

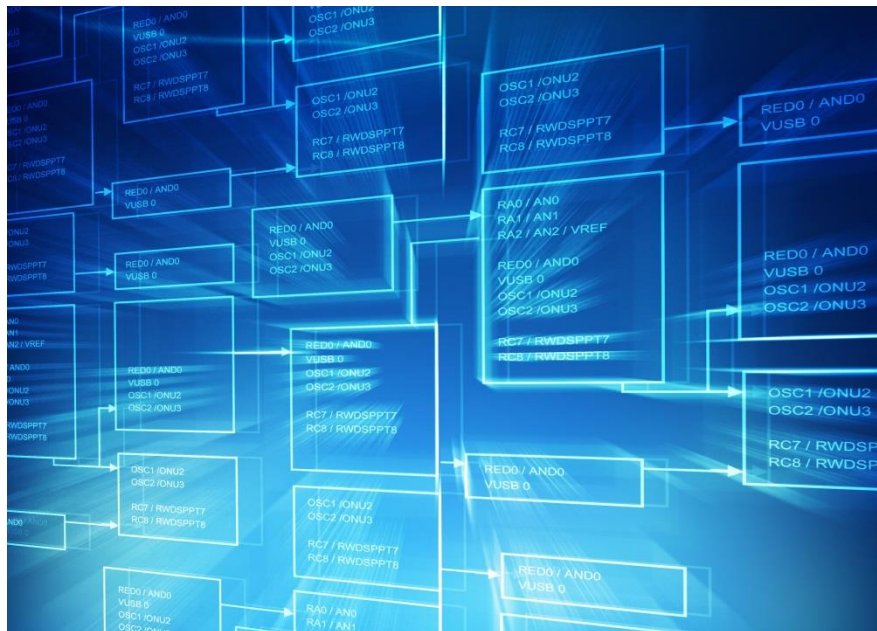
## Département Informatique

MASTER EN SCIENCES ET TECHNIQUES



SYSTÈMES D'INFORMATION DÉCISIONNELS ET IMAGERIE

FACULTÉ DES SCIENCES ET TECHNIQUES ERRACHIDIA



# Bases de données réparties

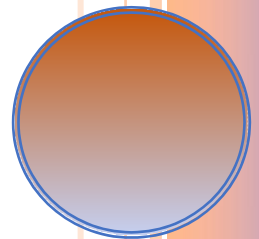
## TP4 - Procédures, Fonctions et Triggers

**ZEKKOURI Hassan**

**Vendredi 31 Decembre 2019**

**Responsable du module :**

**Prof. BAATAOUI**



## REMERCIEMENT

---

Nous tenons à vous remercier monsieur **BAATAOUI** pour votre formation et vos services.

Nous sommes également reconnaissant de nous avoir donné l'occasion de s'ouvrir sur un nouvel aspect de technologie qui est le **PL/SQL** et y mettre en œuvre nos compétences dont nous avons obtenu au cours du module.

Hassan ZEKKOURI

---

# PLAN

## Contents

<i>REMERCIEMENT</i> .....	2
INTRODUCTION .....	4
I. EXERCICE 1.....	5
II. EXERCICE 2.....	10
III. EXERCICE 3.....	19
IV. EXERCICE 4.....	27
FIN.....	48

# INTRODUCTION

## ➤ Objectif :

Dans ce TP nous allons découvrir la programmation avec **PL/SQL** qui est une extension de langage **SQL** permettant d'interroger les bases de données efficacement !

En particulier, pendant ce TP nous allons découvrir les Procédures, Fonctions et Triggers et leurs manipulations !

# I. EXERCICE 1

Création de la base de teste :

```

-- creatio des tables!
CREATE TABLE PILOTE (
    nopilote INTEGER PRIMARY KEY,
    nompil VARCHAR(20),
    adresse VARCHAR(20),
    salaire INTEGER,
    prime INTEGER,
    embauche date
);

INSERT INTO pilote VALUES (1, 'hassan', 'mcissi, alnif', 1000, 200, '13/02/12');
INSERT INTO pilote VALUES (2, 'ALI', 'AZAG, alnif', 1000, 200, '13/02/13');
INSERT INTO pilote VALUES (3, 'haMZA', 'TINGHIR, R565', 1000, 200, '13/02/13');
  
```

1. Le package pack\_pilote comporte les procédures et fonctions publiques suivantes :

```

1 CREATE OR REPLACE PACKAGE pack_pilote AS
2
3     PROCEDURE inser_pil(p_nom IN pilote.nompil%TYPE, p_adresse IN pilote.adresse%TYPE, p_dateEmbauche IN pilote.embauche%TYPE);
4     PROCEDURE sel_pil(p_nom IN pilote.nompil%TYPE, p_adresse OUT pilote.adresse%TYPE, p_salaire OUT pilote.salaire%TYPE);
5     FUNCTION sel_pil(p_nom IN pilote.nompil%TYPE) RETURN pilote.adresse%TYPE;
6     FUNCTION prime_pil(p_nom IN pilote.nompil%TYPE) RETURN pilote.prime%TYPE;
7     PROCEDURE list_pil;
8     PROCEDURE sal_prim_pil;
9
10 END pack_pilote;
11 /
  
```

- a. Inser\_pil (nom,adresse, dateembauche)** qui permet l'insertion d'un nouveau pilote dans la table avec numéro automatique, salaire égal à la moyenne du salaire des pilotes et commission à zéro.

```
-- La procedure qui insere une ligne
PROCEDURE inser_pil(p_nom IN pilote.nompil%TYPE, p_adresse IN pilote.adresse%TYPE, p_dateEmbauche IN pilote.embauche%TYPE) IS
    sal_moy INTEGER;
    N INTEGER;
BEGIN
    -- calcul de salaire moyenne
    sal_moy := sal_moyen_pil();
    -- calcul de nombre d'enregistrement
    SELECT COUNT(*) INTO N
    FROM pilote;
    -- insertion
    INSERT INTO pilote VALUES (N+1, p_nom, p_adresse, sal_moy, 0, p_dateEmbauche);
END inser_pil;
```

- b. Sel\_pil (nom,adresse,salaire)** permettant de récupérer l'adresse et le salaire d'un pilote dont on fournit le nom.

```
-- La procedure qui selection l'adresse et le salaire d'un employer
PROCEDURE sel_pil(p_nom IN pilote.nompil%TYPE, p_adresse OUT pilote.adresse%TYPE, p_salaire OUT pilote.salaire%TYPE) IS
BEGIN
    SELECT adresse, salaire INTO p_adresse, p_salaire
    FROM pilote
    WHERE nompil = p_nom;
END sel_pil;
```

- c. Surcharger la méthode Sel\_pil** pour n'obtenir que l'adresse d'un pilote à partir du nom.

```
-- La fonction qui retourne l'adresse d'un employer
FUNCTION sel_pil(p_nom IN pilote.nompil%TYPE) RETURN pilote.adresse%TYPE IS
    v_adresse pilote.adresse%TYPE;
BEGIN
    SELECT adresse INTO v_adresse
    FROM pilote
    WHERE nompil = p_nom;
    RETURN v_adresse;
END sel_pil;
```

- d. **Prime\_pil(nom,prime)** permettant d'attribuer une prime à un pilote.

```
PROCEDURE prime_pil(p_nom IN pilote.nompil%TYPE, p_prime pilote.prime%TYPE) IS
BEGIN
    UPDATE pilote
    SET prime = p_prime
    WHERE nompil = p_nom;
END prime_pil;
```

- e. **List\_pil ()** permettant d'afficher la liste de tous les pilotes avec adresse, salaire et prime. **Sal\_prim\_pil()** permettant d'obtenir le total des salaires avec les primes.

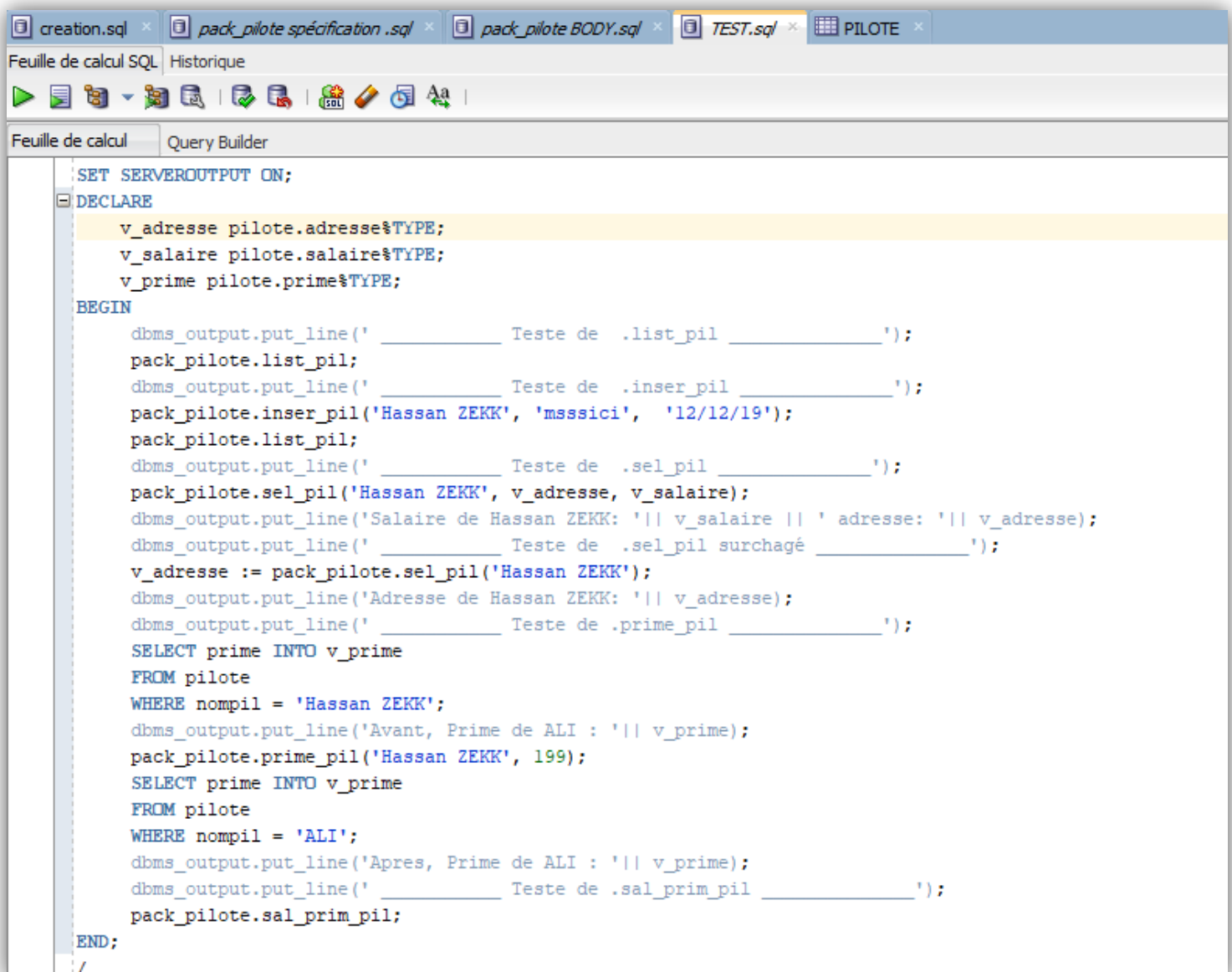
```
-- La procedure qui affiche les pilotes
PROCEDURE list_pil IS
    CURSOR list_pilote IS
        SELECT nompil, adresse, salaire, prime
        FROM pilote;
BEGIN
    dbms_output.put_line('NOM      | ADRESSE      | SALAIRE      | PRIME');
    -- open list_pilote;
    FOR ligne IN list_pilote LOOP
        dbms_output.put_line(ligne.nompil || ' ' || ligne.adresse || ' ' || ligne.salaire || ' ' || ligne.prime);
    END LOOP;
    -- close list_pilote;
END list_pil;
```

```
-- La procedure qui affiche les pilotes (salaire total)
PROCEDURE sal_prim_pil IS
    CURSOR list_pilote IS
        SELECT nompil, adresse, salaire+prime as total_sal
        FROM pilote;
BEGIN
    dbms_output.put_line('NOM      | ADRESSE      | SALAIRE TOTAL');
    -- open list_pilote;
    FOR ligne IN list_pilote LOOP
        dbms_output.put_line(ligne.nompil || ' ' || ligne.adresse || ' ' || ligne.total_sal);
    END LOOP;
    -- close list_pilote;
END sal_prim_pil;
```

## f. Transformer le calcul du salaire moyen des pilotes en une fonction privée.

```
CREATE OR REPLACE PACKAGE BODY pack_pilote AS
    -- La fonction qui retourne le salaire moyenne (privée)
    FUNCTION sal_moyen_pil RETURN INTEGER IS
        sal_moy INTEGER;
    BEGIN
        SELECT AVG(salaire) INTO sal_moy
        FROM pilote;
        RETURN sal_moy;
    END sal_moyen_pil;
```

## 2. Ecrire un bloc PL/SQL pour tester et exécuter les méthodes du package



```
SET SERVEROUTPUT ON;
DECLARE
    v_adresse pilote.adresse%TYPE;
    v_salaire pilote.salaire%TYPE;
    v_prime pilote.prime%TYPE;
BEGIN
    dbms_output.put_line(' _____ Teste de .list_pil _____');
    pack_pilote.list_pil;
    dbms_output.put_line(' _____ Teste de .inser_pil _____');
    pack_pilote.inser_pil('Hassan ZEKK', 'mssici', '12/12/19');
    pack_pilote.list_pil;
    dbms_output.put_line(' _____ Teste de .sel_pil _____');
    pack_pilote.sel_pil('Hassan ZEKK', v_adresse, v_salaire);
    dbms_output.put_line('Salaire de Hassan ZEKK: ' || v_salaire || ' adresse: ' || v_adresse);
    dbms_output.put_line(' _____ Teste de .sel_pil surchargé _____');
    v_adresse := pack_pilote.sel_pil('Hassan ZEKK');
    dbms_output.put_line('Adresse de Hassan ZEKK: ' || v_adresse);
    dbms_output.put_line(' _____ Teste de .prime_pil _____');
    SELECT prime INTO v_prime
    FROM pilote
    WHERE nompil = 'Hassan ZEKK';
    dbms_output.put_line('Avant, Prime de ALI : ' || v_prime);
    pack_pilote.prime_pil('Hassan ZEKK', 199);
    SELECT prime INTO v_prime
    FROM pilote
    WHERE nompil = 'ALI';
    dbms_output.put_line('Après, Prime de ALI : ' || v_prime);
    dbms_output.put_line(' _____ Teste de .sal_prim_pil _____');
    pack_pilote.sal_prim_pil;
END;
```



creation.sql × pack\_pilote spécification .sql × pack\_pilote BODY.sql × TEST.sql × PILOTE ×

Feuille de calcul SQL Historique

Feuille de calcul Query Builder

SET SERVEROUTPUT ON;

Sortie de script x

Tâche terminée en 0,044 secondes

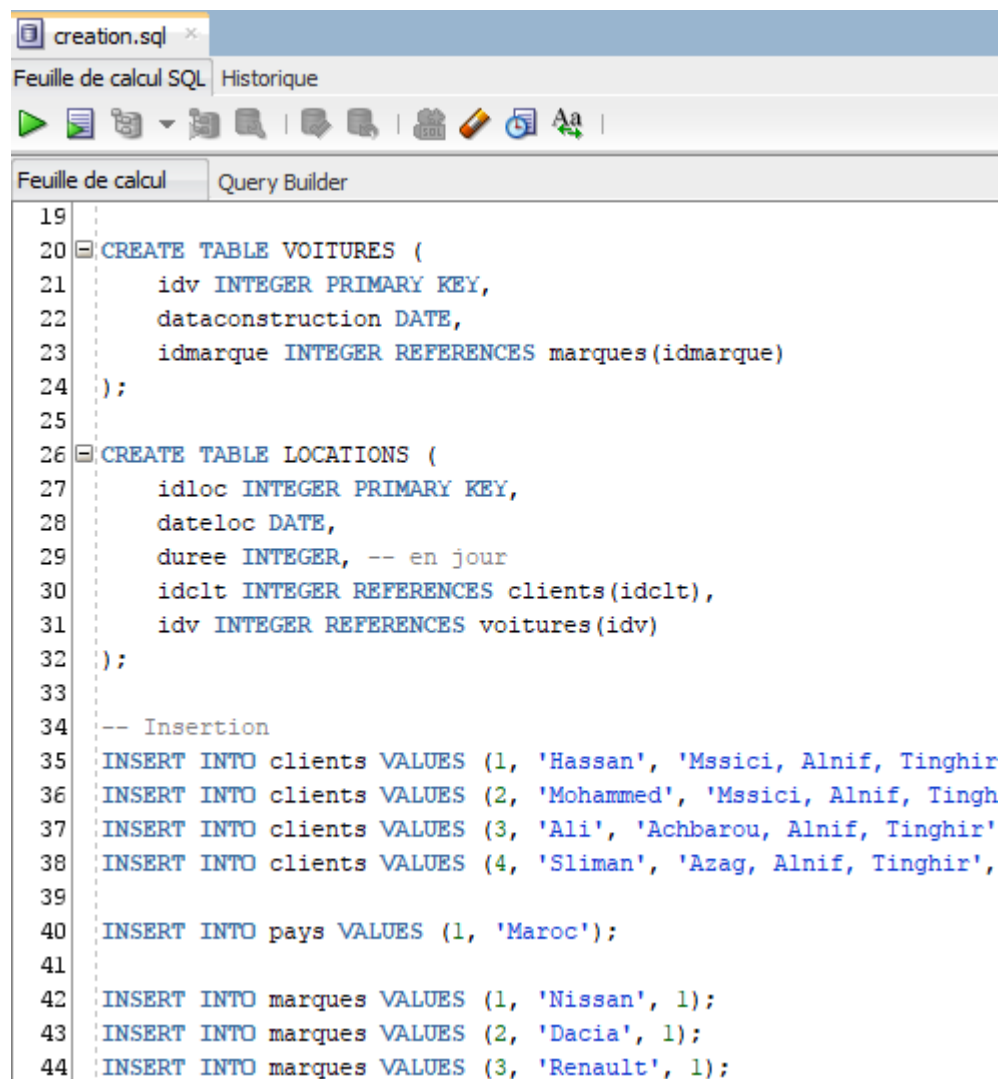
```

Teste de .list_pil
NOM      | ADRESSE      | SALAIRE      | PRIME
hassan   | msssici      | 1000         | 0
hassan   | mcissi, alnif | 1000         | 200
ALI      | AZAG, alnif  | 1000         | 199
hamZA    | TINGHIR, R565 | 1000         | 200
Teste de .inser_pil
NOM      | ADRESSE      | SALAIRE      | PRIME
hassan   | msssici      | 1000         | 0
hassan   | mcissi, alnif | 1000         | 200
ALI      | AZAG, alnif  | 1000         | 199
hamZA    | TINGHIR, R565 | 1000         | 200
Hassan ZEKK | msssici      | 1000         | 0
Teste de .sel_pil
Salaire de Hassan ZEKK: 1000 adresse: msssici
Teste de .sel_pil surchargé
Adresse de Hassan ZEKK: msssici
Teste de .prime_pil
Avant, Prime de ALI : 0
Après, Prime de ALI : 199
Teste de .sal_prim_pil
NOM      | ADRESSE      | SALAIRE TOTAL
hassan   | msssici      | 1000
hassan   | mcissi, alnif | 1200
ALI      | AZAG, alnif  | 1199
hamZA    | TINGHIR, R565 | 1200
Hassan ZEKK | msssici      | 1199

```

## II. EXERCICE 2

Création de la base de teste :



The screenshot shows a SQL IDE window titled 'creation.sql'. The interface includes a toolbar with icons for running queries, saving, and other database operations. The main editor area displays SQL code for creating two tables, 'VOITURES' and 'LOCATIONS', and inserting data into them. The code is as follows:

```
19
20 CREATE TABLE VOITURES (
21     idv INTEGER PRIMARY KEY,
22     dataconstruction DATE,
23     idmarque INTEGER REFERENCES marques(idmarque)
24 );
25
26 CREATE TABLE LOCATIONS (
27     idloc INTEGER PRIMARY KEY,
28     dateloc DATE,
29     duree INTEGER, -- en jour
30     idclt INTEGER REFERENCES clients(idclt),
31     idv INTEGER REFERENCES voitures(idv)
32 );
33
34 -- Insertion
35 INSERT INTO clients VALUES (1, 'Hassan', 'Mssici, Alnif, Tinghir
36 INSERT INTO clients VALUES (2, 'Mohammed', 'Mssici, Alnif, Tingh
37 INSERT INTO clients VALUES (3, 'Ali', 'Achbarou, Alnif, Tinghir'
38 INSERT INTO clients VALUES (4, 'Sliman', 'Azag, Alnif, Tinghir',
39
40 INSERT INTO pays VALUES (1, 'Maroc');
41
42 INSERT INTO marques VALUES (1, 'Nissan', 1);
43 INSERT INTO marques VALUES (2, 'Dacia', 1);
44 INSERT INTO marques VALUES (3, 'Renault', 1);
```

# 1. Définir un programme PL/SQL nommé qui ajoute une ligne à la table locations.

```

DECLARE
  procedure Q1 (p_idloc integer, p_dateloc Date,
    p_duree integer, p_idclt integer, p_idv integer) is
    N1 integer; -- vérifier si p_idloc existe
    N2 integer; -- vérifier si le client existe
    N3 integer; -- vérifier si la voiture existe
  Begin
    Select count(*) into N1 from Locations where idLoc = p_idloc;
    Select count(*) into N2 from Clients where idClct = p_idclt;
    Select count(*) into N3 from Voitures where idv = p_idv;
    If (N1 = 0) and (N2 > 0) and (N3 > 0) Then
      Insert into Locations values (idloc, p_dateloc,
        p_duree, p_idclt, p_idv);
    Else
      Dbms_output.put_line ('Données incorrectes');
    End if;
  End Q1;
BEGIN
  -- Appel de la procédure
  Q1(1, '12/12/19', 4, 1, 1);
END;
  
```

Sortie de script x

Tâche terminée en 0,104 secondes

Procédure PL/SQL terminée.

IDLOC	DATELOC	DUREE	IDCLT	IDV
1	12/12/19	4	1	1

## 2. Définir un bloc PL/SQL nommé qui affiche pour chaque client le numéro et la marque de la dernière voiture louée.

```

1  set SERVEROUTPUT ON;
2  DECLARE
3      numV INTEGER;
4      marqu Varchar(20);
5  procedure Q2 is
6      Cursor dernier is
7          Select C.idclt, C.nom,
8                  max(L.dateloc) as maxDate -- derniere location
9          From Clients C, Locations L
10         Where C.idclt = L.idclt
11         Group by C.idclt, C.nom;
12  Begin
13      Dbms_output.put_line ('Nom          |  num. Voiture | Marque');
14  For ligne in dernier loop
15
16      Select V.idv, M.nomMarque into numV, marqu
17      from Voitures V, Marques M, Locations L
18      Where L.idv = V.idv And V.idmarque = M.idmarque
19      And L.idClt = ligne.idClt And L.dateLoc = ligne.maxDate;
20      Dbms_output.put_line (ligne.nom || '      |      ' || numV || '      |      ' || marqu);
21  End loop;
22  End Q2;
23 BEGIN
24     -- appel
25     Q2();
26 END;

```

```

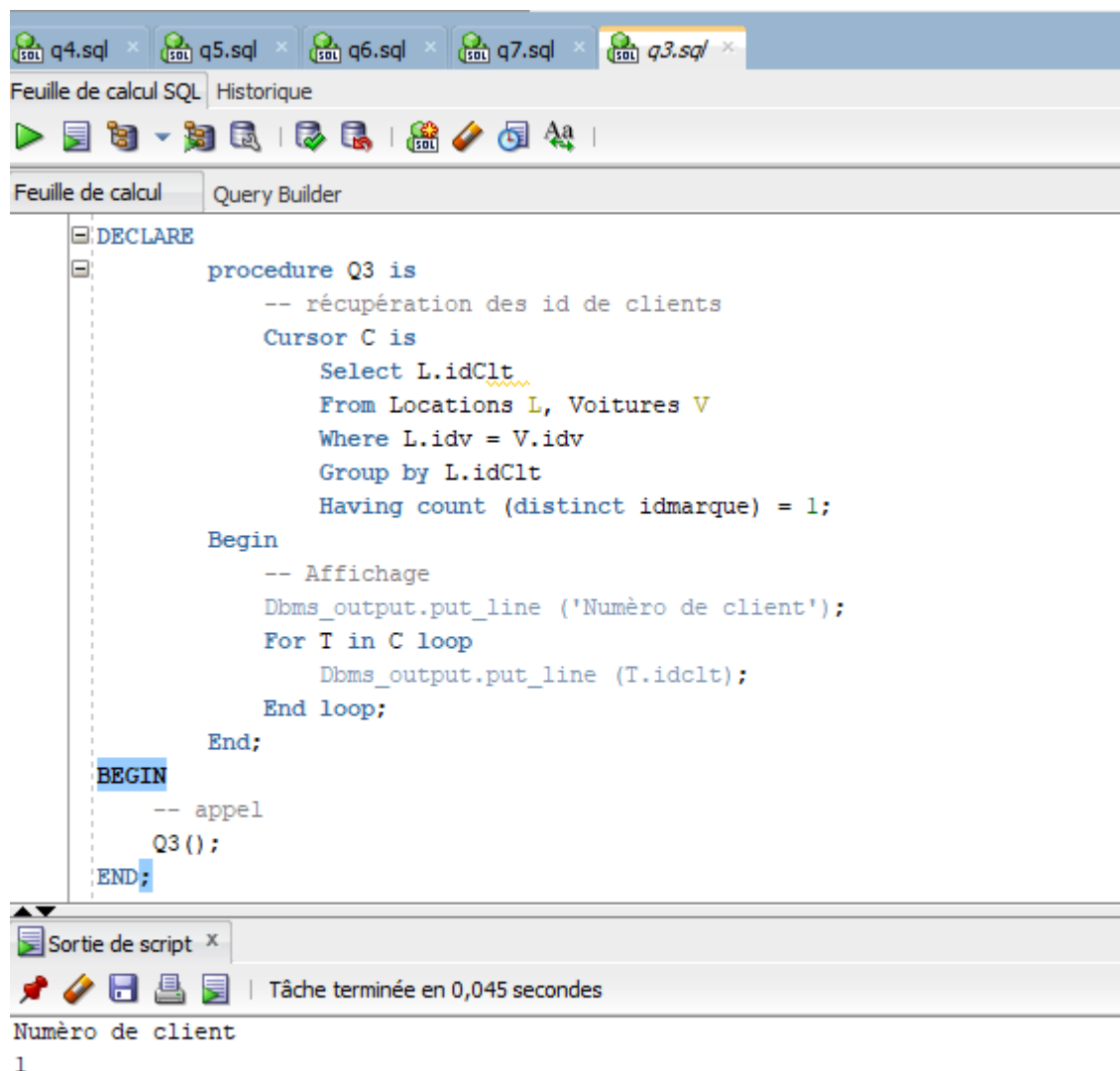
Sortie de script x
Tâche terminée en 0,066 secondes

Nom          |  num. Voiture | Marque
Hassan       |  1            | Nissan

Procédure PL/SQL terminée.

```

### 3. Définir un bloc PL/SQL nommé qui affiche les numéros des clients qui ont réservé seulement des voitures d'une seule marque :



The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains a PL/SQL procedure named Q3. The procedure declares a cursor C to select client IDs from the Locations and Voitures tables, grouped by client ID and filtered to show only those with a single car brand. The procedure then prints the client IDs. The results pane shows the output of the procedure, which is the number 1.

```
DECLARE
procedure Q3 is
  -- récupération des id de clients
  Cursor C is
    Select L.idClt
    From Locations L, Voitures V
    Where L.idv = V.idv
    Group by L.idClt
    Having count (distinct idmarque) = 1;
  Begin
    -- Affichage
    Dbms_output.put_line ('Numéro de client');
    For T in C loop
      Dbms_output.put_line (T.idclt);
    End loop;
  End;
BEGIN
  -- appel
  Q3 ();
END;
```

Sortie de script x

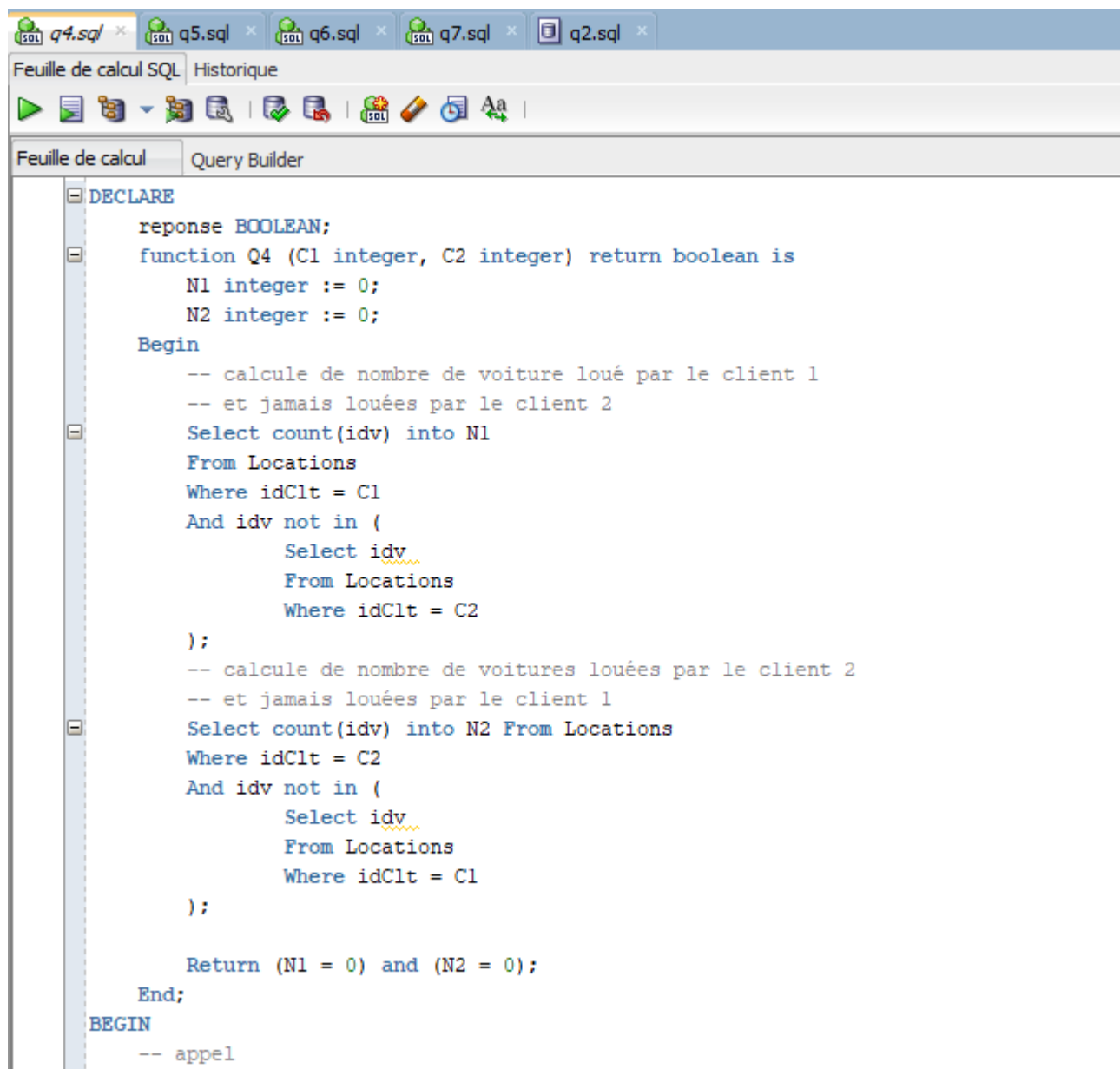
Tâche terminée en 0,045 secondes

Numéro de client

1

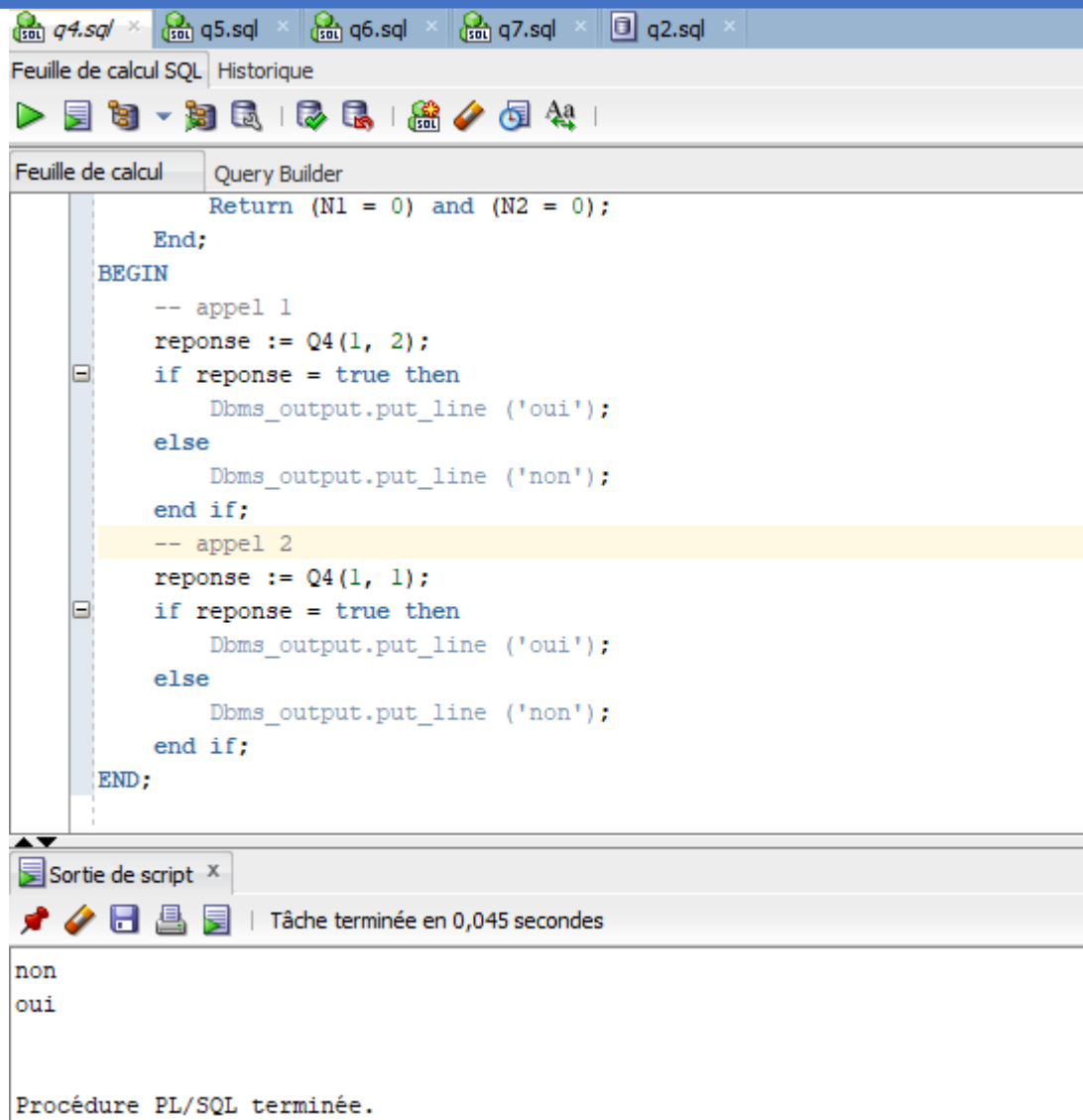
Procédure PL/SQL terminée.

#### 4. Définir un programme PL/SQL nommé permettant de savoir si deux clients ont loué exactement les mêmes voitures :



```
DECLARE
    reponse BOOLEAN;
function Q4 (C1 integer, C2 integer) return boolean is
    N1 integer := 0;
    N2 integer := 0;
Begin
    -- calcule de nombre de voiture loué par le client 1
    -- et jamais louées par le client 2
    Select count(idv) into N1
    From Locations
    Where idClt = C1
    And idv not in (
        Select idv
        From Locations
        Where idClt = C2
    );
    -- calcule de nombre de voitures louées par le client 2
    -- et jamais louées par le client 1
    Select count(idv) into N2 From Locations
    Where idClt = C2
    And idv not in (
        Select idv
        From Locations
        Where idClt = C1
    );

    Return (N1 = 0) and (N2 = 0);
End;
BEGIN
    -- appel
```



The screenshot displays the Oracle SQL Developer environment. At the top, several tabs are open, including 'q4.sql', 'q5.sql', 'q6.sql', 'q7.sql', and 'q2.sql'. The main window is titled 'Feuille de calcul SQL' and 'Historique'. Below this is a toolbar with various icons for execution and editing. The 'Query Builder' tab is active, showing a PL/SQL script. The script includes a 'Return' statement, an 'End;' statement, and a 'BEGIN' block. Inside the 'BEGIN' block, there are two sections: 'appel 1' and 'appel 2'. Each section contains a call to 'Q4' with different arguments, followed by an 'if' statement that checks the 'reponse' and prints 'oui' or 'non' using 'Dbms\_output.put\_line'. The script ends with 'END;'. Below the script editor, the 'Sortie de script' window is open, showing the output of the script execution. The output consists of two lines: 'non' and 'oui', followed by the message 'Procédure PL/SQL terminée.'.

```
Return (N1 = 0) and (N2 = 0);  
End;  
BEGIN  
  -- appel 1  
  reponse := Q4(1, 2);  
  if reponse = true then  
    Dbms_output.put_line ('oui');  
  else  
    Dbms_output.put_line ('non');  
  end if;  
  -- appel 2  
  reponse := Q4(1, 1);  
  if reponse = true then  
    Dbms_output.put_line ('oui');  
  else  
    Dbms_output.put_line ('non');  
  end if;  
END;
```

Sortie de script x

Tâche terminée en 0,045 secondes

non  
oui

Procédure PL/SQL terminée.

## 5. Définir un bloc PL/SQL nommé affichant les noms des clients qui ont effectué des locations les plus longues.

The screenshot shows the Oracle SQL Developer interface. The top pane displays a PL/SQL procedure named Q5. The procedure calculates the total duration for each client, finds the maximum duration, and then lists the names of the clients who have the maximum duration. The bottom pane shows the output of the procedure, which is the name of the client with the maximum duration, Hassan.

```

DECLARE
  procedure Q5 is
    -- récupérer la duree totale par client
    Cursor Somme is
      Select C.idClt, C.nom, sum(duree) as sommeDuree
      From Clients C, Locations L
      Where L.idClt = C.idClt
      Group by C.idClt, C.nom;

    maxDuree integer;
  Begin
    -- trouver la duree max
    Select max(sum(duree)) into maxDuree
    from Locations
    group by idClt;
    -- afficher les client ayant la duree max
    Dbms_output.put_line ('Nom client');
    For T in Somme loop
      If (T.sommeDuree = maxDuree) then
        Dbms_output.put_line (T.nom);
      End if;
    End loop;
  End;
BEGIN
  Q5;
END;

```

Sortie de script x

Tâche terminée en 0,045 secondes

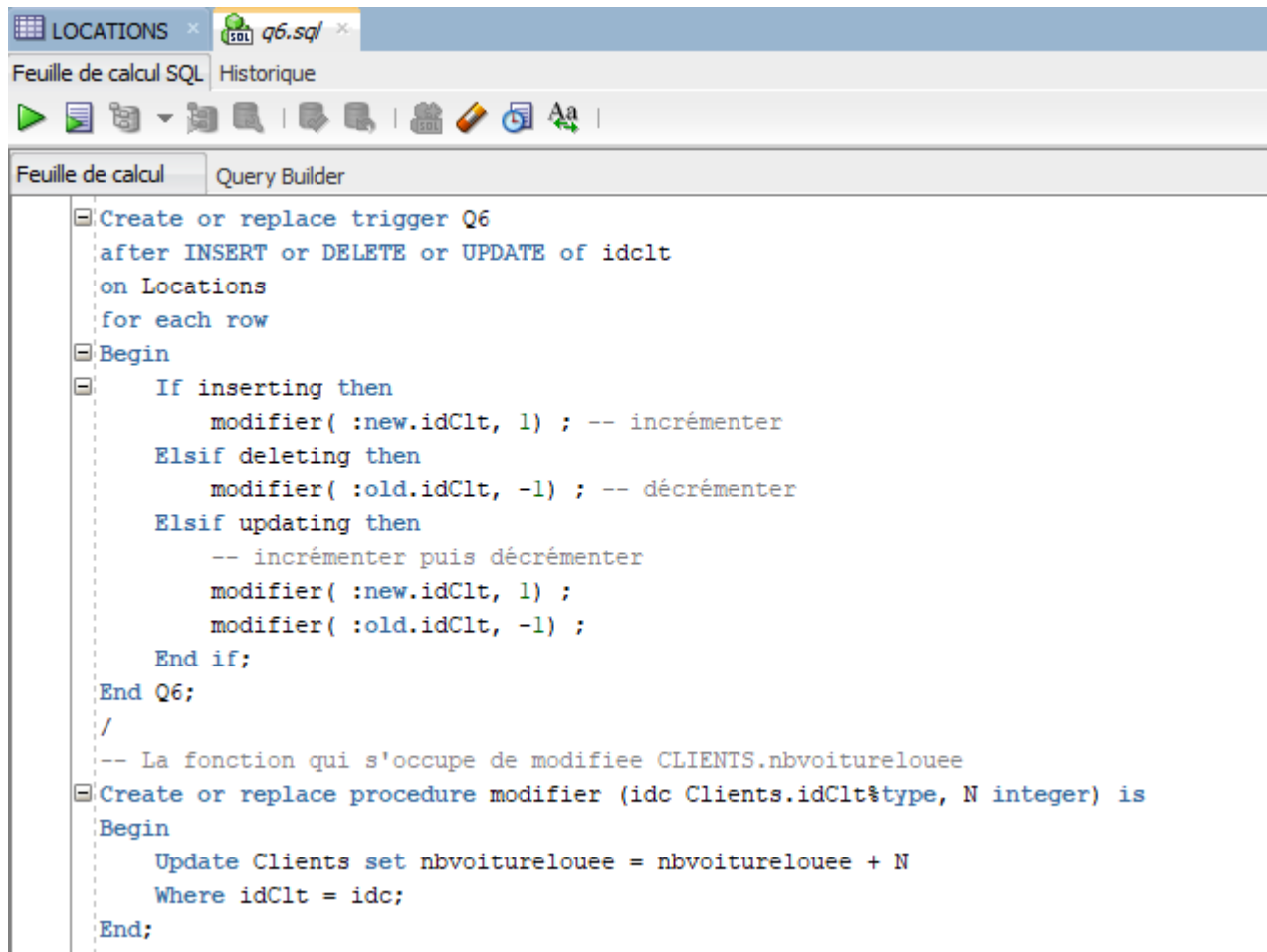
Nom client  
Hassan

Procédure PL/SQL terminée.



6. Lorsque la table locations est manipulée, les nbvoiturelouee des clients doivent rester cohérents avec les données existant dans la table locations. Ecrire le déclencheur assurant cette cohérence dans le cas suivants :

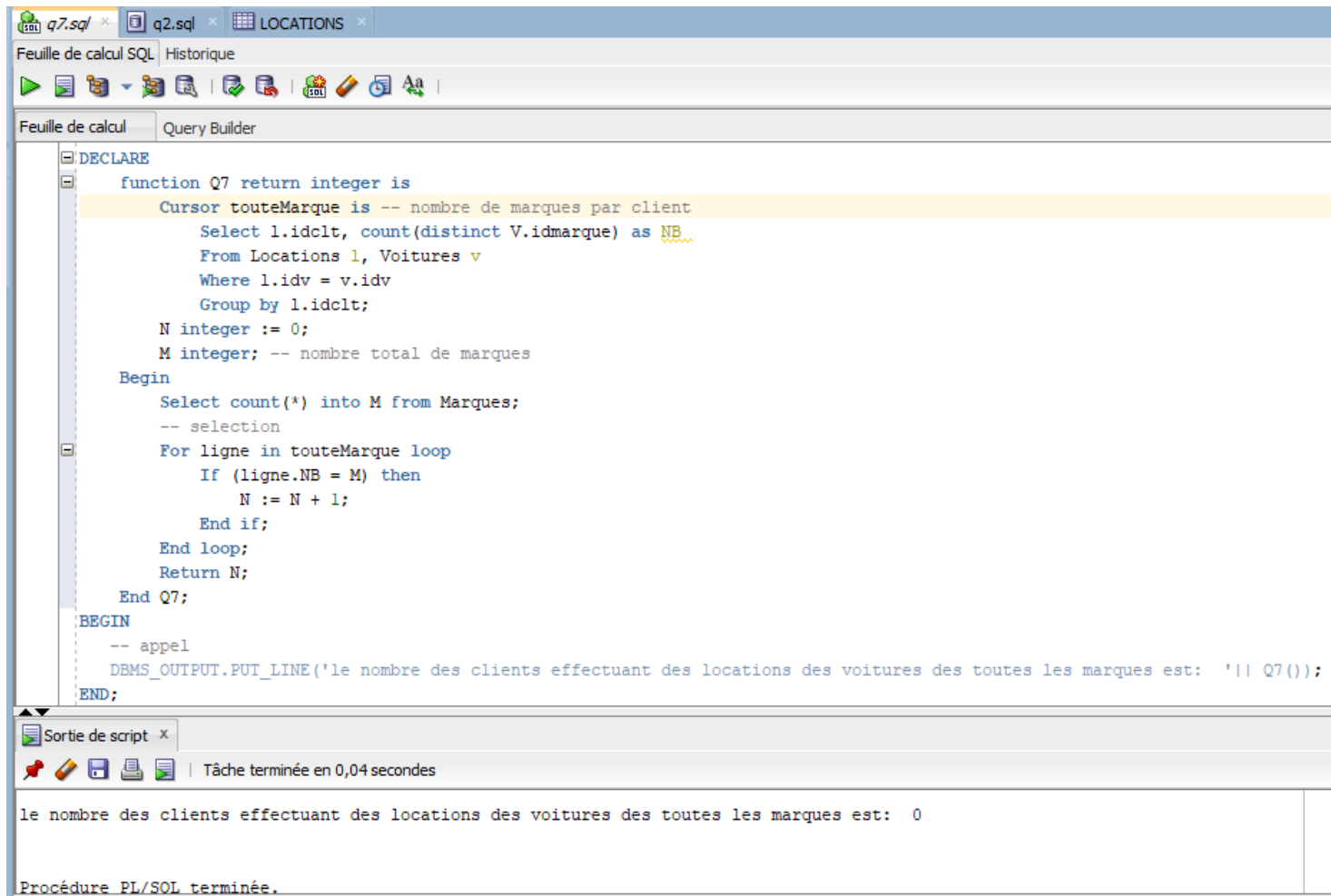
- Lorsqu'on ajoute un tuple.
- Lorsqu'on supprime un tuple.
- Lorsqu'on modifie les valeurs des attributs



```
LOCATIONS x q6.sql x
Feuille de calcul SQL Historique
Feuille de calcul Query Builder

Create or replace trigger Q6
after INSERT or DELETE or UPDATE of idclt
on Locations
for each row
Begin
  If inserting then
    modifier( :new.idClit, 1) ; -- incrémenter
  Elsif deleting then
    modifier( :old.idClit, -1) ; -- décrémenter
  Elsif updating then
    -- incrémenter puis décrémenter
    modifier( :new.idClit, 1) ;
    modifier( :old.idClit, -1) ;
  End if;
End Q6;
/
-- La fonction qui s'occupe de modifier CLIENTS.nbvoiturelouee
Create or replace procedure modifier (idc Clients.idClit%type, N integer) is
Begin
  Update Clients set nbvoiturelouee = nbvoiturelouee + N
  Where idClit = idc;
End;
```

## 7. Définir un bloc PL/SQL nommé permettant de compter le nombre des clients effectuant des locations des voitures des toutes les marques



The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains a PL/SQL function named Q7 that counts the number of clients who have rented cars from all brands. The function uses a cursor to iterate over the results of a query that counts distinct car brands per client. The results pane shows the output of the function call, which is 0.

```
DECLARE
  function Q7 return integer is
    Cursor touteMarque is -- nombre de marques par client
      Select l.idclt, count(distinct V.idmarque) as NB
      From Locations l, Voitures v
      Where l.idv = v.idv
      Group by l.idclt;
    N integer := 0;
    M integer; -- nombre total de marques
  Begin
    Select count(*) into M from Marques;
    -- selection
    For ligne in touteMarque loop
      If (ligne.NB = M) then
        N := N + 1;
      End if;
    End loop;
    Return N;
  End Q7;
BEGIN
  -- appel
  DBMS_OUTPUT.PUT_LINE('le nombre des clients effectuant des locations des voitures des toutes les marques est: '|| Q7());
END;
```

Sortie de script x

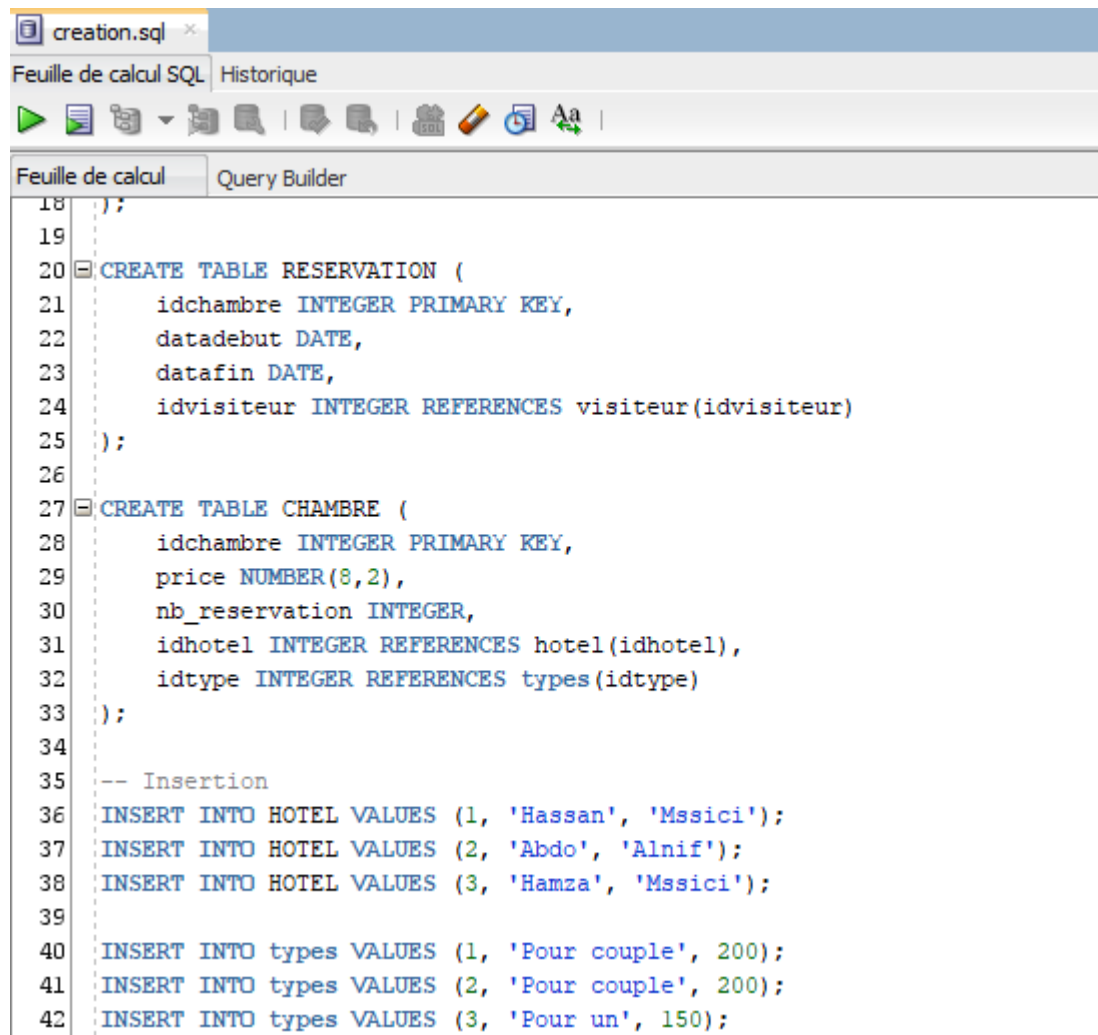
Tâche terminée en 0,04 secondes

le nombre des clients effectuant des locations des voitures des toutes les marques est: 0

Procédure PL/SQL terminée.

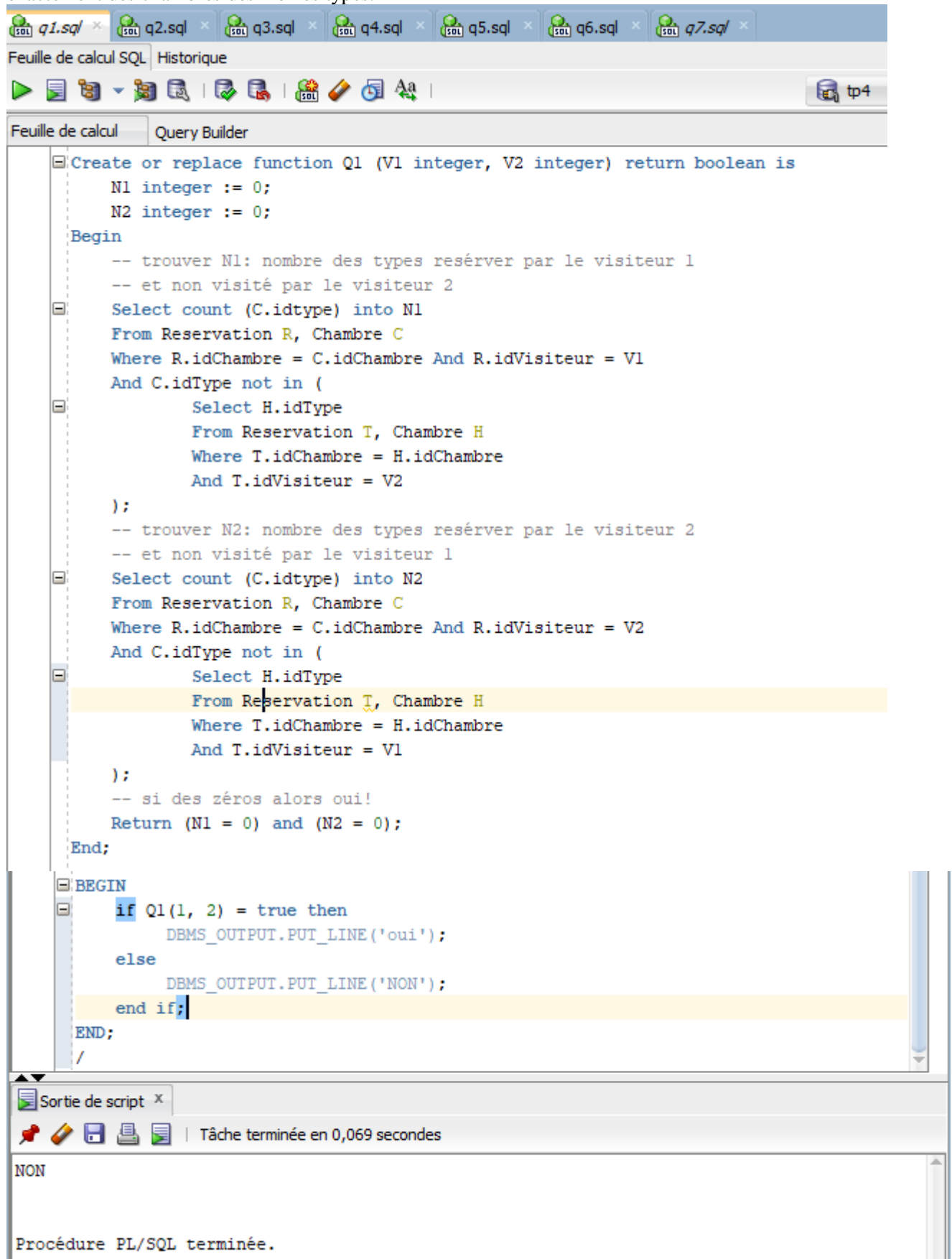
### III. EXERCICE 3

Création de la base de données :



```
18 );
19
20 CREATE TABLE RESERVATION (
21     idchambre INTEGER PRIMARY KEY,
22     datadebut DATE,
23     datafin DATE,
24     idvisiteur INTEGER REFERENCES visiteur(idvisiteur)
25 );
26
27 CREATE TABLE CHAMBRE (
28     idchambre INTEGER PRIMARY KEY,
29     price NUMBER(8,2),
30     nb_reservation INTEGER,
31     idhotel INTEGER REFERENCES hotel(idhotel),
32     idtype INTEGER REFERENCES types(idtype)
33 );
34
35 -- Insertion
36 INSERT INTO HOTEL VALUES (1, 'Hassan', 'Mssici');
37 INSERT INTO HOTEL VALUES (2, 'Abdo', 'Alnif');
38 INSERT INTO HOTEL VALUES (3, 'Hamza', 'Mssici');
39
40 INSERT INTO types VALUES (1, 'Pour couple', 200);
41 INSERT INTO types VALUES (2, 'Pour couple', 200);
42 INSERT INTO types VALUES (3, 'Pour un', 150);
```

1. Définir un programme PL/SQL nommée Q1 permettant de savoir si deux visiteurs ont réservé exactement des chambres des mêmes types.



```

Create or replace function Q1 (V1 integer, V2 integer) return boolean is
  N1 integer := 0;
  N2 integer := 0;
Begin
  -- trouver N1: nombre des types réserver par le visiteur 1
  -- et non visité par le visiteur 2
  Select count (C.idtype) into N1
  From Reservation R, Chambre C
  Where R.idChambre = C.idChambre And R.idVisiteur = V1
  And C.idType not in (
    Select H.idType
    From Reservation T, Chambre H
    Where T.idChambre = H.idChambre
    And T.idVisiteur = V2
  );
  -- trouver N2: nombre des types réserver par le visiteur 2
  -- et non visité par le visiteur 1
  Select count (C.idtype) into N2
  From Reservation R, Chambre C
  Where R.idChambre = C.idChambre And R.idVisiteur = V2
  And C.idType not in (
    Select H.idType
    From Reservation T, Chambre H
    Where T.idChambre = H.idChambre
    And T.idVisiteur = V1
  );
  -- si des zéros alors oui!
  Return (N1 = 0) and (N2 = 0);
End;

BEGIN
  if Q1(1, 2) = true then
    DBMS_OUTPUT.PUT_LINE('oui');
  else
    DBMS_OUTPUT.PUT_LINE('NON');
  end if;
END;
/

```

Sortie de script x

Tâche terminée en 0,069 secondes

NON

Procédure PL/SQL terminée.

```

BEGIN
  if Q1(1, 1) = true then
    DBMS_OUTPUT.PUT_LINE('oui');
  else
    DBMS_OUTPUT.PUT_LINE('NON');
  end if;
END;
/

```

Sortie de script x

Tâche terminée en 0,043 secondes

oui

Procédure PL/SQL terminée.

2. Définir un programme PL/SQL nommée Q2 qui affiche les noms de trois premiers visiteurs qui ont effectué le plus grand nombre des réservations.

```

q2.sql x q3.sql x q4.sql x q5.sql x q6.sql x q7.sql x
Feuille de calcul SQL Historique
0,037 secondes tp4
Feuille de calcul Query Builder
Create or replace procedure Q2 is
  -- ordonné les visiteur par nombre réservation
  Cursor premiers is
    Select V.idVisiteur , V.nom From Visiteur V, Reservation R
    Where V.idVisiteur = R.idVisiteur
    Group by V.idVisiteur , V.nom
    Order by count(*) desc;
  ligne premiers%rowtype;
Begin
  open premiers;
  Loop Fetch premiers into ligne;
    Exit when (premiers%notfound) or (premiers%rowcount) > 3;
    Dbms_output.put_line (ligne.nom);
  End loop;
  close premiers;
End;
/

```

```
-- appel
BEGIN
    Q2;
END;
/
```

Sortie de script x

Tâche terminée en 0,037 secondes

Elément Procedure Q2 compilé

Mohammed Zekkouri  
Hassan Zekkouri

Procédure PL/SQL terminée.

3. Définir un bloc PL/SQL nommée Q3 retournant le nombre des chambres qui n'ont pas été réservé depuis 1 Juin 2009.

q3.sql x q4.sql x q5.sql x q6.sql x q7.sql x RESERVATION x creation.sql x

Feuille de calcul SQL Historique

0,04 secondes tp4

Feuille de calcul Query Builder

```
set SERVEROUTPUT ON;
Create or replace function Q3 return integer is
    N integer;
Begin
    Select count(*) into N From chambre
    Where idChambre not in (
        select idChambre
        from Reservation
        Where dataDebut >= '01/07/2009'
    );
    Return N;
End Q3;
/
-- appel
BEGIN
    DBMS_OUTPUT.put_line('Reponse , ' || Q3());
END;
/
```

Sortie de script x

Tâche terminée en 0,04 secondes

Elément Function Q3 compilé

Reponse , 1

Procédure PL/SQL terminée.

	IDCHAMBRE	DATADEBUT	DATAFIN	IDVISITEUR
1	1	12/02/12	13/02/12	2
2	2	12/02/14	13/02/14	1
3	3	12/02/08	13/02/08	1
4	4	12/02/08	13/02/08	3

4. Définir un bloc PL/SQL nommée Q4 qui affiche les numéros des visiteurs qui ont réservé des chambres de tous les types.

```

SET SERVEROUT ON;
Create or replace procedure Q4 is
  Cursor typesParVisiteur is
    Select R.idVisiteur , count(distinct C.idType) as NB
    From Reservation R, Chambre C
    Where R.idChambre = C.idChambre
    Group by R.idVisiteur;
  N integer;
Begin
  -- nombre total des types
  Select count(*) into N from Types;
  Dbms_output.put_line ('Voici les numéros des visiteurs qui ont réservé des
chambres de tous les types:');
  For ligne in typesParVisiteur loop
    If (ligne.NB = N) then
      Dbms_output.put_line (ligne.idVisiteur);
    End if;
  End loop;
End;
/
-- appel
BEGIN
  Q4;
END;
/

```

5. Définir un bloc PL/SQL nommé Q5 affichant pour chaque chambre, le nom du visiteur qui a effectué la dernière réservation.

```

SET SERVEROUT ON;
Create or replace procedure Q5 is
  -- dernier reservation par chambre
  Cursor dernierReservation is
    Select idChambre , max(dataDebut) as maxDate
    From Reservation
    Group by idChambre;
  NomVisiteur Visiteur.nom$type;
Begin
  Dbms_output.put_line ('pour chaque chambre, le nom du visiteur qui a effectué la dernière réservation. ?');
  Dbms_output.put_line ('id Chambre | nom Visiteur');
  For ligne in dernierReservation loop

    Select V.nom into NomVisiteur
    from Visiteur V, Reservation R
    Where V.idVisiteur = R.idVisiteur
    And R.idChambre = ligne.idChambre And R.dataDebut = ligne.maxDate;

    Dbms_output.put_line (ligne.idChambre || ' ' || NomVisiteur);
  End loop;
End Q5;
/
-- appel
BEGIN
  Q5;
END;
/

```

Sortie de script x

Tâche terminée en 0,056 secondes

```

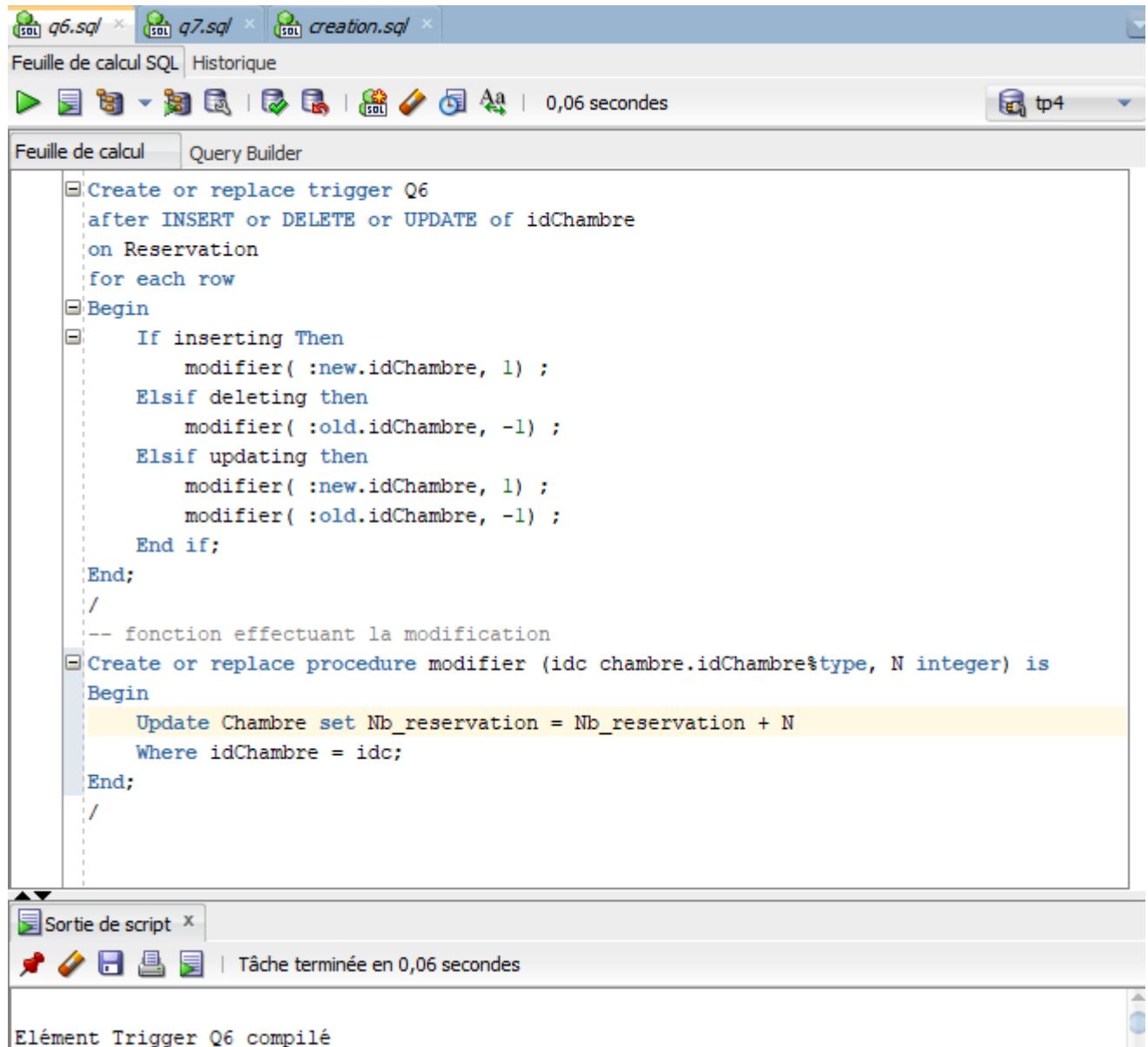
pour chaque chambre, le nom du visiteur qui a effectué la dernière réservation. ?
id Chambre | nom Visiteur
1          Mohammed Zekkouri
2          Hassan Zekkouri
3          Hassan Zekkouri
4          Othmane Zekkouri

Procédure PL/SQL terminée.

```



6. Lorsque la table **Réservation** est manipulée, les *Nb\_réservation* des chambres doivent rester cohérents avec les données existant dans la table **Réservation**. Ecrire le déclencheur assurant cette cohérence dans les cas suivants :
- Lorsqu'on ajoute un tuple.
  - Lorsqu'on supprime un tuple.
  - Lorsqu'on modifie les valeurs des attributs



The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains the following SQL code:

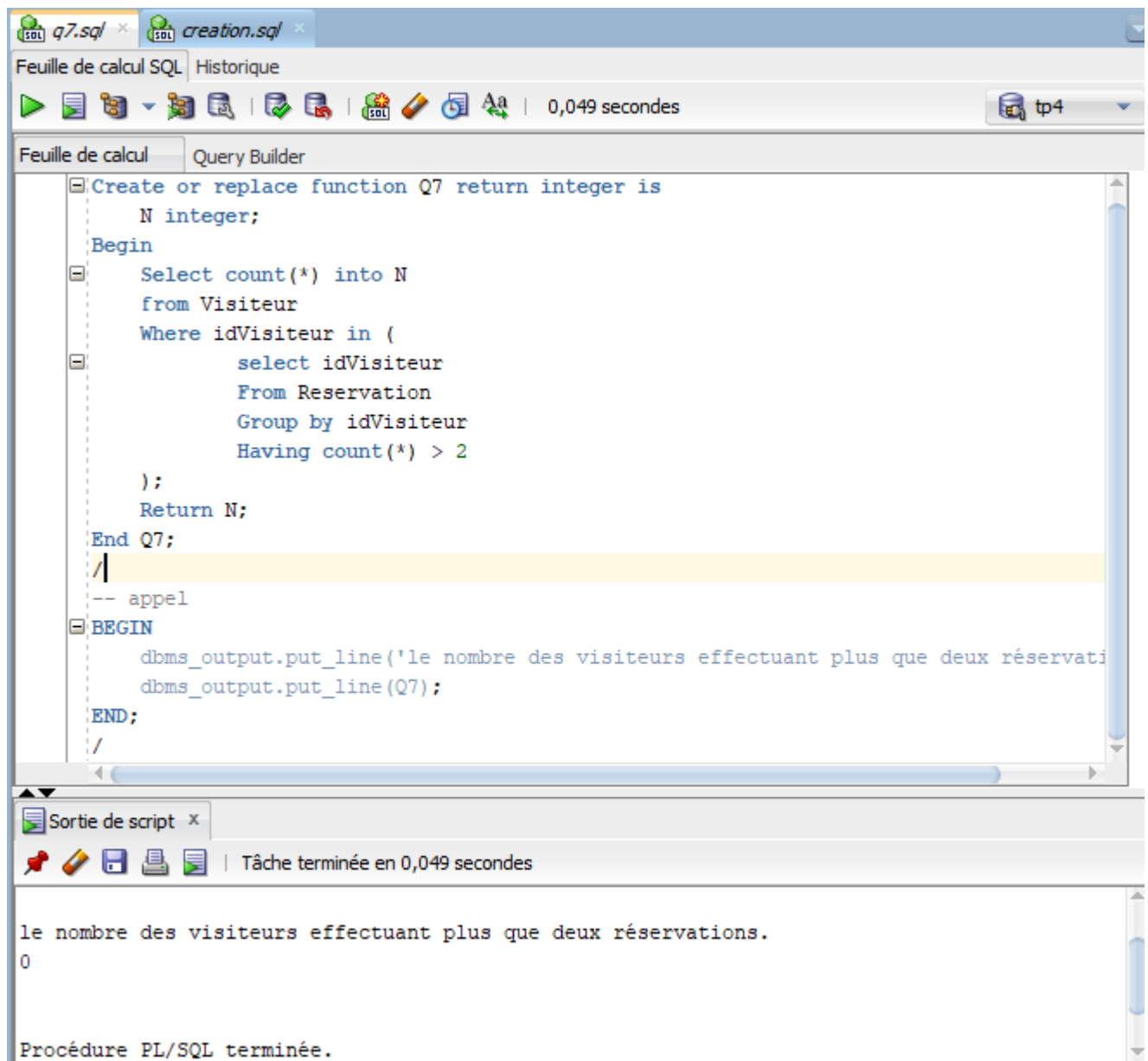
```
Create or replace trigger Q6
after INSERT or DELETE or UPDATE of idChambre
on Reservation
for each row
Begin
  If inserting Then
    modifier( :new.idChambre, 1) ;
  Elsif deleting then
    modifier( :old.idChambre, -1) ;
  Elsif updating then
    modifier( :new.idChambre, 1) ;
    modifier( :old.idChambre, -1) ;
  End if;
End;
/
-- fonction effectuant la modification
Create or replace procedure modifier (idc chambre.idChambre%type, N integer) is
Begin
  Update Chambre set Nb_reservation = Nb_reservation + N
  Where idChambre = idc;
End;
/
```

The results pane shows the output of the script execution:

```
Sortie de script x
Tâche terminée en 0,06 secondes

Elément Trigger Q6 compilé
```

7. Définir un programme PL/SQL nommé Q7 permettant de compter le nombre des visiteurs effectuant plus que deux réservations.



The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Feuille de calcul SQL', shows the SQL script for creating and calling the function Q7. The script defines a function that counts visitors with more than two reservations. The bottom pane, titled 'Sortie de script', shows the output of the script execution.

```
q7.sql x creation.sql x
Feuille de calcul SQL Historique
0,049 secondes tp4

Feuille de calcul Query Builder

Create or replace function Q7 return integer is
  N integer;
Begin
  Select count(*) into N
  from Visiteur
  Where idVisiteur in (
    select idVisiteur
    From Reservation
    Group by idVisiteur
    Having count(*) > 2
  );
  Return N;
End Q7;
/
-- appel
BEGIN
  dbms_output.put_line('le nombre des visiteurs effectuant plus que deux réservations');
  dbms_output.put_line(Q7);
END;
/

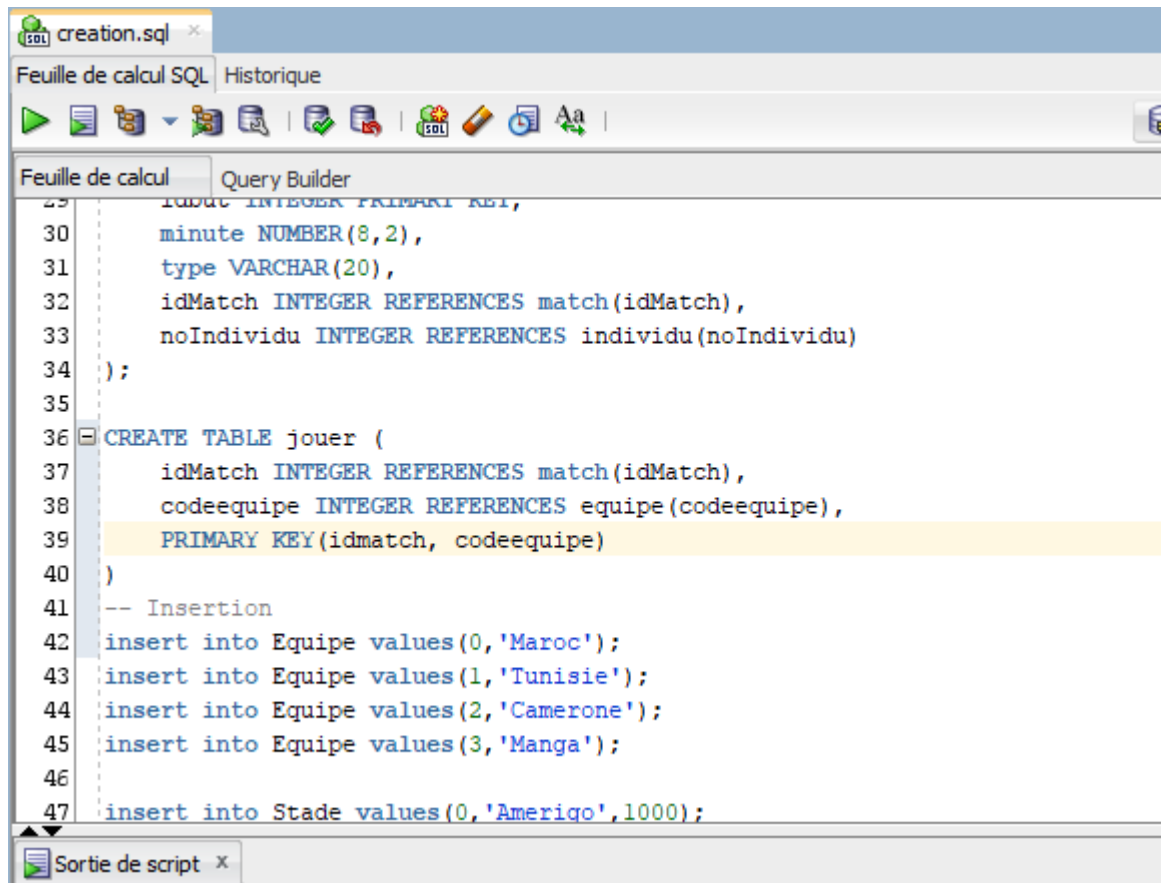
Sortie de script x
Tâche terminée en 0,049 secondes

le nombre des visiteurs effectuant plus que deux réservations.
0

Procédure PL/SQL terminée.
```

## IV. EXERCICE 4

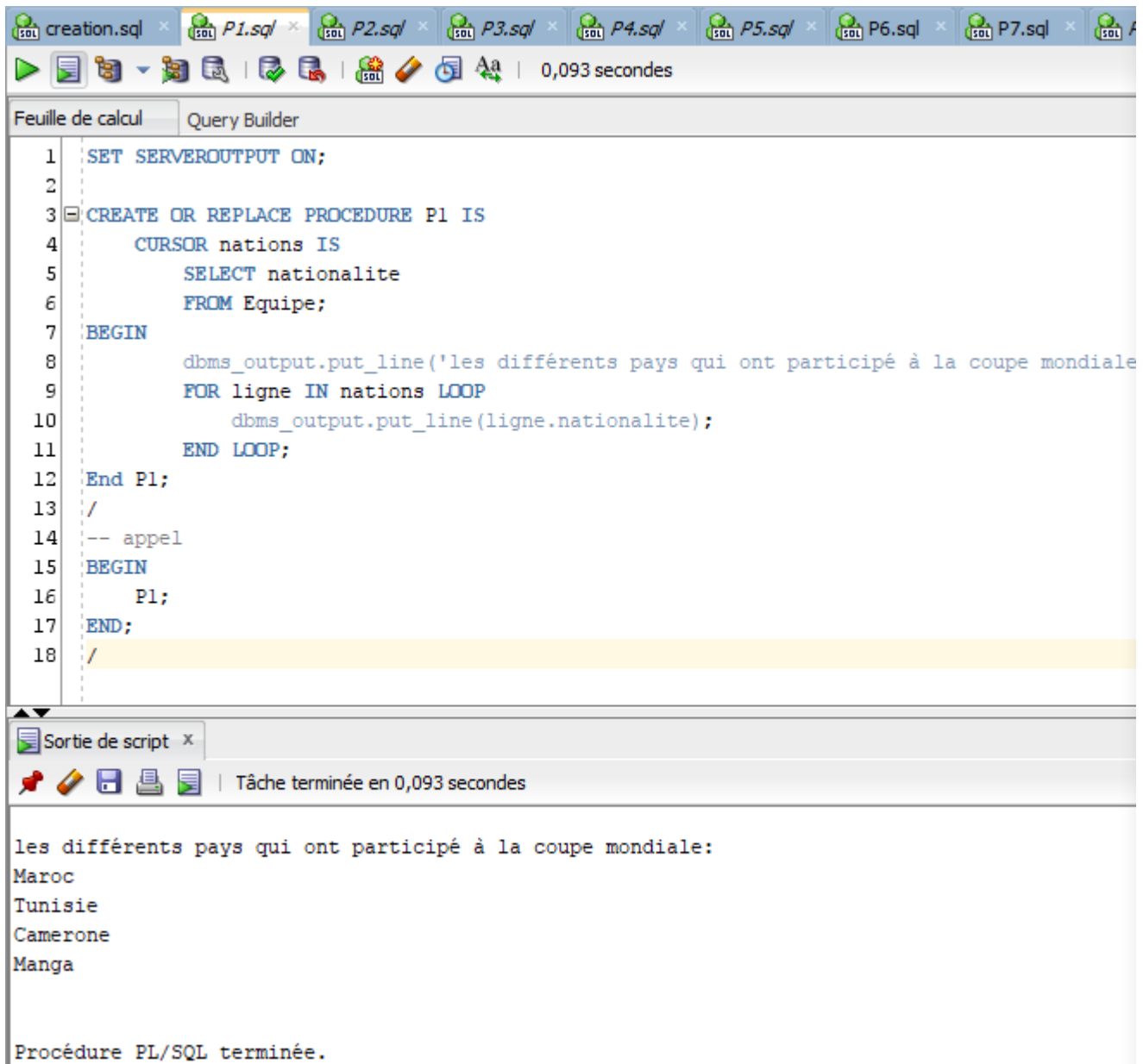
Création de la base de données :



```
29      idMatch INTEGER PRIMARY KEY,  
30      minute NUMBER(8,2),  
31      type VARCHAR(20),  
32      idMatch INTEGER REFERENCES match(idMatch),  
33      noIndividu INTEGER REFERENCES individu(noIndividu)  
34  );  
35  
36  CREATE TABLE jouer (  
37      idMatch INTEGER REFERENCES match(idMatch),  
38      codeequipe INTEGER REFERENCES equipe(codeequipe),  
39      PRIMARY KEY(idmatch, codeequipe)  
40  )  
41  -- Insertion  
42  insert into Equipe values(0,'Maroc');  
43  insert into Equipe values(1,'Tunisie');  
44  insert into Equipe values(2,'Camerone');  
45  insert into Equipe values(3,'Manga');  
46  
47  insert into Stade values(0,'Amerigo',1000);
```

## A. Procédures stockées :

1. Afficher les différents pays qui ont participé à la coupe mondiale.



The screenshot displays the Oracle SQL Developer environment. The top toolbar shows various icons for file operations and execution. The main window is titled 'Feuille de calcul' and 'Query Builder'. It contains a PL/SQL script with the following code:

```
1 SET SERVEROUTPUT ON;
2
3 CREATE OR REPLACE PROCEDURE P1 IS
4     CURSOR nations IS
5         SELECT nationalite
6         FROM Equipe;
7 BEGIN
8     dbms_output.put_line('les différents pays qui ont participé à la coupe mondiale');
9     FOR ligne IN nations LOOP
10        dbms_output.put_line(ligne.nationalite);
11    END LOOP;
12 End P1;
13 /
14 -- appel
15 BEGIN
16     P1;
17 END;
18 /
```

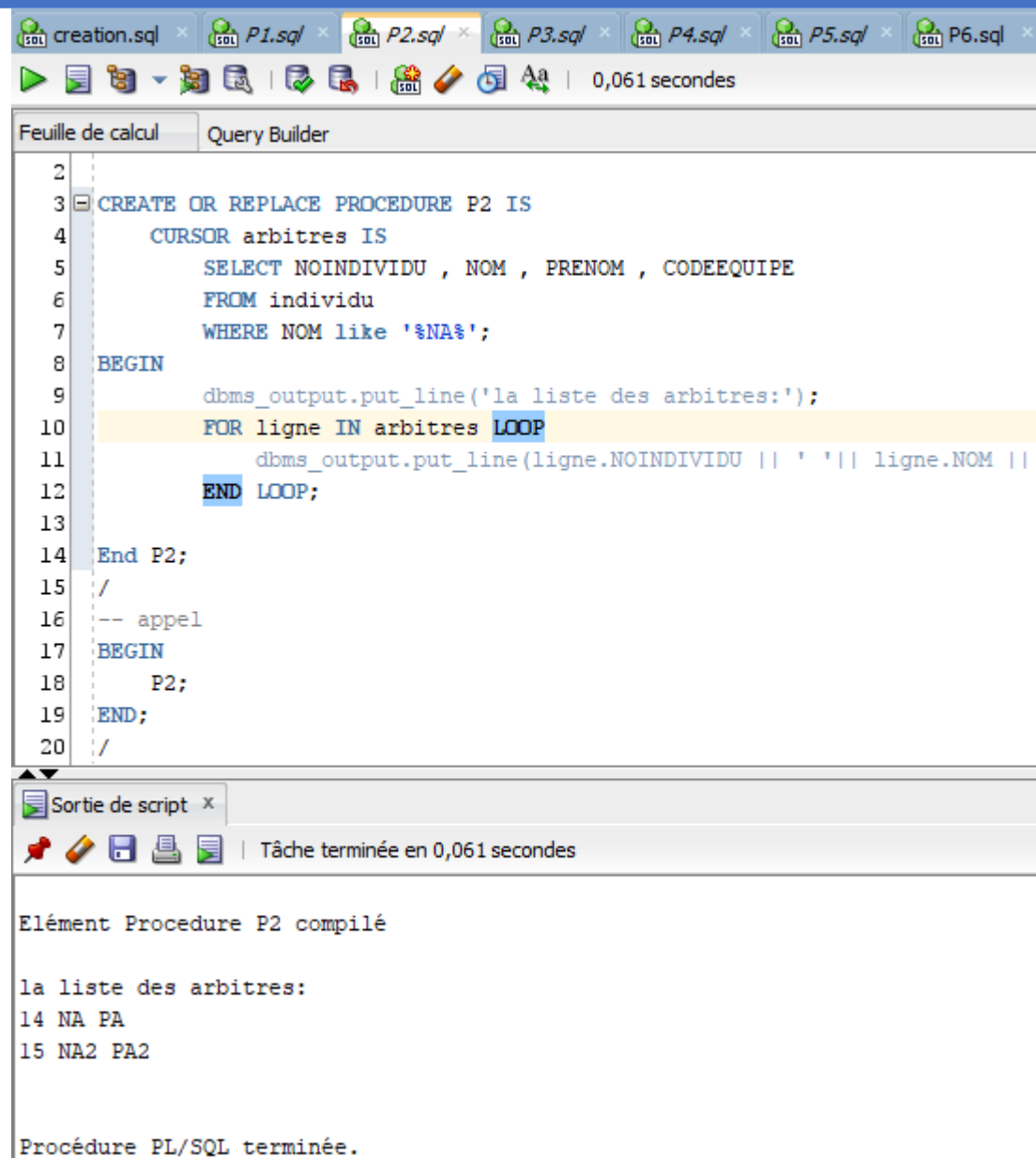
Below the script editor, the 'Sortie de script' window shows the output of the procedure execution:

```
les différents pays qui ont participé à la coupe mondiale:
Maroc
Tunisie
Camerone
Manga

Procédure PL/SQL terminée.
```

The status bar at the bottom of the 'Sortie de script' window indicates 'Tâche terminée en 0,093 secondes'.

2. Afficher la liste des arbitres



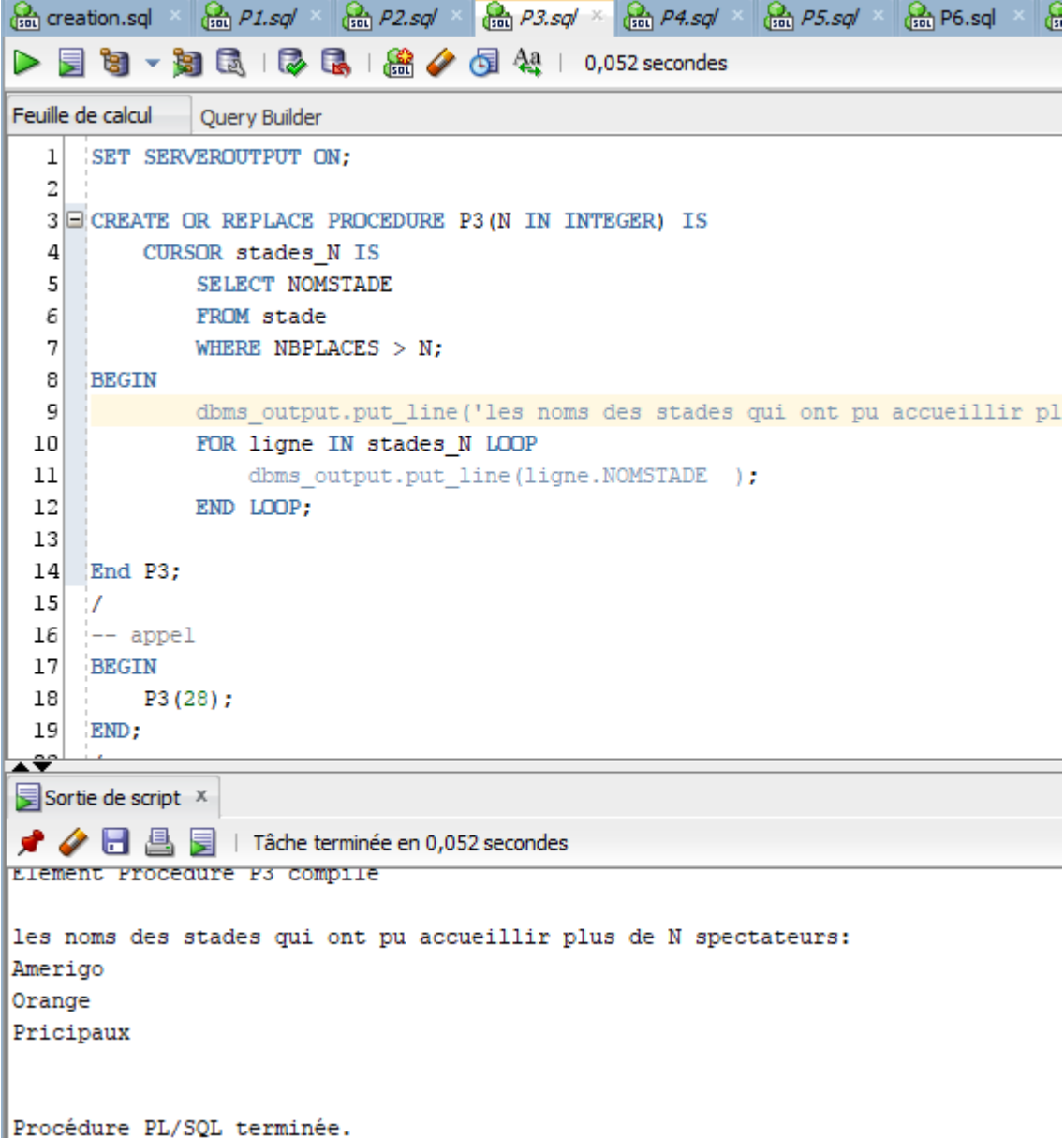
The screenshot displays the Oracle SQL Developer environment. The top toolbar shows various icons for file operations and execution. The main window is titled 'Feuille de calcul' and 'Query Builder'. It contains a PL/SQL script with the following code:

```
2  
3 CREATE OR REPLACE PROCEDURE P2 IS  
4     CURSOR arbitres IS  
5         SELECT NOINDIVIDU , NOM , PRENOM , CODEEQUIPE  
6         FROM individu  
7         WHERE NOM like '%NA%';  
8 BEGIN  
9     dbms_output.put_line('la liste des arbitres:');  
10    FOR ligne IN arbitres LOOP  
11        dbms_output.put_line(ligne.NOINDIVIDU || ' ' || ligne.NOM ||  
12    END LOOP;  
13  
14 End P2;  
15 /  
16 -- appel  
17 BEGIN  
18     P2;  
19 END;  
20 /
```

Below the script editor, the 'Sortie de script' window shows the execution results:

```
Elément Procedure P2 compilé  
  
la liste des arbitres:  
14 NA PA  
15 NA2 PA2  
  
Procédure PL/SQL terminée.
```

### 3. Afficher les noms des stades qui ont pu accueillir plus de N spectateurs



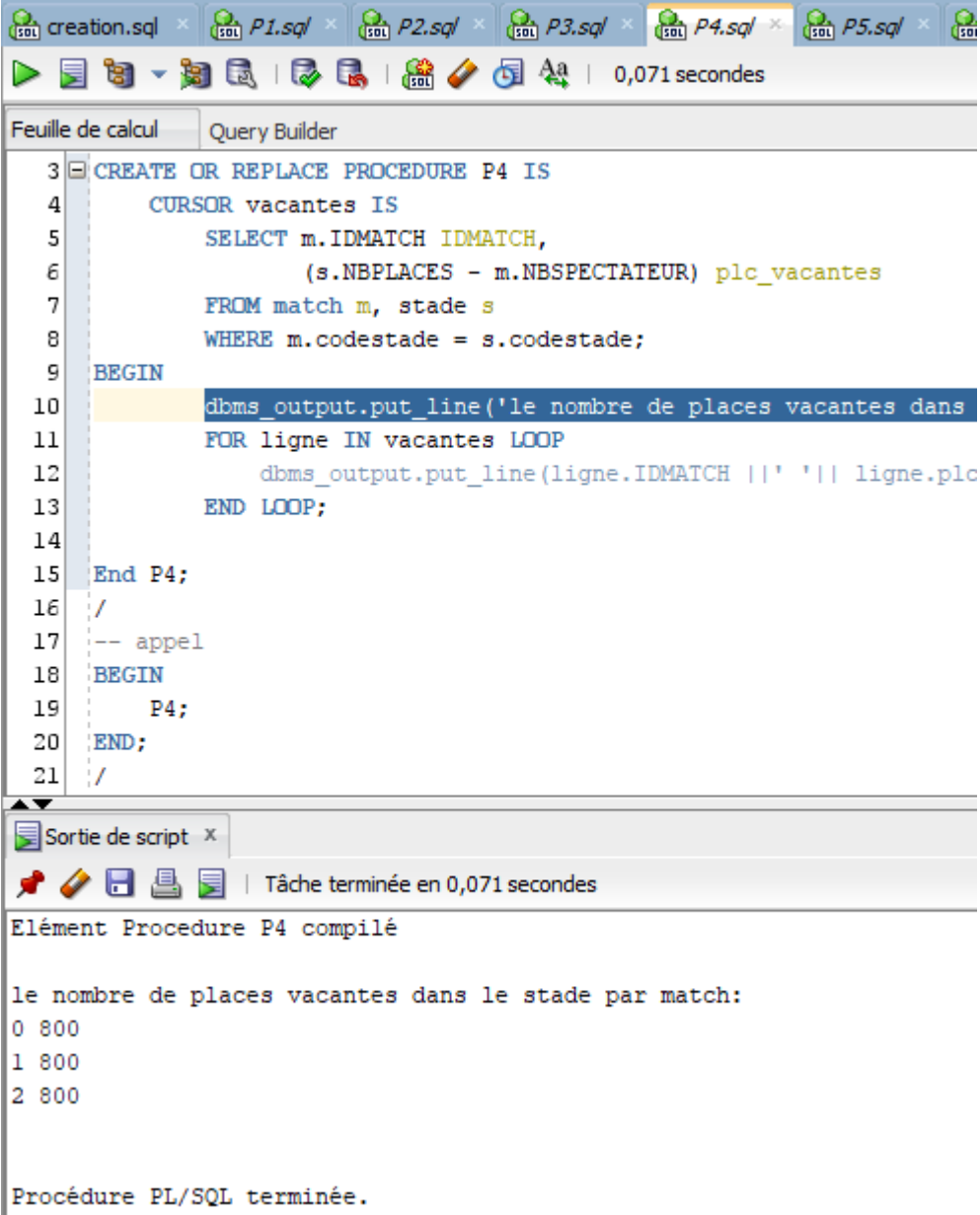
The screenshot displays the Oracle SQL Developer environment. The top toolbar shows various icons for file operations and execution. The main window is titled 'Feuille de calcul' and 'Query Builder'. The script editor contains the following PL/SQL code:

```
1 SET SERVEROUTPUT ON;
2
3 CREATE OR REPLACE PROCEDURE P3(N IN INTEGER) IS
4     CURSOR stades_N IS
5         SELECT NOMSTADE
6         FROM stade
7         WHERE NBPLACES > N;
8 BEGIN
9     dbms_output.put_line('les noms des stades qui ont pu accueillir pl
10     FOR ligne IN stades_N LOOP
11         dbms_output.put_line(ligne.NOMSTADE );
12     END LOOP;
13
14 End P3;
15 /
16 -- appel
17 BEGIN
18     P3(28);
19 END;
```

Below the script editor, the 'Sortie de script' window shows the execution results:

```
Element Procedure P3 compile
les noms des stades qui ont pu accueillir plus de N spectateurs:
Amerigo
Orange
Pricipaux
Procédure PL/SQL terminée.
```

#### 4. Afficher pour chaque match le nombre de places vacantes dans le stade.



The screenshot displays the Oracle SQL Developer environment. The top toolbar shows various icons for file operations and execution. The main window is titled 'Feuille de calcul' and 'Query Builder'. It contains a PL/SQL script for creating and executing a procedure named P4. The script defines a cursor 'vacantes' that selects match IDs and the number of vacant seats (calculated as total seats minus spectators) for each match. The procedure P4 iterates over this cursor and prints the results. The bottom window, titled 'Sortie de script', shows the execution output, indicating that the procedure was compiled and executed successfully, displaying the number of vacant seats for three matches.

```
3 CREATE OR REPLACE PROCEDURE P4 IS
4     CURSOR vacantes IS
5         SELECT m.IDMATCH IDMATCH,
6               (s.NBPLACES - m.NBSPECTATEUR) plc_vacantes
7         FROM match m, stade s
8        WHERE m.codestade = s.codestade;
9 BEGIN
10     dbms_output.put_line('le nombre de places vacantes dans
11     FOR ligne IN vacantes LOOP
12         dbms_output.put_line(ligne.IDMATCH || ' ' || ligne.plc
13     END LOOP;
14
15 End P4;
16 /
17 -- appel
18 BEGIN
19     P4;
20 END;
21 /
```

Sortie de script x | Tâche terminée en 0,071 secondes

Elément Procédure P4 compilé

le nombre de places vacantes dans le stade par match:

```
0 800
1 800
2 800
```

Procédure PL/SQL terminée.

## 5. Afficher tous les noms des joueurs marocains qui ont participé à la coupe mondiale.

```

1  SET SERVEROUTPUT ON;
2
3  CREATE OR REPLACE PROCEDURE P9 IS
4      mx_but INTEGER;
5      CURSOR kuKhlahalBoussoufa IS
6          SELECT i.nom , count(b.idbut) nbr_but
7          FROM individu i, but b, equipe e
8          WHERE i.noindividu = b.noindividu and e.codeequipe = i.codeequip
9                and e.nationalite = 'Maroc'
10         GROUP BY i.nom;
11 BEGIN
12     SELECT max(count(b.idbut)) INTO mx_but
13     FROM individu i, but b, equipe e
14     WHERE i.noindividu = b.noindividu and e.codeequipe = i.codeequip
15           and e.nationalite = 'Maroc'
16     GROUP BY i.nom;
17
18     dbms_output.put_line('les noms des joueurs marocains qui ont mar
19 FOR ligne IN kuKhlahalBoussoufa LOOP
20     if ligne.nbr_but = mx_but then
21         dbms_output.put_line(ligne.nom || ' a marqué ' || ligne.n
22     end if;
23 END LOOP;
24 End P9;
25 /
26
27 -- appel
28 BEGIN
29     P9;
30 END;
31 /
  
```



## 6. Afficher noms ordonnés des arbitres des matchs joués par une équipe donnée.

```

1  creation.sql x P1.sql x P2.sql x P3.sql x P4.sql x P5.sql x P6.sql x
2
3  CREATE OR REPLACE PROCEDURE P6(p_code IN equipe.codeequipe%TYPE) IS
4      CURSOR arbitres IS
5          SELECT i.NOM NOM
6          FROM individu i, jouer j, match m
7          WHERE j.codeequipe = p_code and i.noindividu = m.noindividu
8          ORDER BY 1;
9  BEGIN
10     dbms_output.put_line(' noms ordonnés des arbitres des matchs joués p
11     FOR ligne IN arbitres LOOP
12         dbms_output.put_line(ligne.NOM );
13     END LOOP;
14
15 End P6;
16 /
17
18 -- appel
19 BEGIN
20     P6(0);
21 END;
22 /
  
```

Sortie de script x

Tâche terminée en 0,06 secondes

Elément Procedure P6 compilé

```

noms ordonnés des arbitres des matchs joués par une équipe numéro :0
NA
NA
NA2
  
```

## 7. Afficher les noms des joueurs qui ont marqué au moins N buts d'un type donné.

```

2
3 CREATE OR REPLACE PROCEDURE P7(N IN INTEGER, p_type IN but.type%TYPE) IS
4     CURSOR buteur IS
5         SELECT i.NOM NOM, count(b.idbut) nbBut
6         FROM individu i, but b
7         WHERE i.noindividu = b.noindividu and b.type = p_type
8         GROUP BY i.NOM
9         ORDER BY i.nom;
10 BEGIN
11     dbms_output.put_line('les noms des joueurs qui ont marqué au moins N buts de type
12 FOR ligne IN buteur LOOP
13     if ligne.nbBut >= N then
14         dbms_output.put_line(ligne.NOM );
15     end if;
16 END LOOP;
17
18 End P7;
19 /
20
21 -- appel
22 BEGIN
23     P7(1, 'Tir');
24 END;
25 /

```

Sortie de script x Résultat de requête x

Tâche terminée en 0,058 secondes

Elément Procedure P7 compilé

```

les noms des joueurs qui ont marqué au moins N buts de type :Tir
N1
N2

```

## 8. Afficher pour une équipe le nombre de buts marqués.

```

1  SET SERVEROUTPUT ON;
2
3  CREATE OR REPLACE PROCEDURE P8(p_code IN equipe.codeequipe%TYPE) IS
4      CURSOR butParEquipe IS
5          SELECT e.codeequipe, e.nationalite, count(b.idbut) nbr_but
6          FROM individu i, but b, equipe e
7          WHERE i.noindividu = b.noindividu and e.codeequipe = i.codeequipe
8                and e.codeequipe = p_code
9          GROUP BY e.codeequipe, e.nationalite;
10 BEGIN
11     dbms_output.put_line('le nombre de buts marqués par l'équipe numéro : ' |
12     FOR ligne IN butParEquipe LOOP
13         dbms_output.put_line( 'Nom Equipe: ' || ligne.nationalite || ', Nom
14     END LOOP;
15
16 End P8;
17 /
18
19 -- appel
20 BEGIN
21     P8(0);
22 END;
23 /

```

Sortie de script x

Tâche terminée en 0,054 secondes

```

le nombre de buts marqués par l'équipe numéro : 0
Nom Equipe: Maroc\nNombre buts: 18

```

## 9. Afficher le nom du joueur marocain qui a marqué le maximum de buts.

```

1  SET SERVEROUTPUT ON;
2
3  CREATE OR REPLACE PROCEDURE P9 IS
4      mx_but INTEGER;
5      CURSOR kuKhlahalBoussoufa IS
6          SELECT i.nom , count(b.idbut) nbr_but
7          FROM individu i, but b, equipe e
8          WHERE i.noindividu = b.noindividu and e.codeequipe = i.codeequip
9                and e.nationalite = 'Maroc'
10         GROUP BY i.nom;
11 BEGIN
12     SELECT max(count(b.idbut)) INTO mx_but
13     FROM individu i, but b, equipe e
14     WHERE i.noindividu = b.noindividu and e.codeequipe = i.codeequip
15           and e.nationalite = 'Maroc'
16     GROUP BY i.nom;
17
18     dbms_output.put_line('les noms des joueurs marocains qui ont mar
19 FOR ligne IN kuKhlahalBoussoufa LOOP
20     if ligne.nbr_but = mx_but then
21         dbms_output.put_line(ligne.nom || ' a marqué ' || ligne.n
22     end if;
23 END LOOP;
24 End P9;
25 /
26
27 -- appel
28 BEGIN
29     P9;
30 END;
31 /
  
```

## 10. Ajouter à tous joueur marocain deux buts.

Avant

	IDBUT	MINUTE	TYPE	IDMATCH	NOINDIVIDU
1	1	15	Tir	0	1
2	2	23	Diving header	0	2
3	3	44	Freestyle	0	3
4	4	79	Long rage	0	7
5	5	90	Cross	0	9
6	6	17	Tir	0	1

creation.sql × P5.sql × P6.sql × P7.sql × P8.sql × P9.sql × P10.sql × BUT ×

0,063 secondes

Feuille de calcul Query Builder

```

1 SET SERVEROUTPUT ON;
2 CREATE OR REPLACE PROCEDURE P10 IS
3     CURSOR marocain IS -- id des joueurs
4         SELECT DISTINCT i.noindividu
5         FROM individu i, equipe e
6         WHERE i.codeequipe = e.codeequipe and e.nationalite = 'Maroc';
7     i INTEGER;
8     minut INTEGER := 2; -- fixer à la 2eme minute
9 BEGIN
10
11     SELECT count(*) into i
12     FROM but;
13     i := i + 1;
14
15     FOR ligne IN marocain LOOP
16         FOR fois IN 1..2 LOOP
17             INSERT INTO BUT VALUES (i, minut, 'Tir', 0, ligne.noindividu);
18             i := i + 1;
19             minut := minut + 1;
20         END LOOP;
21     END LOOP;
22
23 End P10;
24 /
25 -- appel
26 BEGIN
27     P10;
28 END;
29 /

```

Sortie de script ×

Tâche terminée en 0,063 secondes

Procédure PL/SQL terminée.

Après

IDBUT	MINUTE	TYPE	IDMATCH	NOINDIVIDU
1	1	15 Tir	0	1
2	2	23 Diving header	0	2
3	3	44 Freestyle	0	3
4	4	79 Long rage	0	7
5	5	90 Cross	0	9
6	6	17 Tir	0	1
7	7	2 Tir	0	1
8	8	3 Tir	0	1
9	9	4 Tir	0	6
10	10	5 Tir	0	6
11	11	6 Tir	0	2
12	12	7 Tir	0	2
13	13	8 Tir	0	4
14	14	9 Tir	0	4
15	15	10 Tir	0	5
16	16	11 Tir	0	5
17	17	12 Tir	0	3
18	18	13 Tir	0	3
19	19	14 Tir	0	0
20	20	15 Tir	0	0

## B. Les fonctions :

1. Ecrire une fonction qui prend en paramètre un numéro stade et qui renvoie une chaîne de caractère contenant le nom du stade et le nombre de match joués sur ce stade.

The screenshot displays the Oracle SQL Developer environment. The top pane shows the 'Query Builder' tab with the following SQL code:

```

1 SET SERVEROUTPUT ON;
2 CREATE OR REPLACE FUNCTION F1(N IN stade.codestade%TYPE) RETURN VARCHAR2 IS
3     nom VARCHAR(20);
4     nbr_match INTEGER;
5 BEGIN
6
7     SELECT s.nomstade, count(m.idmatch) INTO NOM, nbr_match
8     FROM match m, stade s
9     WHERE s.codestade = N AND m.codestade = s.codestade
10    GROUP BY s.nomstade;
11    RETURN ('Nom stade est: ' || nom || ' ET nombre de matche joué est ' || nbr_match);
12 End F1;
13 /
14 -- appel
15 BEGIN
16     DBMS_OUTPUT.put_line(F1(1));
17 END;
18 /
  
```

The bottom pane shows the 'Résultat de requête' (Query Result) tab, indicating that the task was completed in 0,083 seconds. The output of the function call is displayed as follows:

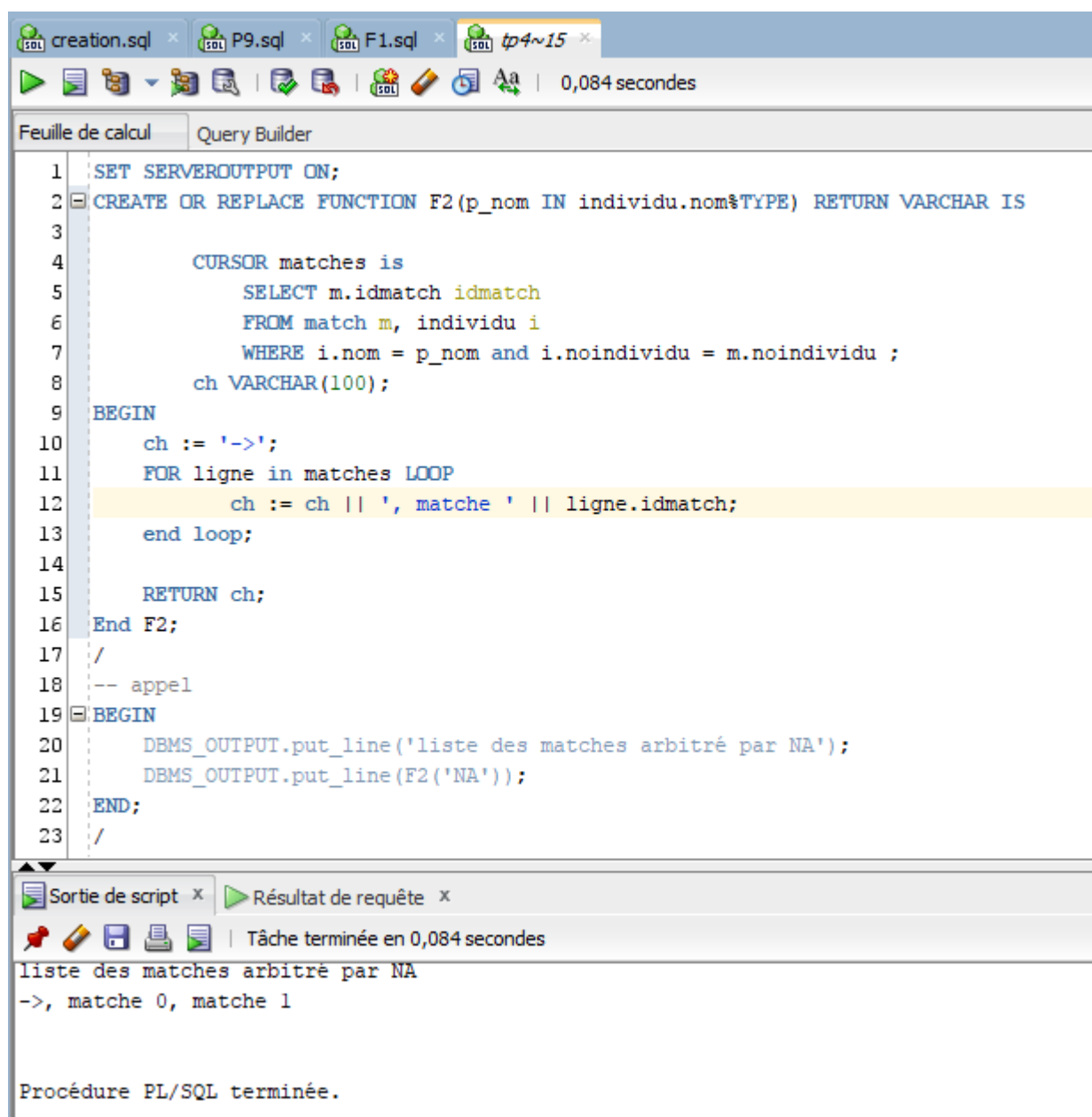
```

Elément Function F1 compilé

Nom stade est: Orange ET nombre de matche joué est 2

Procédure PL/SQL terminée.
  
```

2. Ecrire une fonction qui prend en entrée le nom d'un arbitre et produit une chaîne contenant les matches qu'il arbitre.



```

1  SET SERVEROUTPUT ON;
2  CREATE OR REPLACE FUNCTION F2(p_nom IN individu.nom%TYPE) RETURN VARCHAR IS
3
4      CURSOR matches is
5          SELECT m.idmatch idmatch
6          FROM match m, individu i
7          WHERE i.nom = p_nom and i.noindividu = m.noindividu ;
8      ch VARCHAR(100);
9  BEGIN
10     ch := '->';
11     FOR ligne in matches LOOP
12         ch := ch || ', matche ' || ligne.idmatch;
13     end loop;
14
15     RETURN ch;
16 End F2;
17 /
18 -- appel
19 BEGIN
20     DBMS_OUTPUT.put_line('liste des matches arbitré par NA');
21     DBMS_OUTPUT.put_line(F2('NA'));
22 END;
23 /

```

Sortie de script x    Résultat de requête x

Tâche terminée en 0,084 secondes

```

liste des matches arbitré par NA
->, matche 0, matche 1

Procédure PL/SQL terminée.

```

## C. Les triggers :

1. Lors de l'ajout d'un individu, on met son nom en majuscule ainsi que la première lettre de son prénom.

The screenshot shows the SQL Developer interface. On the left, the 'Query Builder' window displays the following SQL code for creating trigger T1:

```

1 Create or replace trigger T1
2 BEFORE INSERT
3 on INDIVIDU
4 for each row
5 Begin
6     If inserting then
7         :new.nom := UPPER(:new.nom);
8         :new.prenom := INITCAP(:new.prenom);
9     End if;
10 End T1;
11 /
12
13 -- TEST
14 BEGIN
15     insert into Individu values(16,'zekkouri','hassan', null);
16 END;
17 /
18
19

```

On the right, the 'INDIVIDU' table is displayed with the following data:

	NOINDIVIDU	NOM	PRENOM	CODEEQUIPE
1	0	N1	P1	0
2	1	N2	P2	0
3	2	N3	P3	0
4	3	N4	P4	0
5	4	N5	P5	0
6	5	N6	P6	0
7	6	N7	P7	0
8	7	N8	P8	1
9	8	N9	P9	1
10	9	N10	P10	1
11	10	N11	P11	1
12	11	N12	P12	1
13	12	N13	P13	1
14	13	N14	P14	1
15	14	NA	PA	(null)
16	15	NA2	PA2	(null)
17	16	ZEKKOURI	Hassan	(null)

A red arrow points from the 'END;' line of the SQL code to the new row (16) in the table, indicating the result of the trigger execution.

2. Lors de l'ajout d'un match on vérifie si le nombre de spectateurs est inférieur ou égal au nombre de places disponibles dans le stade. Si oui on l'ajoute sinon on affiche un message d'erreur.

The screenshot shows the SQL Developer interface. On the left, the 'Query Builder' window displays the following SQL code for creating trigger T2:

```

1 Create or replace trigger T2
2 BEFORE INSERT on match for each row
3 DECLARE
4     v_nbplaces INTEGER;
5 Begin
6     SELECT nbplaces into v_nbplaces
7     FROM stade
8     WHERE codestade = :new.codestade;
9
10    if :new.nbspectateur > v_nbplaces then
11        RAISE_APPLICATION_ERROR(-20001,'Le nombre de spectateurs est supérieur au nombre de places disponibles dans le stade!');
12    end if;
13 End T2;
14 /
15 -- TEST
16 BEGIN
17     insert into Match values(3,2000,'23/07/2003',14,0);
18 END;

```

Below the code, the 'Sortie de script' window shows the following output:

```

Elément Trigger T2 compilé

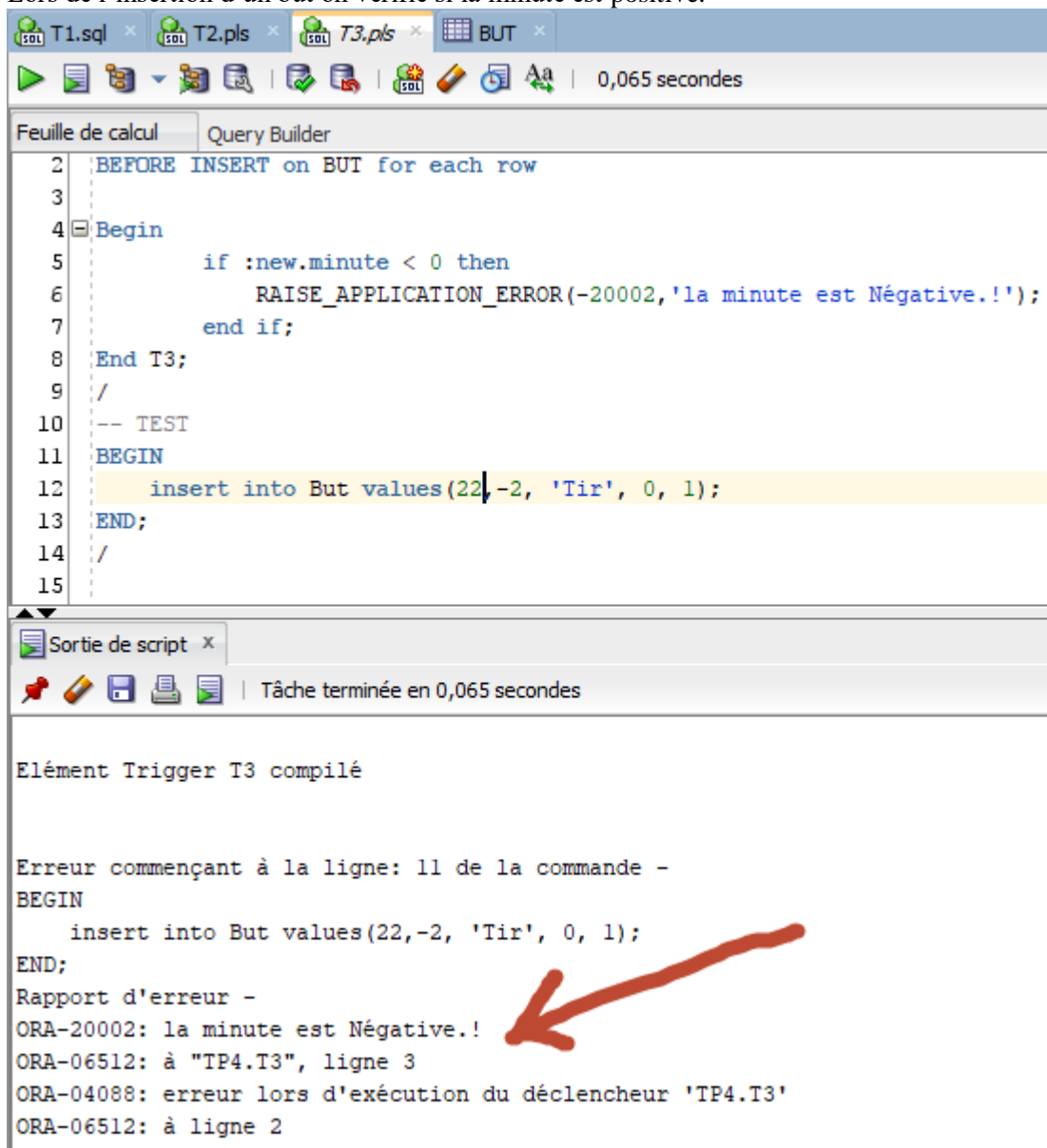
Erreur commençant à la ligne: 16 de la commande -
BEGIN
    insert into Match values(3,2000,'23/07/2003',14,0);
END;
Rapport d'erreur -
ORA-20001: Le nombre de spectateurs est supérieur au nombre de places disponibles dans le stade!
ORA-06512: à "TP4.T2", ligne 9
lors d'exécution du déclencheur 'TP4.T2'

```

A red arrow points from the error message to the 'END;' line of the SQL code, indicating the point of failure.



### 3. Lors de l'insertion d'un but on vérifie si la minute est positive.



The screenshot shows the SQL Developer interface. The top pane displays a PL/SQL script in the 'Query Builder' tab. The script is a trigger for the 'BUT' table, designed to check if the 'minute' column is negative before inserting a new row. The script includes a 'BEGIN' block with an 'if' statement that raises an application error if the minute is less than 0. A test block follows, starting with 'BEGIN' and containing an 'insert into But values(22,-2, 'Tir', 0, 1);' statement. The bottom pane shows the 'Sortie de script' (Script Output) window, which reports that the trigger 'TP4.T3' was compiled successfully. However, it also displays an error message: 'ORA-20002: la minute est Négative.!' (ORA-20002: the minute is negative!). A red arrow points from this error message to the corresponding line in the script above.

```
2 BEFORE INSERT on BUT for each row
3
4 BEGIN
5     if :new.minute < 0 then
6         RAISE_APPLICATION_ERROR(-20002,'la minute est Négative.!!');
7     end if;
8 End T3;
9 /
10 -- TEST
11 BEGIN
12     insert into But values(22,-2, 'Tir', 0, 1);
13 END;
14 /
15
```

Sortie de script x

Tâche terminée en 0,065 secondes

Elément Trigger T3 compilé

Erreur commençant à la ligne: 11 de la commande -  
BEGIN  
 insert into But values(22,-2, 'Tir', 0, 1);  
END;  
Rapport d'erreur -  
ORA-20002: la minute est Négative.!  
ORA-06512: à "TP4.T3", ligne 3  
ORA-04088: erreur lors d'exécution du déclencheur 'TP4.T3'  
ORA-06512: à ligne 2

4. Lors de la suppression d'un joueur, on supprime aussi les buts qu'il a marqués.

T1.sql x T2.pls x T3.pls x BUT x tp4~20 x

Colonnes | Données | Model | Contraintes | Droits | Statistiques | Déclencheurs | Flashback | Dépenda

Trier... | Filtre :

	IDBUT	MINUTE	TYPE	IDMATCH	NOINDIVIDU
1	1	15	Tir	0	1
2	2	23	Diving header	0	2
3	3	44	Freestyle	0	3
4	4	79	Long rage	0	7
5	5	90	Cross	0	9
6	6	17	Tir	0	1
7	7	2	Tir	0	1
8	8	3	Tir	0	1
9	9	4	Tir	0	6
10	10	5	Tir	0	6
11	11	6	Tir	0	2
12	12	7	Tir	0	2
13	13	8	Tir	0	4
14	14	9	Tir	0	4
15	15	10	Tir	0	5
16	16	11	Tir	0	5
17	17	12	Tir	0	3
18	18	13	Tir	0	3
19	19	14	Tir	0	0
20	20	15	Tir	0	0
21	21	2	Tir	0	1

Feuille de calcul Query Builder

```

1 Create or replace trigger T4
2 AFTER DELETE on individu for each row
3 Begin
4     DELETE FROM but
5     WHERE noindividu = :old.noindividu;
6 End T4;
7 /
8 -- TEST
9 BEGIN
10    DELETE FROM INDIVIDU
11    WHERE noindividu = 1;
12 END;
13 /
14

```

Sortie de script x

Tâche terminée en 0,038 secondes

Elément Trigger T4 compilé

Procédure PL/SQL terminée.

Colonnes Données Model Contraintes Droits Statistiques Déclencheurs Flashback Dépendances Détails Partitions Index SQL

Trier... Filtre :

	IDBUT	MINUTE	TYPE	IDMATCH	NOINDIVIDU
1	2	23	Diving header	0	2
2	3	44	Freestyle	0	3
3	4	79	Long rage	0	7
4	5	90	Cross	0	9
5	9	4	Tir	0	6
6	10	5	Tir	0	6
7	11	6	Tir	0	2
8	12	7	Tir	0	2
9	13	8	Tir	0	4
10	14	9	Tir	0	4
11	15	10	Tir	0	5
12	16	11	Tir	0	5
13	17	12	Tir	0	3
14	18	13	Tir	0	3
15	19	14	Tir	0	0
16	20	15	Tir	0	0

On remarque que les buts de l'individu 1 sont supprimés

## 5. Interdire de modifier la date d'un match.

The screenshot shows the Oracle SQL Developer interface. The top toolbar includes icons for running, saving, and other database operations. The main window displays a script with the following SQL code:

```

1 Create or replace trigger T5
2 BEFORE UPDATE OF data on MATCH for each row -- data = date
3 BEGIN
4     RAISE_APPLICATION_ERROR(-20003, 'Impossible de modifier la date d'un match!');
5 End T5;
6 /
7 -- TEST
8 BEGIN
9     UPDATE MATCH
10    SET data = '31/12/19'
11    WHERE idmatch = 1;
12 END;
13 /
14

```

Below the script editor, the 'Sortie de script' (Script Output) window shows the execution results:

```

Erreur commençant à la ligne: 8 de la commande -
BEGIN
    UPDATE MATCH
    SET data = '31/12/19'
    WHERE idmatch = 1;
END;
Rapport d'erreur -
ORA-20003: Impossible de modifier la date d'un match!
ORA-06512: à "TP4.T5", ligne 2
ORA-04088: erreur lors d'exécution du déclencheur 'TP4.T5'
ORA-06512: à ligne 2

```

The error messages are highlighted with a red rectangle, indicating the failure of the trigger test.

## 6. Lors de la suppression d'une équipe on supprime les joueurs.

INDIVIDU

	NOINDIVIDU	NOM	PRENOM	CODEEQUIPE
1	0	N1	P1	0
2	2	N3	P3	0
3	3	N4	P4	0
4	4	N5	P5	0
5	5	N6	P6	0
6	6	N7	P7	0
7	7	N8	P8	1
8	8	N9	P9	1
9	9	N10	P10	1
10	10	N11	P11	1
11	11	N12	P12	1
12	12	N13	P13	1
13	13	N14	P14	1
14	14	NA	PA	(null)
15	15	NA2	PA2	(null)
16	16	ZEKKOURI	Hassan	(null)

Feuille de calcul Query Builder

```

5      WHERE codeequipe = :old.codeequipe;
6
7      -- pour eviter la violation de contrainte d'intégrité
8      -- on supprime sur la table jouer aussi
9      DELETE FROM jouer
10     WHERE codeequipe = :old.codeequipe;
11 End T6;
12 /
13 -- TEST
14 BEGIN
15     DELETE FROM equipe
16     WHERE codeequipe = 1;
17 END;
18 /
19

```

Sortie de script x

Tâche terminée en 0,048 secondes

Elément Trigger T6 compilé

Procédure PL/SQL terminée.

	NOINDIVIDU	NOM	PRENOM	CODEEQUIPE
1	0	N1	P1	0
2	2	N3	P3	0
3	3	N4	P4	0
4	4	N5	P5	0
5	5	N6	P6	0
6	6	N7	P7	0
7	14	NA	PA	(null)
8	15	NA2	PA2	(null)
9	16	ZEKKOURI	Hassan	(null)

Les joueurs de  
l'équipes 1 ont été  
supprimés

7. Lors de la suppression d'une équipe on supprime les joueurs et les buts.  
C'est déjà géré par les deux trigger T6 et T4  
Si on supprime l'équipe 0 les but qui reste seront supprimé avec les joueurs :

```

Feuille de calcul  Query Builder
1 Create or replace trigger T6
2 BEFORE DELETE on equipe for each row
3 Begin
4     DELETE FROM individu
5     WHERE codeequipe = :old.codeequipe;
6
7     -- pour eviter la violation de contrainte d'in
8     -- on supprime sur la table jouer aussi
9     DELETE FROM jouer
10    WHERE codeequipe = :old.codeequipe;
11 End T6;
12 /
13 -- TEST
14 BEGIN
15     DELETE FROM equipe |
16     WHERE codeequipe = 0;
17 END;
18 /
19
Sortie de script x
Tâche terminée en 0,093 secondes

Procédure PL/SQL terminée.

```

IDBUT	MINUTE	TYPE	IDMATCH	NOINDIVIDU
-------	--------	------	---------	------------

	NOINDIVIDU	NOM	PRENOM	CODEEQUIPE
1	14	NA	PA	(null)
2	15	NA2	PA2	(null)
3	16	ZEKKOURI	Hassan	(null)

8. Lors de l'ajout d'une équipe si on essaie d'insérer dans le champ nationalité :
- 'Fr' alors on la modifie en : 'France'
  - 'Mr' alors on la modifie en : 'Maroc'
  - 'Br' alors on la modifie en : 'Brésil'

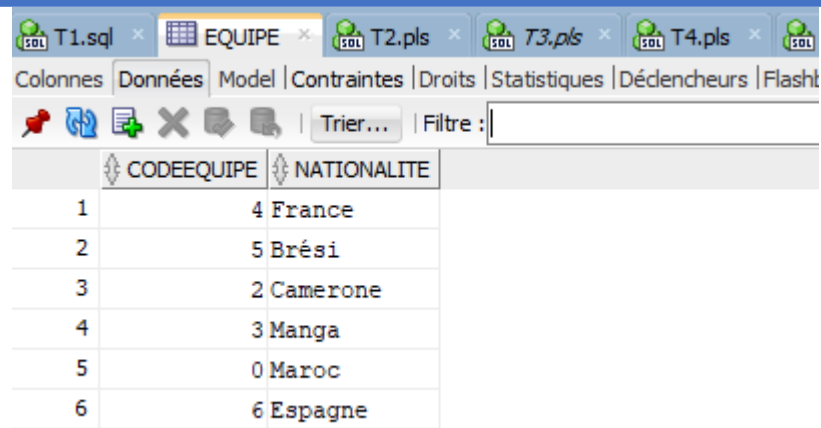
```

1 Create or replace trigger T8
2 BEFORE INSERT on equipe for each row
3 Begin
4     if :new.nationalite = 'Mr' then
5         :new.nationalite := 'Maroc';
6     elsif :new.nationalite = 'Fr' then
7         :new.nationalite := 'France';
8     elsif :new.nationalite = 'Br' then
9         :new.nationalite := 'Brési';
10    elsif :new.nationalite = 'Esp' then
11        :new.nationalite := 'Espagne';
12    end if;
13 End T8;
14 /
15 -- TEST
16 BEGIN
17     insert into Equipe values(0,'Mr');
18     insert into Equipe values(4,'Fr');
19     insert into Equipe values(5,'Br');
20     insert into Equipe values(6,'Esp');
21 END;
22 /
23
Sortie de script x
Tâche terminée en 0,111 secondes

Elément Trigger T8 compilé

Procédure PL/SQL terminée.

```



	CODEEQUIPE	NATIONALITE
1	4	France
2	5	Brési
3	2	Camerone
4	3	Manga
5	0	Maroc
6	6	Espagne

# FIN

Merci pour votre lecture !