EXERCICE 3 (8 points)

Cet exercice porte sur la programmation Python (dictionnaire), les bases de données relationnelles et les requêtes SQL.

Le Tour de France est une course cycliste qui se déroule chaque année. Chaque jour, les coureurs s'affrontent pour remporter l'étape du jour, ce qui détermine un classement d'étape. Le coureur avec le temps cumulé le plus bas sur l'ensemble des étapes mène le classement général. Chaque participant est repéré par un dossard et appartient à une équipe. En 2023, 22 équipes de 8 coureurs, soit 176 cyclistes ont pris le départ du tour.

Partie A

Dans cette partie, nous allons utiliser trois dictionnaires.

Le premier, appelé `participants`, a pour clés les noms complets des coureurs et pour valeurs les numéros de dossard correspondants.

Le deuxième, appelé `temps_etapes`, utilise les numéros de dossard comme clés et contient une liste des temps d'arrivée de chaque étape en seconde.

Le troisième, appelé `classement_general`, utilise également les numéros de dossard comme clés et indique le classement général mis à jour à la fin de chaque nouvelle étape.

Par exemple, à la fin de la quatrième étape, voilà les trois premiers éléments de ces dictionnaires :

```
participants = {"VINGEGAARD Jonas": 1, "BENOOT Tiesj": 2, "KELDERMAN
Wilco": 3, ...}

temps_etapes = {1: [15781, 17199, 16995, 15928], 2: [15960, 17199,
16995, 15928], ...}

classement_general = {1: 6, 2: 30, 3: 13, ...}
```

1. En utilisant ces dictionnaires, écrire une instruction permettant d'obtenir :

- le numéro de dossard de `PHILIPSEN Jasper` ;
- le classement général de `PHILIPSEN Jasper` ;
- le temps, en seconde, mis par le cycliste `PINOT Thibaut` pour courir la quatrième étape.

2. Écrire une fonction `calcul_temps_total` qui a pour paramètre le numéro d'un dossard `d` et qui renvoie le temps total en seconde mis par ce coureur depuis le départ du tour de France.

3. Le dictionnaire `temps_etapes` étant remis à jour après la fin d'une étape, recopier et compléter les lignes 8, 9 et 14 du programme suivant afin que le dictionnaire `classement_general` soit aussi mis à jour.

```
1  classement = []
2
3  for numero_dossard in temps_etapes:
4      element = (numero_dossard,
                  calcul_temps_total(numero_dossard))
5      classement.append(element)
6      pos = len(classement) - 2
7
8      while pos >= 0 and element[...] < classement[pos][...]:
9          classement[pos + 1] = ...
10         pos = pos - 1
11         classement[pos + 1] = element
12
13 for i in range(len(classement)):
14     classement_general[...] = i + 1
```

On suppose qu'on dispose d'un tableau `tableau_temps` composé de tuples contenant le numéro du dossard, le nom, et le temps total en seconde de chaque coureur, trié par ordre croissant de temps. On donne ci-dessous un aperçu du début du tableau :

```
tableau_temps = [(1, "VINGEGAARD Jonas", 65903),
                 (3, "KELDERMAN Wilco", 65987),
                 (2, "BENOOT Tiesj", 66082),
                 ...]
```

On souhaite créer une variable Python `tableau_final` de type liste de listes :

```
[[1, "VINGEGAARD Jonas", 65903],
 [3, "KELDERMAN Wilco", 84],
 [2, "BENOOT Tiesj", 179]]
...
```

Pour le premier, la troisième valeur de la première liste est le temps total mis par le vainqueur. Pour les autres coureurs, la troisième valeur des autres listes est l'écart de temps mis avec le premier.

Par exemple : 84 = 65987 - 65903 et 179 = 66082 - 65903.

4. Recopier et compléter le programme pour qu'il en soit ainsi :

```
tableau_final = []
difference_temps = 0
premier = True
for ligne in tableau_temps:
    coureur = [ligne[0]]
    coureur.append(ligne[1])
    if premier:
        temps_premier = ligne[2]
        coureur.append(temps_premier)
        premier = False
    else:
        difference_temps = ligne[2] - ...
        coureur.append(...)
    tableau_final.append(...)
```

Partie B

Dans cet exercice, on pourra utiliser les mots clés suivants du langage SQL :

`SELECT`, `FROM`, `WHERE`, `JOIN`, `ON`, `INSERT INTO`, `VALUES`, `MIN`, `MAX`, `OR`, `AND` et `ORDER BY`.

Si `propriete` est un des attributs d'une relation, les fonctions d'agrégation `MIN(propriete)`, `MAX(propriete)`, `SUM(propriete)` renvoient, respectivement, la plus petite, la plus grande valeur et la somme des valeurs des attributs sélectionnés.

On considère la base de données du tour de France 2023 dont le schéma relationnel est donné ci-dessous :

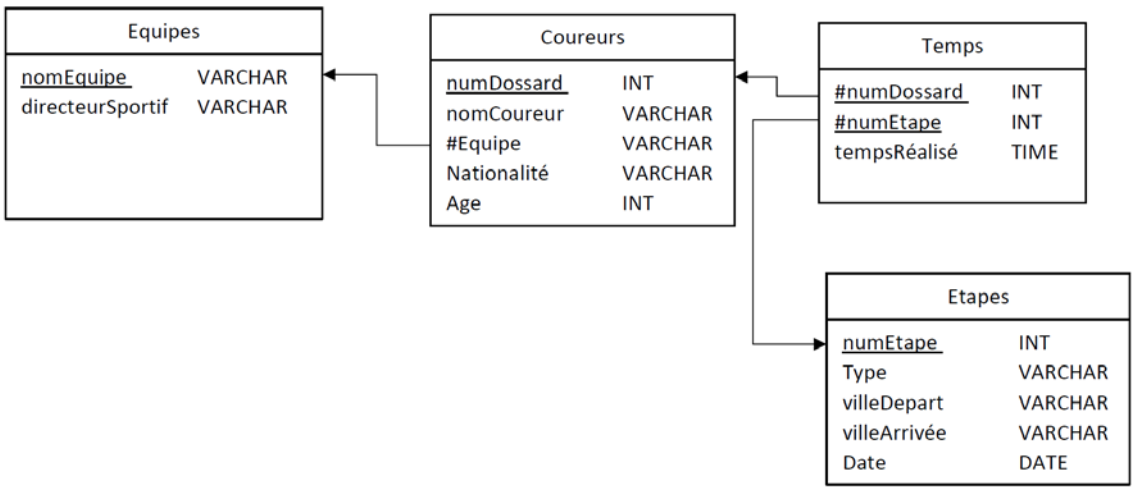

Figure 1. Schéma Relationnel

Dans ce schéma, les clés primaires sont soulignées et les clés étrangères sont précédées du symbole #.

5. Expliquer pourquoi, dans la relation `Temps`, il est nécessaire de prendre le couple (`numDossard`, `NumEtape`) comme clé primaire.

6. Expliquer ce que renvoie la requête SQL suivante :

```
SELECT nomCoureur
FROM Coureurs
WHERE Equipe = 'Cofidis';
```

7. Écrire une requête SQL permettant d'obtenir les dates de toutes les étapes de type `'contre-la-montre'` du tour de France 2023.

8. Écrire une requête SQL permettant d'obtenir le nom du directeur sportif du coureur BARDET Romain.

9. À la fin de la cinquième étape, on veut actualiser la table `Temps` avec les données du jour. Expliquer pourquoi la suite des deux requêtes SQL ci-dessous provoque une erreur.

```
INSERT INTO Temps VALUES (1, 5, 14267);
INSERT INTO Etapes VALUES(5, 'Montagne', 'Pau', 'Laruns',
05/07/2023);
```

10. Expliquer quelle modification est à effectuer pour apporter une solution au problème constaté à la question précédente.

11. Écrire une requête SQL donnant le temps total en course mis par BARDET Romain depuis le départ du tour de France 2023.