Guide MongoDB - Opérations de Requête

1. La Recherche Simple : `.find()`

C'est l'équivalent du `SELECT` en SQL. Elle renvoie un "curseur" (une liste de résultats en attente).

Syntaxe Globale

```
cursor = db.nom_collection.find({FILTRE}, {PROJECTION})
```

Argument 1 : Le Filtre `{}` (Obligatoire, peut être vide)

C'est la condition `WHERE`.

- **Tout prendre** : `{}`
- **Égalité simple** : `{"cuisine": "Italian"}`
- **Opérateurs** : `{"score": {"$gt": 50}}` (Score > 50)
- **Logique ET** : `{"cuisine": "Italian", "borough": "Bronx"}` (Séparé par une virgule)
- **Logique OU** : `{"$or": [{"cuisine": "Italian"}, {"cuisine": "French"}]}`

Argument 2 : La Projection `{}` (Optionnel)

C'est le choix des colonnes `SELECT column1, column2`.

- **Afficher** : `{"name": 1}`
- **Masquer** : `{"_id": 0}` (L'ID est le seul champ qu'on masque explicitement, les autres sont masqués par défaut si on ne les demande pas)

Exemple complet

```
# Trouver les restos Italiens, afficher seulement le nom, sans l'ID
db.restaurants.find({"cuisine": "Italian"}, {"name": 1, "_id": 0})
```

---

2. Le Comptage : `.count_documents()`

Sert uniquement à récupérer un nombre entier.

Syntaxe Globale

```
nombre = db.nom_collection.count_documents({FILTRE})
```

Argument 1 : Le Filtre `{}` (Obligatoire)

- Fonctionne exactement comme le filtre de `.find()`
- **Attention** : Même si tu veux tout compter, tu dois mettre des accolades vides `{}`

Exemple complet

```
# Compter les restos ayant une note > 50
nb = db.restaurants.count_documents({"grades.score": {"$gt": 50}})
```

---

3. Les Valeurs Uniques : `.distinct()`

Sert à connaître les catégories existantes sans doublons.

Syntaxe Globale

```
liste_valeurs = db.nom_collection.distinct("NOM_CHAMP", {FILTRE})
```

Argument 1 : Le Champ (String)

Le nom de la colonne dont tu veux les valeurs uniques (entre guillemets).

Argument 2 : Le Filtre `{}` (Optionnel)

Si tu veux les valeurs uniques seulement pour une partie des données.

Exemple complet

```
# Lister les types de cuisine disponibles dans le quartier "Queens"
types = db.restaurants.distinct("cuisine", {"borough": "Queens"})
```

---

4. L'Agrégation : `.aggregate()`

C'est le plus puissant et le plus complexe. C'est une liste `[]` contenant une suite d'étapes (des dictionnaires `{}`).

Syntaxe Globale

```
resultat = db.nom_collection.aggregate([
    {"$ETAPE_1": {...}},
    {"$ETAPE_2": {...}},
    {"$ETAPE_3": {...}}
])
```

Les Étapes Principales

A. L'étape `$match` (Filtrer)

C'est exactement comme un `find`, mais placé dans le pipeline.

```
{"$match": {"borough": "Queens"}}
```

B. L'étape `$unwind` (Exploser)

Prend un champ tableau (liste) et crée une ligne pour chaque élément.

Syntaxe : Toujours mettre le `$` devant le nom du champ ici (c'est une variable).

```
{"$unwind": "$grades"}  # Attention au '$' devant grades !
```

C. L'étape `$group` (Grouper et Calculer)

C'est ici qu'on fait les maths.

```
{
    "$group": {
        "_id": "$champ_de_regroupement",  # Obligatoire. La clé par
laquelle on groupe.
                                          # Utilise "$" pour dire "la
valeur de ce champ".

        "nom_variable_resultat": {"$OPERATEUR": valeur}  # Les calculs
    }
}
```

Opérateurs courants :

- **Compter** : `{"$sum": 1}`
- **Sommer une colonne** : `{"$sum": "$prix"}`
- **Moyenne** : `{"$avg": "$notes"}`
- **Liste unique** : `{"$addToSet": "$ville"}`

D. L'étape `$project` (Remodeler à la fin)

Permet de nettoyer le résultat du groupe.

```
{
    "$project": {
```

```
        "_id": 0,                                  # Cacher l'ID
        "Ville": "$_id",                           # Renommer l'ID en "Ville"
        "Moyenne": {"$round": ["$moyenne", 2]} # Arrondir
    }
}
```

---

5. Les Modificateurs (Tri et Limite)

Ces méthodes s'enchaînent ("chaining") après un `.find()`.

Syntaxe Tri : `.sort()`

```
# Syntaxe : .sort("champ", DIRECTION)
# 1 = Croissant, -1 = Décroissant

# Tri simple
cursor = db.resto.find().sort("name", 1)

# Tri multiple (Liste de tuples) : D'abord quartier croissant, puis rue
décroissant
cursor = db.resto.find().sort([("borough", 1), ("address.street", -1)])
```

Syntaxe Limite : `.limit()`

```
cursor = db.resto.find().limit(5)  # Ne prend que les 5 premiers résultats
```