Antilles-Guyane – 2026 – sujet1 - Correction

Exercice 1 (6 points)

1. Avantages d'un SGBD sur un tableur

Par exemple (deux suffisent) :

- **cohérence et intégrité des données** : clés primaires et étrangères, types imposés, pas de doublon ni de commande qui renverrait à un client inexistant ;
- **pas de redondance** : le nom du client ou le prix de la pizza sont stockés une seule fois, et non recopiés à chaque commande ;
- **accès simultanés** de plusieurs utilisateurs sans conflit, **gestion des droits** et **sécurité** ;
- **requêtes** puissantes (SQL) pour rechercher, trier, regrouper, même sur un très grand nombre de commandes.

2. Deux pizzas avec le même numéro ?

Non. L'attribut num est la **clé primaire** de la relation pizza : sa valeur identifie une pizza de façon unique, deux lignes ne peuvent pas avoir le même num, quels que soient leur prix et leur couleur.

👉*Commentaire : en revanche, deux pizzas de numéros différents peuvent avoir la même couleur ou le même prix (pizzas 2 et 4 : Bianca).*

3. Une clé étrangère

Dans la relation **commande**, l'attribut **id_client** est une clé étrangère qui référence l'attribut **id** de la relation **client**.

Autre réponse possible : dans commande, **num_pizza** référence l'attribut **num** de la relation **pizza**.

4. Résultat de la requête

prix
11.5
15.5

👉*Commentaire : deux pizzas sont de couleur Bianca : les pizzas 2 et 4.*

5. Couleurs des pizzas à plus de 15 €

```sql
SELECT couleur
FROM pizza
WHERE prix > 15;
```

6. Clients livrés qui n'ont pas payé

```sql
SELECT DISTINCT client.prenom
FROM client
JOIN commande ON client.id = commande.id_client
WHERE commande.livraison = 1 AND commande.paiement = 0;
```

👉 *Commentaire : sur l'extrait, on obtient Raffaele (commande 42364). DISTINCT évite d'afficher plusieurs fois un client qui aurait plusieurs pizzas livrées et non payées.*

7. Prix moyen des pizzas commandées par Esposito

```sql
SELECT AVG(pizza.prix)
FROM pizza
JOIN commande ON pizza.num = commande.num_pizza
JOIN client ON client.id = commande.id_client
WHERE client.nom = 'Esposito';
```

👉 *Commentaire : il faut deux jointures : le nom est dans client, le prix dans pizza, et c'est la table commande qui fait le lien. Sur l'extrait : (11,5 + 12,5 + 11,5) / 3 ≈ 11,83 €.*

8. La commande 42365 est livrée

```sql
UPDATE commande
SET livraison = 1
WHERE code = 42365;
```

9. Nouveau client et sa commande

```sql
INSERT INTO client VALUES (107, 'Girasole', 'Emanuele');
INSERT INTO commande VALUES (42366, 107, 2, '2025-01-03', 0, 0);
```

Il faut d'abord **créer le client**, puis la commande : la clé étrangère id_client de la commande doit référencer un client qui existe déjà.

👉 *Commentaire : la pizza n'est pas encore livrée ni payée au moment de la commande : livraison et paiement valent 0.*

10. Problème posé par la suppression

Des commandes (42364 et 42365) référencent la pizza numéro 1 par la clé étrangère num_pizza. La supprimer violerait la **contrainte d'intégrité référentielle** : ces commandes renverraient à une pizza qui n'existe plus. Le SGBD **refuse** donc la suppression (ou, sans contrôle, la base devient incohérente).

👉*Commentaire : il faudrait d'abord supprimer ou modifier ces commandes, ce qui ferait perdre l'historique. Une meilleure idée : ajouter un attribut « en_vente » à la table pizza plutôt que de supprimer la ligne.*

11. Schéma des deux nouvelles tables

ingredient(id_ingredient : entier, nom : texte)

composition(#num_pizza : entier, #id_ingredient : entier)

La clé primaire de composition est le **couple (num_pizza, id_ingredient)** ; num_pizza est une clé étrangère qui référence pizza.num et id_ingredient une clé étrangère qui référence ingredient.id_ingredient.

👉*Commentaire : une pizza a plusieurs ingrédients et un ingrédient est utilisé par plusieurs pizzas (association « plusieurs à plusieurs ») : c'est pour cela qu'il faut une table de liaison. On peut y ajouter une quantité.*

12. Base accessible depuis Internet

Problèmes : la base devient **exposée aux attaques** (intrusion, vol ou modification des données, injections SQL), et elle contient des **données personnelles** (noms, commandes) protégées par le **RGPD**. Un client ne doit voir que **ses propres** commandes.

Précautions :

- **authentification** des clients (identifiant et mot de passe stocké haché) et **droits limités** (lecture seule de leurs commandes) ;
- communications **chiffrées** (HTTPS) ;
- ne pas laisser la base directement accessible : passer par une **application web** qui utilise des **requêtes préparées** (contre les injections SQL), derrière un **pare-feu** ;
- **sauvegardes** régulières et mises à jour de sécurité.