Requêtes SQL (MySQL compatibles)

A. Liste des reservations avec nom client et ville de l'hotel

```
SELECT r.id_chambre, r.date_debut, r.date_fin, c.nom_complet, h.ville
FROM Reservation r
JOIN Client c ON r.id_client = c.id_client
JOIN Chambre ch ON r.id_chambre = ch.id_chambre
JOIN Hotel h ON ch.id_hotel = h.id_hotel;
```

B. Clients habitant a Paris

```
SELECT * FROM Client WHERE ville = 'Paris';
```

C. Nombre de reservations par client

```
SELECT c.nom_complet, COUNT(*) AS nb_reservations
FROM Reservation r
JOIN Client c ON r.id_client = c.id_client
GROUP BY c.id_client;
```

D. Nombre de chambres par type

```
SELECT tc.nom_type, COUNT(*) AS nb_chambres
FROM Chambre ch
JOIN TypeChambre tc ON ch.id_type = tc.id_type
```

E. Liste des chambres non reservees entre deux dates

```
SELECT ch.*
FROM Chambre ch
WHERE ch.id_chambre NOT IN (
SELECT r.id_chambre
FROM Reservation r
WHERE NOT (r.date_fin < '2025-07-01' OR r.date_debut > '2025-07-10')
```

Requêtes en algèbre relationnelle

A. Reservations avec client et ville hotel

```
π_{id_chambre, date_debut, date_fin, nom_complet, ville} (
Reservation ⋈ Reservation.id_client=Client.id_client Client
⋈ Client.id_client=Chambre.id_chambre Chambre
⋈ Chambre.id_hotel=Hotel.id_hotel Hotel
)
```

B. Clients a Paris

```
σ_{ville='Paris'}(Client)
```

C. Nombre de reservations par client

```
γ_{id_client; count(*) → nb_reservations}(Reservation)
```

D. Nombre de chambres par type

```
γ_{id_type; count(*) → nb_chambres}(Chambre)
```

E. Chambres non reservees entre deux dates

```
Chambre - π_{id_chambre}(
σ_{(date_debut <= date_fin_saisie) ∧ (date_fin >= date_debut_saisie)}(Reservation)
)
```

Expliquez les principales differences entre MySQL et SQLite.

MySQL est un système de gestion de base de données relationnelle (SGBDR) de type client-

serveur. Cela signifie qu'il nécessite l'installation d'un serveur indépendant avec lequel les applications communiquent via le réseau. Il est conçu pour gérer des bases de données volumineuses, des applications multi-utilisateurs et des environnements de production à grande échelle. Il offre également une gestion avancée des utilisateurs, des droits d'accès et de la sécurité.
SQLite, en revanche, est une base de données légère et embarquée. Elle ne nécessite pas de serveur : les données sont stockées directement dans un fichier local avec l'extension .db. Cela la rend très simple à utiliser, rapide à déployer et particulièrement adaptée aux applications locales, mobiles, aux tests ou aux prototypes. Cependant, elle ne propose pas une gestion avancée des utilisateurs ni de mécanismes de sécurité complexes comme MySQL.
En résumé, MySQL est adapté aux projets complexes et collaboratifs, tandis que SQLite est plus simple, autonome et destiné aux petits projets ou aux usages individuels.