I) Python et Data Engineering

Objectif : Réaliser un code clair et proprement structuré. Mettre en avant les éléments considérés comme essentiels pour du code utilisable dans un environnement de production. Mettre l'accent sur vos connaissances en conception de jobs de manipulation de données ainsi que les bonnes pratiques python.

1. Les données

Vous avez à votre disposition les 4 fichiers de données suivants :

drugs.csv : contient les noms de drugs (des médicaments) avec un id (atccode) et un nom (drug)
pubmed.csv : contient des titres d'articles PubMed (title) associés à un journal (journal) à une date donnée (date) ainsi qu'un id (id)
pubmed.json : même structure que pubmed.csv mais en format JSON
clinical_trials.csv : contient des publications scientifiques avec un titre (scientific_title), un id (id), un journal (journal) et une date (date).

2. Le travail à réaliser

L'objectif est de construire une data pipeline permettant de traiter les données définies dans la partie précédente afin de générer le résultat décrit dans la partie 3.

Pour ce faire, vous devez mettre en place un projet en python organisé de la manière qui vous paraît la plus pertinente pour résoudre ce problème. Nous attendons que vous identifiiez une structure de projet et une séparation des étapes nécessaires qui permettent de mettre en évidence vos connaissances autour du développement de jobs data en python.

Il faudra essayer de considérer les hypothèses de travail suivantes :

- Certaines étapes de votre data pipeline pourraient être réutilisées par d'autres data pipelines
- On suppose que votre travail devra être intégré dans un orchestrateur de jobs (de type DAG) par la suite, votre code et la structure choisie doivent donc favoriser cette intégration
- Votre code doit respecter les pratiques que vous mettriez en place dans un cadre professionnel au sein d'une équipe de plusieurs personnes

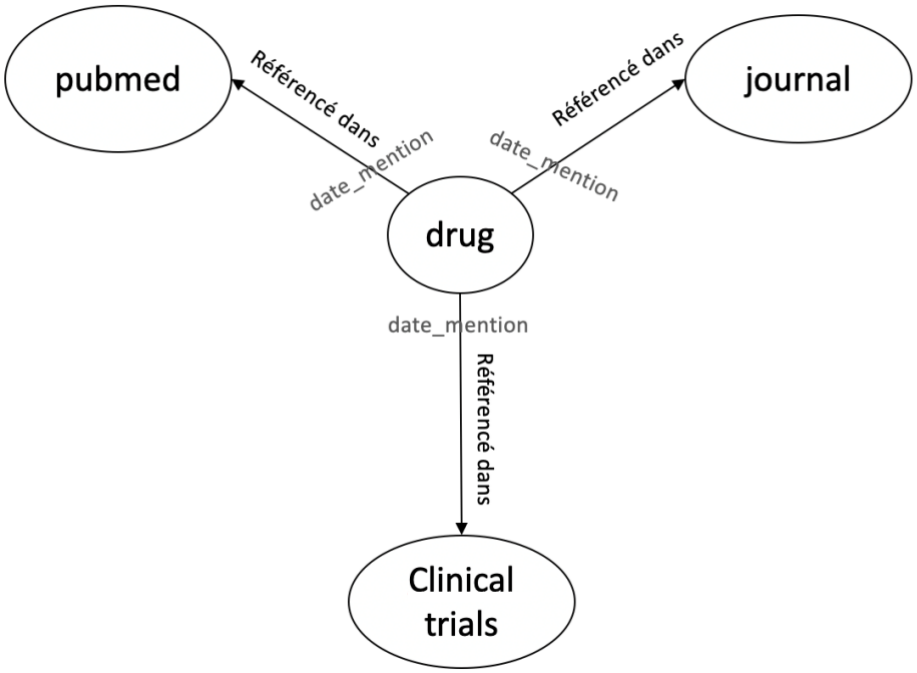
Nous laissons volontairement un cadre assez libre pour voir votre manière de structurer un projet, de rédiger votre code et de mettre en place les éléments qui vous semblent essentiels dans un projet d'équipe. N'hésitez pas à argumenter votre proposition et les choix que vous faites si nécessaire.

3. Data pipeline

Votre data pipeline doit produire en sortie un fichier JSON qui représente un graphe de liaison entre les différents médicaments et leurs mentions respectives dans les différentes publications PubMed, les différentes publications scientifiques et enfin les journaux avec la date associée à chacune de ces mentions. La représentation ci-dessous permet de visualiser ce qui est attendu. Il peut y avoir plusieurs manières de modéliser cet output et vous pouvez justifier votre vision :

Règles de gestion :

- Un drug est considéré comme mentionné dans un article PubMed ou un essai clinique s'il est mentionné dans le titre de la publication.
- Un drug est considéré comme mentionné par un journal s'il est mentionné dans une publication émise par ce journal.



4. Traitement ad-hoc

Vous devez aussi mettre en place (hors de la data pipeline, vous pouvez considérer que c'est une partie annexe) une feature permettant de répondre à la problématique suivante :

- Extraire depuis le json produit par la data pipeline le nom du journal qui mentionne le plus de médicaments différents ?

5. Le rendu

Vous pouvez partager votre proposition sur un repo git hébergé chez le fournisseur de votre choix (github, gitlab ou autre).

6. Pour aller plus loin

Par retour de mail (ou directement sur le repo git si vous le souhaitez), vous pouvez répondre aux questions suivantes (ne nécessite pas d'implémentation dans votre projet) :

Quels sont les éléments à considérer pour faire évoluer votre code afin qu'il puisse gérer de grosses volumétries de données (fichiers de plusieurs To ou millions de fichiers par exemple) ?
Pourriez-vous décrire les modifications qu'il faudrait apporter, s'il y en a, pour prendre en considération de telles volumétries ?

II) SQL

Objectif : Réaliser des requêtes SQL claires et facilement compréhensibles.

1. Les données

Nous avons les tables suivantes :

TRANSACTIONS

Cette table contient des données transactionnelles avec les infos suivantes :

- date : date à laquelle la commande a été passée
- order_id : identifiant unique de la commande
- client_id : identifiant unique du client
- prod_id : identifiant unique du produit acheté
- prod_price : prix unitaire du produit
- prod_qty : quantité de produit achetée

Echantillon de la table TRANSACTION :

date	**order_id**	**client_id**	**prop_id**	**prod_price**	**prod_qty**
01/01/20	1234	999	490756	50	1
01/01/20	1234	999	389728	3,56	4
01/01/20	3456	845	490756	50	2
01/01/20	3456	845	549380	300	1
01/01/20	3456	845	293718	10	6

PRODUCT_NOMENCLATURE

Cette table contient le référentiel produit c'est à dire les méta-données du produit. On y trouve les infos suivantes :

- product_id : identifiant unique du produit
- product_type : type de produit (DECO ou MEUBLE)
- product_name : le nom du produit

Echantillon de la table PRODUCT_NOMENCLATURE :

product_id	**product_type**	**product_name**
490756	MEUBLE	Chaise
389728	DECO	Boule de Noël
549380	MEUBLE	Canapé
293718	DECO	Mug

2. Première partie du test

Je vous propose de commencer par réaliser une requête SQL simple permettant de trouver le chiffre d'affaires (le montant total des ventes), jour par jour, du 1er janvier 2019 au 31 décembre 2019. Le résultat sera trié sur la date à laquelle la commande a été passée.
Je rappelle que la requête doit être claire : n'hésitez pas à utiliser les mot clefs AS permettant de nommer les champs dans SQL.

Echantillon des résultats de la requête :

date	**ventes**
01/01/2020	524240
02/01/2020	520918
03/01/2020	526983
04/01/2020	520987
05/01/2020	524879
06/01/2020	524436

3. Seconde partie du test

Réaliser une requête un peu plus complexe qui permet de déterminer, par client et sur la période allant du 1er janvier 2019 au 31 décembre 2019, les ventes meuble et déco réalisées.

Echantillon des résultats de la requête :

client_id	**ventes_meuble**	**ventes_deco**
999	50	14,24
845	400	60