



ENSA Agadir

GINFO2

Compte Rendu TP N°9:

Fonctions et procédures stockées, Triggers & Packages

PL/SQL

EL IDRISSE NOHAILA

Encadrant : M. Aksasse

Decembre 2024

Introduction :

Dans ce TP, nous allons explorer les fonctionnalités avancées de PL/SQL, un langage procédural étroitement intégré au SQL pour les bases de données Oracle.

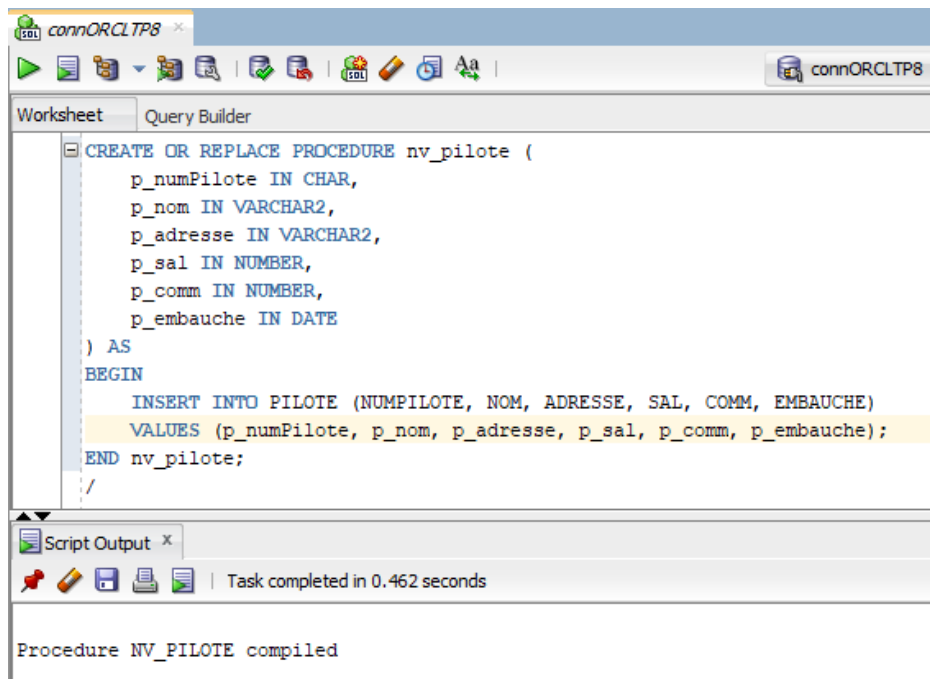
Le travail consiste à manipuler des procédures stockées, des fonctions, des triggers, des packages et des transactions. Ces outils permettent d'optimiser les opérations sur la base de données, de garantir l'intégrité des données, et d'automatiser certaines actions.

1 Procédures Stockées

Une procédure stockée est un bloc de code PL/SQL stocké dans la base de données, qui exécute une série d'opérations prédéfinies. Elle peut accepter des paramètres en entrée/sortie et est appelée pour automatiser des tâches complexes.

1.1 Créer la procédure nv_pilote

Cette procédure permettra d'ajouter un nouveau pilote dans la table PILOTE.



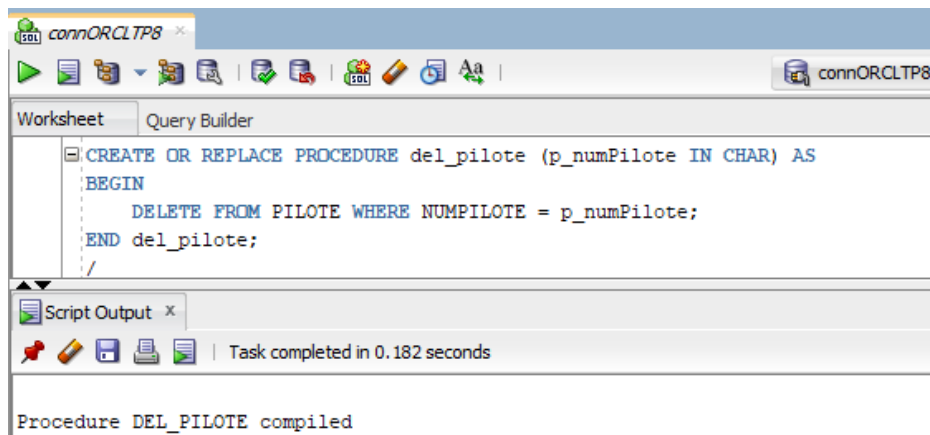
The screenshot shows the SQL Developer interface with a worksheet titled 'connORCLTP8'. The 'Query Builder' tab is active, displaying the following PL/SQL code:

```
CREATE OR REPLACE PROCEDURE nv_pilote (  
    p_numPilote IN CHAR,  
    p_nom IN VARCHAR2,  
    p_adresse IN VARCHAR2,  
    p_sal IN NUMBER,  
    p_comm IN NUMBER,  
    p_embauche IN DATE  
) AS  
BEGIN  
    INSERT INTO PILOTE (NUMPILOTE, NOM, ADRESSE, SAL, COMM, EMBAUCHE)  
    VALUES (p_numPilote, p_nom, p_adresse, p_sal, p_comm, p_embauche);  
END nv_pilote;  
/  
/
```

Below the code editor, the 'Script Output' window shows the message: 'Task completed in 0.462 seconds' and 'Procedure NV_PILOTE compiled'.

1.2 Créer la procédure del_pilote

Cette procédure supprime un pilote à partir de son numéro.



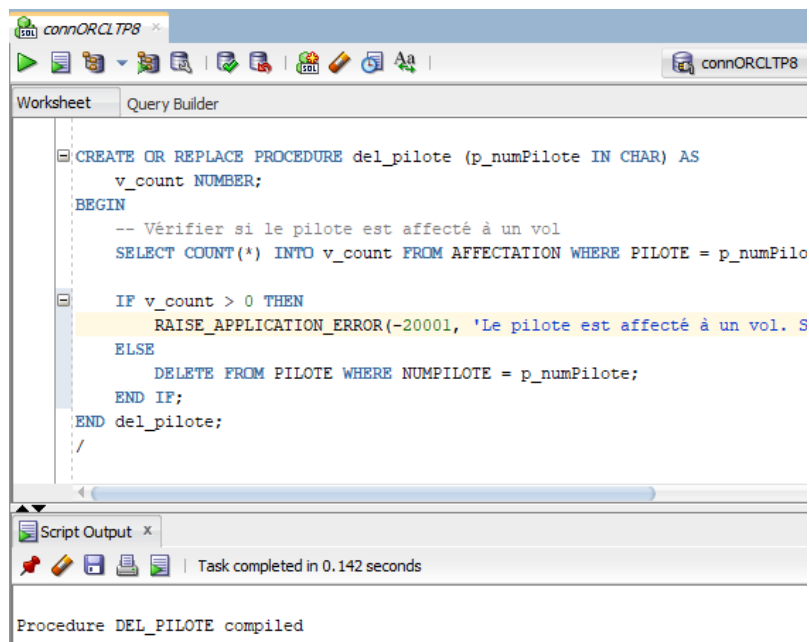
The screenshot shows the SQL Developer interface with a worksheet titled 'connORCLTP8'. The 'Query Builder' tab is active, displaying the following PL/SQL code:

```
CREATE OR REPLACE PROCEDURE del_pilote (p_numPilote IN CHAR) AS  
BEGIN  
    DELETE FROM PILOTE WHERE NUMPILOTE = p_numPilote;  
END del_pilote;  
/  
/
```

Below the code editor, the 'Script Output' window shows the message: 'Task completed in 0.182 seconds' and 'Procedure DEL_PILOTE compiled'.

1.3 Modifier la procédure del_pilote pour vérifier l'affectation du pilote

Avant de supprimer un pilote, on vérifie s'il est affecté à un vol.



The screenshot shows the Oracle SQL Developer interface with the 'Query Builder' tab selected. The main window displays the following PL/SQL code for the 'del_pilote' procedure:

```
CREATE OR REPLACE PROCEDURE del_pilote (p_numPilote IN CHAR) AS
v_count NUMBER;
BEGIN
  -- Vérifier si le pilote est affecté à un vol
  SELECT COUNT(*) INTO v_count FROM AFFECTATION WHERE PILOTE = p_numPilo

  IF v_count > 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Le pilote est affecté à un vol. S
  ELSE
    DELETE FROM PILOTE WHERE NUMPILOTE = p_numPilote;
  END IF;
END del_pilote;
/
```

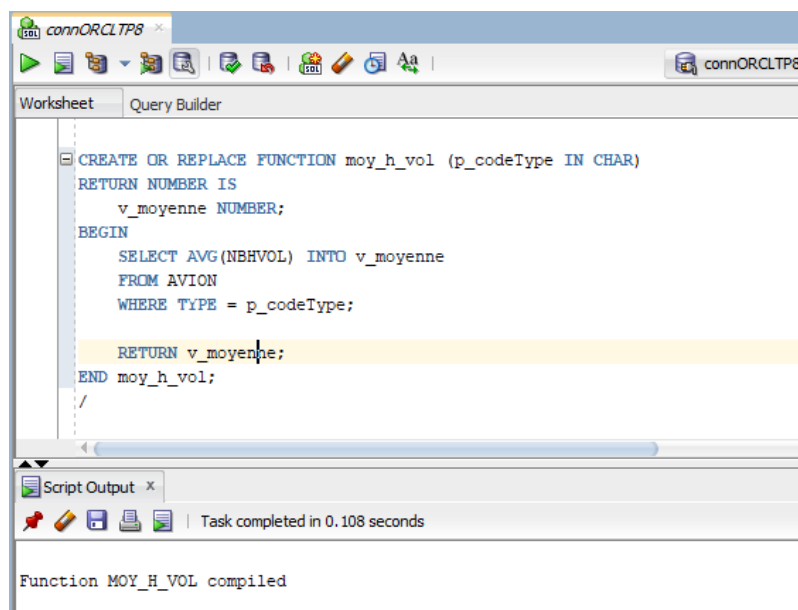
The 'Script Output' window at the bottom shows the message: 'Task completed in 0.142 seconds' and 'Procedure DEL_PILOTE compiled'.

2 Fonctions Stockées

Une fonction est similaire à une procédure, mais elle retourne toujours une valeur. Elle est souvent utilisée pour encapsuler des calculs ou des traitements qui nécessitent un résultat spécifique.

2.1 Créer la fonction moy_h_vol

Cette fonction calcule le nombre moyen d'heures de vol des avions appartenant à une famille donnée.



The screenshot shows the Oracle SQL Developer interface with the 'Query Builder' tab selected. The main window displays the following PL/SQL code for the 'moy_h_vol' function:

```
CREATE OR REPLACE FUNCTION moy_h_vol (p_codeType IN CHAR)
RETURN NUMBER IS
v_moyenne NUMBER;
BEGIN
  SELECT AVG(NBHVOL) INTO v_moyenne
  FROM AVION
  WHERE TYPE = p_codeType;

  RETURN v_moyenne;
END moy_h_vol;
/
```

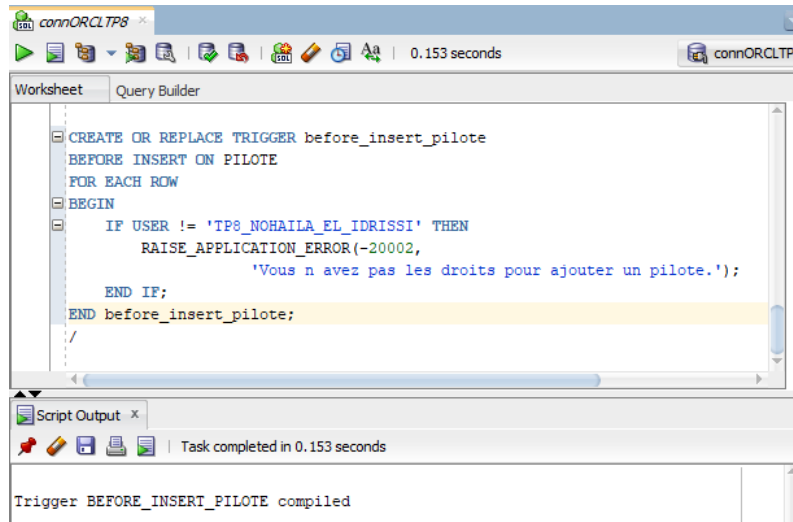
The 'Script Output' window at the bottom shows the message: 'Task completed in 0.108 seconds' and 'Function MOY_H_VOL compiled'.

3 Triggers

Un trigger est un programme PL/SQL associé à une table ou à une vue, qui s'exécute automatiquement en réponse à des événements comme INSERT, UPDATE, ou DELETE. Il permet d'appliquer des règles métier ou de maintenir l'intégrité des données.

3.1 Créer un trigger "Before" pour limiter l'ajout de nouveaux pilotes

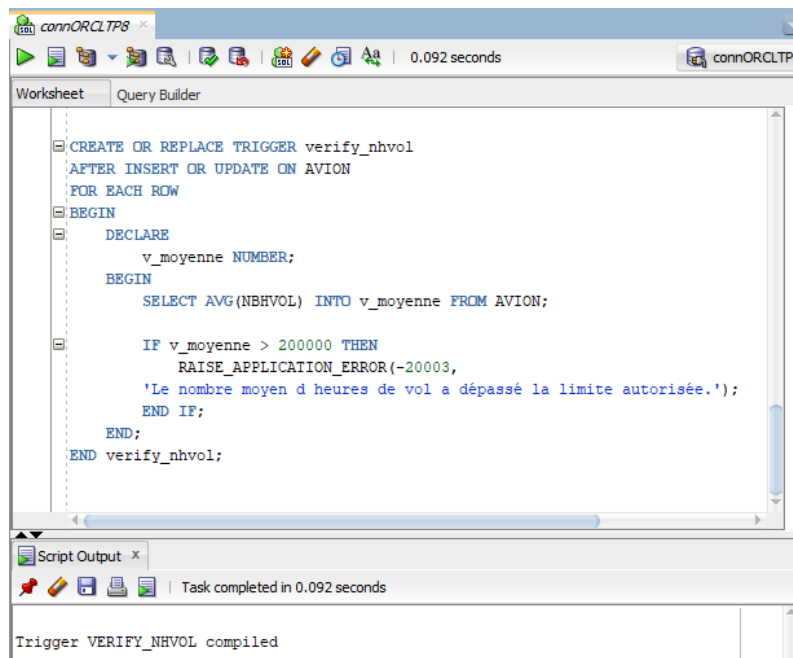
Ce trigger empêche l'ajout d'un pilote sauf pour un utilisateur spécifique.



```
CREATE OR REPLACE TRIGGER before_insert_pilote
BEFORE INSERT ON PILOTE
FOR EACH ROW
BEGIN
    IF USER != 'TP8_NOHAILA_EL_IDRISSI' THEN
        RAISE_APPLICATION_ERROR(-20002,
            'Vous n avez pas les droits pour ajouter un pilote.');
```

3.2 Créer un trigger "After" pour vérifier le nombre d'heures de vol

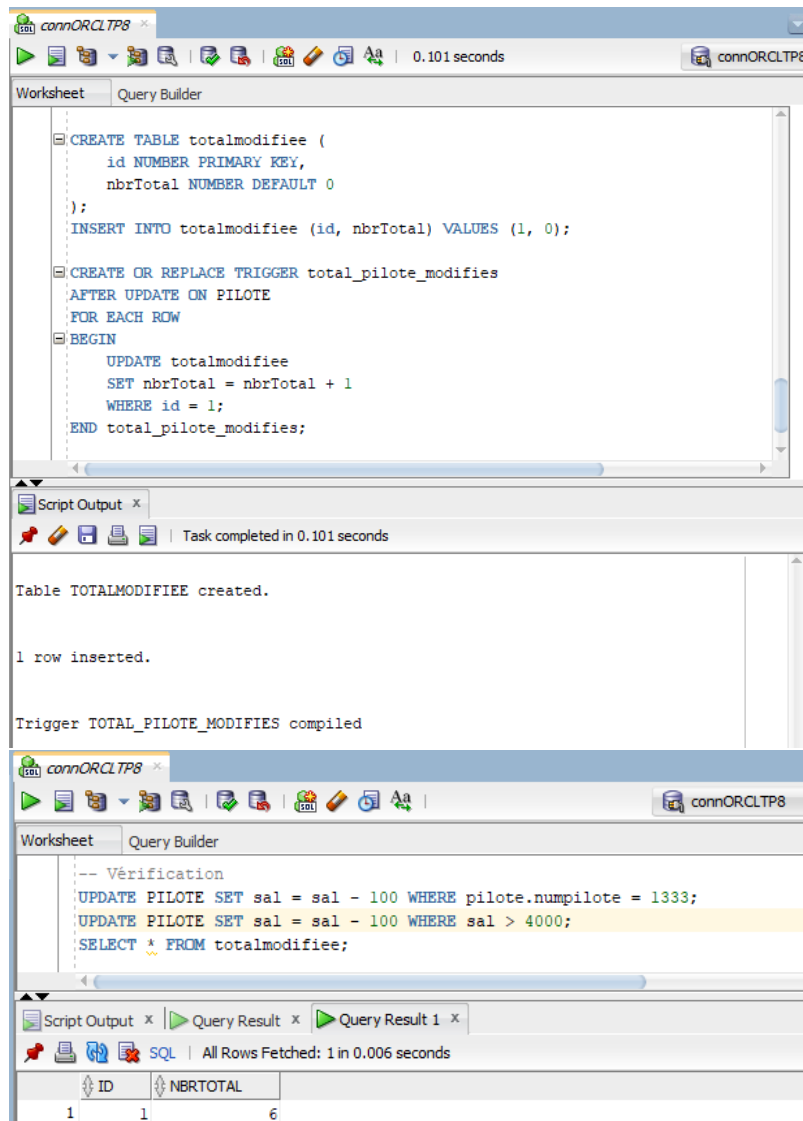
Ce trigger vérifie que le nombre moyen d'heures de vol des avions ne dépasse pas 200000 après une mise à jour ou une insertion.



```
CREATE OR REPLACE TRIGGER verify_nhvol
AFTER INSERT OR UPDATE ON AVION
FOR EACH ROW
BEGIN
    DECLARE
        v_moyenne NUMBER;
```

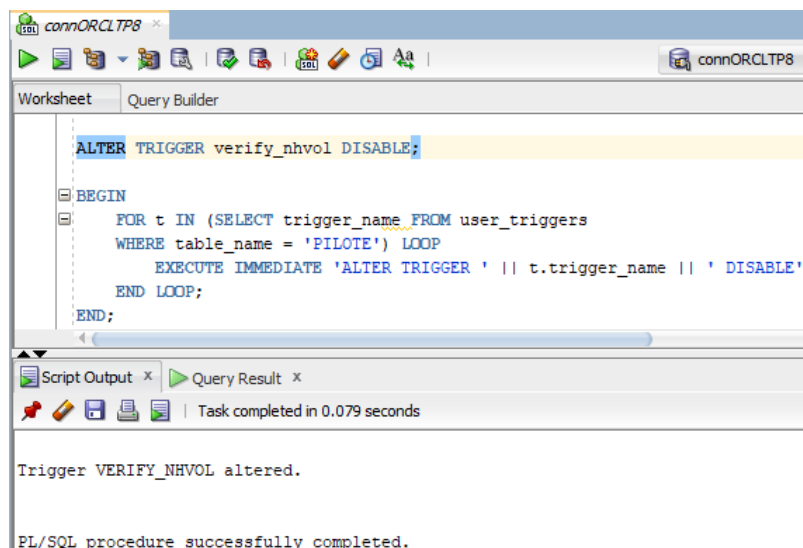
3.3 Créer un trigger pour comptabiliser le nombre de lignes modifiées

Ce trigger comptabilise le nombre de lignes modifiées dans la table PILOTE.



3.4 Désactiver un trigger

Voici comment désactiver un trigger spécifique ou tous les triggers d'une table.

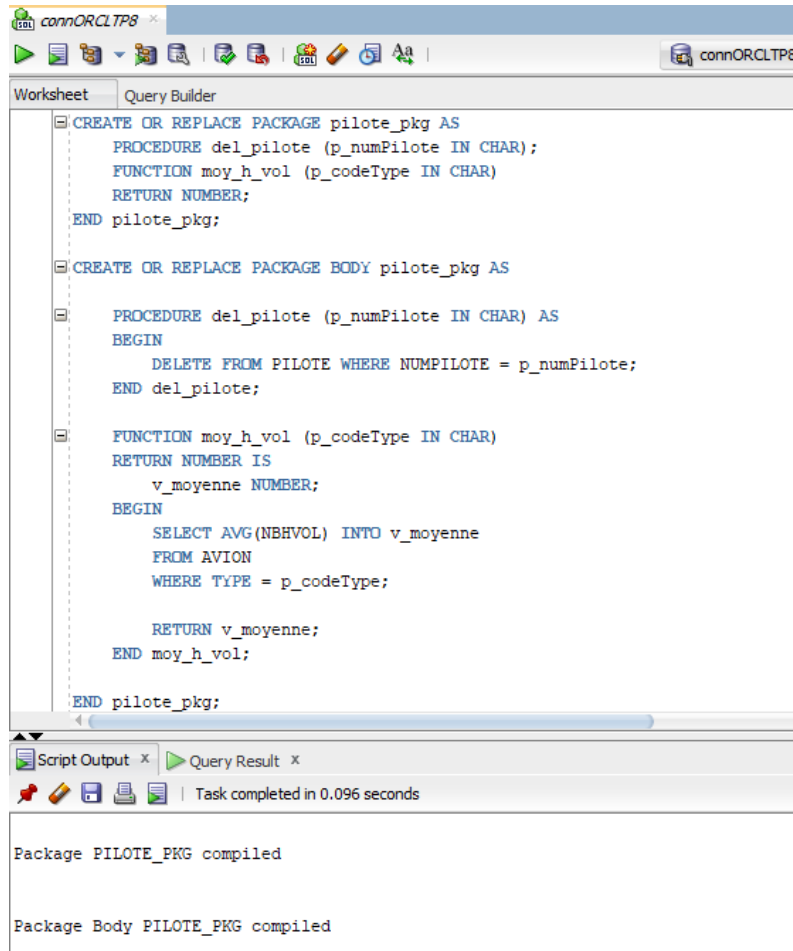


4 Packages

Un package est un conteneur qui regroupe plusieurs procédures, fonctions, variables, et autres éléments PL/SQL dans une seule unité logique. Il facilite l'organisation et la réutilisation du code.

4.1 Créer un package

Le package contiendra des procédures, des fonctions et des traitements d'erreurs pour supprimer un pilote et obtenir le nombre moyen d'heures de vol.



The screenshot shows the Oracle SQL Developer interface with a window titled 'connORCLTP8'. The 'Query Builder' tab is active, displaying the following SQL code:

```
CREATE OR REPLACE PACKAGE pilote_pkg AS
  PROCEDURE del_pilote (p_numPilote IN CHAR);
  FUNCTION moy_h_vol (p_codeType IN CHAR)
  RETURN NUMBER;
END pilote_pkg;

CREATE OR REPLACE PACKAGE BODY pilote_pkg AS

  PROCEDURE del_pilote (p_numPilote IN CHAR) AS
  BEGIN
    DELETE FROM PILOTE WHERE NUMPILOTE = p_numPilote;
  END del_pilote;

  FUNCTION moy_h_vol (p_codeType IN CHAR)
  RETURN NUMBER IS
    v_moyenne NUMBER;
  BEGIN
    SELECT AVG(NBHVOL) INTO v_moyenne
    FROM AVION
    WHERE TYPE = p_codeType;

    RETURN v_moyenne;
  END moy_h_vol;

END pilote_pkg;
```

Below the code editor, the 'Script Output' tab shows the execution results:

```
Package PILOTE_PKG compiled

Package Body PILOTE_PKG compiled
```

A status bar at the bottom indicates 'Task completed in 0.096 seconds'.

5 Transactions

Une transaction est un ensemble d'opérations SQL exécutées de manière atomique. Elle garantit que toutes les opérations s'exécutent avec succès ou qu'aucune d'elles ne prend effet en cas d'échec. Les commandes principales incluent COMMIT, ROLLBACK et SAVEPOINT.

5.1 Visualiser l'effet d'une transaction

1. Insertion de NOHAILA :

- Le pilote NOHAILA a été inséré et validé par un COMMIT avant la création du second point de sauvegarde (savepoint2).
- Une fois le COMMIT effectué, cette modification devient permanente dans la base de données.
- À ce stade, savepoint1 est invalide car un COMMIT invalide tous les points de sauvegarde existants.

The screenshot shows the SQL Developer interface with a script titled 'connORCLTP8'. The script contains the following SQL commands:

```

SAVEPOINT savepoint1;

INSERT INTO pilote
VALUES ('7584', 'NOHAILA', 'AGADIR', 8000, 2000, '12-DEC-2021');

select * from pilote where numpilote = 7584;

commit;

```

The execution output shows the following messages:

```

Savepoint created.

1 row inserted.

>>Query Run In:Query Result

Commit complete.

```

Below the output, a table displays the query results:

	NUMPILOTE	NOM	ADRESSE	SAL	COMM	EMBAUCHE
1	7584	NOHAILA	AGADIR	8000	2000	12-DEC-21

2. Insertion de ELIDRISSI :

- Le pilote ELIDRISSI a été inséré après la création de savepoint2.

The screenshot shows the SQL Developer interface with a script titled 'connORCLTP8'. The script contains the following SQL commands:

```

savepoint savepoint2;

INSERT INTO pilote
VALUES ('5555', 'ELIDRISSI', 'OUARZAZATE', 5000, 100, '30-OCT-2020');

select * from pilote where numpilote = 5555;

```

The execution output shows the following messages:

```

Savepoint created.

1 row inserted.

>>Query Run In:Query Result 1

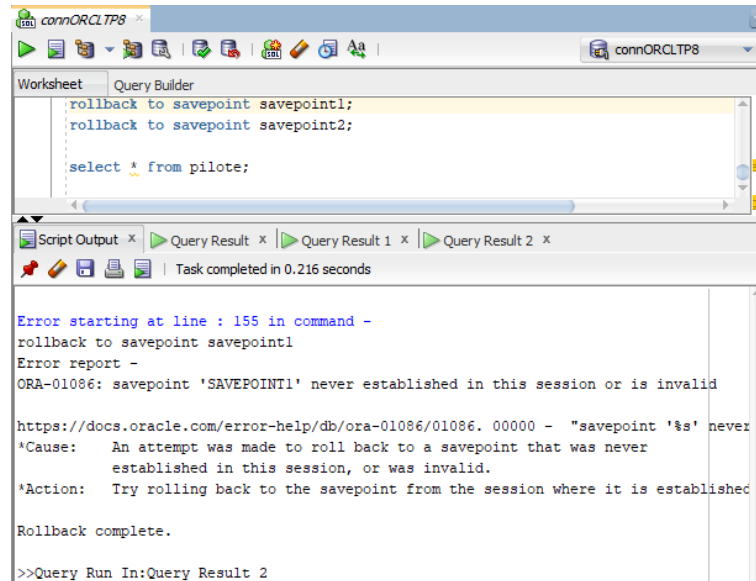
```

Below the output, a table displays the query results:

	NUMPILOTE	NOM	ADRESSE	SAL	COMM	EMBAUCHE
1	5555	ELIDRISSI	OUARZAZATE	5000	100	30-OCT-20

3. Rollback to savepoints :

- Lorsque la commande ROLLBACK TO SAVEPOINT savepoint1 est exécutée, une erreur se produit.
 - Cela s'explique par le fait que savepoint1 a été invalide après le COMMIT.
 - Ainsi, il est impossible de revenir à savepoint1.
- La commande ROLLBACK TO SAVEPOINT savepoint2 fonctionne correctement. Toutes les modifications effectuées après savepoint2 sont annulées. Par conséquent, l'insertion du pilote ELIDRISSI est annulée.



4. Sélection des données :

- Lors de la sélection des données dans la table PILOTE, seul le pilote NOHAILA est présent.
- Cela s'explique par les actions suivantes :
 - NOHAILA a été validé définitivement grâce au COMMIT.
 - ELIDRISSI a été annulé par le ROLLBACK jusqu'au savepoint2.

The screenshot shows the 'Query Result' tab in Oracle SQL Developer, displaying the results of the query 'select * from pilote;'. The table has 12 rows and 6 columns: NUMPILOTE, NOM, ADRESSE, SAL, COMM, and EMBAUCHE. The data is as follows:

	NUMPILOTE	NOM	ADRESSE	SAL	COMM	EMBAUCHE
1	1333	FEDOI	NANTES	3900	2000	01-MAR-00
2	6589	DUVAL	PARIS	4568	3000	12-MAR-00
3	7100	MARTIN	LYON	4400	2550	01-APR-01
4	3452	ANDRE	NICE	4150	2000	12-DEC-00
5	3421	BERGER	REIMS	3200	2000	28-DEC-00
6	6548	BARRE	LYON	3990	3250	01-DEC-00
7	1243	COLLET	PARIS	3840	2000	01-FEB-98
8	5643	DELORME	PARIS	5612	2800	01-FEB-00
9	6723	MARTIN	ORSAY	4100	2000	15-MAY-00
10	8843	GAUCHER	CACHAN	3875	2000	20-NOV-00
11	3465	PIC	TOURS	3890	2000	15-JUL-01
12	7584	NOHAILA	AGADIR	8000	2000	12-DEC-21

Conclusion

Ce TP a permis de mettre en pratique les fonctionnalités avancées du langage PL/SQL, notamment les procédures, fonctions, triggers, packages et transactions. Ces outils facilitent l'automatisation des traitements, assurent l'intégrité des données et contribuent à une meilleure organisation du code.

Les différentes étapes ont renforcé la compréhension des concepts suivants :

- La création et la gestion de procédures et fonctions pour simplifier les opérations complexes.
- L'utilisation de triggers pour appliquer automatiquement des règles et vérifier certaines conditions.
- La structuration du code avec des packages afin de faciliter sa maintenance et sa réutilisation.
- La manipulation des transactions pour assurer la cohérence des données en cas d'opérations critiques.

Ces notions sont essentielles pour développer des applications fiables et optimisées dans un environnement Oracle.