

# Requêtes d'agrégation

Corrigé de TD pour un cours dispensé à l'université de Rennes 2

Romain Tavenard

## 1 Avant-propos

Les bases de données utilisées ici sont les bases `food`, `etudiants` et `keolis`. Vérifiez qu'elles sont bien chargées sur le serveur avant de débiter le TD.

## 2 Agrégations “simples”

1. Affichez, sans utiliser la méthode `count()`, le nombre de restaurants dans la base `food`.

```
> db.NYfood.aggregate([
    {$group: {_id: null, nb_restos: {$sum:1}}}
])
```

2. Affichez le nombre de restaurants par quartier.

```
> db.NYfood.aggregate([
    {$group: {_id: "$borough",
               nb_restos: {$sum:1}}}
])
```

3. Même chose en ignorant les restaurants dont le quartier n'est pas renseigné (valeur `"Missing"`).

```
> db.NYfood.aggregate([
    {$match: {"borough": {$ne: "Missing"}}},
    {$group: {_id: "$borough",
               nb_restos: {$sum:1}}}
])
```

4. Même chose en faisant apparaître les quartiers ayant le plus de restaurants aux premières positions.

```
> db.NYfood.aggregate([
    {$match: {"borough": {$ne: "Missing"}}},
    {$group: {_id: "$borough",
```

```

        nb_restos: {$sum:1}},
    {$sort: {"nb_restos": -1}}
  ])

```

5. Même chose en ne conservant que les quartiers ayant plus de 5000 restaurants.

```

> db.NYfood.aggregate([
  {$match: {"borough": {$ne: "Missing"}}},
  {$group: {_id: "$borough",
    nb_restos: {$sum:1}}},
  {$match: {nb_restos: {$gt: 5000}}},
  {$sort: {"nb_restos": -1}}
])

```

### 3 Groupement par date

Vous savez déjà que les attributs de type date doivent être considérés de façon particulière : il faut notamment toujours utiliser une comparaison sous forme d'intervalle pour ces attributs. Ainsi, si on veut regrouper des documents par date (regrouper ensemble tous les documents concernant un mois donné, puis en faire une moyenne, par exemple), il faudra avoir recours à une syntaxe particulière pour préciser à quel degré de précision sur la date l'agrégation doit se faire (ici on suppose que le champ en question s'appelle `att_date` dans la collection et on fera une agrégation par jour) :

```

> db.coll.aggregate( [
  {
    $group: {
      _id: {
        month: { $month: "$att_date" },
        day: { $dayOfMonth: "$att_date" },
        year: { $year: "$att_date" }
      },
      total: { $sum: "$price" }
    }
  }
] )

```

6. Passez à la base `keolis` et affichez, mois par mois, le nombre de stations de métro vérifiées.

```

> db.metro.aggregate([
  {$group:
    {_id: {month: {$month: "$lastCheckDate"},
      year: {$year: "$lastCheckDate"}},
    nb: {$sum: 1}
  }
])

```

```

    }
  }
])

```

7. Même chose en triant les résultats par date croissante (observez bien la structure du résultat de la requête précédente).

```

> db.metro.aggregate([
  {$group:
    {_id: {month: {$month: "$lastCheckDate"},
           year: {$year: "$lastCheckDate"}},
     nb: {$sum: 1}}},
  {$sort: {"_id.year": 1, "_id.month": 1}}
])

```