Guide MongoDB avec Python

1. Pourquoi utilise-t-on un DataFrame (pandas) ?

MongoDB renvoie les données sous une forme brute appelée un **Curseur** (une sorte de liste de dictionnaires JSON). On "emballe" souvent ce résultat dans un DataFrame Pandas pour trois raisons principales :

La Lisibilité (Excel-like)

Un curseur MongoDB, c'est une liste de dictionnaires illisible à l'œil nu. Le DataFrame transforme cela en un tableau propre avec des lignes et des colonnes. C'est beaucoup plus facile pour vérifier si ta requête a marché.

L'Analyse et le Nettoyage

Python (via Pandas) est parfois plus fort ou plus simple que MongoDB pour nettoyer des données.

Exemple : Dans l'exercice 3 (AirBnB), les prix sont stockés avec des virgules (`"10,50"`). C'est pénible à corriger en pure requête MongoDB. On récupère les données, on les met dans un DataFrame, et on utilise `.str.replace(",", ".")` en Python pour corriger le problème facilement.

La Visualisation

Les librairies de graphiques comme `seaborn` ou `matplotlib` (utilisées dans ton exo 3) ne savent pas lire du MongoDB brut. Elles ont besoin d'un DataFrame pour dessiner des barres ou des courbes.

---

2. Le choix des armes : `.find` vs `.aggregate`

Tu as plusieurs outils, chacun sert à un niveau de complexité différent.

Commande	Quand l'utiliser ?	Analogie SQL
`.find()`	Extraction simple. Quand tu veux juste lire des données, filtrer des lignes, ou choisir des colonnes spécifiques sans faire de calculs.	`SELECT * FROM ... WHERE ...`
`.aggregate()`	Calculs et Transformations. Dès que tu dois grouper, faire des moyennes, exploser des tableaux (`$unwind`) ou enchainer des opérations complexes.	`GROUP BY, SUM, JOIN`
`.count_documents()`	Comptage pur. Quand tu veux juste un chiffre (le nombre de résultats) sans récupérer les données elles-mêmes. C'est beaucoup plus rapide que de faire un find puis de compter la taille de la liste.	`SELECT COUNT(*)`

Commande	Quand l'utiliser ?	Analogie SQL
`.distinct()`	Valeurs uniques. Pour savoir quelles sont les catégories existantes sans doublons.	`SELECT DISTINCT`

3. Les Opérateurs (Le langage du $)

En Python ou SQL, tu utilises des symboles (>, <, =). En JSON (le format de MongoDB), les clés doivent être du texte. On ne peut pas écrire `{"age" > 18}`. On utilise donc des opérateurs qui commencent toujours par `$`.

Comment les utiliser ?

La structure est presque toujours : `{ "champ": { "$operateur": valeur } }`

La Liste des essentiels (à connaître par cœur)

A. Les Comparaisons (dans `find` ou `$match`)

- **`$eq`** (Equal) : Égal à (souvent implicite : `{"a": 1}` équivaut à `{"a": {"$eq": 1}}`).
- **`$ne`** (Not Equal) : Différent de.
- **`$gt`** / **`$gte`** (Greater Than / Equal) : Plus grand que / ou égal.
- **`$lt`** / **`$lte`** (Less Than / Equal) : Plus petit que / ou égal.
- **`$in`** : Est présent dans une liste (ex: `{"quartier": {"$in": ["Bronx", "Queens"]}}`).

B. Les Logiques

- **`$and`** : ET (souvent implicite avec une virgule).
- **`$or`** : OU (ex: `{"$or": [{"prix": 10}, {"prix": 20}]}`).
- **`$regex`** : Recherche de texte (comme le `LIKE` en SQL).

C. Les Étapes de Pipeline (dans `aggregate` uniquement)

Ce sont les verbes d'action au début de chaque étape `{...}`.

- **`$match`** : Filtrer.
- **`$group`** : Grouper.
- **`$project`** : Remodeler (choisir colonnes).
- **`$unwind`** : Déplier un tableau.
- **`$sort`** : Trier.
- **`$limit`** : Limiter le nombre.

D. Les Calculs (dans `$group`)

- **`$sum`** : Somme (ou `$sum: 1` pour compter).
- **`$avg`** : Moyenne.
- **`$min`** / **`$max`** : Minimum / Maximum.
- **`$first`** / **`$last`** : Premier / Dernier élément trouvé.

- `$addToSet` : Créer une liste de valeurs uniques.

---

4. Variables : Python vs MongoDB

C'est le point qui crée le plus de confusion. Il y a deux types de "variables".

A. Les variables Python (Côté Client)

C'est ce que tu connais. Tu stockes le résultat de la requête ou la connexion.

```
db = connex.resto          # Variable qui contient la base de données
requete = db.find(...)     # Variable qui contient le résultat brut
df = pandas.DataFrame(...) # Variable qui contient le tableau propre
```

B. Les variables de Champ MongoDB (Côté Serveur)

Dans une agrégation (`aggregate`), comment dire à Mongo "Utilise la valeur du champ prix" ? On utilise le préfixe `$` devant le nom du champ.

- `"price"` (sans `$`) : C'est juste le nom de la colonne (une chaîne de caractères).
- `"$price"` (avec `$`) : C'est la valeur contenue dans la colonne pour la ligne en cours.

Exemple concret pour comprendre la différence :

Si tu écris ça dans un `$group` :

```
"_id": "cuisine"
# Résultat : Tous les restos sont groupés sous le mot texte "cuisine".
# Tu auras 1 seule ligne de résultat. C'est inutile.
```

Si tu écris ça :

```
"_id": "$cuisine"
# Résultat : Mongo va chercher la VALEUR du champ cuisine (French,
Italian, etc.).
# Tu auras autant de groupes qu'il y a de cuisines différentes.
```

En résumé

> Dans `aggregate`, dès que tu veux faire référence à la valeur d'un champ existant dans tes données, tu dois mettre un `$` devant son nom.