**Question 3 : Créer une fonction à partir d'une table**

Supposons que vous avez une table nommée Produits avec les colonnes suivantes :

* id\_produit (int)
* nom\_produit (varchar(100))
* prix (decimal(10,2))
* stock (int)

On vous demande de créer une fonction nommée appliquer\_remise qui prend un id\_produit et un taux\_remise en entrée et renvoie le nouveau prix du produit après application de la remise.

**Fonction:**

SQL

CREATE FUNCTION appliquer\_remise(produit\_id INT, taux\_remise DECIMAL(2,2))

RETURNS DECIMAL(10,2)

AS $$

BEGIN

DECLARE nouveau\_prix DECIMAL(10,2);

SELECT prix - (prix \* taux\_remise) INTO nouveau\_prix

FROM Produits

WHERE id\_produit = produit\_id;

RETURN nouveau\_prix;

END $$;

LANGUAGE plpgsql;

Utilisez le code [avec prudence](/faq#coding).

**Question 1 : Vrai ou Faux**

**1. VRAI ou FAUX:** Une table PostgreSQL **ne peut pas avoir plusieurs clés primaires.** Une clé primaire est une colonne ou un ensemble de colonnes qui identifie de manière unique chaque ligne de la table. Il ne peut y avoir qu'une seule clé primaire par table.

**2. VRAI ou FAUX:** L'opérateur && **est utilisé pour la conjonction logique (ET)** dans les requêtes SQL. Il permet de combiner deux conditions dans une même clause WHERE. Par exemple, SELECT \* FROM Clients WHERE nom = 'Dupont' && email = 'jean.dupont@exemple.com'; sélectionne tous les clients dont le nom est "Dupont" et l'email est "jean.dupont@exemple.com".

**3. VRAI ou FAUX:** Une vue **est une table virtuelle** qui matérialise les résultats d'une requête SQL. Elle ne stocke pas de données en elle-même, mais renvoie les résultats de la requête chaque fois qu'elle est utilisée. Les vues sont utiles pour créer des perspectives différentes sur les données existantes ou pour simplifier des requêtes complexes.

**Question 2 : Question théorique**

**1. Conception d'une base de données pour une boutique en ligne:**

Pour concevoir une base de données pour une boutique en ligne, on peut utiliser une structure relationnelle avec les tables suivantes :

* **Produits:** Cette table stocke les informations sur les produits, tels que le nom, la description, le prix, le stock, etc.
* **Commandes:** Cette table stocke les informations sur les commandes, tels que le numéro de commande, la date de commande, le client, l'état de la commande, etc.
* **Lignes\_commande:** Cette table stocke les détails des commandes, tels que le produit commandé, la quantité, le prix unitaire, etc.
* **Clients:** Cette table stocke les informations sur les clients, tels que le nom, l'adresse, l'email, le téléphone, etc.
* **Livraisons:** Cette table stocke les informations sur les livraisons, tels que le numéro de commande, l'adresse de livraison, le statut de la livraison, etc.

Ces tables peuvent être reliées entre elles par des clés étrangères pour maintenir l'intégrité des données. Par exemple, une commande est liée à un client par le biais de la colonne client\_id, et une ligne de commande est liée à une commande par le biais de la colonne commande\_id.

**2. Techniques de sauvegarde et de restauration pour PostgreSQL:**

PostgreSQL propose plusieurs techniques de sauvegarde et de restauration pour protéger vos données :

* **Sauvegarde complète de la base de données:** Cette méthode consiste à copier l'ensemble de la base de données, y compris les fichiers de données, les fichiers de configuration et les journaux de transaction.
* **Sauvegarde incrémentielle:** Cette méthode consiste à sauvegarder uniquement les modifications apportées à la base de données depuis la dernière sauvegarde complète.
* **Sauvegarde différentielle:** Cette méthode consiste à sauvegarder uniquement les différences entre la sauvegarde complète la plus récente et la sauvegarde incrémentielle la plus récente.
* **Sauvegarde par point dans le temps:** Cette méthode permet de créer une sauvegarde de la base de données à un moment précis.

La technique de sauvegarde la plus adaptée dépend de vos besoins spécifiques et de votre environnement.

**3. Concepts de normalisation des données:**

La normalisation des données est un processus visant à organiser les données dans une base de données de manière à minimiser les redondances et les anomalies de données. Il s'agit de diviser les données en tables plus petites et plus spécialisées, reliées entre elles par des clés étrangères.

Les avantages de la normalisation des données incluent :

* **Réduction des redondances:** Les données ne sont stockées qu'une seule fois, ce qui réduit l'espace de stockage et le risque d'erreurs de mise à jour.
* **Amélioration de l'intégrité des données:** Les relations entre les tables sont plus strictes, ce qui réduit le risque d'anomalies de données.
* **Facilité de modification et de mise à jour des données:** Les modifications apportées aux données sont plus faciles à effectuer car les données sont organisées de manière logique.

La normalisation des données implique plusieurs niveaux, allant de la normalisation 1NF (première forme normale) à la normalisation 5NF (cinquième forme normale). Le niveau de normalisation approprié dépend des besoins spécifiques de l'application.

N'hésitez pas à me poser des questions supplémentaires si vous avez besoin de plus d'informations.