Exercices type bac : SGBD et POO

Exercice 1 : Base de données

L'exercice porte sur les bases de données et les types construits de données.

On pourra utiliser les mots clés SQL suivants : `AND`, `FROM`, `INSERT`, `INTO`, `JOIN`, `OR`, `ON`, `SELECT`, `SET`, `UPDATE`, `VALUES`, `WHERE`.

On étudie une base de données permettant la gestion de l'organisation d'un festival de musique de jazz, dont voici le schéma relationnel comportant trois relations :

- la relation `groupes` (`idgrp`, `nom`, `style`, `nb_pers`)
- la relation `musiciens` (`idmus`, `nom`, `prenom`, `instru`, `#idgrp`)
- la relation `concerts` (`idconc`, `scene`, `heure_debut`, `heure_fin`, `#idgrp`)

Dans ce schéma relationnel :

- les clés primaires sont soulignées ;
- les clés étrangères sont précédées d'un #.

Ainsi `concerts.idgrp` est une clé étrangère faisant référence à `groupes.idgrp`.

Voici un extrait des tables `groupes`, `musiciens` et `concerts` :

extrait de `groupes`

idgrp	nom	style	nb_pers
12	'Weather Report'	'Latin Jazz'	5
25	'Breckers Brothers'	'Swing Jazz'	4
87	'Return to Forever'	'Latin Jazz'	8
96	'The Jazz Messenger'	'Free Jazz'	3

extrait de `musiciens`

idmus	nom	prenom	instru	idgrp
12	'Parker'	'Charlie'	'trompette'	96
13	'Parker'	'Charlie'	'trombone'	25
58	'Dufler'	'Candy'	'saxophone'	96
97	'Miles'	'Davis'	'saxophone'	87

extrait de `concerts`

idconc	scene	heure_debut	heure_fin	idgrp
10	1	'20h00'	'20h45'	12
24	2	'20h00'	'20h45'	15
36	1	'21h00'	'22h00'	96
45	3	'18h00'	'18h30'	87

Figure 1 : Extrait des tables `groupes`, `musiciens` et `concerts`

1. Citer les attributs de la table `groupes`.

2. Justifier que l'attribut `nom` de la table `musiciens` ne peut pas être une clé primaire.

3. En s'appuyant uniquement sur l'extrait des tables fourni dans la figure 1 écrire ce que renvoie la requête :

```
SELECT nom
FROM groupes
WHERE style = 'Latin Jazz';
```

4. Le concert dont l'`idconc` est 36 finira à 22h30 au lieu de 22h00. Recopier sur la copie et compléter la requête SQL ci-dessous permettant de mettre à jour la relation `concerts` pour modifier l'horaire de fin de ce concert.

```
UPDATE concerts
SET …
WHERE … ;
```

5. Donner une requête SQL permettant de récupérer le nom de tous les groupes qui jouent sur la scène 1.

6. Fournir une requête SQL permettant d'ajouter dans la relation `groupes` le groupe 'Smooth Jazz Fourplay', de style 'Free Jazz', composé de 4 membres. Ce groupe aura un `idgrp` de 15.

Les données sont ensuite récupérées pour être analysées par la société qui produit les festivals de musique. Pour ce faire, elle utilise la programmation en Python afin d'effectuer certaines opérations plus complexes.

Elle stocke les données relatives aux musiciens sous forme d'un tableau de dictionnaires dans laquelle a été ajouté le nombre de concerts effectués par chaque musicien :

```
>>> print(musiciens)
  [{'idmus': 12, 'nom': 'Parker', 'prenom': 'Charlie',
  'instru': 'trompette', 'idgrp' : 96, 'nb_concerts': 5},
  {'idmus': 13, 'nom': 'Parker', 'prenom': 'Charlie',
  'instru': 'trombone', 'idgrp' : 25, 'nb_concerts': 9},
  {'idmus': 58, 'nom': 'Dufler', 'prenom': 'Candy',
  'instru': 'saxophone', 'idgrp' : 96, 'nb_concerts': 4},
  {'idmus': 97, 'nom': 'Miles', 'prenom': 'Davis',
  'instru': 'saxophone', 'idgrp' : 87, 'nb_concerts': 2},
  …
  ]
```

7. Écrire la fonction `recherche_nom` ayant pour unique paramètre un tableau de dictionnaires (comme `musiciens` présenté précédemment) renvoyant un tableau contenant le nom de tous les musiciens ayant participé à au moins 4 concerts.

Exercice 2 : POO

Une entreprise souhaite gérer les colis qu'elle expédie à l'aide d'une application informatique. On sait que chaque colis a un identifiant unique, un poids, une adresse de livraison et un état. Pour chacun d'entre eux, trois états sont possibles : "préparé", "transit" ou "livré".

Pour cela, on a créé une classe `Colis` avec les attributs suivants :

- `id` : un identifiant unique (de type `str`) ;
- `poids` : le poids du colis en kilogrammes (de type `float`) ;
- `adresse` : l'adresse de destination (de type `str`) ;
- `etat` : l'état du colis (de type `str` parmi `'préparé'`, `'transit'`, `'livré'`).

Lorsque l'on crée une instance de la classe `Colis`, l'attribut `etat` est initialisé à `'préparé'` tandis que les valeurs des autres attributs sont passées en paramètres. Voici le début du code Python de la classe `Colis` :

```
1 class Colis:
2     def __init__(self, id, poids, adresse):
3         self.id = id
4         self.poids = poids
5         self.adresse = adresse
6         self.etat = 'préparé'
```

On crée, par exemple, les deux colis suivants :

```
colisA = Colis('AC12', 5.0, '20 rue de la paix 57000 Metz')
colisB = Colis('AF34', 10.25, '32 rue du centre 57000 Metz')
```

1. Écrire la méthode `passer_transit` de la classe `Colis` qui permet de mettre l'état du colis à la valeur `'transit'`.

On dispose de la fonction `ajouter_colis` suivante :

```
1 def ajouter_colis(liste, colis):
2     # ajoute le colis à la fin de la liste
3     liste.append(colis)
```

Par exemple, après l'exécution des trois instructions suivantes, on a ajouté les deux colis crées précédemment à la liste `liste_colis` :

```
1 liste_colis = []
2 ajouter_colis(liste_colis, colisA)
3 ajouter_colis(liste_colis, colisB)
```

2. Dans cette question uniquement, on considère que l'acheminement des colis de plus de 25 kg est refusé par le transporteur.

   Recopier et modifier le code de la fonction `ajouter_colis` afin qu'elle ajoute le colis à la liste si son poids est inférieur ou égal à 25 kg et qu'elle affiche le message "Dépassement du poids maximal autorisé" sinon.

3. Écrire une fonction `nb_colis` qui prend en paramètre une liste d'objets de la classe `Colis` et qui renvoie le nombre de colis présents dans cette liste.

4. Recopier et compléter les lignes 2 et 4 du code ci-après de la fonction `poids_total` qui prend en paramètre une liste d'objets de la classe `Colis` et qui renvoie le poids total de l'ensemble des colis de cette liste.

```
1 def poids_total(liste):
2     total = ...
3     for c in liste :
4         total = ...
5     return total
```

5. Écrire une fonction `liste_colis_etat` qui prend en paramètres une liste d'objets de la classe `Colis` et une chaîne de caractères `statut` (parmi `'préparé'`, `'transit'` ou `'livré'`) et qui renvoie une nouvelle liste contenant l'ensemble des colis de cette liste dont l'état est le même que `statut`.

Correction

Exercice 1

1. attributs de la table groupes : idgrp, nom, style, nb_pers

2. Le même nom peut apparaitre plusieurs fois (par exemple “Parker”), ce qui n’est pas possible avec une clé primaire

3. Cette requête renvoie : ‘Weather Report’ et ‘Return to Forever’

4.
```
UPDATE concerts
SET heure_fin = ‘22h30’
WHERE idconc = 36
```

5.
```
SELECT nom
FROM groupes
JOIN concerts ON concerts.idgrp = groupes.idgrp
WHERE scene = 1
```

6.
```
INSERT INTO groupes
VALUES
(15, 'Smooth Jazz Fourplay', ‘Free Jazz’, 4)
```

7.
```
def recherche_nom(tab):
    t = []
    for d in tab:
        if d['nb_concerts'] >= 4 :
            t.append(d['nom'])
    return t
```

Exercice 2

1.
```
def passer_transit(self):
    self.etat = 'transit’
```
2.
```
def ajouter_colis(liste, colis):
    if colis.poids <= 25:
        liste.append(colis)
    else :
        print('Dépassement du poids maximum autorisé')
```
3.
```
def nb_colis(liste):
    return len(liste)
```
4.
```
def poids_total(liste):
    total = 0
    for c in liste:
        total = total + c.poids
    return total
```
5.
```
def liste_colis(liste, statut):
    lst = []
    for c in liste :
        if c.etat == statut:
            lst.append(c)
    return lst
```