TP 1 : Python Avancé

Problème : Analyse d'un Grand Ensemble de Données de Ventes

Vous travaillez pour une entreprise de commerce électronique qui dispose d'un grand ensemble de données de ventes contenant des informations sur les transactions des clients. Le fichier CSV (sales_data.csv) contient 1 millions de lignes et les colonnes suivantes :

- transaction_id : Identifiant unique de la transaction.
- customer_id : Identifiant unique du client.
- product_id : Identifiant unique du produit.
- quantity : Quantité de produits achetés.
- price : Prix unitaire du produit en DH.
- transaction_date : Date de la transaction.
- region : Région où la transaction a eu lieu.

Le fichier est trop volumineux pour être chargé en mémoire en une seule fois. Votre tâche consiste à analyser ces données en utilisant les techniques apprises dans le cours de python avancé.

Tâches

1. Échantillonnage et Sous-ensemble de Données

- Chargez un échantillon aléatoire de 1 % des lignes du fichier sales_data.csv.
- Sélectionnez uniquement les colonnes customer_id, product_id, quantity, et price.
- Spécifiez les types de données appropriés pour chaque colonne afin de réduire la consommation de mémoire.

2. Conversion en Formats de Fichiers Efficaces

- Convertissez l'échantillon de données chargé en format **Feather** et **Parquet**.
- Comparez la taille des fichiers résultants avec celle du fichier CSV d'origine.
- Mesurez le temps de chargement pour chaque format et expliquez les différences.

3. Utilisation de HDF5

- Créez un fichier HDF5 (sales_data.h5) et stockez l'échantillon de données dans une table appelée sales_sample.
- Ajoutez une deuxième table contenant les transactions dont le prix est supérieur à 100 DH.

- Lisez les données de la table sales_sample et affichez les 5 premières lignes.

4. Lecture par Morceaux

- Lisez le fichier sales_data.csv par morceaux de 100 000 lignes.
- Pour chaque morceau, filtrez les transactions ayant une quantité supérieure à 10.
- Combinez les résultats filtrés en un seul DataFrame et calculez le total des ventes (quantité * prix) pour ces transactions.

5. Chargement dans une Base de Données

- Créez une base de données SQLite (sales.db) et chargez l'intégralité du fichier sales_data.csv dans une table appelée sales.
- Exécutez une requête SQL pour extraire les transactions ayant eu lieu dans la région "Europe" et dont le prix est supérieur à 50 DH.
- Calculez le total des ventes pour ces transactions et affichez le résultat.

Questions pour les Étudiants

1. **Échantillonnage et Sous-ensemble de Données**
   - Pourquoi est-il utile de charger un échantillon aléatoire de données plutôt que l'ensemble complet ?
   - Comment la spécification des types de données réduit-elle la consommation de mémoire ?
2. **Conversion en Formats de Fichiers Efficaces**
   - Quels sont les avantages des formats Feather et Parquet par rapport au format CSV ?
   - Dans quels cas préféreriez-vous utiliser Feather plutôt que Parquet, et vice versa ?
3. **Utilisation de HDF5**
   - Qu'est-ce qu'un fichier HDF5 et comment est-il structuré ?
   - Pourquoi est-il utile de stocker des données dans un fichier HDF5 plutôt que dans un fichier CSV ?
4. **Lecture par Morceaux**
   - Pourquoi est-il nécessaire de lire un fichier volumineux par morceaux ?
   - Comment pouvez-vous filtrer et combiner des données provenant de plusieurs morceaux ?

5. **Chargement dans une Base de Données**

   – Quels sont les avantages de stocker des données dans une base de données SQLite plutôt que dans un fichier CSV ?

   – Comment pouvez-vous exécuter des requêtes SQL sur une base de données SQLite à partir de Python ?

Résultat Attendu

Vous devez exporter votre fichier Jupyter Notebook au format PDF et le soumettre sur la plateforme Google Classroom. Toute soumission en retard sera automatiquement rejetée.

Annexe

Pour créer un jeu de données aléatoire avec 1 million de lignes en utilisant Python et la bibliothèque **pandas**. Ce jeu de données simulera des transactions de vente avec des colonnes telles que **transaction_id, customer_id, product_id, quantity, price, transaction_date**, et **region**.

Code pour générer un jeu de données aléatoire

```
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

# Paramètres pour la génération de données
num_rows = 1_000_000  # 1 million de lignes
regions = ['North America', 'Europe', 'Asia', 'South America', 'Africa', 'Australia']
start_date = datetime(2020, 1, 1)  # Date de début pour les transactions
end_date = datetime(2023, 12, 31)  # Date de fin pour les transactions

# Génération des données
np.random.seed(42)  # Pour la reproductibilité

# transaction_id : Identifiant unique
transaction_ids = np.arange(1, num_rows + 1)

# customer_id : Identifiant client aléatoire entre 1 et 100 000
customer_ids = np.random.randint(1, 100_001, num_rows)

# product_id : Identifiant produit aléatoire entre 1 et 10 000
product_ids = np.random.randint(1, 10_001, num_rows)

# quantity : Quantité aléatoire entre 1 et 10
quantities = np.random.randint(1, 11, num_rows)

# price : Prix aléatoire entre 10 et 500 (avec 2 décimales)
prices = np.round(np.random.uniform(10, 500, num_rows), 2)

# transaction_date : Date aléatoire entre start_date et end_date
date_range = (end_date - start_date).days
transaction_dates = [start_date + timedelta(days=np.random.randint(0, date_range)) for _ in
```

```
range(num_rows)]

# region : Région aléatoire parmi la liste définie
regions_data = np.random.choice(regions, num_rows)

# Création du DataFrame
df = pd.DataFrame({
    'transaction_id': transaction_ids,
    'customer_id': customer_ids,
    'product_id': product_ids,
    'quantity': quantities,
    'price': prices,
    'transaction_date': transaction_dates,
    'region': regions_data
})

# Affichage des 5 premières lignes
print(df.head())

# Sauvegarde du DataFrame dans un fichier CSV
df.to_csv('sales_data.csv', index=False)
print("Fichier 'sales_data.csv' sauvegardé avec succès.")
```

Explication du Code

1. **transaction_id** : Un identifiant unique pour chaque transaction, allant de 1 à 1 000 000.
2. **customer_id** : Un identifiant client aléatoire entre 1 et 100 000.
3. **product_id** : Un identifiant produit aléatoire entre 1 et 10 000.
4. **quantity** : Une quantité aléatoire entre 1 et 10.
5. **price** : Un prix aléatoire entre 10 et 500, avec 2 décimales.
6. **transaction_date** : Une date aléatoire entre le 1er janvier 2020 et le 31 décembre 2023.
7. **region** : Une région aléatoire parmi une liste prédéfinie.

Exemple de Sortie

Voici un aperçu des 5 premières lignes du jeu de données généré :

transaction_id	customer_id	product_id	quantity	price	transaction_date	region
1	12345	5678	3	45.67	2021-03-15	Europe
2	67890	1234	7	120.50	2022-07-22	North America
3	23456	9876	2	89.99	2020-11-30	Asia
4	78901	3456	5	250.00	2023-05-10	South America
5	34567	6789	1	15.99	2021-12-25	Africa

Sauvegarde du Fichier

Le jeu de données est sauvegardé dans un fichier CSV nommé sales_data.csv. Ce fichier peut ensuite être utilisé pour les exercices proposés dans le fichier Jupyter Notebook.

Utilisation du Fichier

- Le fichier sales_data.csv peut être utilisé pour tester les techniques de manipulation de grands ensembles de données, comme l'échantillonnage, la lecture par morceaux, la conversion en formats efficaces (Feather, Parquet), et l'interaction avec une base de données.

Remarque

- La génération de 1 million de lignes peut prendre quelques secondes, selon la puissance de votre machine.