Rakib SHEIKH
I1 EISI, CyberSécurirté, IA et Big Data (Groupe 3 et 4)
rakib.sheikh@cyu.fr

TP 1 : Data Quality avec Soda

Pré-requis :

- Docker
- Python (Via miniconda recommandé)
- Fin du TP 2 de Architecture Décisionnel

Nous supposerons que vous êtes sur un Unix, à vous de trouver les commandes pour Windows pour ceux qui ne sont pas sur Unix.

Introduction : Importation de données

—> Cas 1 : Si vous avez Architecture Décisionnel avec moi

- Réutilisez la base de données sur les taxis de New-York pour continuer la suite du TP.
  - Nous utiliserons les données brutes, sans formattage.

—> Cas 2 : Si vous n'avez pas Architecture Décisionnel avec moi

- Demandez-moi le fichier permettant l'importation des données vers votre base de données.

1. Démarrez Docker
2. Tapez la commande suivante : `docker compose up`
3. Lancez le script d'importation de données du taxi de new-york (environ 20 à 30 minutes)

Commun :

4. Installez les dépendances nécessaires pour le TP `pip install -r requirement.txt`

Exercice 1 : Configuration et prise en main

- Créez un fichier `configuration.yml` avec les paramètres suivants :

```yaml
data_source nyc_warehouse:
  type: postgres
  connection:
    host: localhost
    port: 15432
    username: postgres
    password: admin
  database: nyc_warehouse
  schema: public
```

- Maintenant, testez la bonne configuration de soda en tapant cette ligne de commande

```
soda test-connection -d nyc_warehouse -c configuration.yml
```

Vous devriez avoir ce message

```
Soda Core 3.4.1
Successfully connected to 'nyc_warehouse'.
Connection 'nyc_warehouse' is valid.
```

- Créez un fichier `check.yaml` avec les paramètres suivants et vous devez fournir la liste des colonnes présente dans le dataset.

```
checks for nyc_raw:
    - schema:
        warn:
          when required column missing: [ column_name ]
        fail:
          when forbidden column present: [ column_name, column_name2 ]
```

- Testez la commande ci-dessous, que remarquez-vous ?

```
soda scan -d nyc_warehouse -c configuration.yml check.yml
```

- Maintenant, ajoutez le `-V` en fin de commande. Que remarquez-vous ?
- Changeons le code pour la règle suivante : Nombre de ligne supérieur à 0. (À l'aide de `row_count`)
- Nous allons maintenant récrire le `row_count` sous la forme de warning et de failure. Testez la règle suivante et que remarquez vous ?

```
checks for nyc_raw:
  - row_count:
      warn: when > 90
      fail: when = 0
```

Exercice 2 : Mes premiers checks

Nous allons maintenant rédiger quelque règles pour assurer une certaine qualité des données. Pour ce faire, nous devons nous renseigner sur la connaissance de donnée afin de pouvoir rédiger des règles. La documentation se trouve à ces adresses :

- Yellow Trips Data Dictionary : https://www.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf
- Taxi Zone LookUp Table : https://d37ci6vzurychx.cloudfront.net/misc/taxi_zone_lookup.csv

Pour connaitre la liste des règles que vous pouvez établir :

- Data Contratcs Language de Soda Core : https://github.com/sodadata/soda-core/blob/main/docs/data-contracts-language.md

Votre objectif : Rédigez les règles de qualité de données pour qu'elle soit conforme à la Yellow Trips Data Dictionary

- Faites en de même concernant le TP3, soit après avoir convertie votre base de données en modèle en flocons.

Exercice 3 : Restitution des qualités de données

Votre but est de récupérer l'ensemble des logs afin de pouvoir les stocker dans une autre base de données. Cela nous permettra à l'aide de l'éexercice 4, de réaliser un petit tableau de bord sur la qualitée de données.

- Pour récupérer les logs, voici la marche à suivre :

Exercice 4 : Réalisation de dashboard qualité de donnée

Votre but est de récupérer les données stockés dans votre base de donnée à l'exercice 3 afin de réaliser un tableau de bord pour suivre les indicateurs de performances de votre qualité de données.

Pour cela, nous utiliserons Streamlit afin d'accélérer la génération du tableau de bord.