

1.

Pour optimiser une requête SQL, je commence par vérifier les indexes sur les colonnes les plus souvent utilisées dans les JOIN, WHERE ou ORDER BY. J'évite les SELECT \* quand je n'ai pas besoin de toutes les colonnes. J'essaye aussi de limiter les sous-requêtes trop complexes et de regrouper les opérations qui peuvent l'être. Enfin, j'analyse parfois le plan d'exécution pour comprendre ce qui ralentit.

2.

Une façon simple, c'est d'utiliser DISTINCT, mais ça ne modifie pas la table. Pour vraiment supprimer les doublons, je peux créer une nouvelle table temporaire avec des lignes uniques puis remplacer l'ancienne. Sinon, je peux utiliser une requête avec ROW\_NUMBER() ou GROUP BY selon le contexte.

3.

WHERE filtre les lignes avant les regroupements (GROUP BY), alors que HAVING filtre après. Par exemple, si je veux les catégories qui ont plus de 10 films, j'utilise HAVING COUNT(\*) > 10. Par contre, pour ne garder que les films de plus de 2h, ce sera dans le WHERE.

4.

La normalisation consiste à organiser les données pour éviter les redondances. Par exemple, on sépare les clients, les commandes, les produits, etc. La dénormalisation fait l'inverse : on regroupe les données pour aller plus vite à lire, mais au prix d'un peu de duplication.

5.

DELETE supprime les lignes une par une, donc on peut mettre une condition WHERE. TRUNCATE efface tout d'un coup, plus rapidement, mais sans possibilité de filtrer. Aussi, DELETE garde une trace dans les logs (donc plus sûr pour les rollbacks), alors que TRUNCATE est plus radical.

6.

Je peux utiliser une contrainte UNIQUE sur les colonnes sensibles. Je peux aussi faire un SELECT avant un INSERT, ou utiliser INSERT IGNORE, ON CONFLICT (en PostgreSQL) ou ON DUPLICATE KEY UPDATE (en MySQL) pour gérer les doublons sans planter.

7. Il y a :

- Un-à-un (1:1) → par exemple, un profil utilisateur pour un seul compte.
- Un-à-plusieurs (1:N) → par exemple, un client peut passer plusieurs commandes.
- Plusieurs-à-plusieurs (N:N) → comme un élève qui peut suivre plusieurs cours, et chaque cours avoir plusieurs élèves. Là, on passe par une table intermédiaire.