat" : 48.881893, "lon" : 2.342184 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb81"), "id" : 2921, "name" : "Piscine Paul Valeyre", "address" : "24, rue de Rochechouart", "zipCode" : 75009, "lat" : 48.8781484, "lon" : 2.344948899999996 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb82"), "id" : 2923, "name" : "Piscine Château-Landon", "address" : "31, rue du château-Landon ", "zipCode" : 75010, "lat" : 48.883221, "lon" : 2.363909 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb83"), "id" : 2919, "name" : "Piscine Saint-Germain", "address" : "12, rue Lobineau ", "zipCode" : 75006, "lat" : 48.851669, "lon" : 2.335931 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb84"), "id" : 2924, "name" : "Piscine Georges Rigal", "address" : "115, boulevard de Charonne ", "zipCode" : 75011, "lat" : 48.856609, "lon" : 2.394056 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb85"), "id" : 2925, "name" : "Piscine de la Cour des Lions", "address" : "9, rue Alphonse Baudin ", "zipCode" : 75011, "lat" : 48.860626, "lon" : 2.370507 }
```

// les piscines qui sont dans le 13e arrondissement

db.piscines.find({zipCode: 75013})

```
> db.piscines.find({zipCode: 75013})
{ "_id" : ObjectId("5d9d88f775d6a5051079cb87"), "id" : 2927, "name" : "Piscine de la Butte aux Cailles", "address" : "5, place Paul Verlaine ", "zipCode" : 75013, "lat" : 48.827431, "lon" : 2.352194 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb88"), "id" : 2928, "name" : "Piscine Château des Rentiers", "address" : "184, rue du Château des Rentiers ", "zipCode" : 75013, "lat" : 48.830406, "lon" : 2.3619 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb89"), "id" : 2929, "name" : "Piscine Dunois", "address" : "70, rue Dunois ", "zipCode" : 75013, "lat" : 48.832973, "lon" : 2.366437 }
> █
```

// les piscines qui ne sont pas le 13e arrondissement

db.piscines.find({zipCode: { \$nin: [75013] } })

```
> db.piscines.find( {zipCode: { $nin: [75013] } } )
{ "_id" : ObjectId("5d9d88f775d6a5051079cb7f"), "id" : 2, "name" : "Piscine Saint-Merri", "address" : "16
rue du renard", "zipCode" : 75004, "lat" : 48.859406, "lon" : 2.352479 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb80"), "id" : 2920, "name" : "Piscine Georges Drigny", "address"
: "18, rue Bochart de Saron ", "zipCode" : 75009, "lat" : 48.881893, "lon" : 2.342184 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb81"), "id" : 2921, "name" : "Piscine Paul Valeyre", "address" :
"24, rue de Rochechouart", "zipCode" : 75009, "lat" : 48.8781484, "lon" : 2.344948899999996 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb82"), "id" : 2923, "name" : "Piscine Chateau-Landon", "address"
: "31, rue du chateau-Landon ", "zipCode" : 75010, "lat" : 48.883221, "lon" : 2.363909 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb83"), "id" : 2919, "name" : "Piscine Saint-Germain", "address"
: "12, rue Lobineau ", "zipCode" : 75006, "lat" : 48.851669, "lon" : 2.335931 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb84"), "id" : 2924, "name" : "Piscine Georges Rigal", "address"
: "115, boulevard de Charonne ", "zipCode" : 75011, "lat" : 48.856609, "lon" : 2.394056 }
```

// En affichant uniquement le nom

```
db.piscines.find(
  {zipCode: { $nin: [75013] } },
  {name: 1, _id:0}
)
```

```
> db.piscines.find( {zipCode: { $nin: [75013] } }, {name: 1, _id:0} )
{ "name" : "Piscine Saint-Merri" }
{ "name" : "Piscine Georges Drigny" }
{ "name" : "Piscine Paul Valeyre" }
{ "name" : "Piscine Chateau-Landon" }
{ "name" : "Piscine Saint-Germain" }
{ "name" : "Piscine Georges Rigal" }
{ "name" : "Piscine de la Cour des Lions" }
{ "name" : "Piscine Jean Boiteux ex Reuilly" }
{ "name" : "Piscine Aspirant Dunand" }
{ "name" : "Piscine Didot" }
{ "name" : "Piscine Armand Massard" }
{ "name" : "Piscine Emile Anthoine" }
{ "name" : "Piscine La Plaine" }
{ "name" : "Piscine Blomet" }
{ "name" : "Piscine Ren  et Andre  Mourlon" }
{ "name" : "Piscine Henry de Montherlant" }
{ "name" : "Piscine Bertrand Dauvin" }
{ "name" : "Piscine Bernard Lafay" }
{ "name" : "Piscine des Amiraux" }
{ "name" : "Piscine Hobert" }
Type "it" for more
> █
```


// les piscines qui sont dans le 13e et celles qui sont dans le 14e arrondissement
// Soit avec un \$or

```
db.piscines.find(  
  { $or: [{zipCode: 75013},  
    {zipCode: 75014}] }  
)
```

```
> db.piscines.find( { $or: [{zipCode: 75013}, {zipCode: 75014}] } )  
{ "_id" : ObjectId("5d9d88f775d6a5051079cb87"), "id" : 2927, "name" : "Piscine de la Butte aux Cailles",  
"address" : "5, place Paul Verlaine ", "zipCode" : 75013, "lat" : 48.827431, "lon" : 2.352194 }  
{ "_id" : ObjectId("5d9d88f775d6a5051079cb88"), "id" : 2928, "name" : "Piscine Château des Rentiers", "ad  
dress" : "184, rue du Château des Rentiers ", "zipCode" : 75013, "lat" : 48.830406, "lon" : 2.3619 }  
{ "_id" : ObjectId("5d9d88f775d6a5051079cb89"), "id" : 2929, "name" : "Piscine Dunois", "address" : "70,  
rue Dunois ", "zipCode" : 75013, "lat" : 48.832973, "lon" : 2.366437 }  
{ "_id" : ObjectId("5d9d88f775d6a5051079cb8a"), "id" : 2931, "name" : "Piscine Aspirant Dunand", "address  
" : "20, rue Saillard ", "zipCode" : 75014, "lat" : 48.831699, "lon" : 2.326708 }  
{ "_id" : ObjectId("5d9d88f775d6a5051079cb8b"), "id" : 2932, "name" : "Piscine Didot", "address" : "22, a  
venue Georges Lafenestre ", "zipCode" : 75014, "lat" : 48.824276, "lon" : 2.309616 }
```

// Soit avec un \$in

```
db.piscines.find( { zipCode: { $in: [75013, 75014] } } )
```

```
> db.piscines.find( { zipCode: { $in: [75013, 75014] } } )  
{ "_id" : ObjectId("5d9d88f775d6a5051079cb87"), "id" : 2927, "name" : "Piscine de la Butte aux Cailles",  
"address" : "5, place Paul Verlaine ", "zipCode" : 75013, "lat" : 48.827431, "lon" : 2.352194 }  
{ "_id" : ObjectId("5d9d88f775d6a5051079cb88"), "id" : 2928, "name" : "Piscine Château des Rentiers", "ad  
dress" : "184, rue du Château des Rentiers ", "zipCode" : 75013, "lat" : 48.830406, "lon" : 2.3619 }  
{ "_id" : ObjectId("5d9d88f775d6a5051079cb89"), "id" : 2929, "name" : "Piscine Dunois", "address" : "70,  
rue Dunois ", "zipCode" : 75013, "lat" : 48.832973, "lon" : 2.366437 }  
{ "_id" : ObjectId("5d9d88f775d6a5051079cb8a"), "id" : 2931, "name" : "Piscine Aspirant Dunand", "address  
" : "20, rue Saillard ", "zipCode" : 75014, "lat" : 48.831699, "lon" : 2.326708 }  
{ "_id" : ObjectId("5d9d88f775d6a5051079cb8b"), "id" : 2932, "name" : "Piscine Didot", "address" : "22, a  
venue Georges Lafenestre ", "zipCode" : 75014, "lat" : 48.824276, "lon" : 2.309616 }
```

// les piscines qui ne sont pas dans le 15, 16, 17 et 18e arrondissement

```
db.piscines.find(  
  {zipCode: { $nin: [75015,75016,75017,75018] } },  
  {name: 1, _id:0}  
)
```

```
> db.piscines.find( {zipCode: { $nin: [75015,75016,75017,75018] } }, {name: 1, _id:0} )  
{ "name" : "Piscine Saint-Merri" }  
{ "name" : "Piscine Georges Drigny" }  
{ "name" : "Piscine Paul Valeyre" }  
{ "name" : "Piscine Château-Landon" }  
{ "name" : "Piscine Saint-Germain" }  
{ "name" : "Piscine Georges Rigal" }  
{ "name" : "Piscine de la Cour des Lions" }  
{ "name" : "Piscine Jean Boiteux ex Reuilly" }  
{ "name" : "Piscine de la Butte aux Cailles" }  
{ "name" : "Piscine Château des Rentiers" }  
{ "name" : "Piscine Dunois" }  
{ "name" : "Piscine Aspirant Dunand" }  
{ "name" : "Piscine Didot" }  
{ "name" : "Piscine Rouvet" }  
{ "name" : "Piscine Georges Vallerey" }  
{ "name" : "Piscine Mathis" }  
{ "name" : "Piscine Jean Taris" }  
{ "name" : "Piscine Alfred Nakache" }  
{ "name" : "Piscine Catherine Lagatu ex Parmentier" }  
{ "name" : "Piscine Beaujon" }
```

// En triant par code postal descendant:

```
db.piscines.find( {}, {name: 1, zipCode: 1, _id:0} ).sort({zipCode:-1})
```

```
> db.piscines.find( {}, {name: 1, zipCode: 1, _id:0} ).sort({zipCode:-1})
{ "name" : "Piscine Georges Vallerey", "zipCode" : 75020 }
{ "name" : "Piscine Alfred Nakache", "zipCode" : 75020 }
{ "name" : "Piscine Rouvet", "zipCode" : 75019 }
{ "name" : "Piscine Mathis", "zipCode" : 75019 }
{ "name" : "Piscine Bertrand Dauvin", "zipCode" : 75018 }
{ "name" : "Piscine des Amiraux", "zipCode" : 75018 }
{ "name" : "Piscine Høbert", "zipCode" : 75018 }
{ "name" : "Piscine Bernard Lafay", "zipCode" : 75017 }
{ "name" : "Piscine Henry de Montherlant", "zipCode" : 75016 }
{ "name" : "Piscine d'Auteuil", "zipCode" : 75016 }
{ "name" : "Piscine Armand Massard", "zipCode" : 75015 }
{ "name" : "Piscine Emile Anthoine", "zipCode" : 75015 }
{ "name" : "Piscine La Plaine", "zipCode" : 75015 }
{ "name" : "Piscine Blomet", "zipCode" : 75015 }
{ "name" : "Piscine Renø et Andrø Mourlon", "zipCode" : 75015 }
{ "name" : "Piscine Aspirant Dunand", "zipCode" : 75014 }
{ "name" : "Piscine Didot", "zipCode" : 75014 }
{ "name" : "Piscine de la Butte aux Cailles", "zipCode" : 75013 }
{ "name" : "Piscine Chøteau des Rentiers", "zipCode" : 75013 }
{ "name" : "Piscine Dunois", "zipCode" : 75013 }
```

// les piscines dont le code postal est supérieur ou égal à 75013 triés par code postal descendant

```
db.piscines.find(
  {zipCode: {$gte: 75013}},
  {name: 1, zipCode: 1, _id:0}
).sort({zipCode:-1})
```

```
> db.piscines.find( {zipCode: {$gte: 75013}}, {name: 1, zipCode: 1, _id:0} ).sort({zipCode:-1})
{ "name" : "Piscine Georges Vallerey", "zipCode" : 75020 }
{ "name" : "Piscine Alfred Nakache", "zipCode" : 75020 }
{ "name" : "Piscine Rouvet", "zipCode" : 75019 }
{ "name" : "Piscine Mathis", "zipCode" : 75019 }
{ "name" : "Piscine Bertrand Dauvin", "zipCode" : 75018 }
{ "name" : "Piscine des Amiraux", "zipCode" : 75018 }
{ "name" : "Piscine Høbert", "zipCode" : 75018 }
{ "name" : "Piscine Bernard Lafay", "zipCode" : 75017 }
{ "name" : "Piscine Henry de Montherlant", "zipCode" : 75016 }
{ "name" : "Piscine d'Auteuil", "zipCode" : 75016 }
{ "name" : "Piscine Armand Massard", "zipCode" : 75015 }
{ "name" : "Piscine Emile Anthoine", "zipCode" : 75015 }
{ "name" : "Piscine La Plaine", "zipCode" : 75015 }
{ "name" : "Piscine Blomet", "zipCode" : 75015 }
{ "name" : "Piscine Renø et Andrø Mourlon", "zipCode" : 75015 }
{ "name" : "Piscine Aspirant Dunand", "zipCode" : 75014 }
{ "name" : "Piscine Didot", "zipCode" : 75014 }
{ "name" : "Piscine de la Butte aux Cailles", "zipCode" : 75013 }
{ "name" : "Piscine Chøteau des Rentiers", "zipCode" : 75013 }
{ "name" : "Piscine Dunois", "zipCode" : 75013 }
```

// Les piscines situées à l'ouest de Notre Dame de Paris

db.piscines.find({lon: {\$lt: 2.335198}})

```
> db.piscines.find( {lon: {$lt: 2.335198}} ).pretty()
{
  "_id" : ObjectId("5d9d88f775d6a5051079cb8a"),
  "id" : 2931,
  "name" : "Piscine Aspirant Dunand",
  "address" : "20, rue Saillard ",
  "zipCode" : 75014,
  "lat" : 48.831699,
  "lon" : 2.326708
}
{
  "_id" : ObjectId("5d9d88f775d6a5051079cb8b"),
  "id" : 2932,
  "name" : "Piscine Didot",
  "address" : "22, avenue Georges Lafenestre ",
  "zipCode" : 75014,
  "lat" : 48.824276,
  "lon" : 2.309616
}
```

// Et leur nombre

db.piscines.count({lon: {\$lt: 2.335198}})

```
> db.piscines.count( {lon: {$lt: 2.335198}} )
11
```

// Les piscines dont zipCode=75013 ET id=2929 avec l'opérateur \$and et \$eq

db.piscines.find(
 { \$and: [
 {zipCode: {\$eq: 75013}},
 {id: {\$eq: 2929}}
]}]
)

```
> db.piscines.find( { $and: [ {zipCode: {$eq: 75013}}, {id: {$eq: 2929}} ] } )
{ "_id" : ObjectId("5d9d88f775d6a5051079cb89"), "id" : 2929, "name" : "Piscine Dunois", "address" : "70, rue Dunois ", "zipCode" : 75013, "lat" : 48.832973, "lon" : 2.366437 }
```


// Retrouver les 5 premières piscines par ordre alphabétique (et dont le champ zipCode existe)

db.piscines.find({zipCode: {\$exists:true} }).sort({name: 1}).limit(5)

```
> db.piscines.find({zipCode: {$exists:true} }).sort({name: 1}).limit(5)
{ "_id" : ObjectId("5d9d88f775d6a5051079cb9a"), "id" : 4012, "name" : "Piscine Alfred Nakache", "address" : "4-12, rue Denoyez ", "zipCode" : 75020, "lat" : 48.871498, "lon" : 2.378612 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb8c"), "id" : 2933, "name" : "Piscine Armand Massard", "address" : "66 boulevard du Montparnasse ", "zipCode" : 75015, "lat" : 48.844036, "lon" : 2.323341 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb8a"), "id" : 2931, "name" : "Piscine Aspirant Dunand", "address" : "20, rue Saillard ", "zipCode" : 75014, "lat" : 48.831699, "lon" : 2.326708 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb9c"), "id" : 17349, "name" : "Piscine Beaujon", "address" : "7 Allée Louis de Funès", "zipCode" : 75008, "lat" : 48.8765928, "lon" : 2.307001600000004 }
{ "_id" : ObjectId("5d9d88f775d6a5051079cb93"), "id" : 2940, "name" : "Piscine Bernard Lafay", "address" : "79, rue de la Jonquière ", "zipCode" : 75017, "lat" : 48.894707, "lon" : 2.318883 }
```

// Ajoutez 2 piscines avec un champ nom au lieu de name

db.piscines.insertMany([{ nom: "Giga piscine" }, { nom: "Maxi piscine" }])

```
> db.piscines.insertMany( [ { nom: "Giga piscine" }, { nom: "Maxi piscine" } ] )
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("5d9d97125a2169b088678819"),
    ObjectId("5d9d97125a2169b08867881a")
  ]
}
```

```
> db.piscines.find({}, {_id: 0, nom: 1}).sort({nom: -1}).limit(3)
{ "nom" : "Maxi piscine" }
{ "nom" : "Giga piscine" }
{ }
>
```

// Si je compte mes piscines, j'en ai donc 33

db.piscines.count()

```
> db.piscines.count()
33
```

// Compter uniquement les piscines dont le champ name est présent

db.piscines.count({ name: {\$exists: true} })

```
> db.piscines.count({ name: {$exists: true} })
31
```

```
// Renvoie toutes les piscines ayant effectivement le champ name
// Limite à 5 résultats
// En les triant par ordre alphabétique (case sensitive)
// En plus en limitant les champs retournés au nom
```

```
db.piscines.find(
  {name: {$exists: true}},
  {_id: 0, name: 1}
).sort(
  {name: -1}
).limit(5)
```

```
> db.piscines.find({name: {$exists: true}}, {_id: 0, name: 1}).sort({name: -1}).limit(5)
{ "name" : "Piscine des Amiraux" }
{ "name" : "Piscine de la Cour des Lions" }
{ "name" : "Piscine de la Butte aux Cailles" }
{ "name" : "Piscine d'Auteuil" }
{ "name" : "Piscine Saint-Merri" }
```

II.2

```
// Créer une base de données newyork et une collection restaurants
```

```
use newyork
```

```
db.createCollection("restaurants")
```

```
// Importer le fichier restaurants.json
```

```
mongoimport --db newyork --collection restaurants --file ./restaurants.json
```

```
pierre@Pierre:advanced_database/tp3 (master*)$ mongoimport --db newyork --collection restaurants --file ./restaurants.json
2019-10-09T10:30:48.888+0200    connected to: mongod://localhost/
2019-10-09T10:30:49.380+0200    25359 document(s) imported successfully. 0 document(s) failed to import.
```

```
// Combien y a-t-il de restaurants ?
```

```
db.restaurants.count()
```

```
25359
```

```
// Identique à
```

```
db.restaurants.aggregate( { $count: "name" } )
```

```
> db.restaurants.aggregate( { $count: "name" } )
{ "name" : 25359 }
>
```


// Trouver les restaurants qui sont dans la rue "Morris Park Ave"

> db.restaurants.find({ "address.street" : "Morris Park Ave" })

```
> db.restaurants.find( { "address.street" : "Morris Park Ave" } )
{ "_id" : ObjectId("5d9d9ab8c37579d665d5d090"), "address" : { "building" : "1007", "coord" : [ -73.856077, 40.848447 ], "street" : "Morris Park Ave", "zipcode" : "10462" }, "borough" : "Bronx", "cuisine" : "Bakery", "grades" : [ { "date" : ISODate("2014-03-03T00:00:00Z"), "grade" : "A", "score" : 2 }, { "date" : ISODate("2013-09-11T00:00:00Z"), "grade" : "A", "score" : 6 }, { "date" : ISODate("2013-01-24T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2011-11-23T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2011-03-10T00:00:00Z"), "grade" : "B", "score" : 14 } ], "name" : "Morris Park Bake Shop", "restaurant_id" : "30075445" }
{ "_id" : ObjectId("5d9d9ab8c37579d665d5d457"), "address" : { "building" : "780", "coord" : [ -73.8630529, 40.8456372 ], "street" : "Morris Park Ave", "zipcode" : "10462" }, "borough" : "Bronx", "cuisine" : "Chicken", "grades" : [ { "date" : ISODate("2014-09-09T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2013-07-18T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2013-02-25T00:00:00Z"), "grade" : "A", "score" : 3 }, { "date" : ISODate("2012-09-04T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2012-04-09T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2011-09-27T00:00:00Z"), "grade" : "A", "score" : 2 } ], "name" : "Chick-N-Ribs", "restaurant_id" : "40396053" }
{ "_id" : ObjectId("5d9d9ab8c37579d665d5db83"), "address" : { "building" : "1056", "coord" : [ -73.8542141, 40.848645800000001 ], "street" : "Morris Park Ave", "zipcode" : "10461" }, "borough" : "Bronx", "cuisine" : "Pizza/Italian", "grades" : [ { "date" : ISODate("2014-09-22T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2014-04-17T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2013-03-19T00:00:00Z"), "grade" : "A", "score" : 3 }, { "date" : ISODate("2012-09-13T00:00:00Z"), "grade" : "A", "score" : 5 }, { "date" : ISODate("2012-03-06T00:00:00Z"), "grade" : "A", "score" : 8 } ], "name" : "Captain'S Pizzeria And Restaurant", "restaurant_id" : "40731746" }
{ "_id" : ObjectId("5d9d9ab8c37579d665d5dbd8"), "address" : { "building" : "738", "coord" : [ -73.864632, 40.8452821 ], "street" : "Morris Park Avenue", "zipcode" : "10462" }, "borough" : "Bronx", "cuisine" : "Italian", "grades" : [ { "date" : ISODate("2014-09-09T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2013-07-18T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2013-02-25T00:00:00Z"), "grade" : "A", "score" : 3 }, { "date" : ISODate("2012-09-04T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2012-04-09T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2011-09-27T00:00:00Z"), "grade" : "A", "score" : 2 } ], "name" : "Chick-N-Ribs", "restaurant_id" : "40396053" }
```

// Combien y en-a-t-il ?

```
> db.restaurants.count( { "address.street" : "Morris Park Ave" } )
9
```

// Pour aussi récupérer ceux qui ont pour rue "Morris Park Avenue"

```
db.restaurants.find(
  { $or: [
    { "address.street" : "Morris Park Avenue" },
    { "address.street" : "Morris Park Ave" }
  ] }
)
```

```
> db.restaurants.find( { $or: [{ "address.street" : "Morris Park Avenue" }, { "address.street" : "Morris Park Ave" }] } )
{ "_id" : ObjectId("5d9d9ab8c37579d665d5d090"), "address" : { "building" : "1007", "coord" : [ -73.856077, 40.848447 ], "street" : "Morris Park Ave", "zipcode" : "10462" }, "borough" : "Bronx", "cuisine" : "Bakery", "grades" : [ { "date" : ISODate("2014-03-03T00:00:00Z"), "grade" : "A", "score" : 2 }, { "date" : ISODate("2013-09-11T00:00:00Z"), "grade" : "A", "score" : 6 }, { "date" : ISODate("2013-01-24T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2011-11-23T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2011-03-10T00:00:00Z"), "grade" : "B", "score" : 14 } ], "name" : "Morris Park Bake Shop", "restaurant_id" : "30075445" }
{ "_id" : ObjectId("5d9d9ab8c37579d665d5d457"), "address" : { "building" : "780", "coord" : [ -73.8630529, 40.8456372 ], "street" : "Morris Park Ave", "zipcode" : "10462" }, "borough" : "Bronx", "cuisine" : "Chicken", "grades" : [ { "date" : ISODate("2014-09-09T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2013-07-18T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2013-02-25T00:00:00Z"), "grade" : "A", "score" : 3 }, { "date" : ISODate("2012-09-04T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2012-04-09T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2011-09-27T00:00:00Z"), "grade" : "A", "score" : 2 } ], "name" : "Chick-N-Ribs", "restaurant_id" : "40396053" }
{ "_id" : ObjectId("5d9d9ab8c37579d665d5db83"), "address" : { "building" : "1056", "coord" : [ -73.8542141, 40.848645800000001 ], "street" : "Morris Park Ave", "zipcode" : "10461" }, "borough" : "Bronx", "cuisine" : "Pizza/Italian", "grades" : [ { "date" : ISODate("2014-09-22T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2014-04-17T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2013-03-19T00:00:00Z"), "grade" : "A", "score" : 3 }, { "date" : ISODate("2012-09-13T00:00:00Z"), "grade" : "A", "score" : 5 }, { "date" : ISODate("2012-03-06T00:00:00Z"), "grade" : "A", "score" : 8 } ], "name" : "Captain'S Pizzeria And Restaurant", "restaurant_id" : "40731746" }
{ "_id" : ObjectId("5d9d9ab8c37579d665d5dbd8"), "address" : { "building" : "738", "coord" : [ -73.864632, 40.8452821 ], "street" : "Morris Park Avenue", "zipcode" : "10462" }, "borough" : "Bronx", "cuisine" : "Italian", "grades" : [ { "date" : ISODate("2014-09-09T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2013-07-18T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2013-02-25T00:00:00Z"), "grade" : "A", "score" : 3 }, { "date" : ISODate("2012-09-04T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2012-04-09T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2011-09-27T00:00:00Z"), "grade" : "A", "score" : 2 } ], "name" : "Chick-N-Ribs", "restaurant_id" : "40396053" }
```


// Afficher uniquement (sans l'_id) les champs quartier, type de cuisine et adresse

db.restaurants.find({}, {_id: 0, borough: 1, cuisine: 1, address: 1})

```
> db.restaurants.find({}, {_id: 0, borough: 1, cuisine: 1, address: 1}).limit(1)
{ "address" : { "building" : "469", "coord" : [ -73.961704, 40.662942 ], "street" : "Flatbush Avenue", "zipcode" : "11225" }, "borough" : "Brooklyn", "cuisine" : "Hamburgers" }
>
```

// Trouver la liste des restaurants situés à Staten Island qui font des hamburgers OU de la boulangerie.

// Avec un \$or

**db.restaurants.find({
 borough: "Staten Island",
 \$or: [
 {cuisine: "Hamburger"},
 {cuisine: "Bakery"}
]
}, { _id: 0 })**

```
> db.restaurants.find( { borough: "Staten Island", $or: [{cuisine: "Hamburger"}, {cuisine: "Bakery"}] }, { _id: 0 }).limit(1)
{ "address" : { "building" : "405", "coord" : [ -74.15565029999999, 40.5644155 ], "street" : "Arthur Kill Road", "zipcode" : "10308" }, "borough" : "Staten Island", "cuisine" : "Bakery", "grades" : [ { "date" : ISODate("2014-06-03T00:00:00Z"), "grade" : "A", "score" : 2 }, { "date" : ISODate("2013-06-08T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2012-05-22T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2011-06-10T00:00:00Z"), "grade" : "A", "score" : 4 } ], "name" : "Holtermann's Bakery", "restaurant_id" : "40423364" }
> db.restaurants.find( { borough: "Staten Island", $or: [{cuisine: "Hamburger"}, {cuisine: "Bakery"}] }, { _id: 0 })
{ "address" : { "building" : "405", "coord" : [ -74.15565029999999, 40.5644155 ], "street" : "Arthur Kill Road", "zipcode" : "10308" }, "borough" : "Staten Island", "cuisine" : "Bakery", "grades" : [ { "date" :
```

// Avec un \$in

**db.restaurants.find(
 { borough: "Staten Island", cuisine: {
 \$in: ["Hamburger", "Bakery"]
 }
}, { _id: 0 })**

```
> db.restaurants.find( { borough: "Staten Island", cuisine: { $in: ["Hamburger", "Bakery"] } }, { _id: 0 })
{ "address" : { "building" : "405", "coord" : [ -74.15565029999999, 40.5644155 ], "street" : "Arthur Kill Road", "zipcode" : "10308" }, "borough" : "Staten Island", "cuisine" : "Bakery", "grades" : [ { "date" : ISODate("2014-06-03T00:00:00Z"), "grade" : "A", "score" : 2 }, { "date" : ISODate("2013-06-08T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2012-05-22T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2011-06-10T00:00:00Z"), "grade" : "A", "score" : 4 } ], "name" : "Holtermann's Bakery", "restaurant_id" : "40423364" }
{ "address" : { "building" : "1117", "coord" : [ -74.07979739999999, 40.5987581 ], "street" : "Hylan Boulevard", "zipcode" : "10305" }, "borough" : "Staten Island", "cuisine" : "Bakery", "grades" : [ { "date" : ISODate("2014-12-23T00:00:00Z"), "grade" : "A", "score" : 13 }, { "date" : ISODate("2013-12-02T00:00:00Z"), "grade" : "A", "score" : 13 }, { "date" : ISODate("2012-10-02T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2012-05-21T00:00:00Z"), "grade" : "A", "score" : 13 } ], "name" : "Buono Bakery", "restaurant_id" : "40423440" }
```

```
// Quel est le type de restaurant le plus présent ?  
// La méthode aggregate de mongoDB fait la même chose de manière plus puissante  
// db.collection.aggregate(query, options)
```

```
db.restaurants.aggregate([ {"$group": { "_id": "$cuisine", count: {$sum: 1} } } ])
```

```
> db.restaurants.aggregate([ {"$group": { "_id": "$cuisine", count: {$sum: 1} } } ])  
{ "_id" : "Russian", "count" : 88 }  
{ "_id" : "Cajun", "count" : 7 }  
{ "_id" : "Sandwiches/Salads/Mixed Buffet", "count" : 255 }  
{ "_id" : "Turkish", "count" : 70 }  
{ "_id" : "Barbecue", "count" : 52 }  
{ "_id" : "Soups & Sandwiches", "count" : 51 }  
{ "_id" : "Chicken", "count" : 410 }  
{ "_id" : "Steak", "count" : 86 }  
{ "_id" : "English", "count" : 16 }  
{ "_id" : "Café/Coffee/Tea", "count" : 2 }  
{ "_id" : "Juice, Smoothies, Fruit Salads", "count" : 273 }
```

```
// Pour avoir des détails sur la requête, utiliser explain
```

```
> db.restaurants.explain().aggregate([ {"$group": { "_id": "$cuisine", count: {$sum: 1} } } ])  
{  
  "stages" : [  
    {  
      "$cursor" : {  
        "query" : {  
          },  
        "fields" : {  
          "cuisine" : 1,  
          "_id" : 0  
        },  
        "queryPlanner" : {  
          "plannerVersion" : 1,  
          "namespace" : "newyork.restaurants",  
          "indexFilterSet" : false,  
          "parsedQuery" : {  
            },  
          "queryHash" : "8B3D4AB8",  
          "planCacheKey" : "8B3D4AB8",  
          "winningPlan" : {  
            "stage" : "COLLSCAN",  
            "direction" : "forward"  
          },  
          "rejectedPlans" : [ ]  
        },  
      },  
    },  
    {  
      "$group" : {  
        "_id" : "$cuisine",  
        "count" : {  
          "$sum" : {  
            "$const" : 1  
          }  
        }  
      }  
    }  
  ],  
  "ok" : 1  
}
```

```
// Faire la même requête pour le quartier du Bronx
// En limitant le nombre de retours à 5
```

```
db.restaurants.aggregate([
  {"$group":
    { _id: "$cuisine", count: {$sum: 1} }
  },
  {$limit: 5}
])
```

```
> db.restaurants.aggregate([ {"$group": { _id: "$cuisine", count: {$sum: 1} } }, {$limit: 5} ])
{ "_id" : "Russian", "count" : 88 }
{ "_id" : "Cajun", "count" : 7 }
{ "_id" : "Sandwiches/Salads/Mixed Buffet", "count" : 255 }
{ "_id" : "Turkish", "count" : 70 }
{ "_id" : "Barbecue", "count" : 52 }
```

II.3

```
// Reprendre la base paris
// On ajoute un champ 'acces_handicape' à true aux piscines du 13e
```

```
db.piscines.updateMany( { zipCode: 75013 }, { $set: {acces_handicape: true} } )
```

```
> db.piscines.updateMany( { zipCode: 75013 }, { $set: {acces_handicape: true} } )
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
```

```
// Ajouter un champ verif true et supprimer l'accès handicapé
```

```
db.piscines.updateMany(
  { acces_handicape: {$exists: true} },
  { $set: {verif: true}, $unset: {acces_handicape: ""} }
)
```

```
> db.piscines.updateMany( { acces_handicape: {$exists: true} }, { $set: {verif: true}, $unset: {acces_handicape: ""} } )
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
```

II.4

```
// Dans la liste des restaurants
// Modifier les restaurants dont la cuisine est Hamburgers pour leur ajouter un champ
healthy_food égal à 2
```

```
db.restaurants.updateMany( { cuisine: "Hamburgers" }, { $set: {healthy_food: 2} } )
```

```
> db.restaurants.updateMany( { cuisine: "Hamburgers" }, { $set: {healthy_food: 2} } )
{ "acknowledged" : true, "matchedCount" : 433, "modifiedCount" : 433 }
```


// Pour les végétariens, leur mettre le champ healthy food à 9.

db.restaurants.updateMany({ cuisine: "Vegetarian" }, { \$set: {healthy_food: 9} })

```
> db.restaurants.updateMany( { cuisine: "Vegetarian" }, { $set: {healthy_food: 9} } )
{ "acknowledged" : true, "matchedCount" : 102, "modifiedCount" : 102 }
```

// Vérifier que tous les restaurants ont un tableau grades

db.restaurants.count({grades: {\$exists: true}})

```
> db.restaurants.count()
25359
> db.restaurants.count( {grades: {$exists: true}} )
25359
```

// Supprimer le champ building des restaurants situés dans le Bronx et ajouter un booléen

**db.restaurants.updateMany(
 { borough: "Bronx" },
 {
 \$unset: {"address.building": ""},
 \$set: {isInBronx: true}
 }
)**

```
> db.restaurants.updateMany( { borough: "Bronx" }, { $unset: {"address.building": ""}, $set: {isInBronx: true} } )
{ "acknowledged" : true, "matchedCount" : 2338, "modifiedCount" : 2338 }
```

//Vérifier

**db.restaurants.find(
 { borough: "Bronx" },
 { _id: 0, name: 1, "address.building": 1, isInBronx: 1 }
)**

```
> db.restaurants.find( { borough: "Bronx" }, { _id: 0, name: 1, "address.building": 1, isInBronx: 1 } )
{ "address" : { }, "name" : "Wild Asia", "isInBronx" : true }
{ "address" : { }, "name" : "Carvel Ice Cream", "isInBronx" : true }
{ "address" : { }, "name" : "Happy Garden", "isInBronx" : true }
{ "address" : { }, "name" : "Morris Park Bake Shop", "isInBronx" : true }
{ "address" : { }, "name" : "Happy Garden", "isInBronx" : true }
{ "address" : { }, "name" : "Manhem Club", "isInBronx" : true }
{ "address" : { }, "name" : "The New Starling Athletic Club Of The Bronx", "isInBronx" : true }
{ "address" : { }, "name" : "Yankee Tavern", "isInBronx" : true }
```

// Ajouter un champ rating à 5 à tous les restaurants

db.restaurants.updateMany({}, { \$set: {rating: 5} })

```
> db.restaurants.updateMany( {}, { $set: {rating: 5} } )
{ "acknowledged" : true, "matchedCount" : 25359, "modifiedCount" : 25359 }
>
```

// Multiplier le champ rating par 2 pour les restaurants situés dans le Queens

```
db.restaurants.updateMany(  
  { borough: "Queens" },  
  { $mul: {rating: NumberInt(2)} }  
)
```

```
> db.restaurants.updateMany( { borough: "Queens" }, { $mul: {rating: NumberInt(2)} } )  
{ "acknowledged" : true, "matchedCount" : 5656, "modifiedCount" : 5656 }
```

// Trouver les restaurants de Brooklyn
// Limiter les résultats à 100

```
db.restaurants.find({borough: "Brooklyn"}).limit(100)
```

// Appliquer d'abord un count()

```
db.restaurants.find({borough: "Brooklyn"}).count()
```

// Puis à la place appliquer un size()

```
db.restaurants.find({borough: "Brooklyn"}).size()
```

// Ajouter une entrée au tableau grades pour le restaurant "Tu-Lu'S Gluten-Free Bakery"

```
db.restaurants.updateOne(  
  {name: "Tu-Lu'S Gluten-Free Bakery"},  
  { $push: {grades: {grade: "Z"}} }  
)
```

```
> db.restaurants.updateOne( {name: "Tu-Lu'S Gluten-Free Bakery"}, { $push: {grades: {grade: "Z"}} } )  
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }  
> db.restaurants.find( {name: "Tu-Lu'S Gluten-Free Bakery"} )  
{ "_id" : ObjectId("5d9d9ab9c37579d665d5fafb"), "address" : { "building" : "338", "coord" : [ -73.9846671  
, 40.72952 ], "street" : "East 11 Street", "zipcode" : "10003" }, "borough" : "Manhattan", "cuisine" :  
"Bakery", "grades" : [ { "date" : ISODate("2014-12-01T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date"  
: ISODate("2013-09-24T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2013-05-08T00:00:  
00Z"), "grade" : "B", "score" : 25 }, { "date" : ISODate("2012-04-25T00:00:00Z"), "grade" : "A", "score"  
: 12 }, { "grade" : "Z" } ], "name" : "Tu-Lu'S Gluten-Free Bakery", "restaurant_id" : "41460034", "rating"  
: 5 }  
> db.restaurants.updateOne( {name: "Tu-Lu'S Gluten-Free Bakery"}, { $push: {grades: {grade: "Z"}} } )
```

```
// Modifier le champ rating pour tous les documents pour qu'il soit égal à la moyenne réelle
des grades
// Créer un curseur et le manipuler avec un forEach
// https://docs.mongodb.com/manual/tutorial/iterate-a-cursor/
```

```
var allRestaus = db.restaurants.find()
```

```
allRestaus.forEach( resto => {
```

```
    let totalScores = 0
```

```
    resto.grades.forEach( grade => {
        totalScores += grade.score;
    })
```

```
    db.restaurants.update(
        { _id: resto._id },
        { $set: { rating: (totalScores / resto.grades.length) } }
    )
});
```

```
> db.restaurants.find()
{ "_id" : ObjectId("5d9dd61a94181575cf84ddb7"), "address" : { "building" : "2300", "coord" : [ -73.878611
3, 40.8502883 ], "street" : "Southern Boulevard", "zipcode" : "10460" }, "borough" : "Bronx", "cuisine" :
"American ", "grades" : [ { "date" : ISODate("2014-05-28T00:00:00Z"), "grade" : "A", "score" : 11 }, { '
date" : ISODate("2013-06-19T00:00:00Z"), "grade" : "A", "score" : 4 }, { "date" : ISODate("2012-06-15T00:
00:00Z"), "grade" : "A", "score" : 3 } ], "name" : "Wild Asia", "restaurant_id" : "40357217", "rating" :
6 }
{ "_id" : ObjectId("5d9dd61a94181575cf84ddb8"), "address" : { "building" : "8825", "coord" : [ -73.880382
7, 40.7643124 ], "street" : "Astoria Boulevard", "zipcode" : "11369" }, "borough" : "Queens", "cuisine" :
"American ", "grades" : [ { "date" : ISODate("2014-11-15T00:00:00Z"), "grade" : "Z", "score" : 38 }, { '
date" : ISODate("2014-05-02T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2013-03-02T00
:00:00Z"), "grade" : "A", "score" : 7 }, { "date" : ISODate("2012-02-10T00:00:00Z"), "grade" : "A", "scor
e" : 13 } ], "name" : "Brunos On The Boulevard", "restaurant_id" : "40356151", "rating" : 17 }
{ "_id" : ObjectId("5d9dd61a94181575cf84ddb9"), "address" : { "building" : "6409", "coord" : [ -74.005288
99999999, 40.628886 ], "street" : "11 Avenue", "zipcode" : "11219" }, "borough" : "Brooklyn", "cuisine" :
"American ", "grades" : [ { "date" : ISODate("2014-07-18T00:00:00Z"), "grade" : "A", "score" : 12 }, { '
date" : ISODate("2013-07-30T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2013-02-13T00
:00:00Z"), "grade" : "A", "score" : 11 }, { "date" : ISODate("2012-08-16T00:00:00Z"), "grade" : "A", "sco
re" : 2 }, { "date" : ISODate("2011-08-17T00:00:00Z"), "grade" : "A", "score" : 11 } ], "name" : "Regina
Caterers", "restaurant_id" : "40356649", "rating" : 9.6 }
{ "_id" : ObjectId("5d9dd61a94181575cf84ddba"), "address" : { "building" : "1839", "coord" : [ -73.948260
9, 40.6408271 ], "street" : "Nostrand Avenue", "zipcode" : "11226" }, "borough" : "Brooklyn", "cuisine" :
"Ice Cream, Gelato, Yogurt, Ices", "grades" : [ { "date" : ISODate("2014-07-14T00:00:00Z"), "grade" : "A
", "score" : 12 }, { "date" : ISODate("2013-07-10T00:00:00Z"), "grade" : "A", "score" : 8 }, { "date" : I
SODate("2012-07-11T00:00:00Z"), "grade" : "A", "score" : 5 }, { "date" : ISODate("2012-02-23T00:00:00Z"),
"grade" : "A", "score" : 8 } ], "name" : "Taste The Tropics Ice Cream", "restaurant_id" : "40356731", "r
ating" : 8.25 }
```

```
// Quel est le restaurant qui a la meilleure moyenne
```

```
db.restaurants.find({}, {name: 1}).sort({ rating: -1 })
```

```
> db.restaurants.find({}, {name: 1}).sort({ rating: -1 })
{ "_id" : ObjectId("5d9dd61b94181575cf853bf5"), "name" : "Juice It Health Bar" }
{ "_id" : ObjectId("5d9dd61b94181575cf853e0c"), "name" : "Golden Dragon Cuisine" }
{ "_id" : ObjectId("5d9dd61b94181575cf853b0e"), "name" : "Palombo Pastry Shop" }
{ "_id" : ObjectId("5d9dd61b94181575cf853fd5"), "name" : "Chelsea'S Juice Factory" }
{ "_id" : ObjectId("5d9dd61b94181575cf85226b"), "name" : "Go Go Curry" }
{ "_id" : ObjectId("5d9dd61b94181575cf85369c"), "name" : "Koyla" }
```


II.5

// Vélib

// Récupérer un fichier json des velib chez jcdecaux developper

// Importer dans la base paris, le fichier jcdecaux.json dans une collection velib

```
mongoimport --db paris --collection velib --file ./jcdecaux_velib_paris.json --  
jsonArray
```

// Problème ! On n'a pas de champ codepostal ... On retrouve le code postal dans l'adresse.

// Mettez à jour tous les enregistrements en leur ajoutant un champ zipCode

```
var allVelibs = db.velib.find()
```

```
allVelibs.forEach( velib => {
```

```
    var regex = /[0-9]{5}/g
```

```
    var cp = velib.address.match(regex);
```

```
    db.velib.update(  
        { _id: velib._id },  
        { $set: { zipCode: cp[0] } }  
    )
```

```
});
```

```
> allVelibs.forEach( velib => {  
...  
... var regex = /[0-9]{5}/g  
...  
... var cp = velib.address.match(regex);  
...  
... db.velib.update(  
...  
... { _id: velib._id },  
...  
... { $set: { zipCode: cp[0] } }  
...  
... )  
...  
... });  
> db.velib.find()  
{ "_id" : ObjectId("5d9ddd41e07c390f0310c5e8"), "number" : 20011, "name" : "20011 - PYRÉNÉES-DAGORNO", "a  
ddress" : "103 RUE DES PYRENNEES - 75020 PARIS", "latitude" : 48.85550135398888, "longitude" : 2.40516852  
0639166, "zipCode" : "75020" }  
{ "_id" : ObjectId("5d9ddd41e07c390f0310c5e9"), "number" : 5005, "name" : "05005 - SAINT JACQUES GAY LUSS  
AC", "address" : "27 RUE GAY LUSSAC - 75005 PARIS", "latitude" : 48.844730256132095, "longitude" : 2.3419  
23944866407, "zipCode" : "75005" }  
{ "_id" : ObjectId("5d9ddd41e07c390f0310c5ea"), "number" : 28002, "name" : "28002 - SOLJENITSYNE (PUTEAX  
)", "address" : "BOULEVARD ALEXANDRE SOLJENITSYNE - 92800 PUTEAUX", "latitude" : 48.884478, "longitude" :  
2.24772065, "zipCode" : "92800" }
```

// Quel est l'arrondissement de Paris ou il y a le plus de stations ? (avec un \$in)

```
db.velib.aggregate([
  {"$group": {
    _id: "$zipCode",
    count: {$sum: 1} }
  },
  { $sort: {count: -1} }
])
```

```
> db.velib.aggregate([ {"$group": { _id: "$zipCode", count: {$sum: 1} } }, { $sort: {count: -1} } ])
{ "_id" : "75015", "count" : 87 }
{ "_id" : "75013", "count" : 72 }
{ "_id" : "75020", "count" : 67 }
```

// Quelle est la ville (hors Paris) qui a le plus de stations
// OU plus élégant (avec aggregate)

```
db.velib.aggregate([
  {"$group": { _id: "$zipCode", count: {$sum: 1} } },
  { $regexMatch: { input: "$zipCode", regex: /75[0-9]{3}/ } },
  { $sort: {count: -1} }
])
```

/* Ne fonctionne pas */

// Cherchez la piscine Dunois.

```
db.velib.find( { name: /.*DUNOIS*/ } )
```

```
> db.velib.find( { name: /.*DUNOIS*/ } )
{ "_id" : ObjectId("5d9ddd41e07c390f0310c89c"), "number" : 13043, "name" : "13043 - DUNOIS CLISSON", "address" : "55 RUE DUNOIS - 75013 PARIS", "latitude" : 48.832272458639544, "longitude" : 2.367415866750153, "zipCode" : "75013" }
```

```
// Quelles sont les 5 stations velib les plus proches de la piscine Dunois ?  
// En modifiant la structure de la collection pour qu'elle respecte la norme geoJson et en  
utilisant l'opérateur géographique $near
```

```
db.piscines.updateMany(  
  {},  
  { $set: { location: {  
                                type: "Point",  
                                coordinates: [ $lat, $lon ]  
          } }  
}  
)  
var dunois = db.velib.find( { name: /.*DUNOIS*/ } )  
  
db.piscines.find(  
  location: {  
    $near: {  
      $geometry: {  
        type: "Point",  
        coordinates: [$dunois.latitude, $dunois.longitude]  
      },  
    },  
  },  
).sort(location: -1).limit(5)
```

II.6 Validation de schéma

<https://docs.mongodb.com/manual/core/schema-validation/>

Créez une collection en lui appliquant les restrictions que vous souhaitez.
Expliquez les restrictions mises en place et mettez-moi une capture d'écran

II.7

```
// Importer dans une base us, dans la collection companies le fichier companies.json
```

```
mongoimport --db us --collection companies --file ./companies.json
```

```
// Quelle est la société la plus ancienne ?
```

```
db.companies.find(  
  {founded_year: {$ne: null}},  
  {_id: 0, name: 1, founded_year: 1}  
)  
.sort(  
  { founded_year: 1 }  
)  
.limit(1)
```

```
> db.companies.find({founded_year: {$ne: null}}, {_id: 0, name: 1, founded_year: 1}).sort( { founded_year  
: 1 }).limit(1)  
{ "name" : "Alstrasoft", "founded_year" : 1800 }
```



```
// Quelle est la société qui emploie le plus de personnes ?

// Quelle est la société qui emploie le plus de personnes dans la publicité ?

// Quel est l'effectif cumulé des entreprises de 'network_hosting' ?

// Quelle entreprise est dirigé par Rich Langdale ?

// Supprimer les entreprises de finance

// Mettre à jour les entreprises de publicité en leur ajoutant un champ 'likes'

// Créer un index sur le champ nom de la compagnie

// Supprimer cet index

// Recréer l'index en spécifiant que la valeur doit être unique

// Insérer une société My Little Compagnie en respectant l'organisation actuelle de la base

// Trouver les sociétés qui ont un bureau situé à moins de 20 kilomètres de la statue de la Liberté

// Ajouter un champ phone dans l'adresse du premier bureau des sociétés qui sont situées dans l'état de NY

// Créer une autre collection 'awards', créer quelques récompenses en les reliant à une société en utilisant une référence

// Créer une fonction qui prend en paramètre un _id et qui calcule la moyenne des likes d'une entreprise

// Ajouter quelques likes dans un tableau et tester votre fonction
```