Exercice 1 Une bibliothèque stocke des informations sur ses livres de science-fiction dans la table `livres_SF` suivante :

Titre	Auteur	Emplacement	Numero	Nb_emprunts
Fahrenheit 451	Bradbury Ray	SF BRA	NULL	8
1984	Orwell George	SF ORW	NULL	12
Le meilleur des mondes	Huxley Aldous	SF HUX	NULL	10
La flotte perdue	Campbell Jack	SF CAM	Tome 01 indomptable	2
Le dragon sous la mer	Herbert Frank	SF HER	NULL	4
Le Cycle des Robots	Asimov Isaac	SF ASI	Tome 1 : Les robots	9
Le Cycle des Robots	Asimov Isaac	SF ASI	Tome 2 : Un défilé de robots	8

1. Faire la liste des éléments de l'attribut « Auteur ».
2. La table précédente contient combien d'enregistrements ?

Exercice 2 Une bibliothèque enregistre ses prêts dans une seule table `Prets` :

Abonne_ID	Nom	Prenom	Ville	Livre_ID	Titre	Auteur	DateEmprunt
12	Durand	Alice	Antibes	305	Dune	Herbert Frank	2026-09-14
27	Martin	Karim	Cagnes-sur-Mer	305	Dune	Herbert Frank	2026-09-21
12	Durand	Alice	Antibes	305	Dune	Herbert Frank	2026-10-05
31	Leroy	Sophie	Vence	118	1984	Orwell George	2026-10-08

1. Quelles informations apparaissent plusieurs fois dans cette table ? Quel problème cela pose-t-il quand Alice Durand déménage ?

On répartit alors les données dans trois tables, selon le schéma suivant. `CHAR(n)` : chaîne d'exactement n caractères ; `VARCHAR(n)` : chaîne d'au plus n caractères ; dates au format AAAA-MM-JJ.

Abonne	
Abonne_ID	INT PRIMARY KEY
Nom	VARCHAR(50)
Prenom	VARCHAR(50)
DateNaissance	DATE
Adresse	VARCHAR(255)
Ville	VARCHAR(50)
CodePostal	CHAR(5)

Emprunt	
Abonne_ID	INT
Livre_ID	INT
DateEmprunt	DATE
DateRetourPrevu	DATE
DateRetourReel	DATE

Livre	
Livre_ID	INT PRIMARY KEY
Titre	VARCHAR(100)
Auteur	VARCHAR(100)
Genre	VARCHAR(255)
ISBN	VARCHAR(18)

2. Proposer une clef primaire pour la table `Emprunt`. Le couple (`Abonne_ID`, `Livre_ID`) convient-il ?
3. Le SGBD accepte-t-il les opérations suivantes ? Sinon, quelle contrainte d'intégrité (domaine, relation ou référence) est violée ?
   a. Ajouter un abonné dont l'`Abonne_ID` vaut 12, alors qu'un abonné a déjà cet identifiant.
   b. Ajouter un emprunt pour l'`Abonne_ID` 99, alors qu'aucun abonné n'a cet identifiant.

Exercice 3 On travaille sur la table `Artistes` de la base `cinema.db`, dont le schéma relationnel est :

Artistes(id : Entier, nom : Texte, prenom : Texte, annee_naissance : Entier)

1. Écrire une requête SQL qui permet d'obtenir le nom des artistes dont le prénom est `'David'`.

2. Écrire une requête SQL qui permet d'ajouter le nouvel artiste : `'Christopher Nolan - 1972'` dans la table Artistes (*attention, l'attribut* `id` *doit recevoir une valeur : le plus grand* `id` *actuel est 98*).
3. Écrire une requête SQL qui permet d'ajouter une nouvelle artiste : `'Camille Honette - 1987'` dans la table Artistes.
4. Après vérification, l'année de naissance de Christopher Nolan est 1970. Écrire une requête SQL qui permet de corriger cette erreur.
5. Écrire une requête SQL qui permet d'obtenir le nom des artistes dont le prénom est `'John'` et dont l'id est strictement supérieur à 30.
6. Écrire une requête SQL qui permet d'obtenir le nom et le prénom des artistes nés à partir de 1970 ou dont le prénom est `'James'`.
7. Écrire une requête SQL qui permet d'obtenir le nombre d'artistes nés avant 1920.
8. Écrire une requête SQL qui supprime l'artiste Camille Honette.

Exercice 4 On travaille sur les tables `Films` et `Artistes` de la base `cinema.db`, dont le schéma relationnel est :

Artistes(id : Entier, nom : Texte, prenom : Texte, annee_naissance : Entier)
Films(id : Entier, titre : Texte, annee : Entier, #id_Artistes : Entier)

1. Écrire une requête SQL qui permet d'ajouter un nouveau film : `'19, Psychose, 1960, 1'`.
2. Écrire une requête SQL qui permet d'obtenir, pour tous les films, leur titre, leur année de sortie, ainsi que le nom et le prénom de leur réalisateur.
3. Écrire une requête SQL qui permet d'obtenir, pour tous les films sortis depuis 1980, leur titre, leur année de sortie, ainsi que le nom et le prénom de leur réalisateur.

Récapitulatif SQL à connaître

Lire des données

`SELECT ... FROM` *lire des données*
```
SELECT att1, att2 FROM table;
SELECT * FROM table;
```
`DISTINCT` *supprimer les doublons*
```
SELECT DISTINCT att FROM table;
```
`WHERE` *garder les lignes qui vérifient une condition*
```
SELECT att FROM table
WHERE condition;
```
`ORDER BY` *trier (ASC par défaut, DESC décroissant)*
```
SELECT att FROM table
ORDER BY att1, att2 DESC;
```
`COUNT SUM MIN MAX` *calculer une valeur*
```
SELECT COUNT(*) FROM table
WHERE condition;
```
`JOIN ... ON` *associer deux tables*
```
SELECT att1, att2 FROM table1
JOIN table2
ON table1.clef_etrangere = table2.clef_primaire;
```

Modifier les données

`INSERT INTO ... VALUES` *ajouter une ligne*
```
INSERT INTO table (att1, att2)
VALUES (val1, val2);
```
`UPDATE ... SET ... WHERE` *modifier des lignes*
```
UPDATE table
SET att1 = val1, att2 = val2
WHERE condition;
```
`DELETE FROM ... WHERE` *supprimer des lignes*
```
DELETE FROM table
WHERE condition;
```

Les conditions

`= <>`	égal, différent
`< <= > >=`	comparaisons
`AND OR NOT`	et, ou, non
`IS NULL`	valeur manquante

À retenir

- Une requête se termine par `;` ; chaînes entre apostrophes.
- Ordre : `SELECT, FROM, JOIN ... ON, WHERE, ORDER BY`.
- Sans `WHERE`, `UPDATE` et `DELETE` agissent sur *toutes* les lignes.
- Attribut commun aux deux tables : `Table.attribut`.
- `GROUP BY` et `HAVING` : hors programme.