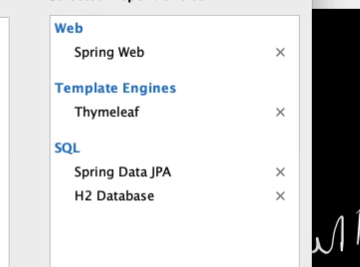
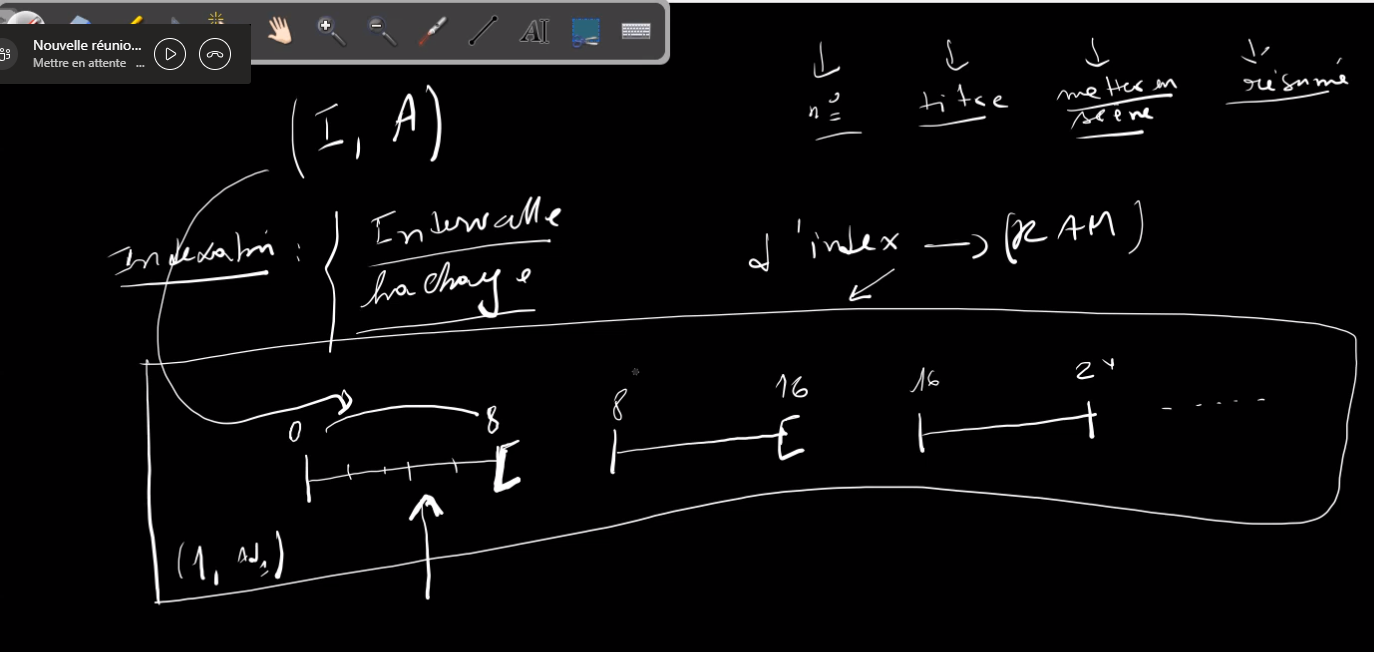
Intellij

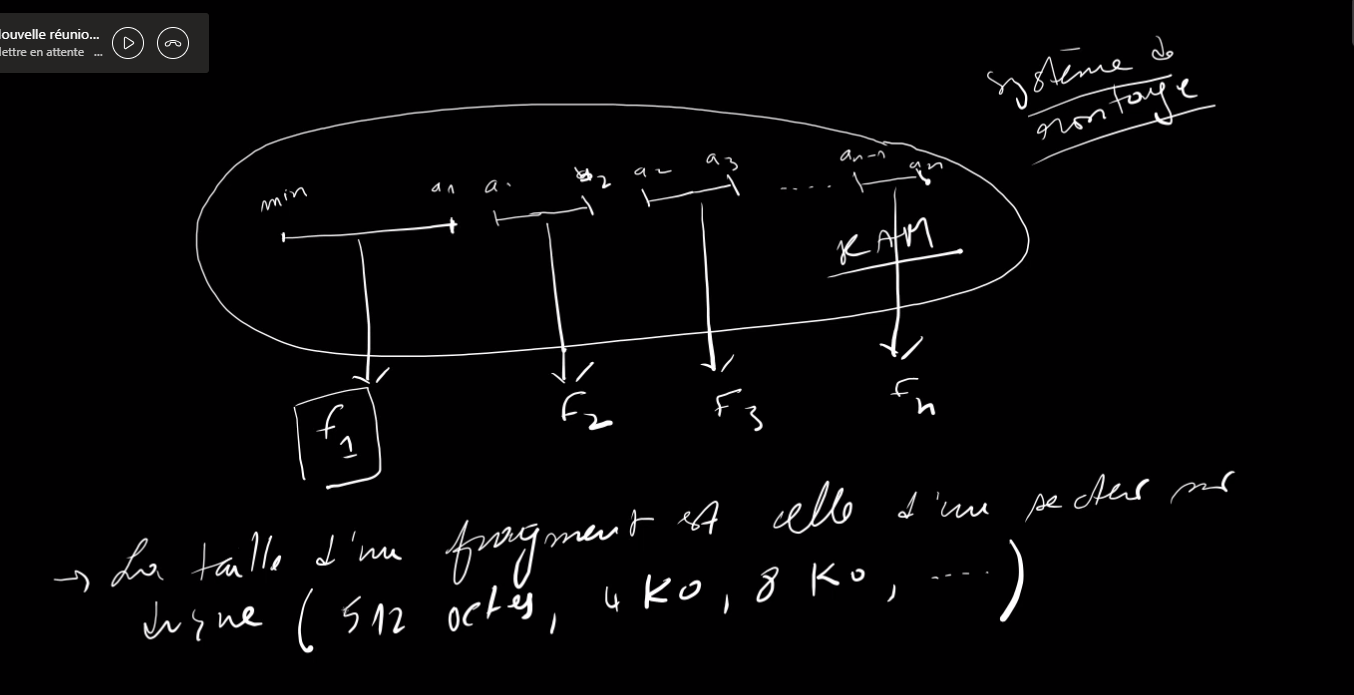


SQL :



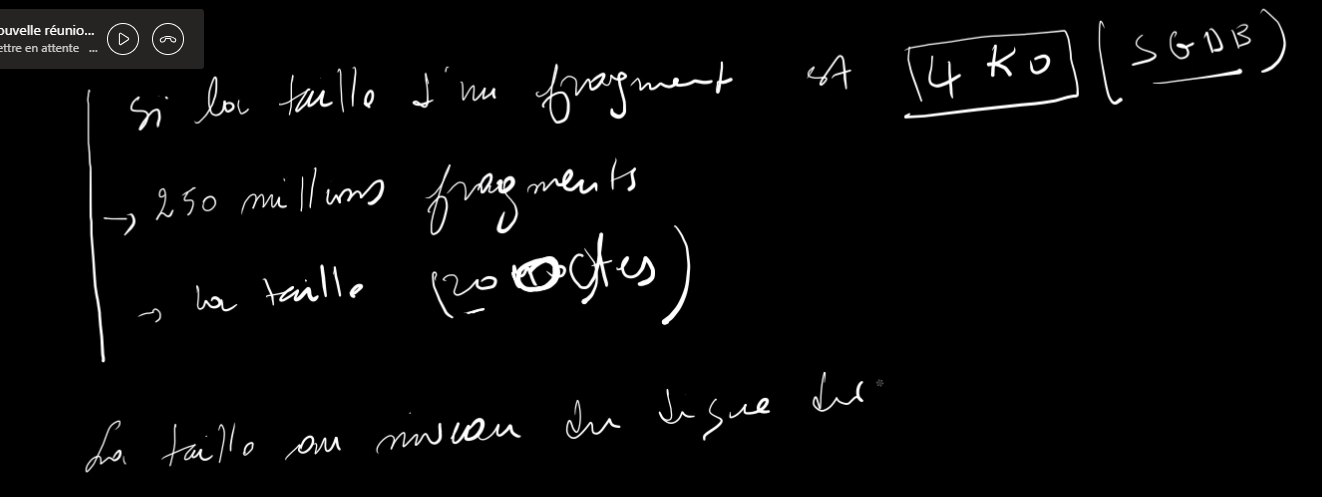
Indexation par hachage pour retrouver rapidement la valeur dans la base (disque).

Les intervalles c’est dans la RAM et ça renvoi à un fragment dans le disque dur.

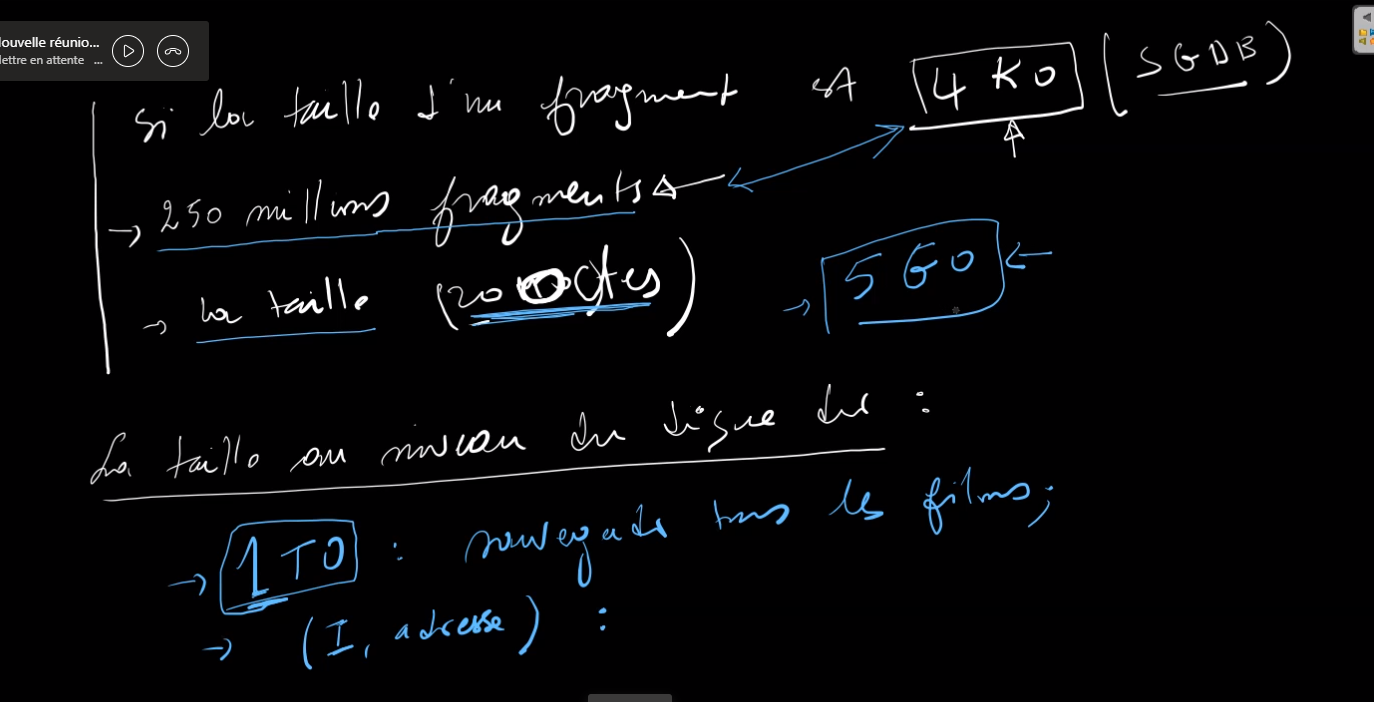


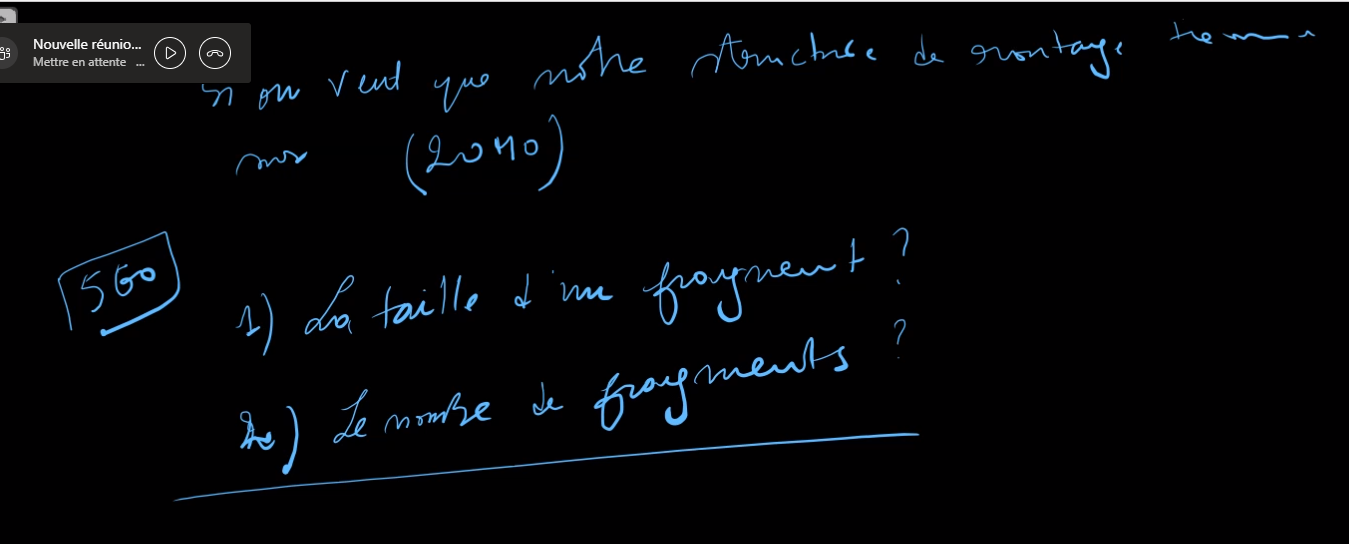
Un secteur disque = 512 octets

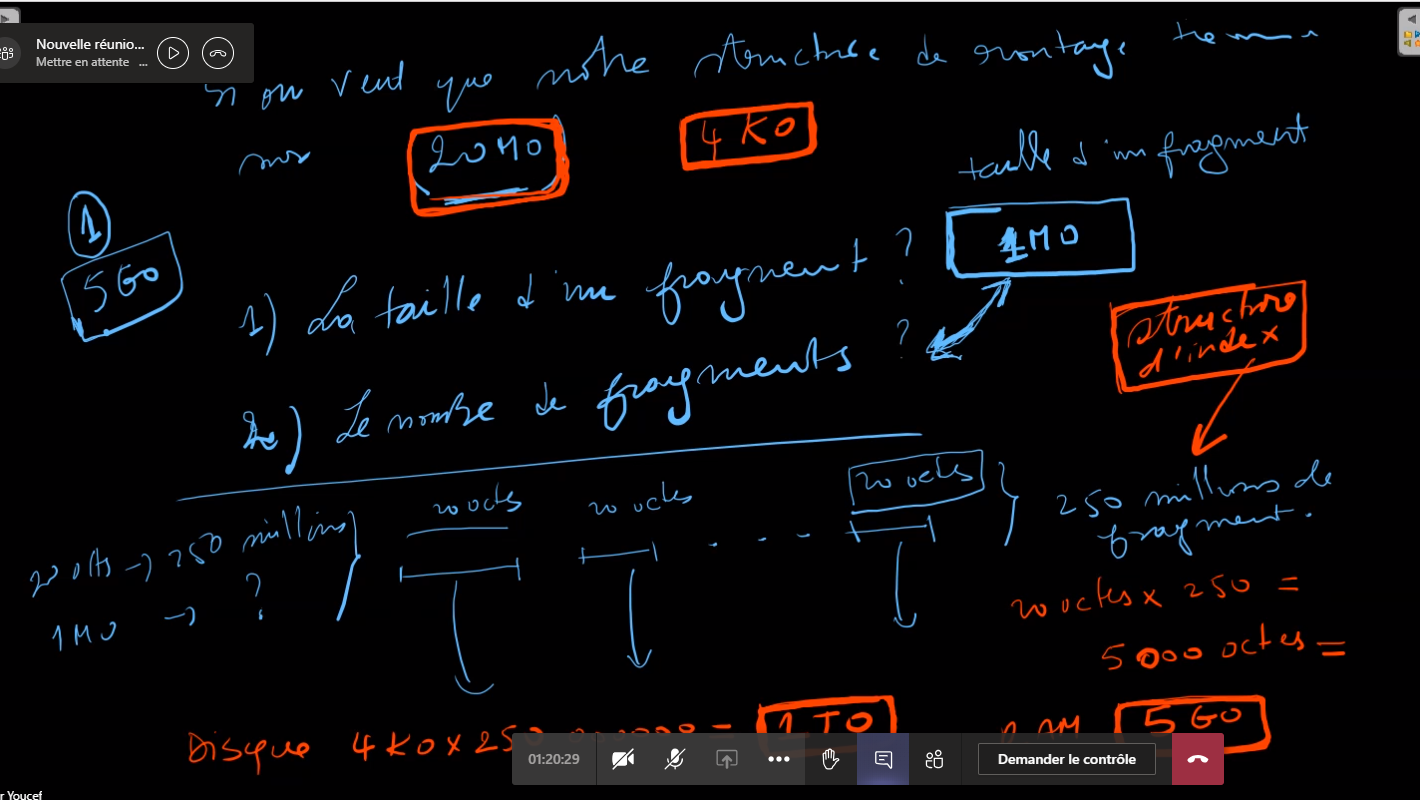
1 fragment = 4 Ko (choix typique pour la BDD)



L’indexation permet donc d’accéder aux données sans parcourir tout le disque. 🡪 (Identifiant, Adresse)



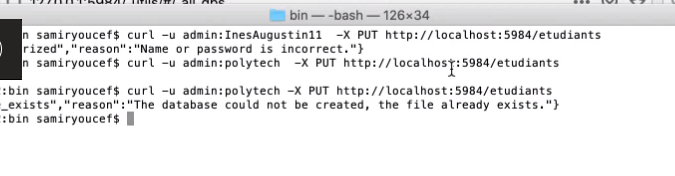




CouchDB :



Créer une base :

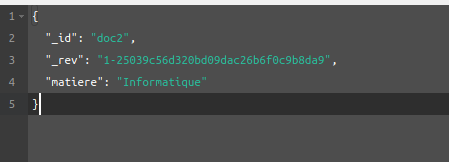


Ajout d’un document dans la base etudiants :

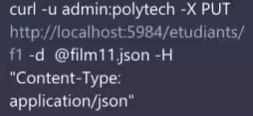


Ici « prenom » est une clé

* Dans couchDB



Ajouter un doc avec un fichier json directement :



Ajouter un base en séparant chaque document :

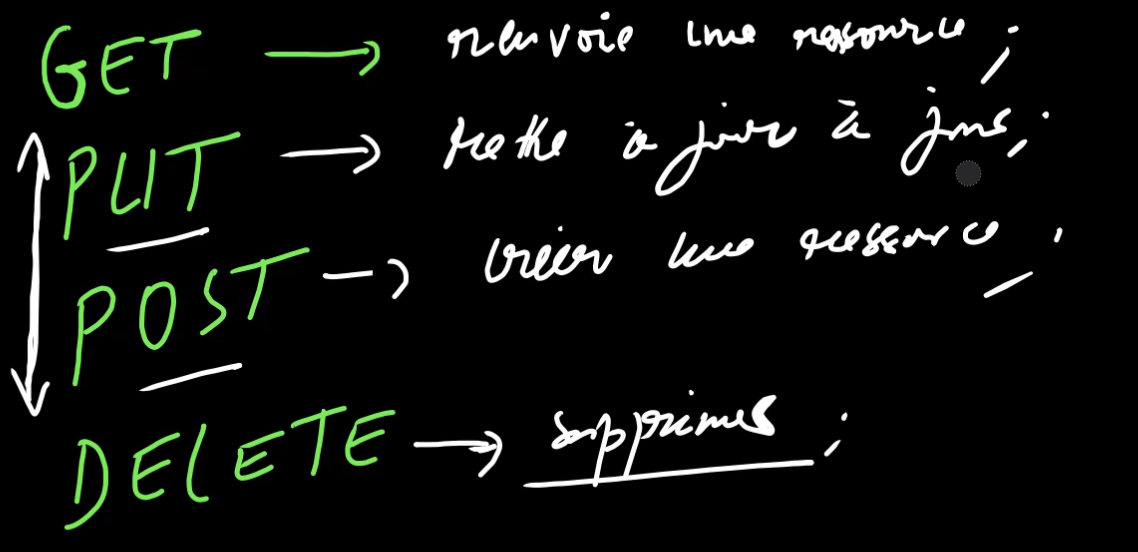


On utilise « \_bulk\_docs » pour séparer les films

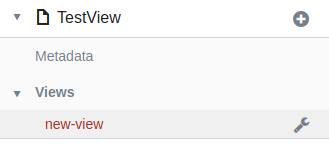
**MAP.REDUCE**

* Permet de traiter en parallèle une requête sur des grandes quantités de données (utilisation de plusieurs machines)
* Ici on créer une nouvelle vue avec un code MAP qui permet de mapper les données, je pense que cela permet de créer un fichier json représentant ce que l’on cherche et dans lequel on peut faire des recherches plus rapidement que si l’on parcourait la bdd entièrement

Remarque :



Dans couchDB : création d’une vue





Notion de clé intermédiaire

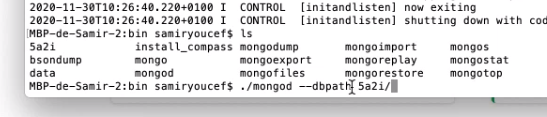
Exemple :

(1921,1),(1927,1),….,(1921,1) Ici on notre clé intermédiaire (liste de toutes les dates)

* REDUCE (\_sum) permet de faire la somme de film pour chaque date

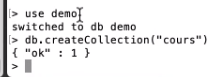
MongoDB :

Serveur :



Client :

./mongo



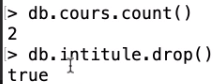






Import d’une BDD :





On peut faire des requêtes sur studio3T :



* Renvoie tous les documents

findOne({}) pour en trouver qu’un seul

*03/12/2020*

**Filtre et projection :**

on peut mettre entre {} un doc json en tant que filtre



on peut ajouter .count() à la fin pour compter les documents



* Ici le / permet de chercher une chaine de caractère dans le champs nom et le /i correspond à « insensible à la casse »

La méthode find peut prendre un ou plusieurs paramètres :

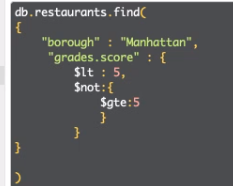


Ici, on renvoi que le nom. Mais il renvoie l’ID, qu’on peut masquer avec « \_id » :0

**Filtrage avec des comparateurs :**

Renvoyer tous les restaurants qui se trouvent dans un quartier ayant un score inférieur à une certaine valeur :



Uniquement des valeurs < 5

Gt = greater than (> à)

Lt = less than (< à)

Or ou and :



Renvoyer les grade : A et score inférieur à 5 :



On obtient alors les restaurants ayant au moins un A (grade) et ayant au moins un score < 5

Renvoyer tous les restaurants dont le dernier score est B :



Afficher toutes les cuisines :



**Aggregation :**

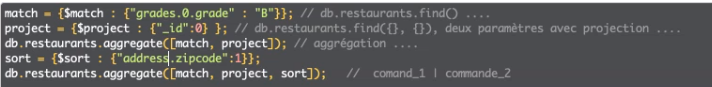


Ici c’est une commande équivalente à Find() (commande précédente) mais c’est utilisé quand on a plusieurs bases (ou tables …)

**Sauvegarde** des résultats intermédiaires :



* Permet de réutiliser les variables, ici match et project.



Ici, on peut rajouter sort pour trier par code postal.

Question : Renvoyer le nombre de restaurant avec comme dernier grade B en utilisant $group.

Sans $group :



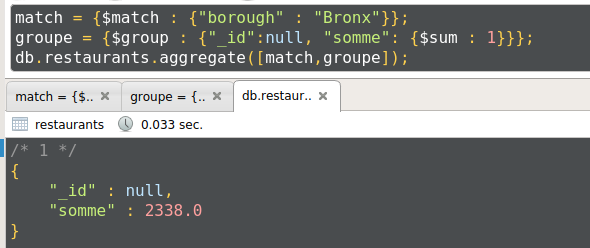
Ou simplement :



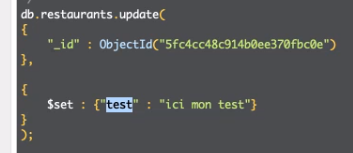
Avec $group :



Question : nombre total de restaurant dans un quartier

🡪2338 restaurants dans le Bronx

**Mise à jour :**



Cela permet de changer la valeur d’un attribut. Pour cela on doit sélectionner la clé primaire et utiliser $set. Ici on ajoute un champ mais on peut aussi modifier la valeur d’autre champs.

Pour supprimer un champ on utilise $unset :



Question : Récupérer les deux fichiers json acteurs et films et faire la jointure entre les deux documents.