

## TP 1 : SQL & SQLite - Corrigé

- Employes (ID, Nom, Prenom, DateNaissance, Adresse, Salaire, #ID\_Departement)
- Departements (ID, Nom, Description)
- Projets (ID, NomProjet, Budget, DateDebut, DateFin)
- EmployesProjets (#ID\_Employe, #ID\_Projet, Role)

### 1 Manipulation des Tables

#### 1.1 Création de Table

##### Questions :

1. Créez la table **Employes** avec les contraintes suivantes : ID (entier, clé primaire avec incrémentation automatique), Nom et Prénom (non null), Adresse ('Tunis' par défaut), Salaire (positif, par défaut 0), id\_Departement (entier, clé étrangère fait référence à la clé primaire de la table Departement).

```
CREATE TABLE Employes (  
    ID INT PRIMARY KEY AUTOINCREMENT,  
    Nom TEXT NOT NULL,  
    Prenom TEXT NOT NULL,  
    DateNaissance DATE,  
    Adresse TEXT DEFAULT 'Tunis',  
    Salaire REAL DEAFULT 0 CHECK(salaire >= 0),  
    ID_Departement INT,  
    FOREIGN KEY (ID_Departement) REFERENCES Departements(ID)  
);
```

2. Créez la table **Departements** avec les contraintes suivantes : ID (entier, clé primaire), Nom (non null, avec des valeurs uniques).

```
CREATE TABLE Departements (  
    ID INT PRIMARY KEY,  
    Nom TEXT NOT NULL UNIQUE,  
    Description TEXT  
);
```

#### 1.2 Modification de Table

3. Ajoutez une nouvelle colonne "HeuresTravail" de type INTEGER à la table "Projets".

```
ALTER TABLE Projets ADD COLUMN HeuresTravail INTEGER;
```

4. Renommez la colonne DateDebut de la table Projets en DebutProjet.

```
ALTER TABLE Projets RENAME COLUMN DateDebut TO DebutProjet;
```

### 2 Manipulation des Données

#### 2.1 Insertion, Mise à Jour et Suppression de Données

5. Ajoutez un nouvel employé avec le nom 'Ali' dans le département numéro 1.

```
INSERT INTO Employes (Nom, Prenom, id_departement)  
VALUES ('Nabli', 'Ali', SELECT ID FROM Departements WHERE Nom = 'Production');
```

6. Mettez à jour le salaire de tous les employés du département 'Ventes' en ajoutant 10%.

```
UPDATE Employes
SET Salaire = Salaire * 1.1
WHERE id_departement = (SELECT ID FROM Departements WHERE Nom = 'Ventes');
```

7. Supprimez tous les employés dont le salaire est inférieur à 4000.

```
DELETE FROM Employes WHERE Salaire < 4000;
```

## 2.2 Recherche de Données

### Exercice

#### 1. Opérations de Base

- (a) Sélectionnez tous les employés dont la deuxième lettre du prénom est 'a'.

```
SELECT * FROM Employes WHERE Prenom LIKE '_a%';
```

- (b) Sélectionnez les noms des employés concaténés avec leur prénoms, séparés par un tiret.

```
SELECT CONCAT(Nom, '-', Prenom) AS NomPrenom FROM Employes;
```

- (c) Sélectionnez les noms uniques de la table des employés.

```
SELECT DISTINCT Nom FROM Employes;
```

#### 2. Jointures

- (a) Sélectionnez les noms des employés et leurs départements correspondants triés par ordre alphabétique décroissant.

```
SELECT Employes.Nom, Employes.Prenom, Departements.Nom AS NomDepartement
FROM Employes
JOIN Departements ON Employes.ID_Departement = Departements.ID
ORDER BY Employes.Nom DESC;
```

- (b) Sélectionnez les noms des employés du département 'Ressources Humaines' et les projets correspondants.

```
SELECT Employes.Nom, Projets.NomProjet
FROM Employes
JOIN EmployesProjets ON Employes.ID = EmployesProjets.ID_Employe
JOIN Projets ON EmployesProjets.ID_Projet = Projets.ID
JOIN Departements ON Employes.ID_Departement = Departements.ID
WHERE Departements.Nom = 'Ressources Humaines';
```

#### 3. Agrégations

- (a) Trouvez le nombre total d'employés.

```
SELECT COUNT(*) AS NombreTotalEmployes FROM Employes;
```

- (b) Trouvez le salaire moyen des employés du département 'Ventes'.

```
SELECT AVG(Salaire) AS SalaireMoyenVentes
FROM Employes
WHERE ID_Departement = (SELECT ID FROM Departements WHERE Nom = 'Ventes');
```

- (c) Trouvez le salaire moyen des employés pour chaque département.

```
SELECT ID_Departement, AVG(Salaire) AS SalaireMoyenParDepartement
FROM Employes
GROUP BY ID_Departement;
```

#### 4. Requêtes SELECT Imbriquées

- (a) Sélectionnez les employés du département 'Ventes' ayant un salaire supérieur à la moyenne des salaires de tous les employés.

```
SELECT *
FROM Employes
WHERE ID_Departement = (SELECT ID FROM Departements WHERE Nom = 'Ventes')
AND Salaire > (SELECT AVG(Salaire) FROM Employes);
```

- (b) Sélectionnez les noms des employés qui ont les salaires les plus élevés parmi tous les départements.

```
SELECT Nom, Prenom
FROM Employes
WHERE Salaire = (SELECT MAX(Salaire) FROM Employes);
```

- (c) Sélectionnez les employés ayant un salaire supérieur à la moyenne des salaires des employés du département 'Ventes'.

```
SELECT *
FROM Employes
WHERE Salaire > (SELECT AVG(Salaire)
                FROM Employes
                WHERE ID_Departement = (SELECT ID
                                         FROM Departements
                                         WHERE Nom = 'Ventes'))
);
```

- (d) Sélectionnez les noms des employés dont le salaire est supérieur à la moyenne des salaires des employés travaillant sur des projets avec un budget supérieur à 50000.

```
SELECT Employes.Nom, Employes.Prenom
FROM Employes
JOIN EmployesProjets ON Employes.ID = EmployesProjets.ID_Employe
JOIN Projets ON EmployesProjets.ID_Projet = Projets.ID
WHERE Employes.Salaire > (SELECT AVG(Salaire)
                        FROM EmployesProjets
                        JOIN Projets ON EmployesProjets.ID_Projet = Projets.ID
                        WHERE Budget > 50000);
```

#### 5. Utilisation de IN, NOT IN, EXISTS, NOT EXISTS et ALL

- (a) Sélectionnez les noms des employés travaillant sur des projets dont le budget est supérieur à 50000.

```
SELECT Employes.Nom, Employes.Prenom
FROM Employes
JOIN EmployesProjets ON Employes.ID = EmployesProjets.ID_Employe
JOIN Projets ON EmployesProjets.ID_Projet = Projets.ID
WHERE Projets.Budget > 50000;
```

- (b) Sélectionnez les noms des employés travaillant sur des projets dont le budget est compris entre 20000 et 50000.

```
SELECT Employes.Nom, Employes.Prenom
FROM Employes
JOIN EmployesProjets ON Employes.ID = EmployesProjets.ID_Employe
JOIN Projets ON EmployesProjets.ID_Projet = Projets.ID
WHERE Projets.Budget BETWEEN 20000 AND 50000;
```

- (c) Sélectionnez les noms des employés qui sont responsables (role ="responsable") de tous les projets.

```
SELECT Employes.Nom, Employes.Prenom
FROM Employes
WHERE NOT EXISTS (SELECT *
                  FROM Projets
                  WHERE NOT EXISTS (SELECT *
                                   FROM EmployesProjets
                                   WHERE EmployesProjets.ID_Employe = Employes.ID
                                   AND EmployesProjets.ID_Projet = Projets.ID
                                   AND EmployesProjets.Role = 'responsable'))
);
```

- (d) Sélectionnez les noms des employés qui ne sont responsables d'aucun projet.

```
SELECT Employes.Nom, Employes.Prenom
FROM Employes
WHERE NOT EXISTS (
    SELECT *
    FROM EmployesProjets
    WHERE EmployesProjets.ID_Employe = Employes.ID
    AND EmployesProjets.Role = 'responsable'
);
```

- (e) Sélectionnez les noms des employés travaillant sur des projets pour lesquels au moins un employé a un salaire supérieur à 7000.

```
SELECT Employes.Nom, Employes.Prenom
FROM Employes
JOIN EmployesProjets ON Employes.ID = EmployesProjets.ID_Employe
JOIN Projets ON EmployesProjets.ID_Projet = Projets.ID
WHERE Projets.ID IN (SELECT ID_Projet
                   FROM EmployesProjets
                   WHERE Role = 'responsable'
                   AND ID_Employe IN (SELECT ID
                                     FROM Employes
                                     WHERE Salaire > 7000));
```

- (f) Sélectionnez les noms des employés travaillant sur des projets pour lesquels tous les employés ont un salaire supérieur à 6000.

```
SELECT Employes.Nom, Employes.Prenom
FROM Employes
JOIN EmployesProjets ON Employes.ID = EmployesProjets.ID_Employe
JOIN Projets ON EmployesProjets.ID_Projet = Projets.ID
WHERE Projets.ID NOT IN (SELECT ID_Projet
                       FROM EmployesProjets
                       WHERE Role = 'responsable'
                       AND ID_Employe IN (SELECT ID
                                         FROM Employes
                                         WHERE Salaire <= 6000));
```

## 2.3 Exemples de requêtes SQL avec SQLite et Python

Voici quelques exemples de requêtes SQL que vous pourriez exécuter à partir de Python en utilisant SQLite.

### 1. Requêtes de Sélection (execute, fetchone, fetchmany, fetchall)

- (a) Sélectionnez tous les employés du département 'Ventes', triez les résultats par salaire de manière décroissante et affichez le résultat avec `fetchall`.
- (b) Sélectionnez le nom et le salaire des employés dont le salaire est supérieur à 5000 et affichez le résultat avec `fetchone`.
- (c) Sélectionnez les employés (triés par leurs salaires) et le nom de leurs département (triés par ordre alphabétique descendant) et affichez le résultat avec `fetchmany` (limitez à 10 résultats).

## 2. Requêtes d'Insertion (`execute`, `executemany`, `commit`)

- (a) Insérez un nouvel employé nommé 'Amir', travaillant dans le département 'Marketing' avec un salaire de 6200.
- (b) Définir le rôle de l'employé numéro 6 en tant que responsable pour tous les projets dont la date de début est postérieure à la date actuelle.

```
import sqlite3

# Connexion à la base de données
conn = sqlite3.connect('ma_base.db')
cursor = conn.cursor()

# Requêtes de Sélection
# 1. a)
cursor.execute("""
    SELECT Employes.* FROM Employes
    JOIN Departements ON Employes.ID_Departement = Departements.ID
    WHERE Departements.Nom = 'Ventes'
    ORDER BY Employes.Salaire DESC
""")
result = cursor.fetchall()
print(result)

# 1. b)
cursor.execute("""
    SELECT Nom, Salaire FROM Employes
    WHERE Salaire > 5000
""")

result = cursor.fetchone()

while result:
    print("Nom: {}, Salaire: {}".format(result[0], result[1]))
    result = cursor.fetchone()

# 1. c)
cursor.execute("""
    SELECT Employes.*, Departements.Nom
    FROM Employes E
    JOIN Departements D ON E.ID_Departement = D.ID
    ORDER BY E.Salaire ASC, D.Nom DESC
    LIMIT 10
""")
result = cursor.fetchmany(10)
print(result)

# Requêtes d'Insertion
```

```
# 2. a)
cursor.execute("""
    INSERT INTO Employes (Nom, Prenom, #ID_Departement, Salaire)
    VALUES ('Amir', '', (SELECT ID FROM Departements WHERE Nom = 'Marketing'), 6200)
""")
conn.commit()

# 2. b)
cursor.execute("""
    UPDATE EmployesProjets
    SET role = responsable
    WHERE id_Employe = 6 AND id_Projet IN (SELECT id FROM Projet WHERE DateDebut > NOW())
""")
conn.commit()

# Fermer la connexion
conn.close()
```