ARTRAGIS

Geste de savoir

Parcourir des sheet

22 février 2021

Table des matières

Aujourd'hui un petit trucs et astuces dans le domaine de l'extraction de données depuis Google Spreadsheet. J'utiliserai python 3.8 pour extraire des données d'une fiche d'un drive privé.

1. Problématique et solution

1.1. Cas d'étude

Dans mon boulot on essaie d'étudier un peu les stats que fournissent nos différents outils ainsi que les documents qu'on produit *via* nos process pour ensuite faire des croisements. Cela est dû à la culture BI qu'on a puisqu'on est un fournisseur d'outils d'administrations, de PRA et de tests de régression pour les outils d'informatiques décisionnelle tels que *Business Objects* (de SAP) ou *Tableau* (de Salesforce).

Une partie de nos documents sont stockés sur drive et le cas du jour consiste à trouver des données stockées dans une spreadsheet google.

Comme on veut faire de l'extraction de données, ici je n'ai pas choisi d'utiliser les macro google spreadsheet mais l'API v2 décrite par google .

1.2. Code de base

De manière périodique, nous créons une feuille qui est toujours basée sur un tableau de 6 colonnes mais un nombre non défini de lignes.

Comme ce n'est qu'un billet, je passe tout de suite au code commenté

```
1 # service est obtenu par le code pour se connecter à spreadsheet
      avec les bons credentials
2 # puis j'ai sélectionné la spreadsheet dont je connais l'id, stocké
      dans la variable wanted_sheet_id
3 sheet_service = service.spreadsheets()
4 result =
      sheet_service.get(spreadsheetId=wanted_sheet_id).execute()["sheets"]
```

```
data = {}
for candidate_sheet in result:
    sheet_name = candidate_sheet['properties']['title']
    headers =
        sheet_service.values().get(spreadsheetId=wanted_sheet_id,
                                                  range=f"{sheet_name}!A1:G1").execu
        "values", [[]])
    data[candidate_sheet["properties"]["title"]] = {
       "values": []
    }
    values =
        sheet_service.values().get(spreadsheetId=wanted_sheet_id,
                                                range=f"{candidate_sheet['propertie
                                  .execute().get("values", [[]])
    for value_row in enumerate(values):
            data[candidate_sheet["properties"]["title"]]["values"].append({})
        for i, header in enumerate(headers[0]):
            # in google value_row is a tuple (index, values)
                data[candidate_sheet["properties"]["title"]]["values"][-1][hea
                = value_row[1][i]
```

Et voilà, on a un dictionnaire avec toutes les valeurs de toutes les sheets.

Seul problème: l'API google possède une limite de requête par heure et là on fait beaucoup de requêtes par feuille.

1.3. Améliorations pour éviter le *query rate*

Un bon moyen est d'utiliser la méthode `batchGet` sur le endpoint `values`. Cela implique de builder les range avant puis on va pouvoir tout traiter d'un coup:

```
ranges = []
# on va retenir les nom des sheet dans un tableau à côté
sheets = []
for candidate_sheet in result:
    sheet_name = candidate_sheet['properties']['title']
    ranges.append(f'{sheet_name}!A1:G1') # get headers
    ranges.append(f'{sheet_name}!A2:G') # values
    sheets.append(sheet_name)
values =
    sheet_service.values().batch_get(spreadsheetId=wanted_sheet_id,
                                     ranges=ranges)
```

```
11 # values est désormais un tableau contenant aux indices pairs les
       headers des feuilles
12 # et aux indices impairs les valeurs à extraire
```

Et voilà le travail.

Liste des abréviations

BI Business Intelligence, en Français Informatique décisionnelle. 1