Exercice: Gestion d'une bibliothèque

Schéma de la base de données :

La base de données contient les tables suivantes :

1. **Livre (Book)**

    - id_livre (INT, PRIMARY KEY)
    - titre (VARCHAR(100))
    - auteur (VARCHAR(100))
    - annee_publication (INT)
    - categorie (VARCHAR(50))
    - disponible (BOOLEAN)

2. **Adherent (Member)**

    - id_adherent (INT, PRIMARY KEY)
    - nom (VARCHAR(100))
    - prenom (VARCHAR(100))
    - date_naissance (DATE)
    - email (VARCHAR(100))
    - telephone (VARCHAR(20))

3. **Emprunt (Loan)**

    - id_emprunt (INT, PRIMARY KEY)
    - id_livre (INT, FOREIGN KEY REFERENCES Livre(id_livre))
    - id_adherent (INT, FOREIGN KEY REFERENCES Adherent(id_adherent))
    - date_emprunt (DATE)
    - date_retour (DATE)

Questions SQL :

1. Création de la base de données

Écrivez les requêtes SQL pour créer les tables avec les contraintes appropriées.

2. Insertion de données

Insérez les données suivantes dans les tables :

- Livre :
    - (1, 'SQL pour les débutants', 'Jean Dupont', 2020, 'Informatique', TRUE)
    - (2, 'Introduction à Python', 'Marie Curie', 2019, 'Programmation', FALSE)
- Adhérent :

    - (1, 'Martin', 'Paul', '1990-05-12', 'paul.martin@email.com', '0612345678')
    - (2, 'Dubois', 'Julie', '1985-09-24', 'julie.dubois@email.com', '0623456789')
- Emprunt :
    - (1, 1, 1, '2024-01-01', '2024-01-15')

3. Sélection de tous les livres

Récupérez tous les livres de la bibliothèque triés par titre par ordre alphabétique.

4. Filtrer les livres par catégorie

Récupérez les livres appartenant à la catégorie 'Informatique'.

5. Vérification de la disponibilité

Affichez les titres des livres qui sont actuellement disponibles.

6. Recherche par auteur

Trouvez tous les livres écrits par 'Victor Hugo'.

7. Détail des emprunts

Affichez les adhérents qui ont emprunté un livre, en indiquant leur nom, prénom et le titre du livre emprunté.

8. Nombre total de livres

Compte combien de livres sont actuellement dans la bibliothèque.

9. Emprunts par adhérent

Affichez le nombre de livres empruntés par chaque adhérent.

10. Emprunts tardifs

Affichez les emprunts dont la date de retour est dépassée par rapport à la date du jour.

11. Adhérents sans emprunt

Trouvez les adhérents qui n'ont jamais emprunté de livre.

12. Mise à jour de la disponibilité

Mettez à jour l'état de disponibilité d'un livre après son emprunt.

13. Suppression d’un adhérent

Supprimez un adhérent qui n’a jamais emprunté de livre.

14. Catégories distinctes

Affichez toutes les catégories de livres distinctes dans la bibliothèque.

15. Emprunts les plus récents

Affichez les 3 derniers emprunts effectués.

16. Livre le plus ancien

Trouvez le livre le plus ancien de la bibliothèque.

17. Longueur du titre

Affichez les livres dont le titre contient plus de 20 caractères.

18. Regroupement par année de publication

Comptez le nombre de livres publiés par année.

19. Transactions SQL

Écrivez une transaction SQL qui insère un nouvel emprunt et met à jour la disponibilité du livre concerné.

20. Adhérents par âge

Affichez la liste des adhérents classés par âge, du plus jeune au plus âgé.

Exercice: Gestion des commandes d'un magasin

Schéma de la base de données :

1. **Produit (Product)**
    - id_produit (INT, PRIMARY KEY)
    - nom (VARCHAR(100))
    - prix (DECIMAL(10, 2))
    - categorie (VARCHAR(50))
    - stock (INT)
2. **Client (Customer)**
    - id_client (INT, PRIMARY KEY)
    - nom (VARCHAR(100))

- prenom (VARCHAR(100))
- email (VARCHAR(100))
- telephone (VARCHAR(20))
- ville (VARCHAR(50))

3. **Commande (Order)**

    - id_commande (INT, PRIMARY KEY)
    - id_client (INT, FOREIGN KEY REFERENCES Client(id_client))
    - date_commande (DATE)
    - statut (VARCHAR(20)) -- Exemple : 'En cours', 'Expédiée', 'Annulée'

4. **Détail_Commande (Order_Detail)**

    - id_detail (INT, PRIMARY KEY)
    - id_commande (INT, FOREIGN KEY REFERENCES Commande(id_commande))
    - id_produit (INT, FOREIGN KEY REFERENCES Produit(id_produit))
    - quantite (INT)

Questions SQL :

1. Création des tables

Écrivez les requêtes SQL pour créer les tables avec les contraintes appropriées.

2. Insertion de données

Ajoutez des données d'exemple pour chaque table :

- Au moins 5 produits (avec des catégories distinctes).
- 5 clients.
- 5 commandes, chacune avec 2 ou 3 produits associés.

3. Liste des commandes

Affichez toutes les commandes avec le nom et prénom du client, la date de commande, et le statut.

4. Total des commandes

Affichez l'ID de chaque commande et le montant total de la commande (quantité × prix).

5. Produits dans une commande

Affichez tous les produits d'une commande donnée avec leur quantité, leur prix unitaire et le total pour chaque produit.

6. Commandes par client

Affichez le nombre total de commandes effectuées par chaque client.

7. Commandes en cours

Affichez les commandes ayant le statut "En cours", avec les informations du client et la date.

8. Produits en rupture de stock

Trouvez tous les produits dont le stock est égal à zéro.

9. Stock total par catégorie

Calculez le stock total disponible pour chaque catégorie de produit.

10. Revenu par catégorie

Calculez le revenu total généré par chaque catégorie de produit.

11. Clients sans commandes

Trouvez les clients qui n'ont jamais passé de commande.

12. Commandes annulées

Affichez les commandes qui ont été annulées, avec les informations des clients.

13. Ville avec le plus de clients

Affichez la ville qui compte le plus de clients.

14. Client ayant dépensé le plus

Trouvez le client qui a dépensé le plus d'argent dans le magasin.

15. Détail des commandes passées un jour donné

Affichez toutes les commandes passées à une date spécifique, avec les détails des produits.

16. Commandes contenant un produit spécifique

Affichez toutes les commandes qui contiennent un produit donné (par exemple, "Laptop").

17. Stock minimal

Affichez les produits dont le stock est inférieur à 5.

18. Produits jamais commandés

Trouvez les produits qui n'ont jamais été commandés.

19. Jointure imbriquée

Affichez le client, la commande, et les produits achetés, triés par date de commande.

20. Suppression d'une commande

Supprimez une commande et mettez à jour le stock des produits correspondants.

Correction: Exercice 1

Correction des questions

1. Création de la base de données

```
CREATE TABLE Livre (
   id_livre INT PRIMARY KEY,
   titre VARCHAR(100),
   auteur VARCHAR(100),
   annee_publication INT,
   categorie VARCHAR(50),
   disponible BOOLEAN
);

CREATE TABLE Adherent (
   id_adherent INT PRIMARY KEY,
   nom VARCHAR(100),
   prenom VARCHAR(100),
   date_naissance DATE,
   email VARCHAR(100),
   telephone VARCHAR(20)
);

CREATE TABLE Emprunt (
   id_emprunt INT PRIMARY KEY,
   id_livre INT REFERENCES Livre(id_livre),
   id_adherent INT REFERENCES Adherent(id_adherent),
   date_emprunt DATE,
   date_retour DATE
);
```

2. Insertion de données

```
INSERT INTO Livre VALUES
(1, 'SQL pour les nuls', 'John Doe', 2018, 'Informatique', TRUE),
(2, 'L’art de la guerre', 'Sun Tzu', 2000, 'Philosophie', TRUE),
(3, 'Python avancé', 'Guido van Rossum', 2021, 'Programmation', FALSE);

INSERT INTO Adherent VALUES
(1, 'Martin', 'Paul', '1990-05-12', 'paul.martin@email.com', '0612345678'),
(2, 'Dubois', 'Julie', '1985-09-24', 'julie.dubois@email.com', '0623456789');

INSERT INTO Emprunt VALUES
(1, 1, 1, '2024-01-01', '2024-01-15');
```

3. Sélection de tous les livres

```
SELECT * FROM Livre ORDER BY titre;
```

4. Filtrer les livres par catégorie

```
SELECT * FROM Livre WHERE categorie = 'Informatique';
```

5. Vérification de la disponibilité

```
SELECT titre FROM Livre WHERE disponible = TRUE;
```

6. Recherche par auteur

```
SELECT * FROM Livre WHERE auteur = 'Victor Hugo';
```

7. Détail des emprunts

```
SELECT A.nom, A.prenom, L.titre
FROM Emprunt E
JOIN Adherent A ON E.id_adherent = A.id_adherent
JOIN Livre L ON E.id_livre = L.id_livre;
```

8. Nombre total de livres

```
SELECT COUNT(*) AS nombre_livres FROM Livre;
```

9. Emprunts par adhérent

```
SELECT id_adherent, COUNT(*) AS nombre_emprunts
FROM Emprunt
GROUP BY id_adherent;
```

10. Emprunts tardifs

```
SELECT * FROM Emprunt WHERE date_retour < CURRENT_DATE;
```

11. Adhérents sans emprunt

```
SELECT * FROM Adherent WHERE id_adherent NOT IN (SELECT DISTINCT id_adherent FROM Emprunt);
```

12. Mise à jour de la disponibilité

```
UPDATE Livre SET disponible = FALSE WHERE id_livre = 1;
```

13. Suppression d'un adhérent

```
DELETE FROM Adherent WHERE id_adherent NOT IN (SELECT DISTINCT id_adherent FROM Emprunt);
```

14. Catégories distinctes

```
SELECT DISTINCT categorie FROM Livre;
```

15. Emprunts les plus récents

```
SELECT * FROM Emprunt ORDER BY date_emprunt DESC LIMIT 3;
```

16. Livre le plus ancien

```
SELECT * FROM Livre ORDER BY annee_publication ASC LIMIT 1;
```

17. Longueur du titre

```
SELECT * FROM Livre WHERE LENGTH(titre) > 20;
```

18. Regroupement par année de publication

```
SELECT annee_publication, COUNT(*) FROM Livre GROUP BY annee_publication;
```

19. Transactions SQL

```
BEGIN TRANSACTION;
INSERT INTO Emprunt (id_emprunt, id_livre, id_adherent, date_emprunt, date_retour)
VALUES (2, 2, 2, '2024-02-01', '2024-02-15');
UPDATE Livre SET disponible = FALSE WHERE id_livre = 2;
COMMIT;
```

20. Adhérents par âge

```
SELECT nom, prenom, date_naissance FROM Adherent ORDER BY date_naissance DESC;
```

Correction des questions

1. Création des tables

```
CREATE TABLE Produit (
   id_produit INT PRIMARY KEY,
   nom VARCHAR(100),
   prix DECIMAL(10, 2),
   categorie VARCHAR(50),
   stock INT
);

CREATE TABLE Client (
   id_client INT PRIMARY KEY,
   nom VARCHAR(100),
   prenom VARCHAR(100),
   email VARCHAR(100),
   telephone VARCHAR(20),
   ville VARCHAR(50)
);

CREATE TABLE Commande (
   id_commande INT PRIMARY KEY,
   id_client INT REFERENCES Client(id_client),
   date_commande DATE,
   statut VARCHAR(20)
);

CREATE TABLE Détail_Commande (
   id_detail INT PRIMARY KEY,
   id_commande INT REFERENCES Commande(id_commande),
   id_produit INT REFERENCES Produit(id_produit),
   quantite INT
```

);

2. Insertion de données

```
INSERT INTO Produit VALUES
(1, 'Laptop', 800.00, 'Informatique', 10),
(2, 'Clavier', 30.00, 'Informatique', 50),
(3, 'Téléphone', 500.00, 'Électronique', 20),
(4, 'Chaise', 100.00, 'Mobilier', 5),
(5, 'Bureau', 300.00, 'Mobilier', 2);

INSERT INTO Client VALUES
(1, 'Martin', 'Paul', 'paul.martin@email.com', '0612345678', 'Paris'),
(2, 'Dubois', 'Julie', 'julie.dubois@email.com', '0623456789', 'Lyon'),
(3, 'Durand', 'Luc', 'luc.durand@email.com', '0634567890', 'Marseille');

INSERT INTO Commande VALUES
(1, 1, '2025-01-01', 'En cours'),
(2, 2, '2025-01-02', 'Expédiée'),
(3, 3, '2025-01-03', 'Annulée');

INSERT INTO Détail_Commande VALUES
(1, 1, 1, 2),
(2, 1, 2, 1),
(3, 2, 3, 1),
(4, 2, 4, 2),
(5, 3, 5, 1);
```

3. Liste des commandes

```
SELECT C.id_commande, CL.nom, CL.prenom, C.date_commande, C.statut
FROM Commande C
JOIN Client CL ON C.id_client = CL.id_client;
```

4. Total des commandes

```
SELECT C.id_commande, SUM(D.quantite * P.prix) AS total
FROM Commande C
JOIN Détail_Commande D ON C.id_commande = D.id_commande
JOIN Produit P ON D.id_produit = P.id_produit
GROUP BY C.id_commande;
```

5. Produits dans une commande

```
SELECT P.nom, D.quantite, P.prix, (D.quantite * P.prix) AS total
FROM Détail_Commande D
JOIN Produit P ON D.id_produit = P.id_produit
WHERE D.id_commande = 1;
```

6. Commandes par client

```
SELECT CL.nom, CL.prenom, COUNT(C.id_commande) AS total_commandes
FROM Client CL
LEFT JOIN Commande C ON CL.id_client = C.id_client
GROUP BY CL.id_client;
```

7. Commandes en cours

```
SELECT CL.nom, CL.prenom, C.date_commande
FROM Commande C
JOIN Client CL ON C.id_client = CL.id_client
WHERE C.statut = 'En cours';
```

8. Produits en rupture de stock

```
SELECT * FROM Produit WHERE stock = 0;
```

9. Stock total par catégorie

```
SELECT categorie, SUM(stock) AS stock_total
FROM Produit
GROUP BY categorie;
```

10. Revenu par catégorie

```
SELECT P.categorie, SUM(D.quantite * P.prix) AS revenu
FROM Détail_Commande D
JOIN Produit P ON D.id_produit = P.id_produit
GROUP BY P.categorie;
```

11. Clients sans commandes

```
SELECT * FROM Client WHERE id_client NOT IN (SELECT DISTINCT id_client FROM
Commande);
```

12. Commandes annulées

```
SELECT CL.nom, CL.prenom, C.date_commande
FROM Commande C
JOIN Client CL ON C.id_client = CL.id_client
WHERE C.statut = 'Annulée';
```

13. Ville avec le plus de clients

```
SELECT ville, COUNT(*) AS total_clients
FROM Client
GROUP BY ville
ORDER BY total_clients DESC
LIMIT 1;
```

14. Client ayant dépensé le plus

```
SELECT CL.nom, CL.prenom, SUM(D.quantite * P.prix) AS total_depense
FROM Commande C
JOIN Client CL ON C.id_client = CL.id_client
JOIN Détail_Commande D ON C.id_commande = D.id_commande
JOIN Produit P ON D.id_produit = P.id_produit
GROUP BY CL.id_client
ORDER BY total_depense DESC
LIMIT 1;
```

15. Détail des commandes passées un jour donné

```
SELECT C.id_commande, C.date_commande, P.nom AS produit, D.quantite, P.prix, (D.quantite *
P.prix) AS total
FROM Commande C
JOIN Détail_Commande D ON C.id_commande = D.id_commande
JOIN Produit P ON D.id_produit = P.id_produit
WHERE C.date_commande = '2025-01-02';
```

16. Commandes contenant un produit spécifique

```
SELECT C.id_commande, CL.nom, CL.prenom, P.nom AS produit, D.quantite, C.date_commande
FROM Commande C
JOIN Détail_Commande D ON C.id_commande = D.id_commande
JOIN Produit P ON D.id_produit = P.id_produit
JOIN Client CL ON C.id_client = CL.id_client
WHERE P.nom = 'Laptop';
```

17. Stock minimal

```
SELECT id_produit, nom, stock
FROM Produit
WHERE stock < 5;
```

18. Produits jamais commandés

```
SELECT P.id_produit, P.nom
FROM Produit P
LEFT JOIN Détail_Commande D ON P.id_produit = D.id_produit
```

```
WHERE D.id_produit IS NULL;
```

19. Jointure imbriquée

```
SELECT CL.nom, CL.prenom, C.id_commande, C.date_commande, P.nom AS produit, D.quantite
FROM Commande C
JOIN Client CL ON C.id_client = CL.id_client
JOIN Détail_Commande D ON C.id_commande = D.id_commande
JOIN Produit P ON D.id_produit = P.id_produit
ORDER BY C.date_commande ASC;
```

20. Suppression d'une commande

```
BEGIN TRANSACTION;

-- Récupérer les quantités de produits à restituer au stock
UPDATE Produit
SET stock = stock + (SELECT D.quantite FROM Détail_Commande D WHERE D.id_produit =
Produit.id_produit AND D.id_commande = 1)
WHERE id_produit IN (SELECT id_produit FROM Détail_Commande WHERE id_commande = 1);

-- Supprimer les détails de la commande
DELETE FROM Détail_Commande WHERE id_commande = 1;

-- Supprimer la commande
DELETE FROM Commande WHERE id_commande = 1;

COMMIT;
```