



# Devoir 1 : Langage SQL



## 1. D'après 2021, Métropole , J2,

L'énoncé de cet exercice utilise les mots du langage SQL suivants : `SELECT FROM, WHERE, JOIN ON, INSERT INTO VALUES, UPDATE, SET, DELETE, COUNT, AND, OR` . On considère dans cet exercice une gestion simplifiée des emprunts des ouvrages d'un CDI. La base de données utilisée sera constituée de trois relations (ou tables) nommées `Eleves` , `Livres` et `Emprunts` selon le schéma relationnel suivant :

- **Livres** (isbn (CHAR 13), titre (CHAR), auteur (CHAR))
- **Emprunts** (idEmprunt (INT), #idEleve (INT), #isbn (CHAR 13), dateEmprunt (DATE), dateRetour (Date))
- **Eleves** (idEleve (INT), nom (CHAR), prenom (CHAR), classe (CHAR))

Dans ce schéma relationnel, un attribut souligné indique qu'il s'agit d'une clé primaire.

Le symbole # devant un attribut indique qu'il s'agit d'une clé étrangère. Ainsi, l'attribut `idEleve` de la relation `Emprunts` est une clé étrangère qui fait référence à la clé primaire `idEleve` de la relation `Eleves` . De même l'attribut `isbn` de la relation `Emprunts` est une clé étrangère qui fait référence à la clé primaire `isbn` de la relation `Livres` .

1. Expliquer pourquoi le code SQL ci-dessous provoque une erreur.

### Requête SQL

```
INSERT INTO Eleves VALUES (128, 'Dupont', 'Jean', 'T1') ;
INSERT INTO Eleves VALUES (200, 'Dupont', 'Jean', 'T1') ;
INSERT INTO Eleves VALUES (128, 'Dubois', 'Jean', 'T2') ;
```

### Réponse

On insère deux entrées dans lesquelles l'attribut `idEleve` est égal à 128 . Or cet attribut est la clé primaire de la table, il ne peut pas exister en doublon.

2. Dans la définition de la relation `Emprunts` , qu'est-ce qui assure qu'on ne peut pas enregistrer un emprunt pour un élève qui n'a pas encore été inscrit dans la relation `Eleves` ?

### Réponse

Il s'agit de la clé étrangère `idEleve` qui doit respecter la contrainte d'intégrité référentielle.

3. Écrire une requête SQL qui renvoie les titres des ouvrages de Molière détenus par le CDI.

### Réponse

#### Requête SQL

```
SELECT titre
FROM Livres
WHERE auteur = 'Molière'
```

4. Décrire le résultat renvoyé par la requête ci-dessous.

#### Requête SQL

```
SELECT COUNT(*)
FROM Eleves
WHERE classe = 'T2' ;
```

### Réponse

On compte les élèves de la table `Eleves` dont la classe est la `'T2'`.

5. Camille a emprunté le livre « *Les misérables* ». Le code ci-dessous a permis d'enregistrer cet emprunt.

#### Requête SQL

```
INSERT INTO Emprunts
VALUES (640, 192, '9782070409228', '2020-09-15', NULL);
```

Camille a restitué le livre le 30 septembre 2020. Recopier et compléter la requête ci-dessous de manière à mettre à jour la date de retour dans la base de données.

#### Requête SQL

```
UPDATE Emprunts SET ..... WHERE .....
```

## Réponse

### Requête SQL

```
UPDATE Emprunts  
SET dateRetour = '2020-09-30'  
WHERE idEmprunt = 640
```

6. Décrire le résultat renvoyé par la requête ci-dessous.

### Requête SQL

```
SELECT DISTINCT nom, prenom  
FROM Eleves, Emprunts  
WHERE Eleves.idEleve = Emprunts.idEleve  
AND Eleves.classe = 'T2' ;
```

## Réponse

On récupère les noms et prénoms des élèves de la classe 'T2' qui ont déjà emprunté un livre.

7. Écrire une requête SQL qui permet de lister les noms et prénoms des élèves qui ont emprunté le livre « *Les misérables* ».

## Réponse

On propose (en utilisant l'ISBN cité dans la question 5):

### Requête SQL

```
SELECT nom, prenom
FROM Eleves
JOIN Emprunts ON Eleves.idEleves = Emprunts.idEleves
WHERE Emprunts.isbn = 192
```

Sans l'ISBN :

### Requête SQL

```
SELECT nom, prenom
FROM Eleves
JOIN Emprunts ON Eleves.idEleves = Emprunts.idEleves
JOIN Livres ON Livres.isbn = Emprunts.isbn
WHERE Livres.titre = 'Les Misérables'
```