Projet data filter

Le but de ce projet est de créer un programme permettant de charger, sauvegarder, filtrer, trier et afficher des données depuis des fichiers au format CSV Json XML ou yaml.

On considérera que les données sont une série(un tableau) de données structurées avec les mêmes champs et pourront donc être représentées par une liste de dictionnaires.

Les champs pourront avoir comme valeur :

- des entiers
- des réels
- des chaînes de caractères
- des booléens
- des listes de valeurs des types ci-dessus

Exemples :

Une structure **student** avec :

- des champs **firstname** et **lastname** de type chaîne de caractère
- un champs **age** de type entier
- un champs **apprentice** de type booléen
- un champs **grades** de type liste d'entiers

Une structure **item** avec :

- des champs **name** et **category** de type chaîne de caractère
- un champs **price** de type réel
- un champs **quantity** de type entier

Chargement/Sauvegarde

Votre programme doit pouvoir charger et sauvegarder les données au minimum aux formats CSV et JSON.

Pour aller plus loin, il pourra également proposer le XML et le YAML

Stats

Votre programme devra pouvoir afficher la structure, c'est à dire les noms des champs, avec :

- pour chacun des champs représentant un nombre : le min, le max, et la valeur moyenne
- pour chacun des champs représentant un booléen : le % de vrai et de faux
- pour chacun des champs représentant une liste : les mêmes stats que pour ceux représentant un nombre, en considérant ici la taille de la liste.

Filtrage

Votre programme doit permettre de filtrer les données, pour ne garder qu'un sous-ensemble des données.

Le filtrage doit au minimum permettre de tester en comparant avec une valeur :

1. dans le cas des chaînes de caractères, on comparera en utilisant l'ordre lexicographique
2. dans le cas des listes, on utilise le nombre d'éléments.

Pour aller plus loin, vous pourrez permettre d'autres critères de filtrage :

- pour les chaînes de caractères : contient, commence/finit par
- pour les listes, des règles plus complexes : tous les éléments, le min/max/moyenne

Pour aller plus loin, vous pouvez aussi permettre de comparer deux champs entre eux, par exemple :

- le prénom est avant le nom dans l'ordre lexicographique
- la note de math est plus haute que la note d'anglais

Ou encore comparer par rapport aux statistiques globales :

- plus vieux que la moyenne
- moins cher que 75% des items

Vous pourrez filtrer sur une combinaison de champs : par exemple, sur les items, vous pouvez filtrer ceux ayant une valeur globale (prix x quantité) supérieure à un seuil.

Tri

Votre programme doit également pouvoir trier les éléments, au minimum par les valeurs de leur champs.

Pour aller plus loin, vous pouvez ensuite proposer de trier via une combinaison de champs (valeur globale pour les items, par exemple), ou de trier sur plusieurs critères successifs, en cas d'égalité (nom, puis prénom pour les étudiants, par exemple)

Interface

La façon d’appliquer les différentes opérations est libre :

- ligne de commande
- menu
- interface graphique

La qualité de l’interface et la facilité à l’utiliser entrent en compte dans la notation.

Améliorations

Pour aller plus loin, vous pouvez également proposer :

- la gestion d’un historique des filtrages, avec des undo/redo
- la possibilité d’ajouter ou retirer des champs

Notation

Pour avoir la moyenne, il faut au minimum faire ce qui est marqué en noir ci-dessus, de façon convenable (pas de bugs, bon découpage du programme).

Le programme doit être un script exécutable en console au moyen de l'interpréteur python, et ne pas dépendre d’un jupyter ou autre.

Pour avoir plus de la moyenne, il faut ajouter des améliorations comme celles proposées en bleu ci-dessus.