TD4

```
--EXERCICE N1
--Q1
-- CREATE OR REPLACE FUNCTION TEST_EXIST(p_empno IN NUMBER) RETURN BOOLEAN
 -- IS
 -- v_nbr NUMBER;
 -- BEGIN
            -- SELECT COUNT(*) INTO v_nbr FROM emp
            -- WHERE empno=p_empno;
            -- IF v_nbr=0 THEN
                  -- RETURN FALSE;
           -- ELSE RETURN TRUE;
            -- END IF;
-- END;
-- /
--Test de la fonction TEST_EXIST
-- SET SERVEROUTPUT ON
-- ACCEPT sempno PROMPT 'DONNER LE NUM DE EMP '
 -- BEGIN
 -- if TEST_EXIST(&sempno) then
  -- DBMS_OUTPUT.PUT_LINE(' l employe existe');
 -- else
 -- DBMS_OUTPUT.PUT_LINE('l employe n existe pas');
 -- end if;
  -- END;
 -- /
 --Q2
-- CREATE OR REPLACE PROCEDURE INSERT_EMP(p_empno NUMBER,p_ename VARCHAR2,p_mgr
NUMBER,p_deptno NUMBER)
-- IS
-- v_valid BOOLEAN;
-- BEGIN
-- v_valid:=NOT TEST_EXIST(p_empno) AND p_empno IS NOT NULL AND TEST_EXIST(p_mgr);
-- IF v_valid THEN
            -- INSERT INTO emp(empno,ename,mgr,deptno)
            -- VALUES(p_empno,p_ename,p_mgr,p_deptno);
-- ELSE
            -- DBMS_OUTPUT.PUT_LINE('Données non valides');
-- END IF;
-- END;
-- /
--Test de la PROCEDURE INSERT_EMP
-- EXECUTE INSERT_EMP(7369,'XXX',7844,20);
----Doit retourner un message d'erreur
--Q3
-- CREATE OR REPLACE PROCEDURE LIST_DEPT(p_deptno NUMBER,p_job VARCHAR)IS
            -- CURSOR emp_cur(pc_deptno NUMBER,pc_job VARCHAR) IS SELECT empno,ename,sal FROM
emp
            -- WHERE deptno=pc_deptno AND job=pc_job;
            -- emp_rec emp_cur%ROWTYPE;
-- BEGIN
```

```
-- OPEN emp_cur(p_deptno,p_job);
-- LOOP
        -- FETCH emp_cur INTO emp_rec;
        -- EXIT WHEN emp_cur%NOTFOUND;
        -- DBMS_OUTPUT.PUT_LINE(emp_rec.empno||emp_rec.ename||emp_rec.sal);
-- END LOOP;
-- CLOSE emp_cur;
-- END;
-- /
--Test du LIST_DEPT
-- EXECUTE LIST_DEPT(20,'CLERK');
 --Q4
-- CREATE OR REPLACE TRIGGER NO_DELETE
-- BEFORE DELETE ON DEPT -- AVANT CHAQUE OPERATION DE SUPPRESSION SUR LA TABLE dept
-- FOR EACH ROW --POUR CHAQUE ENREGISTREMENT
-- BEGIN
        -- IF :OLD.LOC='DALLAS' THEN  -- LA VARIABLE OLD DETERMINE L'ANCIENNE VALEUR
                -- RAISE_APPLICATION_ERROR (-20001, 'Impossible de supprimer un departement localise a
Dallas');
                -- raise_application_error allows you to issue an user-defined error from a code block or stored
program.
-- The error_number is a negative integer with the range from -20999 to -20000.

        -- END IF;
-- END;
-- /
---Test du NO_DELETE
-- DELETE FROM DEPT WHERE LOC='DALLAS';

---------EXERCICE N2--------
--Q1
CREATE OR REPLACE FUNCTION f_nb_cli(p_na IN NUMBER) RETURN NUMBER
IS
        v_nb NUMBER:=0;
BEGIN
        SELECT COUNT( DISTINCT Num_Cli) INTO v_nb
        FROM compte WHERE num_agence=p_na;
        RETURN v_nb;
END;
/
--Test1 du FUNCTION f_nb_cli
ACCEPT s_na PROMPT 'DONNER LE NUMERO DE L AGENCE '
SELECT f_nb_cli(&s_na)from DUAL;
--Test2 du FUNCTION f_nb_cli
VARIABLE NCL NUMBER;
ACCEPT s_na PROMPT 'DONNER LE NUMERO DE L AGENCE '
BEGIN
 :NCL:=f_nb_cli(&s_na);
 END;
 /
 PRINT NCL
```

```
--Q2
 SET SERVEROUTPUT ON
 CREATE OR REPLACE PROCEDURE P_AFF
IS
        CURSOR c_ag IS SELECT num_agence, nom_agence, f_nb_cli(num_agence) nbr FROM agence;
        CURSOR c_cl(p_ag NUMBER) IS SELECT DISTINCT nom_cli, ville_cli FROM client c, compte co
        WHERE c.num_cli=co.num_cli AND num_agence=p_ag;
BEGIN
        FOR rec_ag IN c_ag LOOP
        DBMS_OUTPUT.PUT_LINE('-----------------------------');
                DBMS_OUTPUT.PUT_LINE('Agence : '||rec_ag.nom_agence);
                DBMS_OUTPUT.PUT_LINE('-----------------------------');
                DBMS_OUTPUT.PUT_LINE('Nom du Client          Ville du Client');
                DBMS_OUTPUT.PUT_LINE('----------------------------------------');
                FOR rec_cli IN c_cl(rec_ag.num_agence) LOOP
                        DBMS_OUTPUT.PUT_LINE(rec_cli.nom_cli||'      '||rec_cli.ville_cli);
                END LOOP;
                DBMS_OUTPUT.PUT_LINE('Nombre de clients : '||rec_ag.nbr);
                DBMS_OUTPUT.PUT_LINE('----------------------------------------');
        END LOOP;
END;
/
-- Test du PROCEDURE P_AFF
 EXECUTE P_AFF;

 -- -Q3
  CREATE OR REPLACE TRIGGER T_NO_UP
  BEFORE UPDATE ON compte
  FOR EACH ROW
  BEGIN
  IF(USER!='SYSTEM') THEN
  RAISE_APPLICATION_ERROR(-20002,'Impossible de mettre pour un non SYSTEM...');
  END IF;
  END;
  /
```

```
----------EXERCICE2--TD3
-- /*1*/
 --SET SERVEROUTPUT ON
 --DECLARE
   --CURSOR s_c IS SELECT sname FROM etudiant NATURAL JOIN inscription
     --WHERE corsno=102 AND note >= 10;
--BEGIN
  --FOR s_r IN s_c LOOP
    --DBMS_OUTPUT.PUT_LINE(s_r.sname);
   --END LOOP;
 --END;
 --/

-- /*2*/
--ACCEPT s_cors PROMPT 'Numéro du cours'
 --ACCEPT s_note PROMPT 'Seuil de la note'
 --SET SERVEROUTPUT ON
 --DECLARE
   --CURSOR s_c(p_note NUMBER, p_cors NUMBER) IS SELECT sname FROM etudiant NATURAL JOIN
inscription
     --WHERE corsno=p_cors AND note >= p_note;
             --BEGIN
   --FOR s_r IN s_c(&s_note,&s_cors) LOOP
     --DBMS_OUTPUT.PUT_LINE(s_r.sname);
   --END LOOP;
 --END;
 --/

-- /*3*/
ACCEPT s_cors PROMPT 'Numéro du cours'
 ACCEPT s_note PROMPT 'Seuil de la note'
 SET SERVEROUTPUT ON
 DECLARE
   CURSOR s_c(p_note NUMBER, p_cors NUMBER) IS SELECT stdno, sname FROM etudiant NATURAL JOIN
inscription
     WHERE corsno=p_cors AND note >= p_note
     ORDER BY deptno DESC, stdno ASC;
   e_cz EXCEPTION;
 BEGIN
   IF (&s_cors=0) THEN
    RAISE e_cz;
   END IF;
    FOR s_r IN s_c(&s_note,&s_cors) LOOP
     DBMS_OUTPUT.PUT_LINE(s_r.stdno||' '||s_r.sname);
   END LOOP;
 EXCEPTION
   WHEN e_cz THEN
     DBMS_OUTPUT.PUT_LINE('Le code du cours ne doit pas être égal à zéro');
END;
 /
```