SAE Collecte Web de données

Contextualisation

Dans le cadre de notre SAE Collecte automatisée de données web. Notre but était l'automatisation de la collecte et de l'intégration de données web dans un entrepôt de données. Ce travail nécessite une expertise dans l'analyse des données web et l'utilisation d'API pour récupérer les informations en ligne. Ce projet a duré 2 mois.

Etapes

1. **Acquisition et préparation des données :** Nous avons obtenu deux ensembles de données volumineux du gouvernement. Ces données ont été analysé et préparé en ajustant certaines variables pour les rendre plus exploitables.
2. **Utilisation d'une API :** Nous avons utilisé une API pour obtenir des informations géographiques sur les entreprises en utilisant les numéros SIREN, notamment les coordonnées de longitude et de latitude.
3. **Fusions des bases de données** : Nous avons établi un script python pour fusionner les deux fichiers de données volumineux.
4. **Web scraping** : Mise en place d'un script Python pour collecter automatiquement les données sur Internet (notamment sur Google Maps) en utilisant les coordonnées géographiques, les numéros SIREN et les noms des établissements, procédant par étapes successives pour cette opération.

Contraintes

En raison de la taille importante des bases de données, les opérations de jointure étaient complexes à exécuter. En les abordant par étapes successives, nous avons réussi à surmonter ce défi.

Compétences

- Gestion de projet
- Communication et collaboration
- Analyse et interprétation des données
- Sécurité et confidentialité des données
- ***Compétence 1****: "Traiter des données à des fins décisionnelles"*
- ***Logiciel*** *: Jupyter*

Preuves

SAE Collecte automatisé de données web

1. INTRODUCTION

Nous avons à notre disposition une base de données nommée 'StockEtablissement'. Ce fichier est un extrait de 1000 lignes de la base de données SIRENE. Cette base de données est constituée des informations concernant les entreprises et les différents établissements. Il existe plusieurs variables dont les numéros siret et siren ainsi que les informations concernant leurs adresses.

L'objectif de ce projet était de compléter et collecter les coordonnées géographiques via l'adresse API. Pour cela nous avons préparé un fichier export pour envoyer à l'API. Puis une fois les résultats des coordonnées géographiques complète obtenus, ces données ont permis de compléter les informations concernant les adresses des entreprises et des établissements.

2. DÉMARCHE SUIVI

Lecture du fichier de départ et préparation du fichier export à envoyer à l'API

Dans un premier temps, nous avons développé le code sur 10 lignes. A l'aide de la bibliothèque panda, nous avons commencé par lire le fichier par paquet de 10 lignes et l'insérer dans un dataframe. La base de données étant constituée de nombreuses variables, toutes les colonnes n'étaient pas affichées dans le dataframe. Nous avons sélectionné les principales variables liées à notre projet. Nous avons créé une nouvelle colonne adresse. Les colonnes

Elaboration d'une boucle pour traiter le reste des lignes du fichier

Nous avons mis en place une boucle permettant d'effectuer le traitement de toutes les lignes du fichier de départ. Cette boucle permet de lire le fichier de départ par paquet de lignes.

- Ainsi nous avons créé une boucle pour le fichier export.csv qui contient les colonnes : sirene, siret, adresse, codePostal et Code Libelle. Nous partons du fichier de départ avec les 1000 lignes pour effectuer nos tests avant de faire la boucle sur le fichier volumineux.
- Une fois que la boucle est réussie, nous obtenons le fichier export_API.csv en sortie contenant le bon nombre de lignes et les NaN supprimés. Ce fichier sera envoyé à l'API. Nous rajoutons cette étape dans la boucle.
- Ensuite nous vérifions que la boucle fonctionne automatiquement et qu'elle renvoie un fichier dans notre cas : "export_results_API". Une fois la boucle automatisée, elle sortira ces 2 fichiers automatiquement.
- Passons à la dernière étape. Le but est de créer 2 dataframes : un contenant la colonne "Adresse" du fichier export_API et un contenant les colonnes "latitude" et "longitude" du fichier "export_results_API". Nous faisons une concaténation qui nous sortira un fichier final contenant tous les éléments demandés. Le test a d'abord été effectué sur les 1000 lignes d'un pas de 100 avant d'être testé sur le fichier de 6,84 GO.

Entrée [16]:

```
import pandas as pd
from os import chdir
import requests
import time

# Définir le répertoire de travail
chdir("D:\BUT2\SAE - Collecte Web\StockEtablissement_utf8")

# Chemin du fichier CSV
fichier_csv = "StockEtablissement_utf8.csv"
fichier_export = "D:/BUT2/SAE - Collecte Web/export_API2.csv"
fichier_resultat = "D:/BUT2/SAE - Collecte Web/export_results_API.csv"

# Lecture de la première ligne pour obtenir les noms des colonnes
with open(fichier_csv, "r", encoding="utf-8") as file:
    first_line = file.readline().strip()
    noms_colonnes = first_line.split(",")

# Taille des pas de lecture
P = 10000

# Nombre total de lignes dans le fichier CSV
N = 100000 #Nombre de lignes du fichier à modifier au besoin

i = 0
premiere_iteration = True  # Pour gérer l'en-tête du fichier export_API.csv

# Liste pour stocker les DataFrames Adresse
adresses_dfs = []

while i < N:
    # Lecture des lignes
    sirene = pd.read_csv(fichier_csv, skiprows=i, nrows=P, header=None, names=noms_colonnes)

    # Remplacement des 'nan' par une chaîne vide
    sirene = sirene.fillna('')
```