TP – Traitement de données en tables

Exploration du fichier des prénoms en France

Première NSI

Objectif. Dans ce TP, vous allez explorer un jeu de données réel issu de l'INSEE : le fichier des prénoms attribués aux enfants nés en France depuis 1900. Vous apprendrez à importer, rechercher, trier et fusionner des données organisées en tables. **Durée :** 5-6 heures.

Compétences à acquérir

Compétences	🙁	😐	🙂	😀
Importer une table depuis un fichier CSV				
Rechercher les lignes d'une table vérifiant des critères exprimés en logique propositionnelle				
Recherche de doublons, tests de cohérence				
Trier une table suivant une colonne				
Construire une nouvelle table en combinant les données de deux tables				
Notion de domaine de valeurs				

Conseil. Les données utilisées proviennent de l'INSEE et sont disponibles sur `https://www.insee.fr/fr/statistiques/8595130`. Elles sont placées sous licence ouverte.

Présentation des données

Le fichier `prenoms.csv` contient les prénoms attribués aux enfants nés en France entre 1900 et 2024. Il est composé de quatre colonnes (descripteurs) :

- `sexe` : `1` pour masculin, `2` pour féminin
- `prenom` : le prénom (en majuscules, sans accents)
- `periode` : l'année de naissance
- `valeur` : le nombre de naissances (arrondi au multiple de 5)

Exemple (extrait du fichier).

```
sexe;prenom;periode;valeur
1;AARON;1980;5
1;AARON;1990;25
1;AARON;2000;120
1;AARON;2010;1605
2;EMMA;2000;6845
2;EMMA;2010;6400
```

① Import des données

Exercice 1 ***Chargement du fichier CSV*** Le fichier `prenoms.csv` est à placer dans votre répertoire de travail.

1. Complétez la fonction `charger_csv` ci-dessous. Cette fonction prend en paramètre le nom d'un fichier CSV et renvoie une liste de dictionnaires.

   Remarque. La fonction `csv.DictReader` lit toutes les valeurs comme des chaînes de caractères. Par exemple, `ligne['valeur']` vaut `'295'` (une chaîne) et non `295` (un entier). Il faut donc convertir les valeurs numériques en entiers pour pouvoir faire des calculs.

```python
import csv

def charger_csv(nom_fichier):
    """
    Charge un fichier CSV et renvoie une liste de dictionnaires.
    Chaque dictionnaire représente une ligne du fichier.
    """
    table = []
    with open(nom_fichier, 'r', encoding='utf-8') as fichier:
        lecteur = csv.DictReader(fichier, delimiter=';')
        for ligne in lecteur:
            # À COMPLÉTER : convertir les chaînes en entiers
            ligne['valeur'] = ...
            ligne['periode'] = ...
            table.append(ligne)
    return table

# Chargement des données
prenoms = charger_csv('prenoms.csv')
```

2. Écrivez l'instruction qui affiche le nombre total de lignes dans la table.
3. Écrivez une boucle qui affiche les 5 premières lignes de la table. Que représente chaque ligne ?
4. Écrivez l'instruction qui affiche les descripteurs (les clés) de la première ligne.

Exercice 2 ***Tests de cohérence*** Avant d'exploiter des données, il est important de vérifier leur cohérence.

1. Écrivez une fonction `verifier_sexe(table)` qui prend en paramètre une table et renvoie `True` si toutes les valeurs du descripteur `sexe` sont `'1'` ou `'2'`, et `False` sinon.
2. Écrivez une fonction `annees_presentes(table)` qui prend en paramètre une table et renvoie la liste des années présentes (sans doublons).
3. En utilisant cette fonction, vérifiez s'il y a des années manquantes entre 1900 et 2024.

② Recherche dans une table

Exercice 3 ***Recherche simple***

1. Écrivez une fonction `rechercher_prenom(table, prenom)` qui prend en paramètres une table et un prénom (chaîne de caractères), et renvoie la liste de toutes les lignes dont le descripteur `prenom` est égal au prénom

cherché.

2. Utilisez cette fonction pour récupérer toutes les lignes correspondant au prénom `'CLAUDE'`. Affichez, pour chaque ligne, l'année, le nombre de naissances et le sexe.
3. Écrivez une fonction `total_naissances(table, prenom)` qui prend en paramètres une table et un prénom, et renvoie le nombre total de naissances pour ce prénom (somme des valeurs du descripteur `valeur`).
4. Calculez le total des naissances pour le prénom `'MARIE'` puis pour le prénom `'JEAN'`.
5. **Question personnelle :** recherchez votre propre prénom (en majuscules) et calculez le nombre total de naissances. Que pouvez-vous dire sur la popularité de votre prénom ?

Exercice 4 ***Recherche avec plusieurs critères*** On souhaite rechercher des lignes vérifiant plusieurs critères à la fois.

1. Écrivez une fonction `rechercher(table, sexe, annee)` qui prend en paramètres une table, un sexe (`'1'` ou `'2'`) et une année (entier), et renvoie la liste des lignes vérifiant ces deux critères simultanément.
2. Utilisez cette fonction pour trouver tous les prénoms féminins attribués en 2020. Combien y en a-t-il ?
3. Combien de prénoms masculins différents ont été attribués en 1950 ?
   Indication : construisez d'abord la liste des lignes correspondantes, puis extrayez les prénoms sans doublons.

Exercice 5 ***Recherche de doublons*** Certains prénoms sont attribués à la fois à des garçons et à des filles. On appelle ces prénoms des « prénoms mixtes ».

1. Écrivez une fonction `liste_prenoms_par_sexe(table, sexe)` qui prend en paramètres une table et un sexe, et renvoie la liste des prénoms (sans doublons) correspondant à ce sexe.
2. En utilisant cette fonction, construisez la liste `masculins` des prénoms masculins et la liste `feminins` des prénoms féminins.
3. Écrivez une fonction `prenoms_mixtes(masculins, feminins)` qui prend en paramètres ces deux listes et renvoie la liste des prénoms présents dans les deux listes.
4. Combien y a-t-il de prénoms mixtes ? Donnez quelques exemples.

③ Tri d'une table

Rappel sur les fonctions anonymes (lambda)

La fonction `sorted(liste, key=fonction)` permet de trier une liste. Le paramètre `key` indique la fonction à utiliser pour extraire la valeur de tri de chaque élément.
On utilise souvent une **fonction anonyme** (ou **lambda**) pour définir cette fonction de tri. Par exemple :

```
lambda x: x['periode']
```

est équivalent à :

```
def extraire_annee(x):
    return x['periode']
```

L'avantage de la syntaxe `lambda` est sa concision : on peut définir la fonction directement dans l'appel à `sorted`, sans avoir à lui donner un nom.

Exercice 6 ***Tri suivant une colonne***

1. En utilisant `rechercher_prenom`, récupérez toutes les lignes correspondant au prénom `'EMMA'`.
2. Triez cette liste par année croissante en utilisant :

```
emma_trie = sorted(emma, key=lambda x: x['periode'])
```

3. Affichez l'évolution du prénom EMMA entre 2000 et 2024 (année et nombre de naissances).
4. Écrivez une fonction `top_prenoms(table, annee, sexe, n)` qui prend en paramètres une table, une année, un sexe et un entier `n`, et renvoie les `n` prénoms les plus attribués cette année-là pour ce sexe, triés par nombre de naissances décroissant.
   Indication : utilisez `sorted(..., key=lambda x: x['valeur'], reverse=True)`.
5. Affichez le top 10 des prénoms féminins en 2024.

Exercice 7 ***Analyse de tendances***

1. Écrivez une fonction `annee_pic(table, prenom)` qui prend en paramètres une table et un prénom, et renvoie l'année où ce prénom a été le plus attribué (c'est-à-dire l'année avec le plus grand nombre de naissances).
   Indication : utilisez la fonction `max(liste, key=...)` pour trouver l'élément ayant la plus grande valeur pour un critère donné.
2. Calculez l'année du pic pour les prénoms suivants : `'BRIGITTE'`, `'SYLVIE'`, `'NATHALIE'`, `'STEPHANIE'`, `'LEA'`. Que remarquez-vous sur la succession de ces prénoms dans le temps ?
3. **Question de réflexion :** à votre avis, quels facteurs peuvent influencer les cycles de popularité des prénoms ? Pensez aux médias, aux personnalités célèbres, aux événements historiques...

④ Fusion de tables

On dispose d'un second fichier `origines.csv` (à déposer dans votre répertoire de travail) contenant l'origine étymologique de certains prénoms :

```
prenom;origine;signification
EMMA;germanique;universel ou puissant
GABRIEL;hebreu;force de Dieu
LEA;hebreu;fatiguee ou lionne
LUCAS;grec;lumiere
JADE;espagnol;pierre verte
LOUIS;germanique;illustre au combat
```

Exercice 8 ***Chargement et fusion de deux tables***

1. Chargez le fichier `origines.csv` dans une table `origines` (sans conversion de types, car toutes les valeurs sont des chaînes).

```
def charger_csv_simple(nom_fichier):
    table = []
    with open(nom_fichier, 'r', encoding='utf-8') as fichier:
        lecteur = csv.DictReader(fichier, delimiter=';')
        for ligne in lecteur:
```

```
            table.append(ligne)
    return table

origines = charger_csv_simple('origines.csv')
```

2. Écrivez une fonction `fusionner(table1, table2, cle)` qui prend en paramètres deux tables et le nom d'un descripteur commun (la clé), et renvoie une nouvelle table contenant les lignes de `table1` enrichies des informations de `table2` lorsque les valeurs de la clé correspondent.
   Principe : pour chaque ligne de `table1`, on cherche dans `table2` une ligne ayant la même valeur pour le descripteur `cle`. Si on en trouve une, on crée une nouvelle ligne combinant les informations des deux.

```
def fusionner(table1, table2, cle):
    resultat = []
    for ligne1 in table1:
        # À compléter : parcourir table2
        # À compléter : vérifier si la clé correspond
        # À compléter : créer la nouvelle ligne enrichie
    return resultat
```

3. Fusionnez les tables `prenoms` et `origines` sur la clé `'prenom'`.
4. Pour chaque prénom présent dans `origines`, affichez son origine, sa signification et le nombre de naissances en 2024.

Exercice 9 ***Notion de domaine de valeurs***

1. Écrivez une fonction `valeurs_distinctes(table, descripteur)` qui prend en paramètres une table et le nom d'un descripteur, et renvoie la liste des valeurs distinctes (sans doublons) de ce descripteur dans la table.
2. Appliquez cette fonction au descripteur `'origine'` de la table `origines`. La liste obtenue s'appelle le **domaine de valeurs** du descripteur.
3. Quel est le domaine de valeurs du descripteur `'sexe'` dans la table `prenoms` ?

POUR ALLER PLUS LOIN

Exercice 10 ***Synthèse : portrait d'un prénom*** Écrivez une fonction `portrait_prenom(table, prenom)` qui prend en paramètres une table et un prénom, et affiche un résumé complet de ce prénom :

— le nombre total de naissances ;
— l'année du pic de popularité et le nombre de naissances cette année-là ;
— la répartition entre masculin et féminin (en pourcentage) ;
— la première et la dernière année d'apparition dans les données.

Testez votre fonction avec les prénoms `'CAMILLE'`, `'DOMINIQUE'` et `'CLAUDE'`.