Guide de Survie MongoDB pour Pythonistes 🐍🍃

Ce document résume les concepts clés pour manipuler une base de données MongoDB avec Python (`pymongo`), basé sur les exercices de restaurants et d'AirBnB.

---

1. Le Changement de Paradigme

Pour comprendre MongoDB quand on vient de Python :

- **Un Document** = Un **Dictionnaire** Python (JSON).
- **Une Collection** = Une **Liste** de dictionnaires (équivalent d'une Table SQL).
- **Pas de schéma fixe** : Chaque dictionnaire peut avoir des clés différentes.

---

2. Les Opérations de Base (`find`, `count`, `sort`)

Basé sur le fichier `3-exo1.py`

A. Se connecter

```
import pymongo
connex = pymongo.MongoClient("mongodb://127.0.0.1:27017/")
db = connex.resto # Accès à la base de données
```

B. Rechercher (find)

La méthode `find` prend deux arguments principaux :

- **Le Filtre** (Quelles lignes je veux ?)
- **La Projection** (Quelles colonnes je veux ?)

```
# Syntaxe : db.collection.find({FILTRE}, {PROJECTION})

# Exemple : Je veux le nom et la rue, mais pas l'ID technique
cursor = db.restaurants.find(
    {},                                    # Filtre vide = Tout
prendre
    {"name": 1, "address.street": 1, "_id": 0}   # 1=Afficher, 0=Cacher
)
```

C. Compter (count_documents)

On utilise des opérateurs spéciaux (commençant par `$`) pour les comparaisons, car MongoDB ne comprend pas `>` ou `<` directement dans les clés JSON.

Opérateur	Signification	Exemple
`$eq`	Égal à	`{"cuisine": "French"}` (par défaut)
`$gt` / `$gte`	Plus grand (ou égal)	`{"grades.score": {"$gt": 50}}`
`$lt` / `$lte`	Plus petit (ou égal)	`{"price": {"$lte": 100}}`
`$in`	Est dans une liste	`{"address.street": {"$in": ["Rue A", "Rue B"]}}`
`$regex`	Contient le texte	`{"name": {"$regex": "Burger"}}`

D. Trier (sort)

- `1` : Ordre Croissant (A-Z).
- `-1` : Ordre Décroissant (Z-A).

```
# Trier par quartier (croissant) puis rue (décroissant)
db.restaurants.find(...).sort([("borough", 1), ("address.street", -1)])
```

E. Listes de valeurs uniques (distinct)

Pour obtenir toutes les valeurs uniques d'un champ (comme un SET en Python).

```
styles = db.restaurants.distinct("cuisine")
```

---

3. L'Agrégation : Le Pipeline (aggregate)

Basé sur le fichier `3-exo2.py`

C'est le mode "Expert". On ne fait plus une simple requête, on construit une chaîne de traitement. Les données entrent, subissent des transformations successives, et ressortent.

Syntaxe : `db.collection.aggregate([ {étape1}, {étape2}, ... ])`

Les Étapes Clés du Pipeline

1. $match (Le Filtrage)

C'est le même principe que `find`, mais à l'intérieur du pipeline. Il ne laisse passer que les documents correspondants.

2. $unwind (L'Explosion de Tableau) ⚠ Concept Vital

Si un document contient une liste (ex: `grades: [A, B, A]`), on ne peut pas grouper dessus directement. `$unwind` duplique le document pour chaque élément de la liste.

- **Avant** : `{id: 1, notes: [10, 20]}`

- **Après $unwind** : `{id: 1, notes: 10}` et `{id: 1, notes: 20}`.

3. $group (Le Regroupement)

On définit un identifiant de groupe (`_id`) et on calcule des statistiques.

```
{"$group": {
    "_id": "$borough",              # La clé de regroupement (ex: par
quartier)
    "total": {"$sum": 1},           # Compteur (+1 par document)
    "score_moyen": {"$avg": "$grades.score"}, # Moyenne
    "rues": {"$addToSet": "$street"} # Liste des rues uniques
}}
```

- `$push` : Ajoute tout dans une liste (avec doublons).
- `$addToSet` : Ajoute dans une liste (sans doublons).

4. $project (Le Remodelage)

Utilisé à la fin pour nettoyer le résultat : renommer des champs, cacher l'ID, arrondir des valeurs.

```
{"$project": {
    "_id": 0,
    "Quartier": "$_id",
    "Moyenne": {"$round": ["$score_moyen", 2]}
}}
```

---

4. Astuces Avancées & Pandas

Basé sur le fichier `3-exo3.py`

A. Le raccourci $sortByCount

Remplace la combinaison `$group` + `$sort` pour les comptages simples.

```
# Compter les équipements les plus fréquents et trier le résultat
{"$unwind": "$amenities"},
{"$sortByCount": "$amenities"}
```

B. Grouper sur plusieurs champs (Clé composée)

Si on veut grouper par ID ET par Nom (pour ne pas perdre le nom dans l'affichage final), on utilise un objet comme `_id`.

```
{"$group": {
    "_id": {
        "id": "$reviewer_id",
        "nom": "$reviewer_name"
    },
    "count": {"$sum": 1}
}}
```

C. Champs calculés sans agrégation

Dans une recherche simple (`find`), on peut utiliser `$size` pour obtenir la taille d'un tableau directement.

```
"nb_avis": {"$size": "$reviews"}
```

D. Nettoyage Post-MongoDB (Pandas)

Parfois, MongoDB renvoie des données "sales" (ex: prix en string avec virgules). Il est souvent plus simple de nettoyer avec Pandas après la requête.

```
df["Prix"] = df["Prix"].astype(str).str.replace(",", ".")
df["Prix"] = pandas.to_numeric(df["Prix"])
```

---

5. Memento : SQL vs MongoDB

Action	SQL	MongoDB (Python)
Tout lire	`SELECT *`	`find({})`
Filtrer	`WHERE cuisine = 'French'`	`find({"cuisine": "French"})`
Sélectionner colonnes	`SELECT name, street`	`find({}, {"name": 1, "street": 1})`
Compter	`COUNT(*)`	`count_documents({})`
Trier	`ORDER BY date DESC`	`sort("date", -1)`
Grouper	`GROUP BY`	`aggregate([{"$group": ...}])`
Joindre/Aplatir	Pas d'équivalent direct	`aggregate([{"$unwind": ...}])`