

PL/SQL

Déclarations, itérations, boucles, instructions conditionnelles

Exercice 1. Soit la table suivante :

VOL(Numvol, Heure_départ, Heure_arrivée, Ville_départ, Ville_arrivée)

Écrivez un programme PL/SQL qui insère le vol AF110 partant de Paris à 21h40 et arrivant à Dublin à 23h10 (hypothèse : le vol n'est pas déjà présent dans la table).

Solution :

```
DECLARE
  V VOL%ROWTYPE;
BEGIN

  V.Numvol:='AF110';

  V.heure_depart:= to_date('01/11/2021 21:40', ' dd/mm/yyyy hh24:mi');

  V.heure_arrivee := to_date('01/11/2021 23:10', ' dd/mm/yyyy hh24:mi');

  V.ville_depart := 'Paris';

  V.ville_arrivee := 'Dublin';

  INSERT INTO Vol VALUES V;

END;
```

16	BA985	01/12/21	01/12/21	Londres	Paris
17	AF178	01/12/21	01/12/21	Paris	Londres
18	AF110	01/11/21	01/11/21	Paris	Dublin

Exercice 2. Soit la table RES(NO). Écrivez un bloc PL/SQL qui insère les chiffres de 1 à 100 dans cette table.

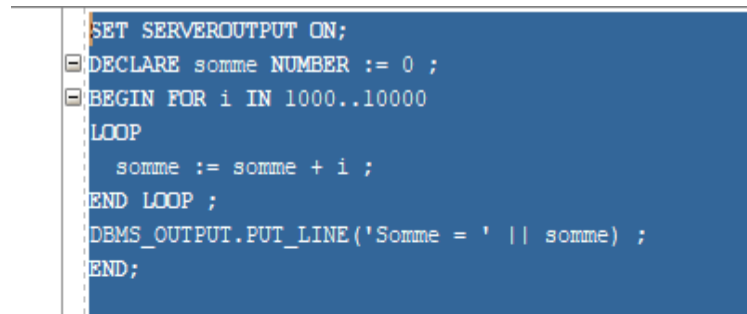
Solution :

```
DECLARE
    NB NUMBER:=1;

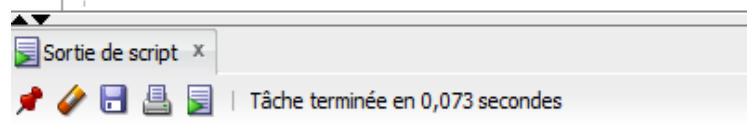
BEGIN
LOOP
INSERT INTO RES values( NB);
NB:=NB+1;
EXIT when NB<=100;
END LOOP;
END;
```

Exercice 3. Écrivez un bloc PL/SQL qui affiche la somme des nombres entre 1000 et 10000.

```
SET SERVEROUTPUT ON;
DECLARE somme NUMBER := 0 ;
BEGIN FOR i IN 1000..10000
LOOP
    somme := somme + i ;
END LOOP ;
DBMS_OUTPUT.PUT_LINE('Somme = ' || somme) ;
END;
```

A screenshot of a SQL Developer script editor window. The code is as follows:

```
SET SERVEROUTPUT ON;
DECLARE somme NUMBER := 0 ;
BEGIN FOR i IN 1000..10000
LOOP
    somme := somme + i ;
END LOOP ;
DBMS_OUTPUT.PUT_LINE('Somme = ' || somme) ;
END;
```



Somme = 49505500

Procédure PL/SQL terminée.

Curseurs, déclencheurs, relations

Exercice 4. On considère la table suivante:
PILOTE(Matricule, Nom, Ville, Age, Salaire).

Écrivez un programme PL/SQL qui calcule la moyenne des salaires des pilotes dont l'âge est entre 30 et 40 ans.

```
DECLARE CURSOR curseur1 IS SELECT salaire FROM pilote WHERE (Age >= 30 AND Age <=40);
salairePilote Pilote.Salaire%TYPE;
sommeSalaires NUMBER(11,2) := 0; moyenneSalaires NUMBER(11,2);
BEGIN OPEN curseur1; LOOP FETCH curseur1 INTO salairePilote;
EXIT
WHEN (curseur1%NOTFOUND OR curseur1%NOTFOUND IS NULL);
sommeSalaires := sommeSalaires + salairePilote;
END LOOP;
moyenneSalaires := sommeSalaires / curseur1%ROWCOUNT; CLOSE curseur1;
DBMS_OUTPUT.PUT_LINE('Moyenne salaires (pilotes de 30 40 ans) : ' || moyenneSalaires);
END;
```

Exercice 5. Soit la table suivante :

METEO(**NOM_VILLE**, **Température**, **Humidité**)

Écrire une fonction PL/SQL qui prends en entrée le nom d'une ville et retourne la température et l'humidité de cette ville. Gérer aussi par une exception le cas où la ville n'existe pas.

```
TYPE HumTemp IS RECORD( HUM METEO.HUMIDITY%TYPE, TEMP METEO.TEMPERATURE%TYPE )
FUNCTION GET_TEMPERATURE (PVILLE IN METEO.NOM_VILLE%TYPE) RETURN HumTemp IS VAL
HUMTEMP;
BEGIN VAL.HUM = -10000; VAL.TEMP = -10000; SELECT HUMIDITE, TEMPERATURE INTO VAL FROM
METEO WHERE VILLE_NOM = PVILLE;
RETURN VAL;
EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Ville n\'existe pas');
RETURN VAL;
END;
```

Exercice 6. Soit la table METEO de l'exercice précédent.

Écrire un déclencheur qui avant l'insertion d'une nouvelle ville dans la table vérifie :

- a. Si la température est la plus grande de toutes les villes, afficher un message d'avertissement.
- b. Si la ville existe déjà dans la table, ne pas l'insérer une nouvelle fois mais faire la mise à jour seulement.

```
CREATE OR REPLACE TRIGGER MTRIGGER BEFORE UPDATE ON METEO FOR EACH ROW DECLARE TMAX
NUMBER;
NB NUMBER := 0;
BEGIN SELECT MAX(Temperature) INTO TMAX FROM METEO ;
SELECT COUNT(*) INTO NB FROM METEO M WHERE M.VILLE_NOM = :NEW.VILLE_NOM;
IF (:NEW.TEMPERATURE > TMAX) THEN DBMS_OUTPUT.PUT_LINE(-20001,'Temperature MAX');
ELSIF NB > 0 THEN UPDATE METEO SET TEMPERATURE =:NEW.TEMPERATURE WHERE VILLE_NOM =
:NEW.VILLE_NOM ;
RAISE_APPLICATION_ERROR(-20001, 'La ville existe déjà');
END IF END;
```