Exercice 1

```
DROP FUNCTION banque."empruntsClient";

CREATE FUNCTION banque."empruntsClient"(idClient integer)
    RETURNS boolean AS $$
    DECLARE
        clienttmp record;
        dataToSend record;

    BEGIN
        SELECT *
        INTO clienttmp
        FROM banque.client
        WHERE ncli = idClient;

        IF NOT FOUND THEN
            RETURN FALSE;
        END IF;

        raise notice 'Client : % %', clienttmp.prenomcli,
clienttmp.nomcli;

        FOR dataToSend IN
            SELECT Emprunt.nemprunt, Emprunt.montant, Compte.ncompte,
Agence.nag
            FROM
            banque.client as Client,
            banque.emprunt as Emprunt,
            banque.compte_client as Compteclient,
            banque.compte as Compte,
            banque.agence as Agence
            WHERE Client.ncli = Compteclient.ncli
            AND Compteclient.ncompte = Emprunt.ncompte
            AND Client.ncli = idClient
            AND Compte.ncompte = Compteclient.ncompte
            AND Compte.nag = Agence.nag
        LOOP

        raise notice 'emprunt numero % de % euros sur le compte % de l
agence %', dataToSend.nemprunt, dataToSend.montant, dataToSend.ncompte,
dataToSend.nag;

        END LOOP;

        RETURN TRUE;
        END;$$
    LANGUAGE 'plpgsql';
```

Exercice 2

Function

```
DROP FUNCTION banque."trigger_compte";

CREATE FUNCTION banque."trigger_compte"()
    RETURNS trigger AS $$
    DECLARE
    clienttmp record;
    dataToSend record;
    BEGIN
    SELECT * INTO clienttmp FROM banque.client WHERE ncli = NEW.ncli;
    IF NOT FOUND THEN
        raise notice 'CLIENT ABSENT';
    END IF;

    raise notice 'Client : % %', clienttmp.prenomcli, clienttmp.nomcli;

    FOR dataToSend IN
        SELECT Compte.ncompte, Compte.typecpte
        FROM
        banque.compte_client as Compteclient,
        banque.compte as Compte
        WHERE Compte.ncompte = Compteclient.ncompte
        AND Compteclient.ncompte = NEW.ncompte
    LOOP
        IF dataToSend.typecpte = 'cpte courant' THEN
            return NEW;
        ELSE
            raise notice 'ERREUR PAS POSSIBLE';
            return false;
        END IF;
    END LOOP;

    return NEW;
    END;$$
    LANGUAGE 'plpgsql';
```

Trigger

```
CREATE TRIGGER before_insert_compte_client
    BEFORE INSERT or UPDATE
    ON banque.compte_client
    FOR EACH ROW
    EXECUTE PROCEDURE banque."trigger_compte"();
```

Exercice 3