TP : MongoDB

L'objectif de ce TP est d'insérer, explorer et préparer des données concernant l'accidentalité routière en France.

Le descriptif des données est disponible ici

Exercice 1 : Lancer MongoDB et installer les outils de connexions

Lancer un service Onyxia mongodb avec la configuration par défaut.

Dans votre instance jupyter lancer un terminal linux et installer les clients python et natif de mongodb avec les commandes suivantes :

1. Récupérer le fichier `.deb`

```
curl -LO https://fastdl.mongodb.org/tools/db/mongodb-database-tools-ubuntu2204-x86_64-100.10.0.deb
```

2. Installer le paquet

```
sudo apt install ./mongodb-database-tools-ubuntu2204-x86_64-100.10.0.deb
```

3. Installer le package python

```
pip install pymongo
```

Exercice 2 : Utilisez `mongoimport` pour importer des données depuis un csv

1. Utiliser la commande `mongoimport` pour importer dans une collection `vehicules` le fichier `vehicules-2022.csv`.

Utiliser la documentation en ligne de l'outil afin de passer à mongodb les différentes options :

- Nom du serveur mongodb : `mongodb-0.mongodb-headless`
- Base de données dans lequel importer : `defaultdb`
- Authentification au serveur mongodb (username et password)
- Type et chemin du fichier à importer (dans notre cas un fichier csv).

> Attention, `mongodb` ne supporte pas l'import quand le séparateur n'est pas soit une tabulation soit une virgule. Pour cela, transformer le fichier avec la commande suivante pour modifier la séparation `;` en une tabulation (format `TSV`) :

```
tr ";" "\t" < vehicules-2022.csv > vehicules-2022.tsv
```

> Il faut également retirer les " dans le fichier afin que mongodb puisse interpéter les types :

```
tr -d "\"" < vehicules-2022.tsv > vehicules-2022.tsv
```

- Préciser que la ligne du nom des colonnes est la première
- Ignorer les valeurs pour les champs vides
- Préciser le nom de la collection

> Passez l'option `--drop` pour réécraser la table à chaque import

2. Vérifier ensuite que les données semblent cohérentes.

Lancer ensuite un notebook et créer une connexion à partir des informations communiqués par le service Onyxia.

Récupérer une ligne de la collection `vehicules` avec la méthode `find_one`

> Pour intéragir avec une collection :

```
db.<collection>.find_one()
```

Exercice 3 : insérer des données

1. Lire le fichier `csv` usagers et ajouter les 10 000 premières ligne des usagers dans la collection `usagers`.

> En utilisant `insert_many`, l'insertion est plus rapide

2. Compter le nombre de documents ajoutés avec la méthode `count_documents` sur la collection `usagers` afin de vérifier

Exercice 4 : dénormaliser les données

L'objectif de cet exercice va être d'associer les véhicules à chaque usager en fonction de la clé `Num_Acc` commune aux deux collections.

1. Faire une fonction qui lit les données de la collection `usagers` et y ajoute les données des véhicules associés dans une nouvelle clé `vehicules`.

2. Supprimer l'ensemble des données de la collection `vehicules`

Exercice 5 : recherche et filtrage de données

1. Rechercher les usagers dans un accident lorsque un véhicule léger (VL) est impliqué. Ne projeter que le `Num_Acc` (sans l'Id généré par mongodb).
2. Rechercher les documents concernant les femmes impliquées dans un accident et lorsque l'un des véhicules est un deux roues motorisé
3. Rechercher les accidents avec 3 ou 4 véhicules impliqués

Exercice 6 : créer une pipeline de données

1. Calculer le nombre total d'accidents par type de véhicule. Quels types de véhicules sont les plus concernés ?
2. Rajouter les données concernant les `lieux` dans une collection et effectuer une jointure afin récupérer le nombre d'accidents dans les autoroutes