

# Rendu final

Réalisé par : Noura Aoujil

## creation des tableaux

Quick SQL

Clear

Settings

Download

Save as Script

Save Model

Shorthand

?

Samples

Generate SQL

Rectangular Snip

```

1 CREATE TABLE PILOTE (
2     NUMPIL INT PRIMARY KEY,
3     NOMPIL VARCHAR(50),
4     ADR VARCHAR(50),
5     SAL INT
6 );
7
8 CREATE TABLE AVION (
9     NUMAV INT PRIMARY KEY,
10    NOMAV VARCHAR(50),
11    CAPACITE INT,
12    LOC VARCHAR(50)
13 );
14
15 CREATE TABLE VOL (
16     NUMVOL INT PRIMARY KEY,
17     NUMPIL INT,
18     NUMAV INT,

```

Output

Copy to Worksheet

```

1 -- create tables
2 v create table create_table_pilote (
3     numpil_int_primary_key    varchar2(4000),
4     nompil                    varchar2(50),
5     adr                       varchar2(50),
6     sal                       integer
7 )
8 ;
9
10 v create table create_table_avion (
11     numav_int_primary_key    varchar2(4000),
12     nomav                    varchar2(50),
13     capacite                 integer,
14     loc                      varchar2(50)
15 )
16 ;
17
18 v create table create_table_vol (

```

# exercice1

## Insertion des Données

```
SQL Worksheet Clear Find Actions Save Run
```

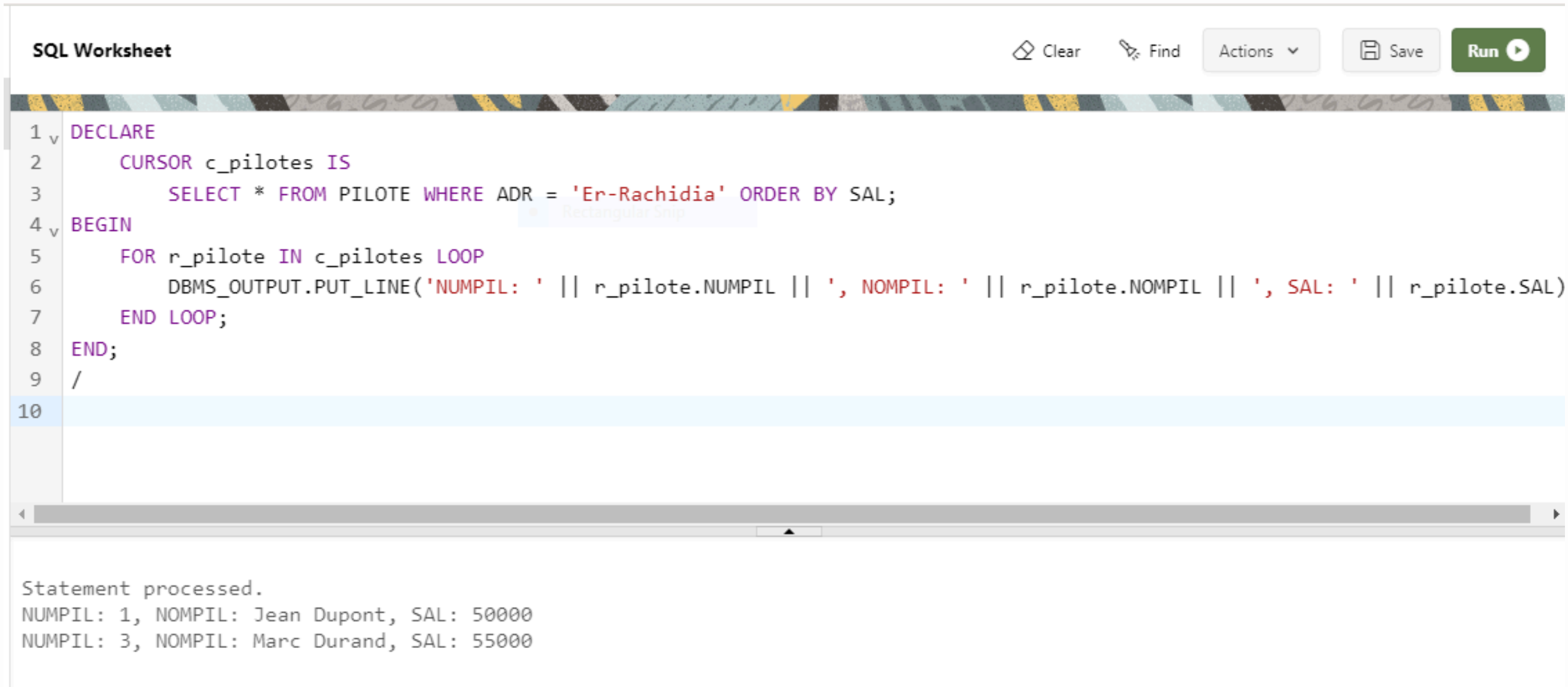
```
1  -- Insert into PILOTE
2  INSERT INTO PILOTE (NUMPIL, NOMPIL, ADR, SAL) VALUES (1, 'Jean Dupont', 'Er-Rachidia', 50000);
3  INSERT INTO PILOTE (NUMPIL, NOMPIL, ADR, SAL) VALUES (2, 'Alice Martin', 'Casablanca', 60000);
4  INSERT INTO PILOTE (NUMPIL, NOMPIL, ADR, SAL) VALUES (3, 'Marc Durand', 'Er-Rachidia', 55000);
5
6  -- Insert into AVION
7  INSERT INTO AVION (NUMAV, NOMAV, CAPACITE, LOC) VALUES (1, 'Airbus A320', 150, 'Casablanca');
8  INSERT INTO AVION (NUMAV, NOMAV, CAPACITE, LOC) VALUES (2, 'Boeing 737', 180, 'Er-Rachidia');
9  INSERT INTO AVION (NUMAV, NOMAV, CAPACITE, LOC) VALUES (3, 'Cessna 172', 4, 'Rabat');
10
11 -- Insert into VOL
12 INSERT INTO VOL (NUMVOL, NUMPIL, NUMAV, VILLE_DEP, VILLE_ARR, H_DEP, H_ARR) VALUES
13 (1, 1, 1, 'Er-Rachidia', 'Casablanca',
```

```
1 row(s) inserted.

1 row(s) inserted.
```

# exercice1

1. Afficher les pilotes d'Er-Rachidia triés par salaire croissant :



The screenshot shows an SQL Worksheet interface with a toolbar at the top containing 'Clear', 'Find', 'Actions', 'Save', and 'Run' buttons. The main area contains a PL/SQL script with line numbers 1 through 10. The script declares a cursor, selects data from the 'PILOTE' table where 'ADR' is 'Er-Rachidia', and loops through the results to print them. The output at the bottom shows the statement was processed successfully and displays two rows of data.

```
1 DECLARE
2     CURSOR c_pilotes IS
3         SELECT * FROM PILOTE WHERE ADR = 'Er-Rachidia' ORDER BY SAL;
4 BEGIN
5     FOR r_pilote IN c_pilotes LOOP
6         DBMS_OUTPUT.PUT_LINE('NUMPIL: ' || r_pilote.NUMPIL || ', NOMPIL: ' || r_pilote.NOMPIL || ', SAL: ' || r_pilote.SAL)
7     END LOOP;
8 END;
9 /
10
```

Statement processed.  
NUMPIL: 1, NOMPIL: Jean Dupont, SAL: 50000  
NUMPIL: 3, NOMPIL: Marc Durand, SAL: 55000

# exercice1

Afficher les informations des pilotes en service :



The screenshot shows an SQL Worksheet interface with a toolbar at the top containing 'Clear', 'Find', 'Actions', 'Save', and 'Run' buttons. The main area contains a PL/SQL program with line numbers 1 through 18. The program declares a cursor 'c\_pilotes\_en\_service' and a record variable 'r\_pilote'. It opens the cursor, loops through the results, and prints the pilot's number, name, and salary. The program ends with 'END;' and a slash. A status bar at the bottom indicates 'Statement processed.'.

```
1 DECLARE
2     CURSOR c_pilotes_en_service IS
3         SELECT DISTINCT p.*
4         FROM PILOTE p
5         JOIN VOL v ON p.NUMPIL = v.NUMPIL;
6
7     r_pilote c_pilotes_en_service%ROWTYPE; -- Declare the record variable to hold the cursor's row data
8
9 BEGIN
10     OPEN c_pilotes_en_service;
11     LOOP
12         FETCH c_pilotes_en_service INTO r_pilote;
13         EXIT WHEN c_pilotes_en_service%NOTFOUND;
14         DBMS_OUTPUT.PUT_LINE('NUMPIL: ' || r_pilote.NUMPIL || ', NOMPIL: ' || r_pilote.NOMPIL || ', SAL: ' || r_pilote.SAL);
15     END LOOP;
16     CLOSE c_pilotes_en_service;
17 END;
18 /
```

Statement processed.

# exercice1

Afficher les informations des avions avec capacité  $\geq 50$  :



The screenshot shows an SQL Worksheet interface with a toolbar at the top containing 'Clear', 'Find', 'Actions', 'Save', and 'Run' buttons. The main area contains a PL/SQL program with line numbers 1 through 18. The program declares a cursor and an exception, then loops through the 'AVION' table to display details for aircraft with a capacity of 50 or more. It includes exception handling for 'NO\_DATA\_FOUND' and a custom exception 'e\_avion\_not\_found'. The output at the bottom shows the statement was processed successfully and displays two rows of aircraft information.

```
1 DECLARE
2     v_avion AVION%ROWTYPE;
3     e_avion_not_found EXCEPTION;
4     PRAGMA EXCEPTION_INIT(e_avion_not_found, -20001);
5 BEGIN
6     BEGIN
7         FOR r_avion IN (SELECT * FROM AVION WHERE CAPACITE >= 50) LOOP
8             DBMS_OUTPUT.PUT_LINE('NUMAV: ' || r_avion.NUMAV || ', NOMAV: ' || r_avion.NOMAV || ', LOC: ' || r_avion.LOC);
9         END LOOP;
10    EXCEPTION
11        WHEN NO_DATA_FOUND THEN
12            RAISE e_avion_not_found;
13    END;
14 EXCEPTION
15     WHEN e_avion_not_found THEN
16         DBMS_OUTPUT.PUT_LINE('Aucun avion trouvé avec une capacité >= 50.');

Statement processed.  
NUMAV: 1, NOMAV: Airbus A320, LOC: Casablanca  
NUMAV: 2, NOMAV: Boeing 737, LOC: Er-Rachidia


```

# exercice1

4-Afficher les pilotes effectuant un vol au départ de leur ville de résidence :



The screenshot shows an SQL Worksheet interface. At the top, there is a title bar "SQL Worksheet" and a toolbar with buttons for "Clear", "Find", "Actions", "Save", and "Run". The main area contains a PL/SQL script with line numbers 1 through 16. The script declares a cursor, performs a join between PILOTE and VOL tables, and uses a loop to output the results. Below the script, a status bar indicates "Statement processed."

```
1 DECLARE
2     CURSOR c_pilotes_vol IS
3         SELECT DISTINCT p.NUMPIL, p.NOMPIL
4         FROM PILOTE p
5         JOIN VOL v ON p.NUMPIL = v.NUMPIL
6         WHERE p.ADR = v.VILLE_DEP;
7 BEGIN
8     FOR r_pilote IN c_pilotes_vol LOOP
9         DBMS_OUTPUT.PUT_LINE('NUMPIL: ' || r_pilote.NUMPIL || ', NOMPIL: ' || r_pilote.NOMPIL);
10    END LOOP;
11 END;
12 /
13
14
15
16
```

Statement processed.

# exercice1

## 5-Insérer un vol et gérer les exceptions :

SQL Worksheet

ClearFindActionsSaveRun

```
1 CREATE OR REPLACE TRIGGER TRGR_UPDATE_TRACK_VOLS
2 AFTER INSERT ON VOL
3 FOR EACH ROW
4 BEGIN
5     UPDATE TRACK_VOLS
6     SET NB_VOLS = NB_VOLS + 1
7     WHERE NUMPIL = :NEW.NUMPIL;
8 EXCEPTION
9     WHEN OTHERS THEN
10         RAISE_APPLICATION_ERROR(-20001, 'Error updating track_vols: ' || SQLERRM);
11 END;
12 /
13 DECLARE
14     v_numvol INT := 101;
15     v_numav INT := 1; -- Replace with the actual input value
16     v_numpil INT := 1; -- Replace with the actual input value
17     e_vol_exists EXCEPTION;
18     e_pilote_not_found EXCEPTION;
19     e_avion_not_found EXCEPTION;
20     v_exists INT;
21     v_pilot_exists INT;
```

Trigger created.



# exercice1

## 5-Insérer un vol et gérer les exceptions :

```
SQL Worksheet
```

```
23 BEGIN
24 -- Vérification si le vol existe déjà
25 BEGIN
26     SELECT 1 INTO v_exists
27     FROM VOL
28     WHERE NUMPIL = v_numpil AND NUMAV = v_numav AND VILLE_DEP = 'Er-Rachidia' AND VILLE_ARR = 'Marrakech'
29           AND H_DEP = TO_TIMESTAMP('2024-06-03 18:00:00', 'YYYY-MM-DD HH24:MI:SS')
30           AND H_ARR = TO_TIMESTAMP('2024-06-03 19:10:00', 'YYYY-MM-DD HH24:MI:SS');
31     RAISE e_vol_exists;
32 EXCEPTION
33 WHEN NO_DATA_FOUND THEN
34     NULL; -- Le vol n'existe pas encore, on peut continuer
35 END;
36
37 -- Vérification de l'existence du pilote
38 BEGIN
39     SELECT NUMPIL INTO v_pilot_exists FROM PILOTE WHERE NUMPIL = v_numpil;
40 EXCEPTION
41 WHEN NO_DATA_FOUND THEN
42     RAISE e_pilote_not_found;
43 END;
44
```

```
Trigger created.
```

# exercice1

## 5-Insérer un vol et gérer les exceptions :

```
SQL Worksheet Clear Find Actions Save Run
```

```
46 BEGIN
47     SELECT NUMAV INTO v_avion_exists FROM AVION WHERE NUMAV = v_numav;
48 EXCEPTION
49     WHEN NO_DATA_FOUND THEN
50         RAISE e_avion_not_found;
51 END;
52
53 -- Insertion du vol
54 INSERT INTO VOL (NUMVOL, NUMPIL, NUMAV, VILLE_DEP, VILLE_ARR, H_DEP, H_ARR) VALUES
55 (v_numvol, v_numpil, v_numav, 'Er-Rachidia', 'Marrakech', TO_TIMESTAMP('2024-06-03 18:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2024-06-03 19:10:00', 'YYYY-MM-DD HH24:MI:SS'));
56 DBMS_OUTPUT.PUT_LINE('Vol inséré avec succès.');
```

```
57
58 EXCEPTION
59     WHEN e_vol_exists THEN
60         DBMS_OUTPUT.PUT_LINE('Erreur : le vol existe déjà.');
```

```
61     WHEN e_pilote_not_found THEN
62         DBMS_OUTPUT.PUT_LINE('Erreur : le numéro de pilote nexiste pas.');
```

```
63     WHEN e_avion_not_found THEN
64         DBMS_OUTPUT.PUT_LINE('Erreur : le numéro d'avion nexiste pas.');
```

```
65     WHEN OTHERS THEN
66         DBMS_OUTPUT.PUT_LINE('Erreur inconnue : ' || SQLERRM);
67 END;
```

# exercice1

5-Insérer un vol et gérer les exceptions :

```
■  
Trigger created.  
  
Statement processed.  
Vol inséré avec succès.
```

# exercice2

## Procédure profilePilote :

**SQL Worksheet**

ClearFindActionsSaveRun

```
1 CREATE OR REPLACE PROCEDURE profilePilote(p_numpil IN INT) IS
2     v_pilote PILOTE%ROWTYPE;
3     e_pilote_not_found EXCEPTION;
4 BEGIN
5     BEGIN
6         SELECT * INTO v_pilote FROM PILOTE WHERE NUMPIL = p_numpil;
7     EXCEPTION
8         WHEN NO_DATA_FOUND THEN
9             RAISE e_pilote_not_found;
10    END;
11
12    DBMS_OUTPUT.PUT_LINE('NUMPIL: ' || v_pilote.NUMPIL || ', NOMPIL: ' || v_pilote.NOMPIL || ', ADR: ' || v_pilote.ADR || ', SAL: ' || v_pilote.SAL);
13
14 EXCEPTION
15     WHEN e_pilote_not_found THEN
16         DBMS_OUTPUT.PUT_LINE('Erreur : le pilote n'existe pas.');
```

Procedure created.

# exercice2

Fonction pilotesErrLibres :

SQL Worksheet

ClearFindActionsSaveRun

```
1 CREATE OR REPLACE FUNCTION pilotesErrLibres RETURN INT IS
2   v_count INT;
3 BEGIN
4   SELECT COUNT(*) INTO v_count
5   FROM PILOTE p
6   WHERE p.ADR = 'Er-Rachidia' AND p.NUMPIL NOT IN (SELECT NUMPIL FROM VOL);
7   RETURN v_count;
8 END;
9 /
10
```

Function created.

# exercice2

Déclencheur trgr\_update\_track\_vols :

SQL Worksheet

ClearFindActionsSaveRun

```
1 CREATE OR REPLACE FUNCTION pilotesErrLibres RETURN INT IS
2   v_count INT;
3 BEGIN
4   SELECT COUNT(*) INTO v_count
5   FROM PILOTE p
6   WHERE p.ADR = 'Er-Rachidia' AND p.NUMPIL NOT IN (SELECT NUMPIL FROM VOL);
7   RETURN v_count;
8 END;
9 /
10
```

Function created.

# exercice2

## 4-Fonction tab\_pilotes :

SQL Worksheet

ClearFindActionsSaveRun

```
1 CREATE OR REPLACE FUNCTION tab_pilotes(fp_sal IN INT) RETURN SYS.ODCINUMBERLIST IS
2   v_pilote_tab SYS.ODCINUMBERLIST := SYS.ODCINUMBERLIST();
3   v_index INT := 0;
4   CURSOR c_pilotes(p_salaire INT) IS
5     SELECT NUMPIL FROM PILOTE WHERE SAL >= p_salaire;
6 BEGIN
7   FOR r_pilote IN c_pilotes(fp_sal) LOOP
8     v_index := v_index + 1;
9     v_pilote_tab.EXTEND;
10    v_pilote_tab(v_index) := r_pilote.NUMPIL;
11  END LOOP;
12  RETURN v_pilote_tab;
13 END;
14 /
15
```

Function created.