Le devoir est écrit en pl/pgsql (posgresql) au lieu de pl/sql (oracle).

Exercice 1 : Blocs PL/SQL anonymes

1. Écrire un bloc PL/SQL calculant le maximum de deux nombres et le renvoie en sortie.

```
ma_bdd=# do $$
ma_bdd$# declare
ma_bdd$#  var1 numeric (7,2);
ma_bdd$#  var2 numeric (7,2);
ma_bdd$# begin
ma_bdd$#  SELECT MIN(sal) INTO var1 FROM EMP;
ma_bdd$#  SELECT MAX(sal) INTO var2 FROM EMP;
ma_bdd$#  raise notice '%', GREATEST(var1, var2);
ma_bdd$# end $$;
NOTICE:  5000.00
DO
ma_bdd=#
```

2. Écrire un bloc PL/SQL qui affiche le nombre d'employés d'un département dont le numéro est donné.

```
ma_bdd=# do $$
ma_bdd$#
ma_bdd$# declare
ma_bdd$#
ma_bdd$#  var3 integer := 30;
ma_bdd$#  var4 numeric(4);
ma_bdd$#
ma_bdd$# begin
ma_bdd$#
ma_bdd$#  SELECT COUNT(noemp) INTO var4 FROM EMP WHERE nodept=var3;
ma_bdd$#  raise notice '%', var4;
ma_bdd$#
ma_bdd$# end $$;
NOTICE:  6
DO
ma_bdd=#
```

3. Créer une table NOMBRE ayant les deux attributs de type NUMBER, A et B. — B est fonction de A
— Ecrire un bloc PL/SQL qui remplit cette table comme suit :

a	b
1	3
2	6
3	9
4	12
5	15
...	...
10	30

```
ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  varA integer := 1;
ma_bdd$#  varB integer := varA * 3;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  CREATE TABLE NOMBRE(
ma_bdd$#   A INTEGER,
ma_bdd$#   B INTEGER);
ma_bdd$#
ma_bdd$#  LOOP
ma_bdd$#   INSERT INTO NOMBRE VALUES (varA, varB);
ma_bdd$#   varA := varA + 1 ;
ma_bdd$#   varB := varA * 3;
ma_bdd$#   EXIT WHEN varA > 10;
ma_bdd$#  END LOOP;
ma_bdd$#
ma_bdd$# END $$;
DO
```

```
ma_bdd=# SELECT * FROM NOMBRE;
 a  | b
----+----
  1 |  3
  2 |  6
  3 |  9
  4 | 12
  5 | 15
  6 | 18
  7 | 21
  8 | 24
  9 | 27
 10 | 30
(10 rows)
```

4. Écrire un programme PL/SQL affichant le reste de la division de n par m sans utiliser la fonction modulo.
— Les nombres n et m sont deux entiers strictement positifs.
— Traiter le cas ou m vaut 0.

```
ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  n integer;
ma_bdd$#  m integer;
ma_bdd$#  s integer;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  n := 500;
ma_bdd$#  m := 7;
ma_bdd$#
ma_bdd$#  IF m < 0 OR n < 0 THEN
ma_bdd$#
ma_bdd$#   RAISE NOTICE E'Au moins un des deux entiers est négatif :\nm = %\nn = %', m, n;
ma_bdd$#
ma_bdd$#  ELSIF m != 0 THEN
ma_bdd$#
ma_bdd$#   s := n - m;
ma_bdd$#   WHILE s >= m LOOP
ma_bdd$#    s := s - m;
ma_bdd$#   END LOOP;
ma_bdd$#
ma_bdd$#   RAISE NOTICE 'Reste de la division : %', s ;
ma_bdd$#
ma_bdd$#  ELSE
ma_bdd$#
ma_bdd$#   RAISE NOTICE 'ON PEUT PAS DIVISER PAR 0 !';
ma_bdd$#
ma_bdd$#  END IF;
ma_bdd$#
ma_bdd$# END $$;
NOTICE:  Reste de la division : 3
DO
ma_bdd=#
```

```
ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  n integer;
ma_bdd$#  m integer;
ma_bdd$#  s integer;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  n := -500;
ma_bdd$#  m := -7;
ma_bdd$#
ma_bdd$#  IF m < 0 OR n < 0 THEN
ma_bdd$#
ma_bdd$#   RAISE NOTICE E'Au moins un des deux entiers est négatif :\nm = %\nn = %', m, n;
ma_bdd$#
ma_bdd$#  ELSIF m != 0 THEN
ma_bdd$#
ma_bdd$#   s := n - m;
ma_bdd$#   WHILE s >= m LOOP
ma_bdd$#    s := s - m;
ma_bdd$#   END LOOP;
ma_bdd$#
ma_bdd$#   RAISE NOTICE 'Reste de la division : %', s ;
ma_bdd$#
ma_bdd$#  ELSE
ma_bdd$#
ma_bdd$#   RAISE NOTICE 'ON PEUT PAS DIVISER PAR 0 !';
ma_bdd$#
ma_bdd$#  END IF;
ma_bdd$#
ma_bdd$# END $$;
NOTICE:  Au moins un des deux entiers est négatif :
m = -7
n = -500
DO
ma_bdd=#
```

```
DO
ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  n integer;
ma_bdd$#  m integer;
ma_bdd$#  s integer;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  n := 500;
ma_bdd$#  m := 0;
ma_bdd$#
ma_bdd$#  IF m < 0 OR n < 0 THEN
ma_bdd$#
ma_bdd$#   RAISE NOTICE E'Au moins un des deux entiers est négatif :\nm = %\nn = %', m, n;
ma_bdd$#
ma_bdd$#  ELSIF m != 0 THEN
ma_bdd$#
ma_bdd$#   s := n - m;
ma_bdd$#   WHILE s >= m LOOP
ma_bdd$#    s := s - m;
ma_bdd$#   END LOOP;
ma_bdd$#
ma_bdd$#   RAISE NOTICE 'Reste de la division : %', s ;
ma_bdd$#
ma_bdd$#  ELSE
ma_bdd$#
ma_bdd$#   RAISE NOTICE 'ON PEUT PAS DIVISER PAR 0 !';
ma_bdd$#
ma_bdd$#  END IF;
ma_bdd$#
ma_bdd$# END $$;
NOTICE:  ON PEUT PAS DIVISER PAR 0 !
DO
ma_bdd=#
```

5. On souhaite afficher pour un département dont le numéro est donné, le message : "Le département NOM se trouve à LOC".

```
ma_bdd=# SELECT * FROM DEPT;
 nodept |   nomdept    |  loc   | budget_in_k€
--------+--------------+--------+--------------
     10 | COMPTABILITE | BREST  |
     20 | RECHERCHE    | RENNES |
     30 | VENTES       | DINAR  |
     40 | GESTION      | DINAN  |
(4 rows)

ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  vnum numeric(3,0) := 30;
ma_bdd$#  vnom DEPT.nomDEPT%TYPE;
ma_bdd$#  vloc DEPT.LOC%TYPE;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  SELECT nomDEPT INTO vnom FROM DEPT WHERE noDEPT = vnum;
ma_bdd$#  SELECT LOC INTO vloc FROM DEPT WHERE noDEPT = vnum;
ma_bdd$#
ma_bdd$#  RAISE NOTICE 'Le département % se trouve à %', vnom, vloc;
ma_bdd$#
ma_bdd$# END $$;
NOTICE:  Le département VENTES se trouve à DINAR
DO
ma_bdd=#
```

Exercice 2 : Procédures et fonctions

1. Écrire une fonction PL/SQL qui calcule et affiche n !.
— n est un entier et doit être considéré comme paramètre de la fonction
— Proposer une solution itérative et une solution récursive.

```
ma_bdd=# CREATE OR REPLACE FUNCTION factorielle(n integer)
ma_bdd-#
ma_bdd-#  RETURNS integer
ma_bdd-#  LANGUAGE plpgsql
ma_bdd-#  as $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  vcompteur integer := 1;
ma_bdd$#  vtotal integer := 1;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  IF n != 0 OR n !=1 THEN
ma_bdd$#
ma_bdd$#   LOOP
ma_bdd$#    vtotal := vtotal * vcompteur;
ma_bdd$#    vcompteur := vcompteur + 1;
ma_bdd$#    EXIT WHEN vcompteur > n;
ma_bdd$#   END LOOP;
ma_bdd$#
ma_bdd$#  END IF;
ma_bdd$#
ma_bdd$#  RETURN vtotal;
ma_bdd$#
ma_bdd$# END $$;
CREATE FUNCTION
ma_bdd=# SELECT factorielle(5);
 factorielle
-------------
         120
(1 row)
```

```
ma_bdd=# CREATE OR REPLACE FUNCTION factorielle(n integer)
ma_bdd-#
ma_bdd-#  RETURNS integer
ma_bdd-#  LANGUAGE plpgsql
ma_bdd-#  as $$
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  IF n = 0 OR n = 1 THEN
ma_bdd$#
ma_bdd$#   RETURN 1;
ma_bdd$#
ma_bdd$#  ELSE
ma_bdd$#
ma_bdd$#   RETURN n * factorielle(n - 1);
ma_bdd$#
ma_bdd$#  END IF;
ma_bdd$#
ma_bdd$# END $$;
CREATE FUNCTION
ma_bdd=# SELECT factorielle(5);
 factorielle
-------------
         120
(1 row)

ma_bdd=#
```

2. Écrire un programme PL/SQL qui contient les éléments suivants :
— Une fonction récursive fFactR (n in number) calculant n !.
— Une procédure pFact (n in number) qui appelle la fonction fFactR.
— Un bloc PL/SQL anonyme qui appelle la procédure pFact.
— n doit être lu en dehors de fFactR et de pFact.

```
ma_bdd=# CREATE OR REPLACE FUNCTION fFactR(n integer)
ma_bdd-#
ma_bdd-#  RETURNS integer
ma_bdd-#  LANGUAGE plpgsql
ma_bdd-#  as $$
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  IF n = 0 OR n = 1 THEN
ma_bdd$#
ma_bdd$#  RETURN 1;
ma_bdd$#
ma_bdd$#  ELSE
ma_bdd$#
ma_bdd$#   RETURN n * factorielle(n - 1);
ma_bdd$#
ma_bdd$#  END IF;
ma_bdd$#
ma_bdd$# END $$;
CREATE FUNCTION
ma_bdd=#
```

```
ma_bdd=# CREATE OR REPLACE PROCEDURE pFact(n integer)
ma_bdd-#
ma_bdd-#  LANGUAGE plpgsql
ma_bdd-#  as $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  result integer := fFactR(n);
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  RAISE NOTICE '%', result;
ma_bdd$#
ma_bdd$# END $$;
CREATE PROCEDURE
ma_bdd=#
```

```
ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  n integer := 5;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  CALL pFact(n);
ma_bdd$#
ma_bdd$# END $$;
NOTICE:  120
DO
ma_bdd=#
```

3. Proposer une procédure affichant les nombres de 1 à n.
— n est un entier et doit être considéré comme paramètre de la procédure
— Proposer deux façons de tester le fonctionnement de cette procédure.

```
ma_bdd=# CREATE OR REPLACE PROCEDURE compteN(n integer)
 LANGUAGE plpgsql
 as $$

DECLARE

 cpt integer := 1;

BEGIN

 LOOP
  RAISE NOTICE '%', cpt;
  cpt := cpt + 1;
  EXIT WHEN cpt > n;
 END LOOP;

END $$;
CREATE PROCEDURE
ma_bdd=# CALL compteN(4);
NOTICE:  1
NOTICE:  2
NOTICE:  3
NOTICE:  4
CALL
ma_bdd=#
```

```
ma_bdd=# CREATE OR REPLACE PROCEDURE compteN(n integer)
ma_bdd-#
ma_bdd-#  LANGUAGE plpgsql
ma_bdd-#  as $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  cpt integer := n;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  LOOP
ma_bdd$#   RAISE NOTICE '%', cpt;
ma_bdd$#   cpt := cpt - 1;
ma_bdd$#   EXIT WHEN cpt = 0 ;
ma_bdd$#  END LOOP;
ma_bdd$#
ma_bdd$# END $$;
CREATE PROCEDURE
ma_bdd=# CALL compteN(4);
NOTICE:  4
NOTICE:  3
NOTICE:  2
NOTICE:  1
CALL
ma_bdd=#
```

4. Proposer une fonction qui calcule et renvoie $2n + m^2$.
— n et m sont des entiers et doivent être considérés comme paramètres de la fonction.
— Aucune autre variable ne doit être déclarée.
— Proposer deux façons de tester le fonctionnement de cette fonction.

```
ma_bdd=# CREATE OR REPLACE FUNCTION calcule(n int, m int)
ma_bdd-#
ma_bdd-#  RETURNS integer
ma_bdd-#  LANGUAGE plpgsql
ma_bdd-#  as $$
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  RETURN 2 * n + m * m;
ma_bdd$#
ma_bdd$# END $$;
CREATE FUNCTION
ma_bdd=# SELECT calcule(2,3);
 calcule
---------
      13
(1 row)

ma_bdd=#
```

5. Écrire un programme PL/SQL qui contient les éléments suivants :
— Une procédure pAug(nEmp number, tAug number) augmentant de tAug% le salaire de l'employé identifié par nEmp.
— Un bloc PL/SQL anonyme qui appelle la procédure pAug.
— nEmp et tAug doivent être lus en dehors de pAug.

```
ma_bdd=# SELECT noemp, nomemp, sal FROM EMP WHERE noemp = 7369;
 noemp | nomemp |  sal
-------+--------+--------
  7369 | SERGE  | 800.00
(1 row)

ma_bdd=# CREATE OR REPLACE PROCEDURE pAug(nEmp EMP.noEMP%TYPE, tAug numeric(3,2))
ma_bdd-#
ma_bdd-#  LANGUAGE plpgsql
ma_bdd-#  AS $$
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  UPDATE EMP SET sal = sal + sal * tAug/100 WHERE  noemp = nEMP;
ma_bdd$#
ma_bdd$# END $$;
NOTICE:  type reference emp.noemp%TYPE converted to numeric
CREATE PROCEDURE
ma_bdd=# DO $$
ma_bdd$# BEGIN
ma_bdd$#  CALL pAug(7369, 10.5);
ma_bdd$# END $$;
DO
ma_bdd=# SELECT noemp, nomemp, sal FROM EMP WHERE noemp = 7369;
 noemp | nomemp |  sal
-------+--------+--------
  7369 | SERGE  | 884.00
(1 row)

ma_bdd=#
```

Exercice 3 : Utilisation de curseurs

1. Écrire un bloc PL/SQL affichant le nombre de départements dont aucun employé ne touche une commission. Un département possédant un ou plusieurs employés ayant une commission >0 ne doit pas être considéré.

```
ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  curseur CURSOR FOR SELECT DISTINCT comm, nodept FROM EMP ORDER BY nodept; --ordre important
ma_bdd$#
ma_bdd$#  comm_curseur EMP.comm%TYPE;
ma_bdd$#  dpt_curseur EMP.noDEPT%TYPE;
ma_bdd$#
ma_bdd$#  dpt_éliminé EMP.noDEPT%TYPE; --stock le dpt éliminé
ma_bdd$#  cpt_ini int;
ma_bdd$#  cpt int; --compteur
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  --SELECT DISTINCT count(nodept) INTO cpt FROM DEPT; --ini du compteur
ma_bdd$#  SELECT COUNT(*) INTO cpt FROM (SELECT DISTINCT nodept FROM EMP) AS SUB;
ma_bdd$#
ma_bdd$#  cpt_ini := cpt;
ma_bdd$#
ma_bdd$#  OPEN curseur;
ma_bdd$#
ma_bdd$#   LOOP
ma_bdd$#
ma_bdd$#    FETCH curseur INTO comm_curseur, dpt_curseur;
ma_bdd$#
ma_bdd$#     IF comm_curseur > 0 THEN
ma_bdd$#
ma_bdd$#      IF dpt_éliminé <> dpt_curseur OR dpt_éliminé IS NULL THEN
ma_bdd$#
ma_bdd$#       dpt_éliminé := dpt_curseur ;
ma_bdd$#
ma_bdd$#       cpt := cpt - 1;
ma_bdd$#
ma_bdd$#       RAISE NOTICE 'Au moins un employé touche une commission dans le département %.', dpt_curseur;
ma_bdd$#
ma_bdd$#      END IF;
ma_bdd$#
ma_bdd$#      --RAISE NOTICE '% % % %', comm_curseur, dpt_curseur, cpt, dpt_éliminé;
ma_bdd$#
ma_bdd$#     END IF;
ma_bdd$#
ma_bdd$#    EXIT WHEN NOT FOUND;
ma_bdd$#
ma_bdd$#   END LOOP;
ma_bdd$#
ma_bdd$#  CLOSE curseur;
ma_bdd$#
ma_bdd$#  RAISE NOTICE 'Aucun employé ne touche de commission dans % départements sur %.', cpt, cpt_ini;
ma_bdd$#
ma_bdd$# END $$;
NOTICE:  Au moins un employé touche une commission dans le département 30.
NOTICE:  Aucun employé ne touche de commission dans 2 départements sur 3.
DO
ma_bdd=#
```

2. Écrire un bloc PL/SQL affichant le nombre de départements dont tous les employés touchent un salaire supérieur à 2000e.

```
ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  curseur CURSOR FOR SELECT sal, nodept FROM EMP ORDER BY nodept; --ordre important
ma_bdd$#
ma_bdd$#  sal_curseur EMP.sal%TYPE;
ma_bdd$#  dpt_curseur EMP.noDEPT%TYPE;
ma_bdd$#
ma_bdd$#  dpt_éliminé EMP.noDEPT%TYPE; --stock le dpt éliminé
ma_bdd$#  cpt_ini int;
ma_bdd$#  cpt int; --compteur
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  --SELECT DISTINCT count(nodept) INTO cpt FROM DEPT; --ini du compteur
ma_bdd$#  SELECT COUNT(*) INTO cpt FROM (SELECT DISTINCT nodept FROM EMP) AS SUB;
ma_bdd$#
ma_bdd$#  cpt_ini := cpt;
ma_bdd$#
ma_bdd$#  OPEN curseur;
ma_bdd$#
ma_bdd$#   LOOP
ma_bdd$#
ma_bdd$#    FETCH curseur INTO sal_curseur, dpt_curseur;
ma_bdd$#
ma_bdd$#     IF sal_curseur <= 2000 THEN
ma_bdd$#
ma_bdd$#      IF dpt_éliminé <> dpt_curseur OR dpt_éliminé IS NULL THEN
ma_bdd$#
ma_bdd$#       dpt_éliminé := dpt_curseur ;
ma_bdd$#
ma_bdd$#       cpt := cpt - 1;
ma_bdd$#
ma_bdd$#       RAISE NOTICE 'Au moins un employé touche un salaire inférieur ou égal à 2000 dans le département %.', dpt_curseur;
ma_bdd$#
ma_bdd$#      END IF;
ma_bdd$#
ma_bdd$#      --RAISE NOTICE '% % % %', comm_curseur, dpt_curseur, cpt, dpt_éliminé;
ma_bdd$#
ma_bdd$#     END IF;
ma_bdd$#
ma_bdd$#    EXIT WHEN NOT FOUND;
ma_bdd$#
ma_bdd$#   END LOOP;
ma_bdd$#
ma_bdd$#  CLOSE curseur;
ma_bdd$#
ma_bdd$#  RAISE NOTICE 'Tous les employés touchent un salaire supérieur à 2000 dans % départements sur %.', cpt, cpt_ini;
ma_bdd$#
ma_bdd$# END $$;
NOTICE:  Au moins un employé touche un salaire inférieur ou égal à 2000 dans le département 10.
NOTICE:  Au moins un employé touche un salaire inférieur ou égal à 2000 dans le département 20.
NOTICE:  Au moins un employé touche un salaire inférieur ou égal à 2000 dans le département 30.
NOTICE:  Tous les employés touchent un salaire supérieur à 2000 dans 0 départements sur 3.
DO
```

3. Proposer un bloc PL/SQL qui affiche les noms des départements qui existent dans les deux tables EMP et DEPT.

```
ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  curseur1 CURSOR FOR SELECT DISTINCT nomdept FROM DEPT, EMP WHERE EMP.nodept = DEPT.nodept;
ma_bdd$#  curseur2 CURSOR FOR SELECT nomdept FROM DEPT;
ma_bdd$#
ma_bdd$#  nomdept_curseur DEPT.nomdept%TYPE;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  RAISE NOTICE 'La table EMP compte les départements suivants :';
ma_bdd$#
ma_bdd$#  OPEN curseur1;
ma_bdd$#
ma_bdd$#   LOOP
ma_bdd$#
ma_bdd$#    FETCH curseur1 INTO nomdept_curseur;
ma_bdd$#
ma_bdd$#    EXIT WHEN NOT FOUND;
ma_bdd$#
ma_bdd$#    RAISE NOTICE '%', nomdept_curseur;
ma_bdd$#
ma_bdd$#   END LOOP;
ma_bdd$#
ma_bdd$#  CLOSE curseur1;
ma_bdd$#
ma_bdd$#  RAISE NOTICE 'La table DEPT compte les départements suivants :';
ma_bdd$#
ma_bdd$#  OPEN curseur2;
ma_bdd$#
ma_bdd$#   LOOP
ma_bdd$#
ma_bdd$#    FETCH curseur2 INTO nomdept_curseur;
ma_bdd$#
ma_bdd$#    EXIT WHEN NOT FOUND;
ma_bdd$#
ma_bdd$#    RAISE NOTICE '%', nomdept_curseur;
ma_bdd$#
ma_bdd$#   END LOOP;
ma_bdd$#
ma_bdd$#  CLOSE curseur2;
ma_bdd$#
ma_bdd$# END $$;
NOTICE:  La table EMP compte les départements suivants :
NOTICE:  RECHERCHE
NOTICE:  COMPTABILITE
NOTICE:  VENTES
NOTICE:  La table DEPT compte les départements suivants :
NOTICE:  COMPTABILITE
NOTICE:  RECHERCHE
NOTICE:  VENTES
NOTICE:  GESTION
DO
ma_bdd=#
```

4. Proposer un bloc PL/SQL qui affiche le résultat suivant : NDEPT NOMEMP MS1 MS2
Avec :
— NDEPT : nom du département
— NOMBEMP : nombre d'employés
— MS1 est la moyenne des salaires par département. Seuls les salaires inférieurs ou égaux à 2500e sont à considérer.
— MS2 est la moyenne des salaires par département. Seuls les salaires supérieurs à 2500e sont à considérer.
— Seuls les employés n'ayant pas de commission sont à considérer.

```
ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  curseur CURSOR FOR SELECT DEPT.nomdept, EMP.nodept, sal, sal, comm
ma_bdd$#         FROM EMP, DEPT
ma_bdd$#         WHERE EMP.nodept = DEPT.nodept
ma_bdd$#         ORDER BY EMP.nodept;
ma_bdd$#
ma_bdd$#  ndept DEPT.nomdept%TYPE;
ma_bdd$#  numdpt EMP.nodept%TYPE;
ma_bdd$#  nombemp int := 0;
ma_bdd$#  ms1 EMP.sal%TYPE;
ma_bdd$#  ms2 EMP.sal%TYPE;
ma_bdd$#
ma_bdd$#
ma_bdd$#  vcomm EMP.comm%TYPE;
ma_bdd$#  ndept_actu DEPT.nomdept%TYPE;
ma_bdd$#  sal1 EMP.sal%TYPE := 0;
ma_bdd$#  sal2 EMP.sal%TYPE := 0;
ma_bdd$#  cpt1 int;
ma_bdd$#  cpt2 int;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  OPEN curseur;
ma_bdd$#
ma_bdd$#  RAISE NOTICE '     NDEPT     |    NOMBEMP    |      MS1      |      MS2      '; --par 15
ma_bdd$#  RAISE NOTICE '---------------+---------------+---------------+---------------';
ma_bdd$#
ma_bdd$#  LOOP
ma_bdd$#
ma_bdd$#   FETCH curseur INTO ndept, numdpt, sal1, sal2, vcomm;
ma_bdd$#
ma_bdd$#   IF vcomm IS NULL OR vcomm = 0 THEN --si la comm est inexistante ou nulle ...
ma_bdd$#
ma_bdd$#    IF ndept = ndept_actu THEN -- tant qu'on est au même numéro de dpt alors ...
ma_bdd$#
ma_bdd$#     nombemp := nombemp + 1; -- on compte le nombre d'employés
ma_bdd$#
ma_bdd$#     IF sal1 <= 2500 THEN -- si <= 2500
ma_bdd$#
ma_bdd$#      cpt1 := cpt1 + 1;
ma_bdd$#      ms1 := (ms1 + sal1)/cpt1;--  on fait la moyenne 1
ma_bdd$#
ma_bdd$#     ELSE -- si >2500
ma_bdd$#
ma_bdd$#      cpt2 := cpt2 + 1;
ma_bdd$#      ms2 := (ms2 + sal2)/cpt2;--  on fait la moyenne 2
ma_bdd$#
ma_bdd$#     END IF;
```

```
ma_bdd$#    ELSE -- dès que le numéro de dpt change...
ma_bdd$#
ma_bdd$#     IF ndept_actu IS NULL THEN
ma_bdd$#
ma_bdd$#      ndept_actu := ndept; -- initialise ndept_actu
ma_bdd$#
ma_bdd$#     ELSE
ma_bdd$#
ma_bdd$#      RAISE NOTICE '    %    |    %    |    %    |    %', ndept_actu, nombemp, ms1, ms2;
ma_bdd$#
ma_bdd$#      ndept_actu := ndept; -- (ré)initialise ndept_actu
ma_bdd$#
ma_bdd$#     END IF;
ma_bdd$#
ma_bdd$#     nombemp := 1; -- réini nb d'employés
ma_bdd$#     cpt1 := 0;
ma_bdd$#     cpt2 := 0;
ma_bdd$#
ma_bdd$#     IF sal1 <= 2500 THEN -- si <= 2500, réini
ma_bdd$#
ma_bdd$#      ms1 := sal1;
ma_bdd$#      ms2 := 0;
ma_bdd$#
ma_bdd$#     ELSE  -- si >2500, réini
ma_bdd$#
ma_bdd$#      ms1 := 0;
ma_bdd$#      ms2 := sal2;
ma_bdd$#
ma_bdd$#     END IF;
ma_bdd$#
ma_bdd$#    END IF;
ma_bdd$#
ma_bdd$#   END IF;
ma_bdd$#
ma_bdd$#  EXIT WHEN NOT FOUND;
ma_bdd$#
ma_bdd$#  END LOOP;
ma_bdd$#
ma_bdd$#  CLOSE curseur;
ma_bdd$#
ma_bdd$# END $$;
NOTICE:       NDEPT     |    NOMBEMP    |      MS1      |      MS2
NOTICE:  ---------------+---------------+---------------+---------------
NOTICE:       COMPTABILITE    |    3    |    1875.00    |    5000.00
NOTICE:       RECHERCHE    |    5    |    1984.00    |    1995.83
NOTICE:       VENTES    |    3    |    2450.00    |    2850.00
DO
ma_bdd=#
```

5. Proposer un bloc PL/SQL affichant les noms des employés du département "VENTES" dont le salaire > 1400 e.

```
ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  curseur CURSOR FOR SELECT DEPT.nomdept, nomemp, sal
ma_bdd$#         FROM EMP, DEPT
ma_bdd$#         WHERE EMP.nodept = DEPT.nodept
ma_bdd$#         ORDER BY EMP.nodept;
ma_bdd$#
ma_bdd$#  vnomdept DEPT.nomdept%TYPE;
ma_bdd$#  vnomemp EMP.nomemp%TYPE;
ma_bdd$#  vsal EMP.sal%TYPE;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  OPEN curseur;
ma_bdd$#
ma_bdd$#  LOOP
ma_bdd$#
ma_bdd$#   FETCH curseur INTO vnomdept, vnomemp, vsal;
ma_bdd$#
ma_bdd$#   IF vnomdept ='VENTES' AND vsal > 1400 THEN
ma_bdd$#
ma_bdd$#    RAISE NOTICE '%', vnomemp;
ma_bdd$#
ma_bdd$#   END IF;
ma_bdd$#
ma_bdd$#  EXIT WHEN NOT FOUND;
ma_bdd$#
ma_bdd$#  END LOOP;
ma_bdd$#
ma_bdd$#  CLOSE curseur;
ma_bdd$#
ma_bdd$# END $$;
NOTICE:  ROI
NOTICE:  BENJAMIN
NOTICE:  BRAHIM
DO
ma_bdd=#
```

6. Proposer un bloc PL/SQL permettant de recopier le contenu de la table EMP avec augmentation de la commission de 50% dans la table EMPBIS possédant le même schéma que EMP. On suppose que la table EMPBIS existe et est vide.

```
ma_bdd=# CREATE TABLE EMPBIS(
ma_bdd(# NoEMP NUMERIC(4) NOT NULL,
ma_bdd(# NomEMP VARCHAR(10),
ma_bdd(# EMPLOI VARCHAR(15),
ma_bdd(# MGR NUMERIC(4),
ma_bdd(# DateEMB DATE,
ma_bdd(# SAL NUMERIC(7,2),
ma_bdd(# COMM NUMERIC(7,2),
ma_bdd(# NoDEPT NUMERIC(2),
ma_bdd(# CONSTRAINT EMPBIS_clé_primaire PRIMARY KEY (NoEMP));
CREATE TABLE
ma_bdd=# \d EMPBIS;
                    Table "public.empbis"
 Column  |         Type          | Collation | Nullable | Default
---------+-----------------------+-----------+----------+---------
 noemp   | numeric(4,0)          |           | not null |
 nomemp  | character varying(10) |           |          |
 emploi  | character varying(15) |           |          |
 mgr     | numeric(4,0)          |           |          |
 dateemb | date                  |           |          |
 sal     | numeric(7,2)          |           |          |
 comm    | numeric(7,2)          |           |          |
 nodept  | numeric(2,0)          |           |          |
Indexes:
    "empbis_clé_primaire" PRIMARY KEY, btree (noemp)

ma_bdd=# SELECT * FROM EMPBIS;
 noemp | nomemp | emploi | mgr | dateemb | sal | comm | nodept
-------+--------+--------+-----+---------+-----+------+--------
(0 rows)
```

```
ma_bdd=# SELECT * FROM EMPBIS;
 noemp | nomemp | emploi | mgr | dateemb | sal | comm | nodept
-------+--------+--------+-----+---------+-----+------+--------
(0 rows)

ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  curseur CURSOR FOR SELECT * FROM EMP;
ma_bdd$#
ma_bdd$#  vnoemp EMP.noemp%TYPE;
ma_bdd$#  vnomemp EMP.nomemp%TYPE;
ma_bdd$#  vemploi EMP.emploi%TYPE;
ma_bdd$#  vmgr EMP.mgr%TYPE;
ma_bdd$#  vdateemb EMP.dateemb%TYPE;
ma_bdd$#  vsal EMP.sal%TYPE;
ma_bdd$#  vcomm EMP.comm%TYPE;
ma_bdd$#  vnodept EMP.nodept%TYPE;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  OPEN curseur;
ma_bdd$#
ma_bdd$#  LOOP
ma_bdd$#
ma_bdd$#   FETCH curseur INTO vnoemp, vnomemp, vemploi, vmgr, vdateemb, vsal, vcomm, vnodept;
ma_bdd$#
ma_bdd$#   EXIT WHEN NOT FOUND;
ma_bdd$#
ma_bdd$#   INSERT INTO EMPBIS VALUES (vnoemp, vnomemp, vemploi, vmgr, vdateemb, vsal, (vcomm * 1.5), vnodept);
ma_bdd$#
ma_bdd$#  END LOOP;
ma_bdd$#
ma_bdd$#  CLOSE curseur;
ma_bdd$#
ma_bdd$# END $$;
DO
ma_bdd=# SELECT * FROM EMPBIS;
 noemp |  nomemp  |    emploi     | mgr  |  dateemb   |   sal   |  comm   | nodept
-------+----------+---------------+------+------------+---------+---------+--------
  7499 | BRAHIM   | VENDEUR       | 7698 | 1981-02-20 | 1600.00 |  450.00 |     30
  7521 | NASSIMA  | VENDEUR       | 7698 | 1981-02-22 | 1250.00 |  750.00 |     30
  7566 | LUCIE    | GESTIONNAIRE  | 7839 | 1981-04-02 | 2975.00 |         |     20
  7654 | MARTIN   | VENDEUR       | 7698 | 1981-09-28 | 1250.00 | 2100.00 |     30
  7698 | BENJAMIN | GESTIONNAIRE  | 7839 | 1981-05-01 | 2850.00 |         |     30
  7782 | DAYANE   | GESTIONNAIRE  | 7839 | 1981-06-09 | 2450.00 |         |     10
  7788 | ARIJ     | ANALYSTE      | 7566 | 1982-12-09 | 3000.00 |         |     20
  7839 | MAYAR    | PRESIDENT     |      | 1981-11-17 | 5000.00 |         |     10
  7844 | ROI      | VENDEUR       | 7698 | 1981-09-08 | 1500.00 |    0.00 |     30
  7876 | VIRGINIE | FONCTIONNAIRE | 7788 | 1983-01-12 | 1100.00 |         |     20
  7900 | LYNA     | FONCTIONNAIRE | 7698 | 1981-12-03 |  950.00 |         |     30
  7902 | ASMA     | ANALYSTE      | 7566 | 1981-12-03 | 3000.00 |         |     20
  7934 | SIMONE   | FONCTIONNAIRE | 7782 | 1982-01-23 | 1300.00 |         |     10
  7369 | SERGE    | FONCTIONNAIRE | 7902 | 1980-12-17 |  884.00 |         |     20
(14 rows)
```

7. Proposer un bloc PL/SQL qui vérifie à partir de son nom donné, l'existence d'un département (par son numéro) dans la table EMP.

```
ma_bdd=# DO $$
ma_bdd$#
ma_bdd$# DECLARE
ma_bdd$#
ma_bdd$#  curseur CURSOR FOR SELECT nodept, nomdept FROM DEPT;
ma_bdd$#
ma_bdd$#  vnodept DEPT.nodept%TYPE;
ma_bdd$#  vnomdept DEPT.nomdept%TYPE;
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  OPEN curseur;
ma_bdd$#
ma_bdd$#  LOOP
ma_bdd$#
ma_bdd$#   FETCH curseur INTO vnodept, vnomdept;
ma_bdd$#
ma_bdd$#   EXIT WHEN NOT FOUND;
ma_bdd$#
ma_bdd$#   IF vnodept IN (SELECT DISTINCT nodept FROM EMP) THEN
ma_bdd$#
ma_bdd$#    RAISE NOTICE 'Le département % % existe dans EMP.',vnodept, vnomdept;
ma_bdd$#
ma_bdd$#   ELSE
ma_bdd$#
ma_bdd$#    RAISE NOTICE 'Le département % % n''existe pas dans EMP.',vnodept, vnomdept;
ma_bdd$#
ma_bdd$#   END IF;
ma_bdd$#
ma_bdd$#  END LOOP;
ma_bdd$#
ma_bdd$#  CLOSE curseur;
ma_bdd$#
ma_bdd$# END $$;
NOTICE:  Le département 10 COMPTABILITE existe dans EMP.
NOTICE:  Le département 20 RECHERCHE existe dans EMP.
NOTICE:  Le département 30 VENTES existe dans EMP.
NOTICE:  Le département 40 GESTION n'existe pas dans EMP.
DO
```

Exercice 4 : Déclencheurs/Triggers

1. On souhaite interdire toute opération n'ayant pas pour objet la simple consultation de la table DEPT après 17H00. Proposer un déclencheur assurant cette tâche.

```
ma_bdd=# CREATE OR REPLACE FUNCTION log_no_change_after_5pm()
ma_bdd-#
ma_bdd-#  RETURNS TRIGGER LANGUAGE PLPGSQL AS $$
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  IF DATE_PART('hour', now()) >= 17 THEN
ma_bdd$#
ma_bdd$#   --RAISE NOTICE 'ÉCHEC';
ma_bdd$#   RAISE EXCEPTION E'Tentative à % h %.\nDéclencheur = %\nTable ciblée = %\nopération = %',
ma_bdd$#   DATE_PART('hour', now()), DATE_PART('minute', now()), TG_NAME, TG_TABLE_NAME, TG_OP
ma_bdd$#      USING DETAIL = 'Mise à jour interdite après 17h.'; --E important
ma_bdd$#
ma_bdd$#  ELSE
ma_bdd$#
ma_bdd$#   RETURN NEW;
ma_bdd$#
ma_bdd$#  END IF;
ma_bdd$#
ma_bdd$# END $$;
CREATE FUNCTION
```

```
ma_bdd=# CREATE TRIGGER no_change_after_5pm
ma_bdd-#
ma_bdd-# BEFORE INSERT OR UPDATE OR DELETE
ma_bdd-#  ON EMP
ma_bdd-# FOR EACH STATEMENT
ma_bdd-# EXECUTE PROCEDURE log_no_change_after_5pm();
CREATE TRIGGER
ma_bdd=#
```

```
ma_bdd=# UPDATE EMP SET sal = 800 WHERE nomemp = 'SERGE';
ERROR:  Tentative à 18 h 15.
Déclencheur = no_change_after_5pm
Table ciblée = emp
opération = UPDATE
DETAIL:  Mise à jour interdite après 17h.
CONTEXT:  PL/pgSQL function log_no_change_after_5pm() line 8 at RAISE
ma_bdd=#
```

2. On souhaite interdire de manière automatique toute insertion d'un salaire < 500e. Proposer un déclencheur assurant cette tâche.

```
ma_bdd=# CREATE OR REPLACE FUNCTION log_no_sal_under_500e()
ma_bdd-#
ma_bdd-#  RETURNS TRIGGER LANGUAGE PLPGSQL AS $$
ma_bdd$#
ma_bdd$# BEGIN
ma_bdd$#
ma_bdd$#  IF NEW.sal < 500 THEN
ma_bdd$#
ma_bdd$#   RAISE EXCEPTION E'Tentative à % h %.\nDéclencheur = %\nTable ciblée = %\nopération = %',
ma_bdd$#   DATE_PART('hour', now()), DATE_PART('minute', now()), TG_NAME, TG_TABLE_NAME, TG_OP
ma_bdd$#      USING DETAIL = 'Insertion d''un salaire inférieur à 500 interdite.';
ma_bdd$#
ma_bdd$#  ELSE
ma_bdd$#
ma_bdd$#   RAISE NOTICE E'%\n', NEW.sal;
ma_bdd$#   RETURN NEW;
ma_bdd$#
ma_bdd$#  END IF;
ma_bdd$#
ma_bdd$# END $$;
CREATE FUNCTION
ma_bdd=#
```

```
ma_bdd=# CREATE TRIGGER no_sal_under_500e
ma_bdd-#
ma_bdd-# AFTER INSERT ON EMP
ma_bdd-# FOR EACH ROW
ma_bdd-# EXECUTE PROCEDURE log_no_sal_under_500e();
CREATE TRIGGER
ma_bdd=#
```

```
ma_bdd=# INSERT INTO EMP VALUES (7953, 'JIMMY', 'VENDEUR', 7839, '20-MAY-2015', 400, NULL, 10);
ERROR:  Tentative à 15 h 48.
Déclencheur = no_sal_under_500e
Table ciblée = emp
opération = INSERT
DETAIL:  Insertion d'un salaire inférieur à 500 interdite.
CONTEXT:  PL/pgSQL function log_no_sal_under_500e() line 7 at RAISE
ma_bdd=#
```