Projet BackEND Python DIA 2 : Analyse et Visualisation de données avec MongoDB, FastAPI et Streamlit

Objectifs du projet :

- Installer et configurer MongoDB.
- Importer des données dans MongoDB à partir de fichiers CSV.
- Manipuler et interroger les données avec Python.
- Créer une API avec FastAPI pour calculer des KPI.
- Visualiser les données et les KPI avec Streamlit.
- Organiser le projet sur un dépôt GitHub.
- Soutenir et défendre ses choix technologiques lors d'une soutenance.

Étapes du projet :

1. Installation et Configuration de MongoDB

1. **Installation de MongoDB :**
    - Téléchargez et installez MongoDB Community Edition ici.
    - Démarrez le service MongoDB :
      mongod
2. **Installation de MongoDB Compass :**
    - Téléchargez MongoDB Compass ici.
    - Lancez Compass et connectez-vous au serveur MongoDB local (`mongodb://localhost:27017`).
3. **Créer une base de données :**
    - Nommez la base de données : `ecommerce`.

---

2. Importer des Données dans MongoDB

1. **Préparation des données :**
    - Vous disposez des fichiers suivants :
        - `Location.csv`

        - `Orders.csv`
        - `Products.csv`
        - `Customers.csv`
    - Placez ces fichiers dans un répertoire accessible.
2. **Importer les données dans MongoDB :**

Utilisez la commande `mongoimport` pour importer les fichiers CSV :
mongoimport --db ecommerce --collection locations --type csv --headerline --file Location.csv
mongoimport --db ecommerce --collection orders --type csv --headerline --file Orders.csv
mongoimport --db ecommerce --collection products --type csv --headerline --file Products.csv

    - mongoimport --db ecommerce --collection customers --type csv --headerline --file Customers.csv
3. **Vérification des données dans MongoDB Compass :**
    - Ouvrez Compass, naviguez vers la base `ecommerce` et vérifiez que les collections sont correctement peuplées.

---

3. Explorer et Manipuler les Données

1. **Joindre les collections dans MongoDB Compass :**
    - Réalisez des jointures entre les collections pour relier :
        - Les commandes (`orders`) avec les produits (`products`).
        - Les clients (`customers`) avec leurs commandes.
        - Les localisations (`locations`) des clients.
2. **Exemple de jointure :**

Utilisez l'agrégation MongoDB pour joindre les collections :

```
db.orders.aggregate([
 {
  $lookup: {
   from: "customers",
   localField: "customer_id",
   foreignField: "customer_id",
   as: "customer_info"
  }
 }
```

    -])

---

4. Importer les Données dans Python

1. **Installer les dépendances :**
    - Installez les bibliothèques nécessaires :
      pip install pymongo pandas fastapi uvicorn streamlit
2. **Connexion à MongoDB :**

Exemple de script Python pour se connecter à MongoDB et charger une collection :

```
from pymongo import MongoClient
import pandas as pd

client = MongoClient("mongodb://localhost:27017/")
db = client['ecommerce']

# Charger la collection "orders" en DataFrame
orders = pd.DataFrame(list(db.orders.find()))
```

- print(orders.head())

---

5. Créer une API avec FastAPI

1. **Créer un fichier FastAPI :**
    - Nommez-le `main.py`.

Exemple de code pour calculer des KPI :

```
from fastapi import FastAPI
from pymongo import MongoClient

app = FastAPI()

client = MongoClient("mongodb://localhost:27017/")
db = client['ecommerce']

@app.get("/kpi/orders-per-customer")
async def orders_per_customer():
    pipeline = [
        {"$group": {"_id": "$customer_id", "total_orders": {"$sum": 1}}},
        {"$sort": {"total_orders": -1}}
    ]
    result = list(db.orders.aggregate(pipeline))
```

2. return {"data": result}
3. **Lancer l'API :**
   uvicorn main:app --reload
4. **Tester l'API :**

- Ouvrez `http://127.0.0.1:8000/docs` pour tester l'API avec Swagger UI.

---

6. Visualisation avec Streamlit

1. **Créer une application Streamlit :**
   - Nommez-le `app.py`.

Exemple de code pour visualiser les KPI :

```
import streamlit as st
import requests
import pandas as pd

st.title("Tableau de bord eCommerce")

# Récupérer les KPI depuis l'API
response = requests.get("http://127.0.0.1:8000/kpi/orders-per-customer")
if response.status_code == 200:
    data = response.json()["data"]
    df = pd.DataFrame(data)
    st.bar_chart(df.set_index("_id")["total_orders"])
else:
```

2. st.error("Erreur lors du chargement des données")
3. **Lancer Streamlit :**
   streamlit run app.py

Livrables :

1. Un dépôt GitHub bien organisé contenant :
   - Les fichiers de code (`main.py`, `app.py`, scripts pour importer les données).
   - Une documentation claire sur l'installation et l'utilisation.
2. Une soutenance comprenant :
   - Une démonstration du projet fonctionnel (API et visualisation).
   - Une défense des choix technologiques effectués (veille technologique).

Barème de notation :

1. Rendu technique (60 points)

- **Organisation du dépôt GitHub (10 points)** :
    - Clarté de la structure et de la documentation.
- **Importation des données dans MongoDB (10 points)** :
    - Données correctement importées et jointures fonctionnelles.
- **API FastAPI (20 points)** :
    - Fonctionnalités implémentées (calcul des KPI).
    - Code propre et fonctionnel.
- **Visualisation Streamlit (20 points)** :
    - Application fonctionnelle et ergonomique.

2. Soutenance (40 points)

- **Démonstration du projet (20 points)** :
    - Projet fonctionnel et bien présenté.
- **Défense des choix technologiques (20 points)** :
    - Veille technologique pertinente.
    - Argumentation claire et cohérente.

Note finale : /100 points