AI2 - Bootcamp
Module 2 - Programmation Avancée et Gestion des données
Auteur : Yannick OREAL

Rapport de projet - Pipeline ETL

INTRODUCTION

La source des données provient du musée d'art The Art Institute of Chicago. Il s'agit d'un musée situé à Chicago aux Etats-Unis. Deuxième plus grand musée d'art du pays après le Metropolitan Museum of Art de New York, il abrite l'une des plus importantes collections d'art des Etats-Unis. Le musée met à disposition une api afin de parcourir la très grande quantité des œuvres exposées ou stockées. Une documentation complète explique comment interroger l'API : https://api.artic.edu/docs/#introduction

Cette API ne nécessite pas d'authentification, ce qui en fait un bon exemple pour une expérimentation simple. Dans le cadre du projet, nous récupérons un échantillon aléatoire d'œuvres, avec pour paramètre, le nombre d'œuvres souhaitées. Le code et le rapport de projet sont récupérables sur le github https://github.com/aXee808/AI2-module2-project

Les données récupérées dans les réponses json sont sous les structures suivantes :

```
{data : {id,                              → id de l'oeuvre
        title,                            → le titre de l'oeuvre
        image_id,                         → id de la photo de l'oeuvre
        artist_title,                     → le nom/prénom de l'auteur
        artist_display,                   → informations sur l'auteur
        artist_id,                        → id de l'auteur de l'oeuvre
        dimensions_detail:{...},          → détails des dimensions
        detail_medium,                    → informations sur l'oeuvre
        place_of_origin,                  → information sur le lieu de conception
        main_reference_number}}           → référence de l'année de rentrée stock


{data : {birth_date,                      → date de naissance de l'auteur
        death_date}}                      → date de décès de l'auteur
```

La structure du code s'organise autour de plusieurs fonctions dédiées à des tâches distinctes

Pour la partie extraction :

get_artic_edu_artwork_json(art_id)	interroge le endpoint artwork de l'API
get_artic_edu_artist_json(artist_id)	interroge le endpoint artist de l'API
display_artwork_informations(dic)	affiche les données récupérées par artwork
get_artwork_image(image_id,display_mode,export_mode,n)	récupère l'image au format jpeg, et permet un enregistrement local
image_to_byte_array(image)	convertie l'image en tableau de bits (pour intégration dans le dataframe)
get_n_random_artwork(n,csv_save,display_mode,export_mode,verbose_mode)	fonction principale qui appelle les autres fonctions, et qui gère la récupération aléatoire, retourne un objet DataFrame pandas

La boucle de récupération aléatoire de n oeuvres (sans commentaires du code) :

```
for i in range(1,n+1):
    id_valide = False
    while id_valide == False:
         artwork_id = random.randint(1,1000000)
         artwork_information_dic = get_artic_edu_artwork_json(artwork_id)

        if artwork_information_dic!=None:
             id_valide = True

        artist_add_info = get_artic_edu_artist_json(artwork_information_dic['artist_id'])
```

Pour la partie nettoyage :

clean_data(df)	prends le dataframe extrait en entrée et retire les œuvres sans titre, sans images, et dont on ignore la date de naissance de l'artiste.

Pour la partie transformation :

transform_data(df,with_image)	prends le dataframe nettoyé en entrée, et procède à un certain nombre de transformation : - nettoyage du champs 'medium_display' - ajout d'un champs 'support_type' à partir de 'medium_display' - ajout d'un champs 'type' à partir de 'medium_display'

	- ajout d'un champs 'stock_year' à partir de 'ref_number' - jointure avec un dataframe 'countries.csv' pour récupérer le pays et le continent (à partir du champs 'place_of_origine' - rationalisation des champs 'country' et 'continent' et remplacement des NaN par des valeurs "Other" - conversion des champs 'birth_date' et 'death_date' en entier - suppression des champs ref_number, place_of_origin

Pour la partie chargement (dans une base SQLite3) :

load_dataframe_to_sqlite_db(df)	prends le dataframe nettoyé en entrée, et le sauvegarde dans une base SQLite en local (fichier .sqlite)

Et on peut lancer le pipeline complet avec la fonction :

start_pipeline_etl(n)	prends le nombre d'œuvres à récupérer, et enchaîne les 4 étapes, l'extraction, le nettoyage, la transformation, et le chargement.