Correction : 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`.

> 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-2022-2.tsv
> ```

Lancer ensuite la commande suivante en remplacant le mode de passe et l'username par les valeurs du compte onyxia :

```
mongoimport --db defaultdb  --collection vehicules --file vehicules-2022-2.tsv --host mongodb-0.mongodb-headless --type tsv --headerline --username user-<> --password xxx --drop  --ignoreBlanks
```

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

```
db.vehicules.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`.

```
df = pd.read_csv("usagers-2022.csv", sep=';')
df.replace(r"\xa0", "", regex=True)
for n, i in df.iterrows():
    row = i.to_dict()
    db.usagers.insert_one(row)
```

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

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

```
db.usagers.count_documents({})
```

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`.

```
for usagers in db.usagers.find():
    num_acc = usagers["Num_Acc"]
    veh = list(db.vehicules.find({"Num_Acc": num_acc}))
    db.usagers.update_one({"_id": usagers["_id"]}, {"$set": {"vehicules": veh}})
```

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

```
db.vehicules.delete_many({})
```

Exercice 5 : recherche et filtrage de données

1. Rechercher les accidents lorsque un véhicule léger (VL) est impliqué. Ne projeter que le `Num_Acc` (sans l'Id généré par mongodb).

```
for usager in db.usagers.find({"vehicules.catv": 7},{"_id": 0, "Num_Acc":
1}):
    print(usager)
```

2. Rechercher les documents concernant les femmes impliquées dans un accident et lorsque l'un des véhicules est un deux roues motorisé

```
for usager in db.usagers.find({
    "$and": [
        {"sexe": 2},
        {"vehicules.catv": {"$in": [30, 31, 32, 33, 34]}}
    ]
}):
    print(usager)
```

3. Rechercher les accidents avec 3 ou 4 véhicules impliqués

```
for usager in db.usagers.find({
    "$or": [
        {"vehicules": {"$size": 4}},
        {"vehicules": {"$size": 3}}
    ]
}):
    print(usager)
```

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 ?

```
pipeline = [
    {"$unwind": "$vehicules"},
    {"$group": {"_id": "$vehicules.catv", "total_accidents": {"$sum":
1}}},
```

```
    {"$sort": {"total_accidents": -1}}
]

results = db.usagers.aggregate(pipeline)
for result in results:
    print(result)
```

Ce sont les véhicules légers

2. Rajouter les données concernant les `lieux` dans une collection et effectuer une jointure afin récupérer le nombre d'accidents dans les autoroutes

Importer la collection `lieux`avec `mongoimport`

```
mongoimport --db defaultdb  --collection lieux --file lieux-2022-2.tsv --
host mongodb-0.mongodb-headless --type tsv --headerline --username user-<>
--password xxx --drop  --ignoreBlanks
```

La pipeline de jointure est la suivante :

```
pipeline = [
    {
        "$lookup": {
            "from": "lieux",
            "localField": "Num_Acc",
            "foreignField": "Num_Acc",
            "as": "lieu_info"
        }
    },
    {"$unwind": "$lieu_info"},
    {"$match": {"lieu_info.catr": 1 }},
]

results = db.usagers.aggregate(pipeline)

for result in results:
    print(result)
    break
```