Baccalauréat Général – Enseignement de Spécialité
Numérique et Sciences Informatiques

Livret d'entrainement sur les bases de données

By DarkSATHI Li

Exercice 1 (3 points)

Cet exercice porte sur les bases de données et le langage SQL.

On considère une gestion simplifiée des voyages dans l'espace. La base de données utilisée est constituée de quatre relations nommées `Astronaute`, `Fusee`, `Equipe` et `Vol`. Voici le contenu des tables `Astronaute`, `Fusee`, `Equipe` et `Vol`.

Les clés primaires sont soulignées et les clés étrangères sont précédées d'un `#`.

Tables de la base de données

Astronaute				
id_astronaute	nom	prenom	nationalite	nb_vols
1	'PESQUET'	'Thomas'	'français'	2
2	'AMSTRONG'	'Neil'	'américain'	8
3	'MAURER'	'Mathias'	'allemand'	1
4	'MCARTHUR'	'Megan'	'américain'	5

Fusee			
id_fusee	modele	constructeur	nb_places
1	'Falcon 9'	'SpaceX'	6
2	'Starship'	'SpaceX'	100
3	'Soyouz'	'TsSKB Progress'	2
4	'SLS'	'Boeing'	6

Equipe	
#id_vol	#id_astronaute
1	1
1	2
1	3
2	1
2	3
3	1
3	2
3	4
4	2
4	4

Vol		
id_vol	#id_fusee	Date
1	1	'12/09/2022'
2	4	'25/10/2022'
3	3	'18/11/2022'
4	2	'23/12/2022'

Questions

1. On s'intéresse ici à la notion de clés primaire et étrangère.
   (a) Donner la définition d'une clé primaire.
   (b) Dans la table `Astronaute`, la clé primaire est `id_astronaute`. Expliquer pourquoi cette requête SQL renvoie une erreur :

   ```
   INSERT INTO Astronaute
   VALUES (3, 'HAIGNERE', 'Claudie', 'français', 3);
   ```

   (c) Le schéma relationnel de la table `Astronaute` est :

   ```
   Astronaute (id_astronaute : INT, nom : TEXT, prenom : TEXT,
               nationalite : TEXT, nb_vols : INT)
   ```

   Écrire le schéma relationnel de la table `Fusee` en précisant le domaine de chaque attribut.
2. On s'intéresse ici à la récupération d'informations issues de la base de données.
   (a) Écrire le résultat que la requête suivante renvoie :

   ```
   SELECT COUNT(*)
   FROM Fusee
   WHERE constructeur = 'SpaceX';
   ```

   (b) Écrire une requête SQL qui renvoie le modèle et le constructeur des fusées ayant au moins quatre places.
   (c) Écrire une requête SQL qui renvoie les noms et prénoms des astronautes dans l'ordre alphabétique du nom.
3. (a) Recopier et compléter les requêtes SQL suivantes permettant d'ajouter un cinquième vol avec la fusée 'Soyouz' le 12/04/2023 avec l'équipage composé de PESQUET Thomas et MCARTHUR Megan. On ne s'intéresse pas ici à la mise à jour qui suivra.

   ```
   INSERT INTO Vol VALUES(.............................);
   INSERT INTO Equipe VALUES(.....................);
   INSERT INTO ............... VALUES(.....................);
   ```

   (b) Écrire une requête SQL permettant d'obtenir le nom et le prénom des astronautes ayant décollé le '25/10/2022'.

Exercice 2 (3 points)

Cet exercice porte sur les bases de données relationnelles et le langage SQL.

L'énoncé de cet exercice utilise les mots-clés du langage SQL suivants : `SELECT`, `FROM`, `WHERE`, `JOIN...ON`, `UPDATE...SET`, `DELETE`, `INSERT INTO...VALUES`, `ORDER BY`.

- La clause `ORDER BY` suivie d'un attribut permet de trier les résultats par ordre croissant des valeurs de l'attribut.

Radio France souhaite créer une base de données relationnelle contenant les podcasts des émissions de radio. Pour cela, elle utilise le langage SQL. Elle crée :

- une relation (ou table) `podcast` qui contient le thème et l'année de diffusion ;
- une relation `emission` qui contient les émissions (`id_emission`, `nom`), la radio de diffusion et l'animateur ;
- une relation `description` qui contient un résumé et la durée du podcast en minutes.

Relations de la base de données

Relation `podcast`

id_podcast	theme	annee	id_emission
1	Le système d'enseignement supérieur français est-il juste et efficace ?	2022	10081
2	Trois innovations pour la croissance future (1/3) : La révolution blockchain.	2021	10081
3	Travailleurs de plateformes : vers un nouveau prolétariat ?	2021	10175
4	Le poids de la souveraineté numérique française	2019	10183
40	Le poids de la souveraineté numérique française	2019	10183
5	Dans le cloud en Islande, terre des data center	2019	10212

Relation `emission`

id_emission	nom	radio	animateur
10081	Entendez-vous l'éco ?	France culture	Tiphaine De R.
10175	Le Temps du débat	France culture	Léa S.
10183	Soft power	France culture	Frédéric M.
10212	La tête au carré	France inter	Mathieu V.

Relation `description`

id_description	resume	duree	id_emission
101	Autrefois réservé à une élite, l'enseignement supérieur français s'est profondément démocratisé : donne-t-il pour autant les mêmes chances à chacun ?	4	10081
102	Quelles sont leurs conditions de travail et quels sont leurs moyens de contestation ?	58	10175
103	La promesse de la blockchain, c'est la suppression des intermédiaires et la confiance à grande échelle.	4	10081

Questions

1. Écrire le schéma relationnel de la relation `description`, en précisant les attributs et leurs types probables, la clé primaire et la ou les clé(s) étrangère(s) éventuelle(s).
2. (a) Écrire ce qu'affiche la requête suivante appliquée aux extraits précédents :

   ```
   SELECT theme, annee FROM podcast WHERE id_emission = 10081
   ```

   (b) Écrire une requête SQL permettant d'afficher les thèmes des podcasts de l'année 2019.
   (c) Écrire une requête SQL affichant la liste des thèmes et des années de diffusion des podcasts dans l'ordre chronologique des années.
3. (a) Décrire simplement le résultat obtenu avec cette requête SQL :

   ```
   SELECT DISTINCT theme FROM podcast
   ```

   (b) Écrire une requête SQL supprimant la ligne contenant l'`id_podcast` = 40 de la relation `podcast`.
4. (a) Une erreur de saisie a été faite dans la relation `emission`. Écrire une requête SQL permettant de changer le nom de l'animateur de l'émission "Le Temps du débat" en "Emmanuel L.".
   (b) Écrire une requête SQL permettant d'ajouter l'émission "Hashtag" sur la radio "France inter" avec "Mathieu V.". On lui donnera un `id_emission` égal à 12850.
5. Écrire une requête permettant de lister les thèmes, le nom des émissions et le résumé des podcasts pour lesquels la durée est strictement inférieure à 5 minutes.

Exercice 3 (2 points)

L'énoncé de cette partie utilise les mots du langage SQL suivant : `SELECT`, `FROM`, `WHERE`, `UPDATE`, `JOIN`, `ON`, `INSERT`, `INTO`, `VALUES`, `AND`, `COUNT`, `DISTINCT`.

Une compétition internationale de rugby féminin est organisée en France. Une base de données relationnelle est mise en place afin de gérer l'organisation. Son schéma relationnel est donné ci-dessous :

- `JOUEUSE(idjoueuse, #pays, nom, prenom, age, numero)`
- `SELECTION(#idjoueuse, #idmatch, points)`
- `MATCH(idmatch, stade, jour, horaire)`

Dans ce schéma, les clés primaires sont soulignées et les clés étrangères sont précédées du symbole #.
On donne ci-dessous un extrait de la relation `JOUEUSE` :

idjoueuse	pays	nom	prenom	age	numero
101	"France"	"Lefevre"	"Charline"	23	15
108	"Australie"	"Porteur"	"Cindy"	31	7
305	"Argentine"	"Gomez"	"Laëtitia"	35	8
318	"Tunisie"	"Char"	"Jo"	30	2

1. (a) Expliquer ce que renvoie la requête SQL suivante :

```
SELECT nom, prenom, numero
FROM JOUEUSE
WHERE pays = "France";
```

   (b) Écrire une requête permettant d'obtenir le nom et le prénom de toutes les joueuses de l'équipe d'Argentine ayant au moins 30 ans.

2. (a) Une erreur de saisie a été commise : la joueuse Charline Lefevre n'a pas 23 ans mais 33 ans. Écrire une requête SQL permettant de mettre à jour son âge.

   (b) Écrire la requête permettant d'insérer la joueuse dont l'identifiant `idjoueuse` est 105, le nom "Warm", le prénom "Suzanna", l'âge 29 ans, le pays "Angleterre" et le numéro 6.

3. On suppose que la compétition est terminée et que les tables sont correctement remplies.

   (a) Recopier en intégralité sur la copie la requête SQL ci-dessous et la compléter de façon à ce qu'elle renvoie les noms des joueuses qui ont marqué au moins 10 points lors d'un match.

```
SELECT ...
FROM JOUEUSE
JOIN SELECTION ON ...
WHERE ...
```

   (b) On rappelle qu'en langage SQL, la fonction d'agrégation `COUNT` permet de compter un nombre d'enregistrements. Par exemple, pour déterminer le nombre de joueuses dans la table `JOUEUSE`, on peut utiliser la requête suivante :

```
SELECT COUNT(idjoueuse)
FROM JOUEUSE;
```

   Écrire une requête permettant de connaître le nombre de sélections de la joueuse Laëtitia Gomez, dont l'identifiant `idjoueuse` est 305, lors de cette compétition.

   (c) Écrire une requête permettant de connaître les noms des stades dans lesquels la joueuse Laëtitia Gomez, dont l'identifiant `idjoueuse` est 305, a joué lors de cette compétition.

Exercice 4 (5 points)

Cet exercice porte sur le traitement des données en table et les bases de données. Il est constitué de deux parties indépendantes.

Un étudiant souhaite développer une application permettant de faciliter le covoiturage pour les déplacements du quotidien. Dans cet objectif, il étudie des données extraites de la Base Nationale des Lieux de Covoiturage (BNLC), disponible sur le site data.gouv.fr.

Dans un premier temps (partie A), les données d'une table décrivant des lieux de covoiturage (adresse postale, nombre de places, ...) sont manipulées à l'aide d'un tableau contenant des dictionnaires en langage Python.

Dans un second temps (partie B), une base de données contenant deux tables (les sites de covoiturage et les caractéristiques des communes de France) est exploitée à l'aide du langage SQL.

Partie A : traitement de données en table

Une table est implémentée par un tableau nommé `tab_lieux` contenant des dictionnaires en langage Python. Chaque dictionnaire correspond à un lieu de stationnement pour le covoiturage.

Les clés des dictionnaires, communes à tous les dictionnaires, correspondent aux descripteurs utilisés pour cette table :

- `id_lieu` : identifiant du lieu, la donnée est un entier (chaque lieu possède un identifiant unique) ;
- `ad_lieu` : adresse du lieu, la donnée est une chaîne de caractères ;
- `insee` : code INSEE de la commune où se trouve le lieu, la donnée est une chaîne de caractères ;
- `nb_places` : nombre de places du lieu, la donnée est un entier ;
- `type` : nature du parking (supermarché, parking municipal, aire de stationnement située en sortie d'autoroute, ...), la donnée est une chaîne de caractères.

On donne en illustration les trois premiers éléments de ce tableau de dictionnaires :

```
tab_lieux = [
    {"id_lieu": 1, "ad_lieu": "Place De La Fontaine", "insee": "1024",
     "nb_places": 5, "type": "Supermarché"},
    {"id_lieu": 2, "ad_lieu": "La Boisse", "insee": "1049",
     "nb_places": 100, "type": "Parking municipal"},
    {"id_lieu": 3, "ad_lieu": "Château-Gaillard", "insee": "1089",
     "nb_places": 15, "type": "Sortie autoroute"},
    ...
]
```

1. Accès aux informations du tableau :
   (a) Donner, sans justifier, la valeur à laquelle on accède avec l'instruction `tab_lieux[0]["insee"]`.
   (b) Écrire l'instruction qui permet d'obtenir la valeur "La Boisse".
2. On propose trois blocs d'instructions pour parcourir le tableau et afficher le nombre de places des lieux de covoiturage. Parmi ces trois propositions, deux seulement sont correctes. Indiquer, sans justifier, les deux propositions correctes.

Proposition 1

```
for i in range(len(tab_lieux)):
    print(dico["nb_places"])
```

Proposition 2

```
for i in range(len(tab_lieux)):
    print(tab_lieux[i]["nb_places"])
```

Proposition 3

```
for dico in tab_lieux:
    print(dico["nb_places"])
```

3. On dispose de la fonction ci-dessous :

```
def fonction_inconnue(table, n, nom_type):
    """
    Entrée : un tableau de dictionnaires, un entier, une
    chaîne de caractères
    Sortie : un entier
    """
    k = 0
    for dico in table:
        if dico["nb_places"] >= n and dico["type"] == nom_type:
            k = k + 1
    return k
```

Décrire, dans le contexte de l'exercice, ce que renvoie cette fonction, lors de l'appel : `fonction_inconnue(tab_lieux, 100, "Sortie autoroute")`.

4. La fonction, dont le code est proposé ci-après, permet de renvoyer le code INSEE du lieu de covoiturage possédant le plus grand nombre de places.

Recopier et compléter ce code :

```
def insee_max_places(table):
    """
    Entrée : un tableau de dictionnaires
    Sortie : une chaîne de caractères (le code INSEE du lieu
    possédant le plus grand nombre de places)
    """
    maxi = ...
    code_insee = ...
    for dico in table:
        if ...
            maxi = ...
            code_insee = ...
    return code_insee
```

5. La fonction dont les spécifications sont données ci-après, prend en paramètres une table de lieux de covoiturage au format tableau contenant des dictionnaires et le nom d'un type de lieu de covoiturage.
   Recopier et compléter le code de cette fonction. Dans le code de la fonction, les trois points (. . .) peuvent correspondre à une ou plusieurs lignes de programme.

```
def moyenne_par_type(table, nom_type):
    """
    Entrée : un tableau de dictionnaires et une chaîne de
    caractères (type de lieu)
    Cette fonction renvoie, parmi les lieux de covoiturage
    dont le type est nom_type, la moyenne du nombre de places
    pour le type choisi.
    Sortie : un flottant
    """
    ...
    return moyenne
```

   Exemple, avec la table `tab1` ci-après, contenant uniquement trois enregistrements :

```
tab1 = [
    {"id_lieu":1, "ad_lieu":"Place Du marché", "insee":"1032",
     "nb_places":10, "type":"Supermarché"},
    {"id_lieu":2, "ad_lieu":"La Pesse", "insee":"1058",
     "nb_places":100, "type":"Parking municipal"},
    {"id_lieu":3, "ad_lieu":"Le Pautet", "insee":"1075",
     "nb_places":20, "type":"Supermarché"}
]
```

   `moyenne_par_type(tab1, "Supermarché")` vaut 15.

Partie B : base de données

On pourra utiliser les mots du langage SQL suivants : `SELECT`, `FROM`, `WHERE`, `JOIN`, `INSERT INTO`, `VALUES`, `COUNT`.

Afin de pouvoir gérer un site de covoiturage, on utilise une base de données contenant les relations `LIEU` et `COMMUNE`.

Le schéma relationnel, où les clés primaires sont soulignées et les clés étrangères sont précédées du symbole #, est le suivant :

- `LIEU(id_lieu : entier, ad_lieu : texte, #code_insee : texte, nb_places : entier, type : texte)`
- `COMMUNE(code_insee : texte, nom : texte, departement : texte, region : texte, latitude : décimal, longitude : décimal)`

On rappelle qu'en langage SQL la fonction d'agrégation `COUNT` permet de compter un nombre d'enregistrements.

Par exemple, pour déterminer le nombre de lieux dans la table `LIEU`, on peut utiliser la requête suivante :

```
SELECT COUNT(id_lieu)
FROM LIEU;
```

On donne un extrait des deux premières lignes de ces relations :

Relation `LIEU`

id_lieu	ad_lieu	code_insee	nb_places	type
1	"Place De La Fontaine"	"01001"	5	"Supermarché"
2	"La Boisse"	"01049"	100	"Parking municipal"

Relation `COMMUNE`

code_insee	nom	departement	region	lat	long
"01001"	"L'ABERGEMENT-CLEMENCIAT"	"AIN"	"RHONE-ALPES"	46.1534	4.9261
"01002"	"L'ABERGEMENT-DE-VAREY"	"AIN"	"RHONE-ALPES"	46.0092	5.4280

1. On se place dans la relation `COMMUNE`. Expliquer pourquoi deux communes ne peuvent pas posséder le même code INSEE `code_insee`.
2. Écrire une requête SQL permettant d'obtenir l'identifiant `id_lieu` et le code INSEE `code_insee` des lieux de covoiturage de type "Sortie autoroute".
3. Écrire une requête SQL permettant de compter le nombre de lieux de covoiturage se trouvant dans le département "JURA".
4. La commune de code INSEE "40714" vient d'être intégrée à la commune voisine "40146". En conséquence, il faut mettre à jour la base de données.
   (a) La requête suivante a été saisie. Une erreur a été renvoyée par le logiciel. Expliquer pourquoi.

   ```
   DELETE FROM COMMUNE
   WHERE code_insee = "40714";
   ```

   (b) Les informations liées à la commune de code INSEE "40714" devront désormais être rattachées à la commune de code INSEE "40146". Écrire une requête permettant de mettre à jour la table `LIEU`.

Exercice 5 (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, voici 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.

```
classement = []
for numero_dossard in temps_etapes:
    element = (numero_dossard, calcul_temps_total(numero_dossard))
    classement.append(element)
    pos = len(classement) - 2

    while pos >= 0 and element[...] < classement[pos][...]:
        classement[pos + 1] = ...
        pos = pos - 1
    classement[pos + 1] = element

for i in range(len(classement)):
    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 :

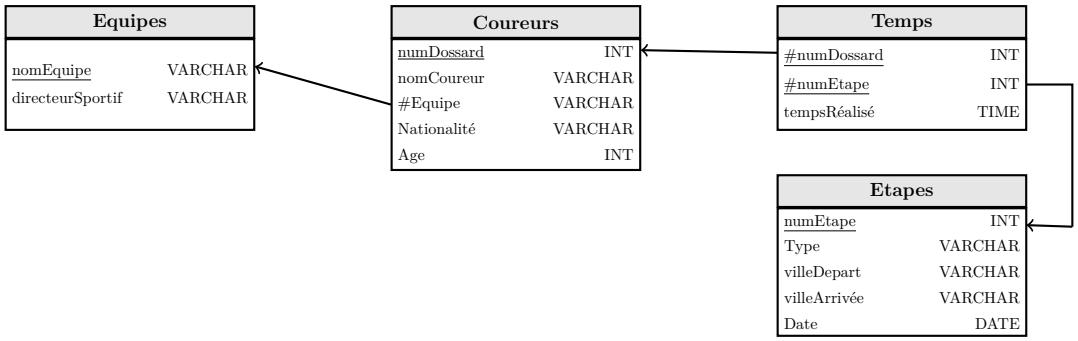
```
tableau_final = []
difference_temps = 0
premier = True
for ligne in tableau_temps:
    coureur = [ligne[0]]
    coureur.append(ligne[1])
    if premier:
        temps_premier = ligne[2]
        coureur.append(temps_premier)
        premier = False
    else:
        difference_temps = ligne[2] - ...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



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

1. Expliquer pourquoi, dans la relation `Temps`, il est nécessaire de prendre le couple (`numDossard`, `numEtape`) comme clé primaire.
2. Expliquer ce que renvoie la requête SQL suivante :

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

3. Écrire une requête SQL permettant d'obtenir les dates de toutes les étapes de type 'contre-la-montre' du tour de France 2023.
4. Écrire une requête SQL permettant d'obtenir le nom du directeur sportif du coureur BARDET Romain.
5. 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');
```

6. Expliquer quelle modification est à effectuer pour apporter une solution au problème constaté à la question précédente.
7. Écrire une requête SQL donnant le temps total en course mis par BARDET Romain depuis le départ du tour de France 2023.