EXERCICE 1 : LES FONCTIONS

1. *Ecrire la fonction **stock_cereale(refCereale)** qui retourne le stock d'une céréale passée en paramètre.. La céréale doit exister dans la bdd, message d'erreur si la céréale n'existe pas.*
   ***Paramètre** : **refCereale** : référence céréale ;*

```
CREATE OR REPLACE FUNCTION stock_cereale(p_refCereale varchar )  RETURNS NUMERIC AS $$
DECLARE
    v_qtestock numeric;
BEGIN
    -- on récupére le stock
    SELECT sum(qtestock) INTO v_qtestock FROM silo
                where codecereale= p_refCereale;
    -- si retourne NULL : la céréale n'existe pas
    IF v_qtestock is NULL
      THEN
         RAISE EXCEPTION 'la céréale ref  % n''existe pas', p_refCereale;
    END IF;

    RETURN v_qtestock;
END
$$ LANGUAGE plpgsql;
```

```
-- Jeux de test
SELECT stock_cereale('MS440')  ; -- =>656
SELECT stock_cereale('CZ120')  ; -- =>556
SELECT stock_cereale('BL500')  ; -- =>580
SELECT stock_cereale('BE500')  ; -- => la céréale ref  BE500 n'existe pas
```

2. *Ecrire la fonction **verif_contrat(refCereale,qte)** qui permet de vérifier si le stock d'une céréale permet de satisfaire un contrat (une commande) pour une quantité donnée*
   Si la quantité en stock est suffisante la fonction doit retourner TRUE sinon FALSE.
   La céréale doit exister dans la bdd, message d'erreur si la céréale n'existe pas.
   ***Paramètre** : **refCereale** : référence céréale;*
   ***qte** : Quantité demandée ;*

```
CREATE OR REPLACE FUNCTION verif_contrat(p_refCereale varchar,p_qte int) RETURNS boolean AS
$$
DECLARE
BEGIN
    -- on utilise la fonction qui retourne la qte en stock d'une céréale passés en
    --paramétre, cette fonction vérifie que la céréale existe

    RETURN (stock_cereale(p_refCereale) >= p_qte);
END
$$ LANGUAGE plpgsql;
```

```
-- Jeux de test
SELECT verif_contrat('MS440',700)  ; -- => FALSE
SELECT verif_contrat('BL500',45)  ; -- => TRUE
SELECT verif_contrat('BE500',980)  ; -- => ERREUR:  la céréale ref  BE500 n'existe pas
```

3. Ecrire la fonction **ajout_cereale*(refCereale,qte)*** qui permet d'ajouter une céréale dans les silos. Cette fonction cherchera les silos qui contiennent cette céréale et remplira le silo le plus plein et complétera le stockage dans les autres silos toujours en complétant les silos les plus pleins. La fonction retourne la quantité de céréale qui n'a pas pu être stockée.
La céréale doit exister dans la bdd, message d'erreur si la céréale n'existe pas

Paramètre : ***refCereale*** : référence céréale;
qte : Quantité à ajouter ;

```
CREATE OR REPLACE FUNCTION ajout_cereale(p_refCereale varchar,p_qte int) RETURNS integer AS
$$
DECLARE
    v_qtelibre numeric; -- quantité restante dans le silo courant
    v_qteAstocker numeric; -- quantité à stocker dans le silo courant

    v_qte_restante numeric := p_qte; -- quantité restante à stocker
    v_ref varchar;
    rec_silo silo%ROWTYPE;

    -- création d'un curseur paramétré qui recherche les silos qui stockent la céréale
    -- trié par ordre décroissant (du plus rempli au moins rempli)
    Curs_ListeSilo CURSOR(ref varchar) IS SELECT * from silo
                    where codecereale= ref
                    order by qtestock desc;

BEGIN
    -- on vérifie que la céréale existe
    Select codecereale INTO v_ref from cereale where codecereale= p_refCereale;
    IF not found
       THEN RAISE EXCEPTION 'la céréale ref  % n''existe pas', p_refCereale;
    END IF;

    OPEN Curs_ListeSilo(p_refCereale);
    -- parcours du curseur paramétré
    LOOP
       FETCH Curs_ListeSilo INTO rec_silo;
       EXIT WHEN NOT FOUND; -- sort de la boucle si le curseur ne retourne plus rien

       -- calcul de la place qu'il reste dans le silo courant
       IF  v_qte_restante > 0
          THEN
          -- calcul de la place qu'il reste dans le silo courant
              v_qtelibre:= rec_silo.qtemax - rec_silo.qtestock ;
          -- si la qte libre est > à la qte restante on ajoute la qte restante
          -- sinon la quantité libre
              IF v_qtelibre > v_qte_restante
                  THEN v_qteAstocker := v_qte_restante;
                  ELSE v_qteAstocker := v_qtelibre;
              END IF;
          -- mise à jour du silo
              Update silo
                  Set qtestock = qtestock+ v_qteAstocker
                  Where codesilo = rec_silo.codesilo;
          -- mise à jour de la quantité restante
              v_qte_restante := v_qte_restante - v_qteAstocker;
          ELSE EXIT; -- on sort si il n'y a plus rien à socker
       END IF;
    END LOOP;

    CLOSE Curs_ListeSilo;
    Return v_qte_restante;
END
$$ LANGUAGE plpgsql;
```

codesilo	codecereale	qtestock	qtemax
A	BL500	580	700
I	CZ120	78	400
G	CZ120	478	700
H	CZ130	471	700
E	MS440	48	500
D	MS440	608	700
F	MS450	456	700
C	OR180	255	450
B	OR180	120	500
J	SG010	95	700

Vous utiliserez les valeurs de paramètre suivantes pour faire vos tests :

```
SELECT ajout_cereale('MS440',92)  ; -- 0
SELECT ajout_cereale('MS440',300) ; -- 0
SELECT ajout_cereale('MS440',200) ; -- 48
SELECT ajout_cereale('MS440',200) ; -- 200
SELECT ajout_cereale('MY440',200) ; --  ERREUR:  La céréale MY440 est inconnue !
```

EXERCICE 2 : LES TRIGGERS

1. *Ecrire le trigger **trig_insert_contrat()** qui lors de l'insertion d'un contrat vérifie :*
   - *Que la céréale existe, message d'erreur le cas contraire*
   - *Que le négociant existe, message d'erreur le cas contraire*
   - *Que la quantité de céréale demandée est supérieur à 0, message d'erreur le cas contraire*
   - *Que la quantité de céréale demandée peut être satisfaite (il y en a assez en stock), message d'erreur le cas contraire*

   Lors de l'insertion la date du contrat n'est pas renseignée ni le numéro du contrat, le trigger doit mettre cette date à la date courante et générer un numéro de contrat valide, avant l'insertion effective.
   ***Attention** ce contrat n'est pas encore livré, il ne faut donc pas modifier la quantité en stock de céréale.*

Définition de la fonction triggers

```
CREATE or replace FUNCTION funct_insert_contrat() RETURNS trigger AS $$

BEGIN
    -- on vérifie que la céréale existe
    perform codecereale from cereale where codecereale=NEW.codecereale;

    -- si aucune ligne retournée ou pas de codecereale
    IF NEW.codecereale IS NULL OR NOT FOUND
        THEN RAISE EXCEPTION 'la céréale ref  % n''existe pas', NEW.codecereale;
    END IF;

    -- on vérifie que le negociant existe
    perform no_negociant from negociant where no_negociant=NEW.no_negociant;

    IF NOT FOUND
        THEN RAISE EXCEPTION 'la négotiant N°  % n''existe pas', NEW.no_negociant;
    END IF;

    -- Vérifie que la qte>0
    IF NEW.qtecde <=0
        THEN RAISE EXCEPTION 'la qte (%) doit être > 0', NEW.qtecde;
    END IF;

    -- vérifie que le stock est suffisant pour satisfaire la commande
    -- appel de la fonction verif_contrat()
    If NOT verif_contrat(NEW.codecereale, NEW.qtecde)
        THEN RAISE EXCEPTION 'pas assez de cérale % en stock', NEW.codecereale;
    END IF;

    NEW.datecontrat:= current_timestamp;
    NEW.nocontrat := (SELECT max(nocontrat)+1 from contrat);

RETURN NEW;
END; $$ LANGUAGE plpgsql;
```

Définition du trigger

```
CREATE TRIGGER trig_insert_contrat BEFORE INSERT ON contrat
                FOR EACH ROW EXECUTE PROCEDURE funct_insert_contrat();
```

```
INSERT INTO contrat (codecereale,no_negociant,qtecde) VALUES('CA120',48,400);
            -- ERREUR:  la céréale ref  CA120 n'existe pas
INSERT INTO contrat (codecereale,no_negociant,qtecde) VALUES('CZ120',80,400);
            -- ERREUR:  la négotiant N°  80 n'existe pas
INSERT INTO contrat (codecereale,no_negociant,qtecde) VALUES('CZ120',48,800);
            -- ERREUR:  pas assez de cérale CZ120 en stock
INSERT INTO contrat (codecereale,no_negociant,qtecde) VALUES('CZ120',48,-100);
            -- ERREUR:  la qte (-100) doit être > 0
INSERT INTO contrat (codecereale,no_negociant,qtecde) VALUES('CZ120',48,400);
            -- 1 ligne affectée
```