**Création de notre première collection**

MongoDB est structuré autour de 3 éléments : le document, la collection et la database.

* Une database contient une ou plusieurs collections.
* Une collection regroupe un ensemble de documents.

show databases // Pour visualiser les différentes databases

use database. // Pour aller sur une database particulière ou la créer si elle n’existe pas

show collections // Pour visualiser les collections de la database

La commande db.nomDeMaCollection.insert permet de créer notre premier document. Cela crée automatiquement la collection si elle n’existe pas. La commande db.nomDeMaCollection.find permet de retrouver tous les documents d’une collection.

**Insérer des documents dans MongoDB**

On va commencer par insérer un document grâce à la fonction insertOne().

db.stagiaire.insertOne**({**"nom"**:**"BYADI"**})**

**{**

"acknowledged" **:** true**,**

"insertedId" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**)**

**}**

Quand on regarde la collection, on peut constater que [**MongoDB**](https://www.mongodb.com/fr), a bien inséré le document. Il a créé également une clé, nommée *id*

**db**.**stagiaire**.**find()**

**{**

"\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),**"nom" **:** "BYADI"

**}**

Si l’on refait la même insertion, on obtient le résultat suivant:

**db**.**stagiaire**.**find()**

**{**

"\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),**"nom" **:** "BYADI"

**}**

**{**

"\_id" **:** ObjectId**(**"6370da5bd9b9de9229c6938e"**),**"nom" **:** "BYADI"

**}**

MongoDB a inséré un deuxième document. Le système n’a pas détecté de doublon: il a généré une clé différente pour les deux enregistrements.

On peut choisir nous même la clé *id.*

db.stagiaire.insertOne**({**"\_id"**:**1**,**"nom"**:**"BYADI"**})**

**db**.**stagiaire**.**find()**

**{**"\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),**"nom" **:** "BYADI"**}**

**{**"\_id" **:** ObjectId**(**"6370da5bd9b9de9229c6938e"**),**"nom" **:** "BYADI"**}**

**{**"\_id" **:** NumberInt**(**1**),** "nom" **:** "KHADIRI"**}**

Si j’insère un document avec le même *id*, cela sera sans effet car la clé doit rester unique.

Je peux également insérer un document avec une structure différente dans la même collection sans que cela ne pose de problème. (Par exemple avec un champ âge supplémentaire).

db.stagiaire.insertOne**({**"nom"**:**"TAGHZOUTI"**,**"age"**:**"21"**})**

**{**"acknowledged"**:** true**,**"insertedId" **:** ObjectId**(**"6370e7f9d9b9de9229c6938f"**)**

**}**

db.stagiaire.find**()**

**{**"\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),**"nom" **:** "BYADI"**}**

**{**"\_id" **:** ObjectId**(**"6370da5bd9b9de9229c6938e"**),**"nom" **:** "BYADI"**}**

**{**"\_id" **:** NumberInt**(**1**),**"nom" **:** "KHADIRI"**}**

**{**"\_id" **:** ObjectId**(**"6370e7f9d9b9de9229c6938f"**),**"nom" **:** "TAGHZOUTI"**,**"age" **:** "21"**}**

MongoDB fait partie de la grande famille des [**bases NoSQL**](https://www.journaldunet.com/solutions/dsi/1194284-base-nosql-laquelle-choisir-pour-quels-besoins/). Nous n’avons pas eu besoin de déclarer de structure pour notre collection avant d’insérer des documents. Cette souplesse permet à MongoDB de s’adapter très facilement aux évolutions d’applications.

Il est possible d’insérer plusieurs documents en même temps. On va passer en argument un tableau de documents à la fonction insert().

db.stgiaire.insert**([{**"name"**:**"OUBOUAZZA"**},{**"name"**:**"OUJILALI"**,**"ville"**:[**"Casa"**,**"Rabat"**]}])**

**{**

"acknowledged" **:** true**,**

"insertedIds" **:** **{**

"0" **:** ObjectId**(**"6370ebc9d9b9de9229c69390"**),**

"1" **:** ObjectId**(**"6370ebc9d9b9de9229c69391"**)**

**}**

**}**

**{**"\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),**"nom" **:** "BYADI"**}**

**{**"\_id" **:** ObjectId**(**"6370da5bd9b9de9229c6938e"**),**"nom" **:** "BYADI"**}**

**{** "\_id" **:** NumberInt**(**1**),**"nom" **:** "KHADIRI"**}**

**{** "\_id" **:** ObjectId**(**"6370e7f9d9b9de9229c6938f"**),**"nom" **:** "TAGHZOUTI"**,**"age" **:** "21"**}**

**{** "\_id" **:** ObjectId**(**"6370ed03d9b9de9229c69394"**),**"nom" **:** "OUBOUAZZA"**}**

**{**"\_id" **:** ObjectId**(**"6370ed03d9b9de9229c69395"**),**"nom" **:** "OUJILALI"**,**"ville" **:** **[**"Casa"**,**"Rabat"**]}**

**Supprimer des documents dans MongoDB**

Pour supprimer une collection, on utilise la fonction drop().  
On peut vérifier ensuite que la collection ne contient plus aucun document.

db.stagiaire.drop**()**

**true**

db.stagiaire.find**()**

Si on ne souhaite pas supprimer la collection mais uniquement certains documents, on va utiliser la fonction deleteOne() ou deleteMany().

Comment choisir les documents à supprimer ? Par une condition que l’on va passer comme argument de notre requête. Nous allons peupler de nouveau notre collection stagiaire par les mêmes document comme précédemment .

Voilà les documents présents :

db.stagiaire.find**()**

**{**"\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),**"nom" **:** "BYADI"**}**

**{**"\_id" **:** ObjectId**(**"6370da5bd9b9de9229c6938e"**),**"nom" **:** "BYADI"**}**

**{** "\_id" **:** NumberInt**(**1**),**"nom" **:** "KHADIRI"**}**

**{** "\_id" **:** ObjectId**(**"6370e7f9d9b9de9229c6938f"**),**"nom" **:** "TAGHZOUTI"**,**"age" **:** "21"**}**

**{** "\_id" **:** ObjectId**(**"6370ed03d9b9de9229c69394"**),**"nom" **:** "OUBOUAZZA"**}**

**{**"\_id" **:** ObjectId**(**"6370ed03d9b9de9229c69395"**),**"nom" **:** "OUJILALI"**,**"ville" **:** **[**"Casa"**,**"Rabat"**]}**

Imaginons que nous voulons supprimer le deuxième document portant le nom BYADI.  
La fonction deleteOne()jouera ce rôle.

db.stagiaire.deleteOne**({**\_id**:**ObjectId**(**"6370da5bd9b9de9229c6938e"**)})**

**{**"acknowledged" **:** true**,** "deletedCount" **:** 1.0**}**

db.stagiaire.find**()**

**{** "\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),** "nom" **:** "BYADI"**}**

**{** "\_id" **:** NumberInt**(**1**),** "nom" **:** "KHADIRI"**}**

**{**"\_id" **:** ObjectId**(**"6370e7f9d9b9de9229c6938f"**),**"nom" **:** "TAGHZOUTI"**,**"age" **:** "21"**}**

**{** "\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69396"**),** "nom" **:** "OUBOUAZZA"**}**

**{**"\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69397"**),**"nom" **:** "OUJILALI"**,**"ville" **:** **[**

"Casa"**,**"Rabat"**]}**

**Mettre à jour des documents dans MongoDB**

MongoDB met à disposition la fonction update avec différents opérateurs en fonction du type de mise à jour souhaité. La fonction update prend deux arguments obligatoires :

* un document représentant la condition de recherche des documents de la collection
* un document représentant la mise à jour souhaitée

**Ajouter ou remplacer un champ existant avec $set**

**db**.**stagiaire**.**find({**"nom"**:**"OUBOUAZZA"**})**

**{** "\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69396"**),** "nom" **:** "OUBOUAZZA"**}**

db.stagiaire.update**({**"nom"**:**"OUBOUAZZA"**},{**$set**:{**"ville"**:**"Khemisset"**}})**

db.stagiaire.find**()**

**{** "\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),** "nom" **:** "BYADI"**}**

**{** "\_id" **:** NumberInt**(**1**),** "nom" **:** "KHADIRI"**}**

**{** "\_id" **:** ObjectId**(**"6370e7f9d9b9de9229c6938f"**),** "nom" **:** "TAGHZOUTI"**,**

"age" **:** "21"**}**

**{** "\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69396"**),** "nom" **:** "OUBOUAZZA"**,** "ville" **:** "Khemisset"**}**

**{** "\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69397"**),** "nom" **:** "OUJILALI"**,**

"ville" **:** **[**"Casa"**,**"Rabat"**]}**

Dans cet exemple, on a simplement rajouté un champ ville dans le document de OUBOUAZZA.

Rajoutons un document dans notre collection :

db.stagiaire.insertOne**({**"nom"**:**"BYADI"**,**"age"**:**60**})**

db.stagiaire.find**()**

**{** "\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),** "nom" **:** "BYADI"**}**

**{** "\_id" **:** NumberInt**(**1**),** "nom" **:** "KHADIRI"**}**

**{**"\_id" **:** ObjectId**(**"6370e7f9d9b9de9229c6938f"**),**"nom" **:** "TAGHZOUTI"**,**"age" **:** "21"**}**

**{** "\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69396"**),** "nom" **:** "OUBOUAZZA"**,** "ville" **:** "Khemisset"**}**

**{** "\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69397"**),**"nom" **:** "OUJILALI"**,**

"ville" **:** **[** "Casa"**,** "Rabat" **]}**

**{**"\_id" **:** ObjectId**(**"6370fc9dd9b9de9229c69398"**),**"nom"**:**"BYADI"**,**"age"**:**NumberInt**(**20**)}**

Faisons une mise à jour sur les documents portant le nom BYADI en ajoutant ville.

db.stagiaire.update**({**"nom"**:**"BYADI"**},{**$set**:{**"ville"**:**"Fes"**}})**

db.stagiaire.find**()**

**{** "\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),** "nom" **:** "BYADI", "ville" **:** "Fes"**}**

**{** "\_id" **:** NumberInt**(**1**),** "nom" **:** "KHADIRI"**}**

**{**"\_id" **:** ObjectId**(**"6370e7f9d9b9de9229c6938f"**),**"nom" **:** "TAGHZOUTI"**,**"age" **:** "21"**}**

**{** "\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69396"**),** "nom" **:** "OUBOUAZZA"**,** "ville" **:** "Khemisset"**}**

**{** "\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69397"**),**"nom" **:** "OUJILALI"**,**

"ville" **:** **[** "Casa"**,** "Rabat" **]}**

**{**"\_id" **:** ObjectId**(**"6370fc9dd9b9de9229c69398"**),**"nom"**:**"BYADI"**,**"age"**:**NumberInt**(**20**)}**

Le résultat de la commande montre qu’un seul document a été mis à jour. La commande find montre que le premier document avec le nom « BYADI » a bien été modifié mais pas le second. C’est une protection dans MongoDB pour empêcher par défaut la mise à jour sur de multiples documents. Si c’est ce que l’on souhaite, il faut rajouter une instruction multi: true dans notre fonction update.

db.stagiaire.update**({**"nom"**:**"BYADI"**},{**$set**:{**"ville"**:**"Fes"**}},{**multi**:**true**})**

db.stagiaire.find**()**

**{** "\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),** "nom" **:** "BYADI", "ville" **:** "Fes"**}**

**{** "\_id" **:** NumberInt**(**1**),** "nom" **:** "KHADIRI"**}**

**{**"\_id" **:** ObjectId**(**"6370e7f9d9b9de9229c6938f"**),**"nom" **:** "TAGHZOUTI"**,**"age" **:** "21"**}**

**{** "\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69396"**),** "nom" **:** "OUBOUAZZA"**,** "ville" **:** "Khemisset"**}**

**{** "\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69397"**),**"nom" **:** "OUJILALI"**,**

"ville" **:** **[** "Casa"**,** "Rabat" **]}**

**{**"\_id" **:** ObjectId**(**"6370fc9dd9b9de9229c69398"**),**"nom"**:**"BYADI"**,**"age"**:**NumberInt**(**20**),** "ville" **:** "Fes"**}**

**Incrémenter un champ numérique existant avec $inc**

Dans certains cas, on peut avoir envie de faire une mise à jour en se basant sur la valeur actuelle du champ. $inc permet de rajouter une valeur à une donnée numérique. Cette valeur peut être positive ou négative.  
Si on souhaite par exemple incrémenter l’âge de TAGHZOUTI, on peux exécuter la commande suivante :

db.stagiaire.update**({**"nom"**:**"TAGHZOUTI"**},{**$inc**:{**"age"**:**2**}})**

db.stagiaire.find**()**

**{**"\_id" **:** ObjectId**(**"6370d733d9b9de9229c6938d"**),** "nom" **:** "BYADI", "ville" **:** "Fes"**}**

**{**"\_id" **:** NumberInt**(**1**),** "nom" **:** "KHADIRI"**}**

**{**"\_id" **:** ObjectId**(**"6370e7f9d9b9de9229c6938f"**),**"nom" **:** "TAGHZOUTI"**,**"age" **:** "23"**}**

**{**"\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69396"**),** "nom" **:** "OUBOUAZZA"**,** "ville" **:** "Khemisset"**}**

**{**"\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69397"**),**"nom" **:** "OUJILALI"**,**

"ville" **:** **[**"Casa"**,**"Rabat"**]}**

**{**"\_id" **:** ObjectId**(**"6370fc9dd9b9de9229c69398"**),**"nom"**:**"BYADI"**,**"age"**:**NumberInt**(**20**),** "ville" **:** "Fes"**}**

**Mettre à jour un tableau avec $push ou $pull**

Si on utilise $set sur un tableau, va remplacer le tableau existant par un nouvel élément. Comment mettre à jour le tableau sans écraser les données existantes ?  
L’opérateur **$push** permet de rajouter un nouvel élément à un tableau.

db.stagiaire.update**({**"nom"**:**"OUJILALI"**},{**$push**:{**"ville"**:**"Fes"**}})**

db.stagiaire.find**()**

**.**

**.**

**.**

**{**

"\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69397"**),**

"nom" **:** "OUJILALI"**,**

"ville" **:** **[**

"Casa"**,**

"Rabat"**,**

"Fes"

**]**

**}  
.  
.  
.**

Dans cet exemple, on a rajouté « Fes » dans le tableau contenant déjà «Casa » et « Rabat ». Il faut noter que si « Fes » était déjà présent, l’élément aurait été quand même inséré. Si l’on ne souhaite pas de doublon, il existe l’opérateur **$addToSet** qui assure cette fonction.

Pour supprimer un élément, on peut utiliser **$pull**. Ainsi, si on souhaite supprimer la ville de « Fes », on lance la commande suivante :

db.stagiaire.update**({**"nom"**:**"OUJILALI"**},{**$pull**:{**"ville"**:**"Fes"**}})**

db.stagiaire.find**()**

**.**

**.**

**.**

**{**

"\_id" **:** ObjectId**(**"6370ef11d9b9de9229c69397"**),**

"nom" **:** "OUJILALI"**,**

"ville" **:** **[**

"Casa"**,**

"Rabat"**,**

"Meknes"

**]**

**}**

### Projection

La projection permet de sélectionner les informations à renvoyer. Si, par exemple, on s’intéresse uniquement au nom du stagiaire, à son age et aux noms des villes, on va limiter les informations retournées en précisant les champs souhaités dans un document JSON. Et, également passer ce document comme deuxième argument de la recherche find.

**db**.**stastagiaire**.**find({**"ville"**:**"Fes"**},{**"nom"**:**1**,**"age"**:**1**,**"\_id"**:**0**})**

### Affiche les noms et ages des stagiaire de Fes. Pour conserver un champ, il suffit de le préciser dans ce document et de lui affecter la valeur 1.

### Recherche dans un tableau

Les documents contiennent des champs simples comme le nom ou l’age mais aussi des tableaux pour stocker le nom des villes. Comment faire alors des recherches dans un tableau ? Tout simplement de la même manière que pour un champ simple :

**db**.**stastagiaire**.**find({**"ville"**:**"Fes"**},{**"nom"**:**1**,**"age"**:**1**,** "**ville**" : 1,"\_id"**:**0**})**.pretty()

On peut combiner plusieurs filtres.

**db**.**stastagiaire**.**find({**"ville"**:**"Fes"**,**"age"**:**23**},{**"nom"**:**1**,**"age"**:**1**,**"ville"**:**1**,**"\_id"**:**0**}).**

Affiche les noms, âges et viles des stagiaires de Fes âgés de 23 ans

Comment faire si on veut filtrer les stagiaires avec Khemisset **ou** Fes ? Pour cela, on utilise alors l’opérateur **$in**.

**db**.**stastagiaire**.**find({**"ville"**:{**$in**:[**"Khemisset"**,**"Fes"**]}},{**"nom"**:**1**,**"age"**:**1**,**"ville"**:**1**,**"\_id"**:**0**})**

Pour avoir uniquement les stagiaires avec Casa **et** Rabat, il existe l’opérateur **$all**.

**db**.**stastagiaire**.**find({**"ville"**:{**$all**:[**"Casa"**,**"Rabat"**]}},{**"nom"**:**1**,**"age"**:**1**,**"ville"**:**1**,**"\_id"**:**0**})**

Les documents **MongoDB** peuvent contenir des documents imbriqués. Dans notre collection stagiaire, supposons que chacun des documents a un sous document filiere :

**{**"\_id" **:** ObjectId**(**"63749aa94d60b886a468a0d3"**),**"nom" **:** "BYADI"**,**"ville" **:** "Fes"**,**"filiere" **:** **{**"id" **:** "DevOWFS"**,**"nom" **:** "Devlop. FS"**}}**

Peut-on faire des recherches sur des éléments de ce document ? Oui, bien entendu ! Il suffit pour cela d’utiliser la notation nomDocument.nomChamp.

On fera attention de bien mettre cet élément entre quottes dans notre requête.

db.stastagiaire.find**({})**

**{** "\_id" **:** ObjectId**(**"63749aa94d60b886a468a0d3"**),** "nom" **:** "BYADI"**,** "ville" **:** "Fes"**,** "filiere" **:** **{**"id" **:** "DevOWFS"**,** "nom" **:** "Devlop. FS"**}}**

**{** "\_id" **:** NumberInt**(**1**),** "nom" **:** "BYADI"**,** "ville" **:** "Fes"**}**

**{**

"\_id" **:** ObjectId**(**"63749d4f4d60b886a468a0d5"**),**

"nom" **:** "TOUGHZAOUI"**,**

"age" **:** NumberInt**(**23**),**

"ville" **:** "Fes"**,**

"filiere" **:** **{** "id" **:** "DevOWFS"**,** "nom" **:** "Devlop. FS" **}**

**}**

**{**

"\_id" **:** ObjectId**(**"6374a2ad4d60b886a468a0d6"**),**

"nom" **:** "OUBOUAZA"**,**

"ville" **:** "Khemisset"**,**

"filiere" **:** **{** "id" **:** "DevOWFS"**,** "nom" **:** "Devlop. FS" **}**

**}**

**{**

"\_id" **:** ObjectId**(**"6374a2ad4d60b886a468a0d7"**),**

"nom" **:** "BELAZIZ"**,**

"ville" **:** **[**

"Rabat"**,**

"Casa"**,**

"Fes"

**],**

"filiere" **:** **{** "id" **:** "DevOAM"**,** "nom" **:** "Devlop. AM" **}**

**}**

**{**

"\_id" **:** ObjectId**(**"6374b3b74d60b886a468a0d8"**),**

"nom" **:** "BYADI"**,**

"age" **:** NumberInt**(**60**),**

"ville" **:** "Fes"**,**

"filiere" **:** **{**

"id" **:** "DevOAM"**,**

"nom" **:** "Devlop. AM"

**}**

**}**

**db**.**stastagiaire**.**find({**"filiere.id"**:**"DevOWFS"**},{**"nom"**:**1**,**"age"**:**1**})**

**{**

"\_id" **:** ObjectId**(**"63749aa94d60b886a468a0d3"**),**"nom" **:** "BYADI"**}**

**{**

"\_id" **:** ObjectId**(**"63749d4f4d60b886a468a0d5"**),**"nom" **:** "TOUGHZAOUI"**,**"age" **:** NumberInt**(**23**)}**

**{**

"\_id" **:** ObjectId**(**"6374a2ad4d60b886a468a0d6"**),**

"nom" **:** "OUBOUAZA"

**}**

Dans toutes nos recherches, nous avons utilisé des conditions d’égalité avec l’opérateur :.  
Il n’est pas possible (contrainte JSON ?) d’utiliser les signes habituels (>, >=, <, <=, !=). **MongoDB** propose à la place des opérateurs dédiés ($gt, $gte, $lt, $lte, $ne). Prenons un exemple et faisons une recherche sur tous les stagiaires ayant au moins 23 ans.

**db**.**stastagiaire**.**find({**"age"**:{**$gte**:**23**}},{**"nom"**:**1**,**"age"**:**1**,**"filiere.id"**:**1**})**

Cherchons les stagiaires qui ont un champs âge (operateur $exists)

**db**.**stastagiaire**.**find({**"age"**:{**$exists**:**true**}},{**"nom"**:**1**,**"age"**:**1**,**"filiere.id"**:**1**})**

**{**

"\_id" **:** ObjectId**(**"63749d4f4d60b886a468a0d5"**),**

"nom" **:** "TOUGHZAOUI"**,**

"age" **:** NumberInt**(**23**),**

"filiere" **:** **{**

"id" **:** "DevOWFS"

**}**

**}**

**{**

"\_id" **:** ObjectId**(**"6374b3b74d60b886a468a0d8"**),**

"nom" **:** "BYADI"**,**

"age" **:** NumberInt**(**60**),**

"filiere" **:** **{**

"id" **:** "DevOAM"

**}**

**}**

Cherchons les stagiaires qui n’ont pas de champs âge (operateur $exists)

**db**.**stastagiaire**.**find({**"age"**:{**$exists**:**false**}},{**"nom"**:**1**,**"age"**:**1**,**"filiere.id"**:**1**})**

**{**

"\_id"**:**ObjectId**(**"63749aa94d60b886a468a0d3"**),**"nom"**:**"BYADI"**,**"filiere"**:{**"id"**:**"DevOWFS" **}**

**}**

**{**

"\_id" **:** NumberInt**(**1**),** "nom" **:** "BYADI"**}**

**{**

"\_id" **:** ObjectId**(**"6374a2ad4d60b886a468a0d6"**),** "nom" **:** "OUBOUAZA"**,**"filiere" **:** **{**

"id" **:** "DevOWFS"**}}**

**{**

"\_id" **:** ObjectId**(**"6374a2ad4d60b886a468a0d7"**),**"nom" **:** "BELAZIZ"**,**"filiere" **:** **{**"id"**:**"DevOAM"**}}**

Quand les recherches renvoient beaucoup de documents, il peut être utile de les trier.  
On dispose pour cela de la fonction sort qui va permettre de trier les résultats sur un champ par ordre croissant ou décroissant.

L’usage est très simple. Nous allons donc reprendre notre requête précédente et trier les résultats par ordre décroissant sur les noms

**db**.**stastagiaire**.**find({**"age"**:{**$exists**:**false**}},{**"nom"**:**1**,**"age"**:**1**,**"filiere.id"**:**1**})**.**sort({**"nom"**:**-1**})**

remplacer -1 par 1 pour avoir un tri croissant

Agrégation simple

Compter le nombre de stagiaire par filière :

**db**.**stastagiaire**.**aggregate({**$group**:{**\_id**:**"$filiere.id"**,**Nbr**:{**$count**:{}}}})**

Calculer la moyenne d’âge par filière :

**db**.**stastagiaire**.**aggregate({**$group**:{**\_id**:**"$filiere.id"**,**MoyenneAge**:{**$avg**:**"$age"**}}})**

Calculer la somme d’âge par filière :

**db**.**stastagiaire**.**aggregate({**$group**:{**\_id**:**"$filiere.id"**,**SommeAge**:{**$sum**:**"$age"**}}})**

Calculer la Moyenne d’âge de tous les stagiaires :

db.stastagiaire.aggregate**({**$group**:{**\_id**:**"Tout"**,**MoyenneAge**:{**$avg**:**"$age"**}}})**

Calculer la moyenne d’âge par filière  en éliminant les document qui n’ont pas d’âge:

db.stastagiaire.aggregate**({**$match**:{**"age"**:{**$exists**:**true**}}},{**$group**:{**\_id**:**"$filiere.id"**,**MoyenneAge**:{**$avg**:**"$age"**}}})**

Calculer le nombre de stagiaires par filière trié par ordre décroissant

**db**.**stastagiaire**.**aggregate({**$match**:{**"age"**:{**$exists**:**true**}}},{**$group**:{**\_id**:**"$filiere.id"**,**Nbr**:{**$sum**:**1**}}},{**$sort**:{**Nbr**:**-1**}})**