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 débuter 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}}
                    ])
```

4 Manipulations de listes

8. Affichez le nombre minimum et maximum de notes obtenues par les restaurants de la base `food`.

```
> db.NYfood.aggregate([
                        {$project: {taille: {$size: "$grades"}}},
                        {$group: {_id: null,
                                  nb_min: {$min: "$taille"},
                                  nb_max: {$max: "$taille"}}
                        }
                    ])
```

9. Affichez le nombre moyen de notes obtenues par les restaurants.

```
> db.NYfood.aggregate([
                        {$project: {taille: {$size: "$grades"}}},
                        {$group: {_id: null,
                                  nb_moyen: {$avg: "$taille"}}}
                    ])
```

10. Même chose en ne considérant que les restaurants du quartier `"Manhattan"`.

```
> db.NYfood.aggregate([
                        {$match: {"borough": "Manhattan"}},
                        {$project: {taille: {$size: "$grades"}}},
                        {$group: {_id: null,
                                  nb_moyen: {$avg: "$taille"}}}
                    ])
```

11. Même chose en affichant la moyenne quartier par quartier.

```
> db.NYfood.aggregate([
                        {$project: {taille: {$size: "$grades"},
                                    quartier: "$borough"}},
                        {$group: {_id: "$quartier",
                                  nb_moyen: {$avg: "$taille"}}}
                      ])
```

12. Affichez, pour chaque valeur possible de note, le nombre de fois qu'elle a été attribuée.

```
> db.NYfood.aggregate([
                        {$unwind: "$grades"},
                        {$group: {_id: "$grades.grade", nb: {$sum: 1}}}
                      ])
```

13. Même question en ne considérant que les restaurants du quartier "Brooklyn".

```
> db.NYfood.aggregate([
                        {$match: {"borough": "Brooklyn"}},
                        {$unwind: "$grades"},
                        {$group: {_id: "$grades.grade", nb: {$sum: 1}}}
                      ])
```

14. Répétez la manipulation précédente en n'affichant que les notes données plus de 1000 fois dans ce quartier.

```
> db.NYfood.aggregate([
                        {$match: {"borough": "Brooklyn"}},
                        {$unwind: "$grades"},
                        {$group: {_id: "$grades.grade", nb: {$sum: 1}}},
                        {$match: {nb: {$gt: 1000}}}
                      ])
```

15. Au lieu de cette dernière contrainte (note donnée plus de 1000 fois), affichez les 3 notes les plus données.

```
> db.NYfood.aggregate([
                        {$match: {"borough": "Brooklyn"}},
                        {$unwind: "$grades"},
                        {$group: {_id: "$grades.grade", nb: {$sum: 1}}},
                        {$sort: {nb: -1}},
                        {$limit: 3}
                      ])
```

16. Affichez, mois par mois, le nombre d'évaluations effectuées.

```
> db.NYfood.aggregate([
                        {$unwind: "$grades"},
```

```
                        {$group: {_id: {month: {$month: "$grades.date"},
                                        year: {$year: "$grades.date"}},
                                  nb: {$sum: 1}}
                        }
                    ])
```

17. Même chose en triant les dates par ordre croissant.

```
> db.NYfood.aggregate([
                        {$unwind: "$grades"},
                        {$group: {_id: {month: {$month: "$grades.date"},
                                        year: {$year: "$grades.date"}},
                                  nb: {$sum: 1}}
                        },
                        {$sort: {"_id.year": 1, "_id.month": 1}}
                    ])
```

5 Questions avancées

18. Référez-vous à l'aide du mot-clé $out pour générer une nouvelle collection, appelée `summary`, dans la base `food` qui contienne les statistiques pour chaque année et chaque quartier des nombres moyen, minimum et maximum de notes obtenues par restaurant. Les données seront triées pour faire apparaître les années les plus récentes en premières et, pour une même année, les quartiers par ordre alphabétique.

```
> db.NYfood.aggregate([
                        {$unwind: "$grades"},
                        {$group: {_id: {year: {$year: "$grades.date"},
                                        name: "$name",
                                        borough: "$borough"},
                                  nb: {$sum: 1}}
                        },
                        {$group: {_id: {year: "$_id.year",
                                        borough: "$_id.borough"},
                                  nb_min: {$min: "$nb"},
                                  nb_max: {$max: "$nb"},
                                  nb_avg: {$avg: "$nb"}}},
                        {$sort: {"_id.year": -1, "_id.borough": 1}},
                        {$out: "summary"}
                    ])
```

19. Passez à la base `etudiants` et calculez, pour chaque étudiant, sa moyenne.

```
> db.notes.aggregate([
                        {$unwind: "$notes"},
```

```
                        {$group: {_id: "$nom", moyenne: {avg: "$notes"}}}
                    ])
```

20. Quel est le jour durant lequel ont été données le plus de notes pour des restaurants de `"Manhattan"` ?

```
> db.NYfood.aggregate([
                        {$match: {"borough": "Manhattan"}},
                        {$unwind: "$grades"},
                        {$group: {_id: {day: {$dayOfMonth: "$grades.date"},
                                        month: {$month: "$grades.date"},
                                        year: {$year: "$grades.date"}},
                                  n_notes: {$sum: 1}}},
                        {$sort: {"n_notes": -1}},
                        {$limit: 1}
])
```