Fonctions stockées (scalaires et d'agrégation)

1. **Total des achats d'un client**
   Fonction scalaire `fn_total_achats(client_id INT) RETURNS NUMERIC` qui calcule la somme des montants de tous les paniers achetés d'un client.
2. **Chiffre d'affaires mensuel**
   Fonction d'agrégation personnalisée `agg_ca_mensuel(date DATE) RETURNS NUMERIC` qui retourne le CA du mois de `date`.
3. **Top 5 des jeux les plus vendus**
   Fonction `fn_top_ventes(limit INT DEFAULT 5) RETURNS TABLE(id_jeu INT, quantite BIGINT)`.
4. **Nom complet du client**
   Fonction scalaire `fn_nom_complet(client_id INT) RETURNS TEXT` concaténant prénom et nom.
5. **Validation du numéro de téléphone**
   Fonction `fn_valide_numero(num TEXT) RETURNS BOOLEAN` qui lève une exception si le format n'est pas conforme.

Procédures stockées

6. **Ajout d'un nouveau jeu**
   Procédure `proc_ajouter_jeu(nom TEXT, date_sortie DATE, prix NUMERIC, stock INT, editor_id INT)`.
7. **Mise à jour du stock**
   Procédure `proc_update_stock(jeu_id INT, delta INT)` qui incrémente/décrémente le stock.
8. **Application d'une promotion**
   Procédure `proc_appliquer_promo(panier_id INT, promo_id INT)` qui met à jour le total du panier.
9. **Validation d'un panier (transactionnelle)**
   Procédure `proc_valider_panier(panier_id INT, mode_paiement TEXT)` :
   - vérifie le stock pour chaque choix
   - met `achete = TRUE`, fixe `date_achat`
   - rollback si stock insuffisant
10. **Batch d'augmentation tarifaire**
    Procédure `proc_augmenter_prix(pourcentage NUMERIC, seuil_age INT)` qui augmente le prix des jeux sortis depuis plus de `seuil_age` années.

Scripts avec curseurs

11. **Jeux en rupture de stock**
    Script PL/pgSQL utilisant un curseur `CURSOR c_stock IS SELECT id, nom FROM jeu WHERE stock = 0;`.
12. **Commandes non achetées d'un client**
    Script avec curseur sur `panier` pour lister les paniers `achete = FALSE`.
13. **Parcours des promotions actives**
    Script curseur sur `promotion` pour appliquer un test unitaire.

Gestion d'exceptions

14. **Prix négatif**
    Bloc `BEGIN … EXCEPTION WHEN check_violation THEN … END;` pour intercepter tout `prix < 0`.
15. **Doublon d'email client**
    Exception personnalisée `RAISE EXCEPTION 'Email déjà existant : %', NEW.email` dans un trigger BEFORE INSERT sur `client`.

Triggers

16. **BEFORE INSERT sur panier**
    Calcule automatiquement `total` à partir des `choix` associés avant insertion.
17. **AFTER INSERT sur choix**
    Décrémente le stock du jeu correspondant.
18. **BEFORE DELETE sur jeu**
    Empêche la suppression si le stock n'est pas à zéro (`RAISE EXCEPTION`).
19. **AFTER UPDATE du prix**
    Insère un enregistrement dans une table de log `prix_log(jeu_id, ancien_prix, nouveau_prix, date_modif)`.

Cas récursif

20. **Recommandations de jeux**
    Fonction récursive (ou `WITH RECURSIVE`) `fn_recommandations(client_id INT, profondeur INT)` qui :
    - Identifie les autres clients ayant acheté au moins un même jeu
    - Remonte les jeux achetés par ces clients, à une profondeur donnée
    - Retourne une liste de jeux "similaires" recommandés