MongoDB – Atelier sur l'agrégation

Pour effectuer les exercices ci-dessous, vous pouvez utiliser le Mongo Shell de compass ou de votre serveur MongoDB sous Docker ou un Playground de VSCode.

Vous disposez d'une base de données appelée « **db_bird** » avec une collection « **sightings** » et « **birds** ».

Prérequis

Veuillez restaurer dans votre serveur MongoDB la base de données « **db_bird** » depuis l'archive « **db_bird.gz** » transmises par votre enseignant en utilisant la commande suivante (dans un terminal CMD) :

```
docker exec -i mongo mongorestore
   --uri=mongodb://root:admin@localhost:27017
   --authenticationDatabase=admin
   --drop --gzip --db=db_bird
   --archive=/backupdb/db_bird.gz
```

Le fichier db_bird.gz doit préalablement être mit dans le répertoire C:\165_docker\backupdb !!

Type d'opérateurs d'agrégation:

Étape $match

Nous souhaitons utiliser ces données pour savoir où nous devrions aller voir notre oiseau préféré, le Merlebleu de l'Est. Nous utiliserons les coordonnées du lieu (latitude et longitude) et le nombre d'observations à chaque endroit pour identifier les meilleurs endroits pour observer les Merlebleus de l'Est.

Instruction :

Vous êtes désormais connecté à la base de données « db_bird ». Utilisez la collection « **sightings** ».

A. Tout d'abord, créez un pipeline d'agrégation, qui contiendra deux étapes. (Vous avez oublié la méthode d'agrégation ? Consultez l'astuce ci-dessous !)

B. Créez une étape **$match** qui filtre les documents sur notre oiseau cible avec la valeur « species_common » qui équal à « Eastern Bluebird ».

C. Créez une étape **$group** qui regroupe les documents en fonction de la latitude et de la longitude de l'observation, stockées dans le champ « location.coordinates ».

D. Au sein des groupes, créez un champ pour afficher le nombre de documents dans chaque groupe, appelé "number_of_sightings" en utilisant **$count**.

E. Exécutez votre pipeline d'agrégation et découvrez où chercher les merlebleus de l'Est !

Une fois terminé, comparez votre réponse à la bonne réponse dans la section « Solution et Code résolu ».

Astuces :

N'oubliez pas qu'un pipeline d'agrégation doit commencer par `db.<collection>.aggregate([])` dans le shell MongoDB.

- La première étape sera une étape **$match**, qui nécessitera une requête pour trouver des documents spécifiques.
- La deuxième étape sera une étape **$group**, qui nécessitera :
  - **_id**, le champ permettant de regrouper les documents
  - **$count**, l'opérateur accumulateur utilisé pour compter le nombre de documents dans chaque groupe.

Le schéma des documents de la collection d'observations utilise un tableau de coordonnées pour suivre l'emplacement de l'observation de l'oiseau, le premier élément (0) est la longitude et le deuxième élément (1) est la latitude, `{ location: { coordinates : [x, y ] } }`.

Solution :

Vous pouvez accéder à la collection que vous souhaitez utiliser avec la notation par points :

```
db.sightings
```

Utilisez la méthode « aggregate » pour configurer le pipeline d'agrégation.

La première étape sera **$match** qui acceptera une simple requête pour filtrer des documents spécifiques. Dans ce cas, nous utilisons le champ « species_common » pour rechercher notre oiseau préféré, le merlebleu de l'Est. (Eastern Bluebird) :

```
db.sightings.aggregate([
  {
    $match: {
        species_common: 'Eastern Bluebird'
    }
  }
])
```

Ensuite, nous voulons voir combien d'observations ont eu lieu à chaque endroit, afin de pouvoir nous rendre à l'endroit où nous sommes le plus susceptibles de voir un Merlebleu de l'Est. Pour ce faire, nous utiliserons **$group** et définirons le champ **_id** (obligatoire pour $group) pour être les coordonnées de localisation, et créerons un champ pour le nombre d'observations, en utilisant **$count** pour totaliser le nombre d'enregistrements à chacun des points de coordonnées.

Code résolu :

```
db.sightings.aggregate([
  {
    $match: {
        species_common: 'Eastern Bluebird'
    }
  }, {
    $group: {
        _id: '$location.coordinates',
        number_of_sightings: {
            $count: {}
        }
    }
  }
])
```

Étapes $sort et $limit

Nous souhaitons trouver les oiseaux observés le plus au nord.

Instruction :
Utilisez la collection « **sightings** ».

A. Créez un pipeline d'agrégation. (Vous avez oublié les étapes de commande ou d'agrégation ? Consultez l'astuce ci-dessous !)

B. Utilisez une étape **$sort** pour trier les données du nord au sud. Pour ce faire, utilisez « location.coordinates.1 », en notant que le schéma est `{ location : { coordinates : [x, y] } }`, où la valeur de latitude la plus élevée est la valeur la plus au nord. Le y dans le schéma représente la latitude.

C. Utilisez une étape **$limit** pour limiter le nombre de documents afin qu'uniquement les 4 premiers documents s'affichent.

D. Exécutez votre pipeline d'agrégation et découvrez quels oiseaux ont été aperçus dans le Nord !

E. Une fois cet atelier terminé, comparez votre réponse à la bonne réponse dans la section « Solution et Code résolu ».

Astuces :

N'oubliez pas qu'un pipeline d'agrégation doit commencer par `db.<collection>.aggregate([])` dans le shell MongoDB.

Utilisez **$sort** pour la première étape, qui prend :

- Un champ
- Une valeur de tri, soit :
    - -1, pour indiquer un tri de la valeur la plus élevée à la valeur la plus faible
    - 1, pour indiquer un tri de la valeur la plus basse à la valeur la plus élevée

Utilisez une étape **$limit**, qui prend simplement le nombre de documents que vous souhaiteriez renvoyer en même temps.

Solution :

La première étape sera **$sort** qui accepte un champ à trier, et -1 pour indiquer que le tri doit afficher la valeur la plus élevée jusqu'à la valeur la plus faible. Dans ce cas, « **location.coordinates.1** » sera le champ par lequel nous trierons. Dans un tableau de coordonnées, le premier élément (0) est la longitude et le deuxième élément (1) est la latitude. Les valeurs latitudinales sont plus grandes plus au nord, nous les trions donc de la plus élevée à la plus basse.

```
db.sightings.aggregate([
  {
    $sort: {
        'location.coordinates.1': -1
    }
  }
])
```

Ensuite, utilisez **$limit** pour renvoyer uniquement les 4 observations les plus au nord dans l'ordre nouvellement trié.

Code résolu :

```
db.sightings.aggregate([
  {
    $sort: {
        'location.coordinates.1': -1
    }
  }, {
    $limit: 4
  }
])
```

Étape $projet

Il y a beaucoup de données dans chaque document, mais nous souhaitons renvoyer uniquement une liste de l'heure « date » de l'observation et le nom commun « species_common » de l'oiseau observé.

Instruction :

Utilisez la collection « **sightings** ».

A. Créez un pipeline d'agrégation sur la collection. (Vous avez oublié les étapes de commande ou d'agrégation ? Consultez l'astuce ci-dessous !)

B. Créez une étape **$project** qui nous montre simplement les champs "date" et "species_common". Projetez le champ "_id".

C. Exécutez votre pipeline d'agrégation et consultez la liste des observations !

D. Une fois cet atelier terminé, comparez votre réponse à la bonne réponse dans la section « Solution et Code résolu ».

Astuce :

L'étape d'agrégation sera une étape **$project**, qui nous montre uniquement les champs spécifiés pour faciliter la visualisation des données.

Solution :

L'étape d'agrégation sera **$project** qui montre les champs indiqués dans chaque enregistrement de notre collection. Dans ce cas, *species_common* et *date*. Nous n'avons pas besoin de voir le *_id*, nous pouvons donc projeter ce champ en définissant sa valeur sur 0 à l'étape **$project**.

Code résolu :

```
db.sightings.aggregate([
  {
    $project: {
        _id: 0,
        species_common: 1,
        date: 1
    }
  }
])
```

Étape $set

Vous disposez d'une base de données appelée « **db_bird** » avec une collection d'informations sur les espèces d'oiseaux « **birds** ». À l'avenir, nous ajouterons de nouveaux animaux à la base de données et à la collection. Avant de faire cela, nous devons ajouter le champ « **classe** » et définir ce champ à « **bird** » dans tous les documents existants de la collection.

Instruction :

Utilisez la collection « **birds** » pour cet exercice.

A. Créez un pipeline d'agrégation sur la collection. (Vous avez oublié les étapes de commande ou d'agrégation ? Consultez l'astuce ci-dessous !)

B. Créez un nouveau champ appelé « classe » avec comme valeur « bird » pour chacun de ces oiseaux.

C. Exécutez votre pipeline d'agrégation et vérifiez que le champ « classe » a été ajouté et défini sur « bird ».

D. Une fois cet atelier terminé, comparez votre réponse à la bonne réponse dans la section « Solution et Code résolu ».

Astuces :

N'oubliez pas qu'un pipeline d'agrégation doit commencer par `db.<collection>.aggregate([])` dans le shell MongoDB.
Utilisez l'étape **$set** pour ajouter un nouveau champ avec une valeur à chaque document.

Solution :

Vous pouvez accéder à la collection que vous souhaitez utiliser avec la notation par points :

```
db.birds
```

Utilisez la méthode « aggregate » pour configurer le pipeline d'agrégation.

Nous utilisons l'étape **$set** pour ajouter un nouveau champ aux documents de la collection. Nous ajoutons un champ appelé "classe" et définissons la valeur de ce champ sur "oiseau".

Code résolu :

```
db.birds.aggregate([
  {
    $set: {
      'class': 'bird'
    }
  }
])
```

Étape $count

Nous souhaitons connaître le nombre total d'observations de Merlebleus de l'Est en 2022.

Instruction :

Utilisez la collection « **sightings** ».

A. Créez un pipeline d'agrégation sur la collection. (Vous avez oublié les étapes de commande ou d'agrégation ? Consultez l'astuce ci-dessous !)

B. Créez une étape qui recherche les correspondances où « species_common » est « Eastern Bluebird » et où le champ de date est une valeur comprise entre le 1er janvier 2022 0h00 et le 1er janvier 2023 0h00.

C. Créez une étape pour compter le nombre d'observations que cela comprend avec $count.

D. Exécutez votre pipeline d'agrégation et voyez combien d'observations de merlebleus de l'Est ont eu lieu en 2022.

E. Une fois cet atelier terminé, comparez votre réponse à la bonne réponse dans la section « Solution et Code résolu ».

Astuce :

N'oubliez pas qu'un pipeline d'agrégation doit commencer par `db.<collection>.aggregate([])` dans le shell MongoDB.

- Utilisez l'étape **$match** pour rechercher des documents spécifiques. $match prend une requête, où vous devriez :
  - Spécifiez les dates après le premier jour de 2022 à l'aide de $gt : ISODate("2022-01-01T00:00:00.0Z")
  - Spécifiez les dates avant le premier jour de 2023 en utilisant $lt: ISODate("2023-01-01T00:00:00.0Z")
  - Spécifiez que « species_common » est « Eastern Bluebird ».
- Utilisez l'étape **$count** pour afficher le nombre d'observations de Bluebird dans un champ appelé "Bluebird_sightings_2022".

Solution :

La première étape d'agrégation sera **$match** qui trouvera les documents dont le champ date contient une valeur supérieure à ISODate("2022-01-01T00:00:00.0Z") et inférieure à ISODate("2023-01-01T00:00: 00.0Z"). L'étape **$match** trouvera également les documents dont le champ « species_common » est "Eastern Bluebird".

```
db.sightings.aggregate([
  {
    $match: {
        date: {
            $gt: ISODate('2022-01-01T00:00:00.0Z'),
            $lt: ISODate('2023-01-01T00:00:00.0Z')
        },
        species_common: 'Eastern Bluebird'
    }
  }
])
```

La dernière étape que nous incluons est **$count**. L'étape **$count** crée un champ appelé *bluebird_sightings_2022*, qui inclut la valeur du nombre de documents actuellement dans le pipeline.

Code résolu :

```
db.sightings.aggregate([
{
  $match: {
    date: {
      $gt: ISODate('2022-01-01T00:00:00.000Z'),
      $lt: ISODate('2023-01-01T00:00:00.000Z')
    },
    species_common: 'Eastern Bluebird'
  }
}, {
  $count: 'bluebird_sightings_2022'
}
])
```