1. **Question 1: VRAI ou FAUX** a. Les vues sont des objets de base de données physiques. b. Un déclencheur peut être utilisé pour valider les données avant leur insertion dans une table. c. Une procédure stockée peut retourner des résultats.
2. **Question 2: Question théorique** Expliquez la différence entre une vue et une table temporaire en SQL. Quand est-il approprié d'utiliser chacune de ces structures ?
3. **Question 3: Fonction à partir d'une table donnée** Écrivez une fonction en SQL qui prend en entrée un identifiant de client et retourne le nombre total de commandes passées par ce client à partir d'une table "Commandes".
4. **Question 4: Utilisation d'une fonction** Utilisez la fonction que vous avez écrite dans la question 3 pour afficher le nombre total de commandes passées par le client avec l'identifiant 1001.
5. **Question 5: Utilisation d'un déclencheur** Créez un déclencheur qui, lorsqu'une nouvelle commande est insérée dans la table "Commandes", met à jour automatiquement la table "Clients" en incrémentant le nombre total de commandes pour le client correspondant.
6. **Question 6: Utilisation d'une procédure** Écrivez une procédure stockée en SQL qui prend en entrée le nom d'un produit et retourne le nombre total de fois que ce produit a été commandé à partir de la table "DétailsCommandes".
7. **Question 7: Utilisation d'une vue** Créez une vue qui affiche les informations des clients (identifiant, nom, email) ainsi que le nombre total de commandes passées par chaque client à partir des tables appropriées de la base de données de l'examen.

 **Question 1: VRAI ou FAUX** a. FAUX - Les vues ne sont pas des objets physiques, ce sont des requêtes stockées. b. VRAI - Les déclencheurs peuvent être utilisés pour valider les données avant leur insertion. c. VRAI - Les procédures stockées peuvent retourner des résultats.

 **Question 2: Question théorique** Une vue est une requête stockée qui crée une "vue" virtuelle des données, tandis qu'une table temporaire est une structure temporaire utilisée pour stocker des données de manière temporaire. Les vues sont appropriées pour simplifier les requêtes complexes ou limiter l'accès aux données, tandis que les tables temporaires sont utiles pour stocker des résultats intermédiaires ou temporaires lors de traitements complexes.

 **Question 3: Fonction à partir d'une table donnée** Voici un exemple de fonction en SQL qui retourne le nombre total de commandes pour un client donné à partir d'une table "Commandes" :

sql

 CREATE FUNCTION GetTotalCommandesForClient (@ClientId INT)

RETURNS INT

AS

BEGIN

DECLARE @TotalCommandes INT;

SELECT @TotalCommandes = COUNT(\*) FROM Commandes WHERE ClientId = @ClientId;

RETURN @TotalCommandes;

END;

 **Question 4: Utilisation d'une fonction** Pour utiliser la fonction créée dans la question 3 pour afficher le nombre total de commandes passées par le client avec l'identifiant 1001, vous pouvez exécuter cette requête :

sql

 SELECT dbo.GetTotalCommandesForClient(1001) AS TotalCommandesClient1001;

 **Question 5: Utilisation d'un déclencheur** Voici un exemple de déclencheur en SQL qui met à jour automatiquement le nombre total de commandes pour un client lorsqu'une nouvelle commande est insérée dans la table "Commandes" :

sql

 CREATE TRIGGER UpdateTotalCommandes

ON Commandes

AFTER INSERT

AS

BEGIN

UPDATE Clients

SET TotalCommandes = TotalCommandes + 1

WHERE ClientId = (SELECT ClientId FROM inserted);

END;

 **Question 6: Utilisation d'une procédure** Voici un exemple de procédure stockée en SQL qui retourne le nombre total de fois qu'un produit a été commandé à partir de la table "DétailsCommandes" :

sql

 CREATE PROCEDURE GetTotalCommandesProduit (@ProductName NVARCHAR(50))

AS

BEGIN

SELECT COUNT(\*) AS TotalCommandesProduit

FROM DétailsCommandes

WHERE NomProduit = @ProductName;

END;

 **Question 7: Utilisation d'une vue** Voici un exemple de vue en SQL qui affiche les informations des clients ainsi que le nombre total de commandes passées par chaque client :

sql

CREATE VIEW VueInfoClients AS

SELECT c.ClientId, c.Nom, c.Email, COUNT(\*) AS TotalCommandes

FROM Clients c

INNER JOIN Commandes cmd ON c.ClientId = cmd.ClientId

GROUP BY c.ClientId, c.Nom, c.Email;

**1. Fonction :**

sql

-- Création d'une fonction qui calcule le carré d'un nombre

CREATE FUNCTION calculer\_carre(num INT) RETURNS INT AS $$

BEGIN

RETURN num \* num;

END;

$$ LANGUAGE plpgsql;

Vous pouvez appeler cette fonction comme ceci :

sql

SELECT calculer\_carre(5); -- Résultat : 25

**2. Déclencheur :**

sql

-- Création d'un déclencheur qui met à jour la date de modification d'une ligne

CREATE OR REPLACE FUNCTION maj\_date\_modif()

RETURNS TRIGGER AS $$

BEGIN

NEW.date\_modif := NOW();

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER declencheur\_maj\_date\_modif

BEFORE INSERT OR UPDATE ON ma\_table

FOR EACH ROW

EXECUTE FUNCTION maj\_date\_modif();

Dans cet exemple, ma\_table est le nom de la table sur laquelle le déclencheur est appliqué. Le déclencheur mettra à jour la colonne date\_modif avant chaque insertion ou mise à jour.

**3. Procédure :**

sql

-- Création d'une procédure stockée qui affiche le nom et l'âge d'un utilisateur

CREATE PROCEDURE afficher\_infos\_utilisateur(in\_id INT)

LANGUAGE plpgsql

AS $$

DECLARE

nom\_utilisateur TEXT;

age\_utilisateur INT;

BEGIN

SELECT nom, age INTO nom\_utilisateur, age\_utilisateur

FROM utilisateurs

WHERE id = in\_id;

RAISE NOTICE 'Nom: %, Âge: %', nom\_utilisateur, age\_utilisateur;

END;

$$;

Pour appeler cette procédure :

sql

CALL afficher\_infos\_utilisateur(1);

**4. Vue :**

sql

-- Création d'une vue qui affiche les informations des utilisateurs avec leur département

CREATE VIEW vue\_utilisateurs\_departement AS

SELECT u.nom, u.age, d.nom\_departement

FROM utilisateurs u

JOIN departements d ON u.departement\_id = d.id;

Vous pouvez ensuite interroger cette vue comme une table :

sql

SELECT \* FROM vue\_utilisateurs\_departement;