EXERCICE 3 (8 points)

Cet exercice porte sur la programmation Python, la programmation orientée objet, les bases de données relationnelles et les requêtes SQL.

Partie A

Une entreprise, présente sur différents sites en France, attribue à chacun de ses employés un numéro de badge unique.

Dans le tableau ci-dessous, on donne le numéro de badge, le nom, le prénom et les années de naissance et d'entrée dans l'entreprise de quelques salariés.

numéro badge	nom	prénom	année de naissance	année d'entrée
112	LESIEUR	Isabelle	1982	2005
2122	VASSEUR	Adrien	1962	1980
135	HADJI	Hakim	1992	2015

Pour chaque personne, on souhaite stocker les informations dans un objet de la classe `Personne` définie ci-dessous :

```
1 class Personne():
2     def __init__(self, num, n , p , a_naiss, a_entree):
3         self.num_badge = num
4         self.nom = n
5         self.prenom = p
6         self.annee_naissance = a_naiss
7         self.annee_entree = a_entree
```

1. Écrire à l'aide du tableau précédent, l'instruction permettant de créer l'objet `personneA` de la première personne du tableau : LESIEUR Isabelle.

2. Donner l'instruction permettant d'obtenir le numéro de badge de l'objet `personneA` instancié à la question précédente.

On souhaite ajouter une méthode `annee_anciennete` à la classe `Personne` qui donne le nombre d'années d'ancienneté d'une personne au sein de l'entreprise. Par exemple : Madame LESIEUR Isabelle a une ancienneté dans l'entreprise de 19 ans en considérant que nous sommes en 2024.

3. Recopier et compléter le code suivant de la méthode `annee_anciennete` :

```
1     def annee_anciennete(self):
2         return ...
```

On considère la classe `Personnel` qui modélise la liste du personnel d'une entreprise et dont le début de l'implémentation est la suivante :

```
1 class Personnel:
2     def __init__(self):
3         self.liste = []
```

4. Écrire la méthode `ajouter` permettant d'ajouter un objet de type `Personne` à la liste du personnel de l'entreprise de la classe `Personnel`.

5. Écrire la méthode `effectif` de la classe `Personnel`. Cette méthode devra renvoyer le nombre de personnes présentes dans l'entreprise.

6. Recopier et compléter la méthode `donne_nom` de la classe `Personnel`. Cette méthode prend en paramètre le numéro de badge d'une personne et renvoie le nom de la personne correspondant à ce badge si elle existe, ou `None` sinon.

```
1       def donne_nom(..., num):
2           for elt in self.liste:
3               if ... == num:
3                   return ...
4           return ...
```

7. Lors de la célèbre cérémonie des vœux, l'entreprise souhaite mettre à l'honneur les personnes ayant exactement 10 ans d'ancienneté dans l'entreprise. Écrire une méthode de la classe `Personnel nb_personne_honneur` qui prend en paramètre l'année de la cérémonie et qui retourne le nombre de personne(s) à mettre à l'honneur.

8. Écrire une méthode `plus_anciens` de la classe `Personnel` qui retourne la liste des numéros de badge des personnes ayant la plus grande ancienneté dans l'entreprise.

Partie B

On utilise maintenant une base de données relationnelle. La table `Personnel` dont un extrait est donné ci-dessous contient toutes les données importantes sur le personnel de l'entreprise. L'attribut `num_centre` désigne le numéro du centre dans lequel travaille une personne.

Personnel					
num_badge	nom	prenom	num_centre	annee_naiss	annee_debut
112	LESIEUR	Isabelle	1	1982	2005
2122	VASSEUR	Adrien	2	1962	1980
135	HADJI	Hakim	1	1992	2015

L'attribut `num_badge` est la clé primaire pour la table `Personnel`.

9. Décrire par une phrase en français le résultat de la requête SQL suivante :

```
SELECT nom, prenom
FROM Personnel
WHERE num_centre = 2;
```

10. Monsieur HADJI Hakim vient d'obtenir une mutation pour le centre numéro `3`. Donner la requête permettant de modifier son numéro de centre sachant que son numéro de badge est 135.

On souhaite proposer plus d'informations sur les différents centres de l'entreprise. Pour cela, on crée une deuxième table `Centre` avec les attributs suivants :

- `num` de type INT ;
- `nom` de type TEXT ;
- `num_tel` de type TEXT ;
- `ville` de type TEXT.

Table `Centre`			
`num`	`nom`	`num_tel`	`ville`
1	Normandie	0450646859	Caen
2	PACA	0450646859	Marseille

11. Expliquer l'intérêt d'utiliser deux tables (`Personnel` et `Centre`) au lieu de regrouper toutes les informations dans une seule table.

12. Expliquer comment les tables `Centre` et `Personnel` sont mises en relation.

13. Écrire une requête permettant d'avoir les noms des personnes travaillant dans le centre de Lille et ayant été embauchées entre 2015 (inclus) et 2020 (inclus).

Le centre de Normandie vient d'être fermé, mais les personnes de ce centre n'ont pas encore été affectées dans leur nouveau centre. On souhaite mettre à jour la table `Centre` en premier à l'aide de la requête suivante.

```
DELETE *
FROM Centre
WHERE nom = 'Normandie';
```

14. Expliquer pourquoi cette requête a renvoyé une erreur.