TP – MongoDB : CRUD & Opérateurs

Objectifs

- Manipuler une collection MongoDB avec les opérations CRUD.
- Formuler des requêtes avec opérateurs de comparaison et logiques.

Pré-requis

- mongosh installé et lancé.
- Aucune base préalable requise.

Consignes

- Écrire **les commandes exactes** que vous exécutez.
- Ne pas modifier l'énoncé (les _id sont imposés).
- Sauvegarder vos commandes dans un fichier texte nommé TP1_CRUD_<Nom>.txt.

Partie A : Mise en place (Create)

A1. Sélectionner (ou créer) la base entreprise.
A2. Créer la collection employes.
A3. Insérer **en une seule commande** la liste suivante dans employes :

{ _id: 1, nom: "Dupont", prenom: "Sarah", age: 28, ville: "Paris", poste: "Développeuse", salaire: 3200 }

{ _id: 2, nom: "Martin", prenom: "Alex", age: 35, ville: "Lyon", poste: "Chef de projet", salaire: 4800 }

{ _id: 3, nom: "Nguyen", prenom: "Lina", age: 24, ville: "Paris", poste: "Data Analyst", salaire: 3000 }

{ _id: 4, nom: "Moreau", prenom: "Lucas", age: 41, ville: "Marseille", poste: "Architecte", salaire: 6200 }

{ _id: 5, nom: "Bernard", prenom: "Emma", age: 29, ville: "Lille", poste: "Développeuse", salaire: 3400 }

A4. Donner la commande qui permet de **compter** les documents insérés.

Partie B : Lecture (Read)

B1. Afficher **tous** les employés avec un rendu lisible.
B2. Afficher **uniquement** nom, poste, salaire (sans _id).
B3. Lister les employés **travaillant à Paris**.
B4. Lister les employés dont le **salaire est strictement supérieur à 4000**.
B5. Trier les employés **par salaire décroissant**, ne retourner que les **3 premiers** (commande unique).

Partie C : Mise à jour (Update)

C1. Augmenter de **10%** le salaire **de tous** les employés **à Paris**.
C2. Modifier le poste d'**Emma Bernard** en Lead Developer.
C3. Ajouter le champ booléen remote: true **uniquement** aux employés de **Paris**.
C4. Créer un champ prime et le fixer à 500 **pour les salaires < 3500**. (une seule commande)

Partie D : Suppression (Delete)

D1. Supprimer **un** employé nommé **Nguyen**.
D2. Supprimer **tous** les employés de **Marseille**.
D3. Donner la commande qui **supprime** la **collection** (ne pas l'exécuter si vous enchaînez les parties suivantes).

Partie E : Opérateurs de comparaison & logiques

E1. Trouver les employés **âgés de plus de 30 ans et** gagnant **moins de 5000**.
E2. Trouver les employés **de Lyon ou de Lille**.
E3. Trouver les employés **qui ne travaillent pas à Paris**.
E4. Trouver les employés **dont le poste n'est pas** Chef de projet.
E5. Requête unique : salaires **entre 3200 et 6000 inclus**, **à Paris**, et **poste** dans ["Développeuse","Lead Developer"], triés par **nom croissant**.

Partie F : Questions de réflexion (réponses en 2–3 lignes)

F1. Différence entre updateOne() et updateMany() (cas d'usage).
F2. Conséquence d'une insertion avec un _id déjà existant.
F3. Pourquoi la base est-elle visible après use entreprise seulement si des données ont

été créées ?
F4. Quels sont les risques d'ajouter des champs à la volée (schéma flexible) et comment les limiter ?

Partie H : Requêtes combinant projection, tri & pagination

H1. Afficher tous les employés, triés par nom croissant, en ne retournant que : nom, prenom, ville, salaire (sans _id).

H2. Afficher les employés classés par salaire croissant, ne retourner que les 2 moins bien payés.

H3. Afficher les employés triés par salaire décroissant, mais *ignorer* le premier (le mieux payé) et afficher les 2 suivants (utiliser skip() et limit()).

H4. Afficher les employés dont le nom commence par "D" ou "M", triés par nom croissant (utiliser une expression régulière).

H5. Afficher uniquement les employés ayant un salaire pair (indice : utiliser $expr et un opérateur arithmétique).