CORRIGÉ

Tle

Corrigé détaillé
4 exercices

Bases de données relationnelles — SQL

> Ce qui est noté au contrôle, ce n'est pas le résultat seul : c'est le raisonnement posé, puis le calcul mené jusqu'au bout avec l'unité.

1 Vocabulaire. Répondre en une phrase précise.

1. a. La **clé primaire** identifie de façon **unique** chaque ligne ; la **clé étrangère référence** la clé primaire d'une autre table.
2. b. La garantie qu'une **clé étrangère** pointe toujours vers une ligne qui **existe**.
3. c. Une information n'est stockée qu'à **un seul endroit** : la modifier ne demande qu'une correction, et deux versions contradictoires ne peuvent pas coexister.
4. d. *WHERE* filtre les **lignes** ; *HAVING* filtre les **groupes**, après **GROUP BY**.
5. e. Le fait de faire interpréter comme du **code** ce qui était censé être une **donnée**.

2 Une table Clients compte 1 000 lignes, une table Commandes 5 000 lignes. Une requête les joint sans condition de rapprochement.

1. a. Le produit des deux tailles :
$$1000 \times 5000 = 5000000$$
2. Soit **cinq millions** de lignes.
3. b. Au plus **5 000** : chaque commande est rattachée à **un** client.
4. c. Rapport :
$$5000000 \div 5000 = 1000$$
5. Soit **mille fois** plus de lignes que nécessaire.
6. d. Parce que la requête est **syntaxiquement valide** : le moteur n'a aucun moyen de deviner quelles colonnes devaient se correspondre. Il exécute ce qu'on lui demande — toutes les combinaisons.
7. e. Parce que sur de **petites tables de test**, le produit reste modeste et le résultat paraît plausible. C'est sur les **données réelles** que le volume explose — donc après la mise en production.

3 Construire des requêtes. Décrire en français ce que chacune doit faire, puis sa structure.

1. a. **SELECT** des colonnes voulues, *FROM* Clients, *WHERE* sur la colonne ville. Aucun **JOIN** : une seule table suffit.
2. b. **SELECT** le client et un agrégat de comptage, *FROM* Clients, **JOIN** Commandes **sur la condition** reliant la **clé étrangère** à la **clé primaire**, puis **GROUP BY** le client.
3. c. La même requête, complétée d'un *HAVING* portant sur le **comptage** supérieur à trois.
4. d. Parce que *WHERE* s'applique **avant** le regroupement : au moment où il est évalué, les **groupes n'existent pas encore**, donc le comptage non plus. C'est l'**ordre d'exécution** qui impose *HAVING*.

4 Injection SQL. Analyser.

1. a. De la **façon d'écrire la requête** : de la **concaténation** d'une saisie d'utilisateur dans une chaîne qui sera interprétée. Le langage n'y est pour rien.
2. b. La lecture de données auxquelles l'utilisateur n'a pas droit, leur **modification**, leur suppression, ou le contournement d'une authentification.
3. c. À déclarer d'abord la **structure** de la requête, puis à y passer les valeurs **séparément**. Le moteur sait alors que ces valeurs sont des **données** et ne les interprète jamais comme du code.
4. d. Parce qu'elle suppose qu'on a **pensé à tous les cas** — tous les caractères, tous les encodages, toutes les variantes. C'est une hypothèse qu'on ne peut pas démontrer, et il suffit d'un oubli.
5. e. Qu'il ne faut **jamais mélanger** ce qui relève de la **structure** d'une instruction et ce qui relève des **données**. Le même principe vaut hors des bases : c'est lui qui fonde l'échappement en HTML, la séparation des arguments dans un appel système, et le rejet des formats interprétés.

Corrige-toi toi-même. Compte tes points exercice par exercice : c'est ainsi qu'on voit où l'on perd, et ce n'est presque jamais là où on le croit.

Ce qui est évalué	Points	Ce qui fait perdre le point
Définir clés primaire et étrangère	3 pts	Confondre les deux rôles

Écrire une jointure avec sa condition	5 pts	Joindre deux tables sans condition
Distinguer WHERE et HAVING par l'ordre d'exécution	4 pts	Mettre un agrégat dans un WHERE
Vérifier le volume du résultat	3 pts	Accepter un résultat mille fois trop volumineux
Expliquer l'injection SQL et sa parade	5 pts	Proposer le filtrage des caractères comme solution complète