Topics

1. SQL procedures review
2. PL/SQL Conditional Statements
3. Triggers
4. Iteration Statements

Review

- What is server side application logic?
- What program units are stored with the database server
- Procedures, functions, packages, triggers
- What is the difference between a procedure and a function

SQL PROCEDURES

- To run a procedure called PROC1 in DB2 type
- CALL PROC1
- SQL/PL
- In ORACLE type
- EXECUTE PROC1
- How does a procedure run another procedure
- PROC1
- PL/SQL

SQL PROCEDURES

- DIFFERENT TYPES OF ARGUMENTS FOR A PROCEDURE ARE
- IN
- OUT
- IN OUT

PL/SQL Conditional Statements

- IF statement
- IF THEN
- IF THEN ELSE
- IF THEN ELSIF ELSE
- CASE
- Compare a given expression to different values
- Evaluates multiple conditions and choose the first condition that is true

```
SET SERVEROUTPUT ON;
DECLARE semester CHAR(1);
BEGIN semester := 'T';
CASE
    WHEN semester = 'F' THEN DBMS_OUTPUT.PUT_LINE('Fall Term');
    WHEN semester = 'W' THEN DBMS_OUTPUT.PUT_LINE('Winter Term');
    WHEN semester = 'S' THEN DBMS_OUTPUT.PUT_LINE('Summer Term');
    ELSE DBMS_OUTPUT.PUT_LINE('Wrong Value');
END CASE;

DBMS_OUTPUT.PUT_LINE('Something Else To Do');

EXCEPTION WHEN CASE_NOT_FOUND
THEN
DBMS_OUTPUT.PUT_LINE('No Semester Found');
END;
```

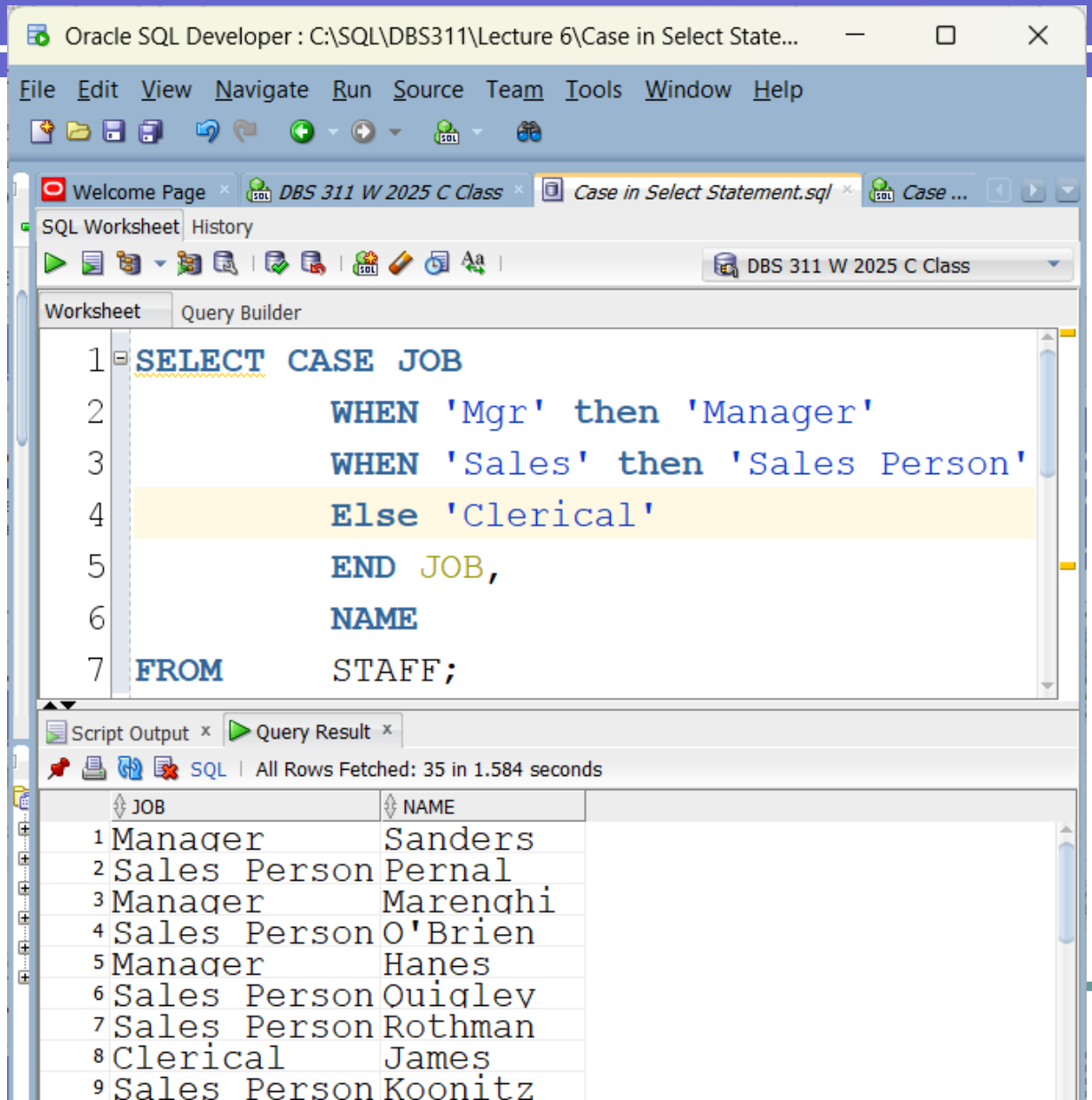
- What is an alternative version of Case with slightly different syntax for this problem

Alternative Case Syntax

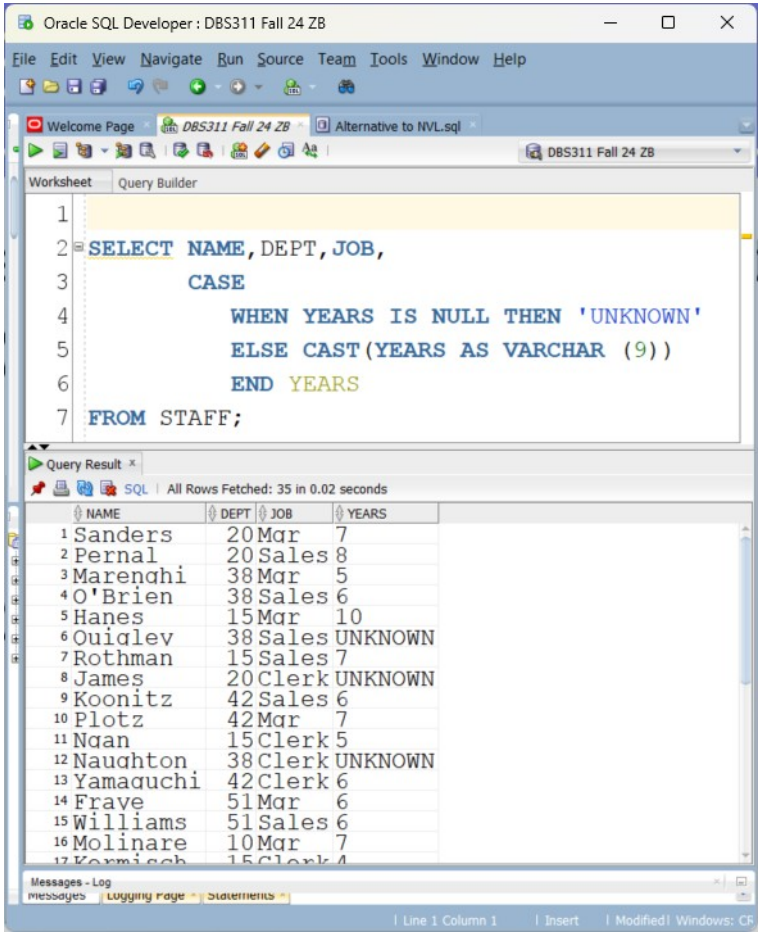
```
DECLARE semester CHAR(1);
BEGIN semester := 'T';
CASE semester
   WHEN 'F' THEN DBMS_OUTPUT.PUT_LINE('Fall Term');
   WHEN 'W' THEN DBMS_OUTPUT.PUT_LINE('Winter Term');
   WHEN 'S' THEN DBMS_OUTPUT.PUT_LINE('Summer Term');
   ELSE DBMS_OUTPUT.PUT_LINE('Wrong Value');
END CASE;

DBMS_OUTPUT.PUT_LINE('Something Else To Do');

EXCEPTION WHEN CASE_NOT_FOUND
THEN
DBMS_OUTPUT.PUT_LINE('No Semester Found');
END;
```

- Can you use an if statement in an Oracle select statement?
- **There's no if keyword in SQL**. If you want to do if-else-then logic in select , where or anywhere else in a statement, you need a case expression.

CASE in a SELECT statement



CASE in a SELECT statement



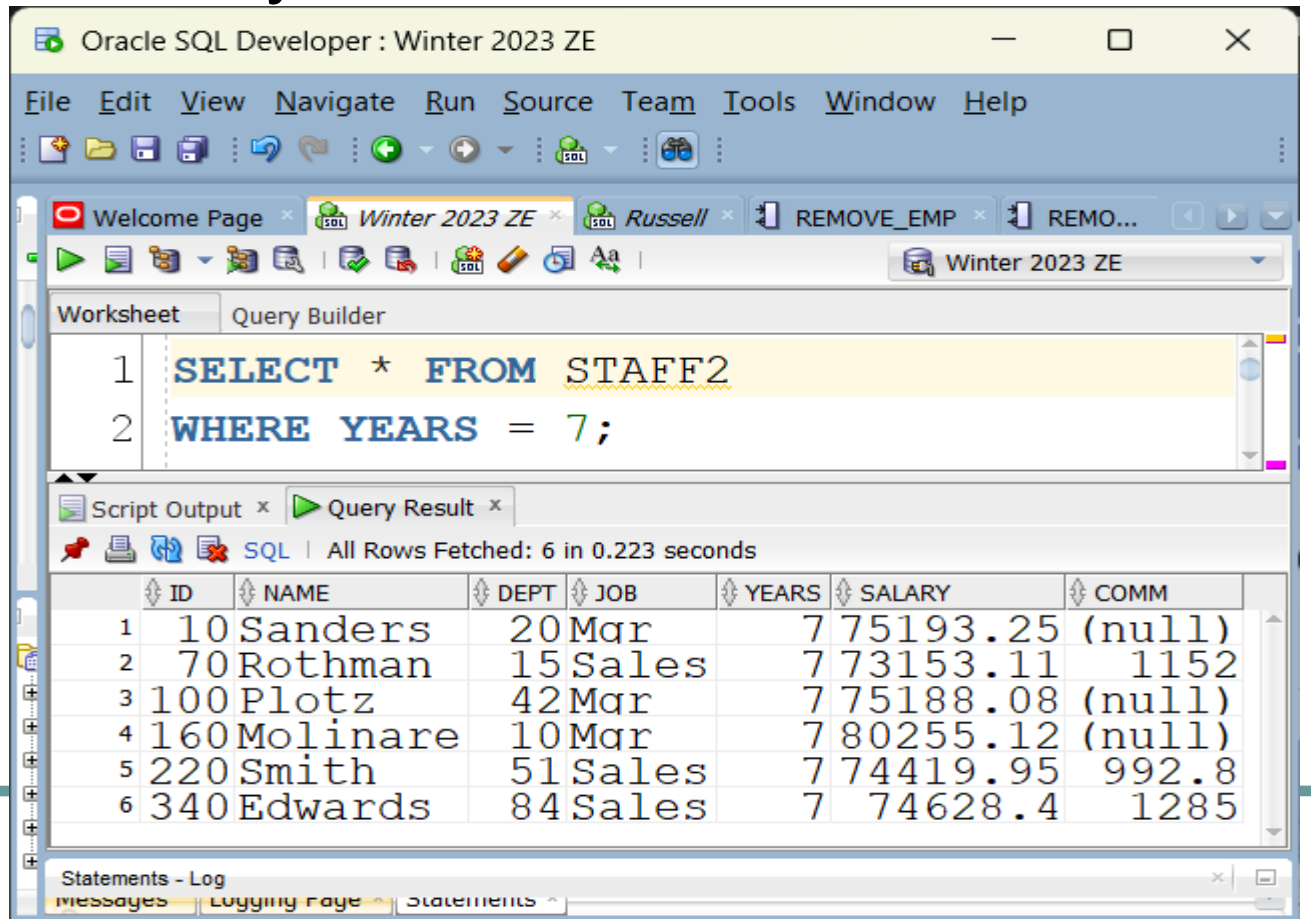
- What are the three possible blocks in a procedure?
- Declarative, Executable, Exception
- What is an anonymous block
- Not saved as a stored procedure on server, no name required, used for testing a stored procedure's arguments
- Where can an anonymous block be saved?
- As a file on your Desktop or Laptop

PL/SQL Review

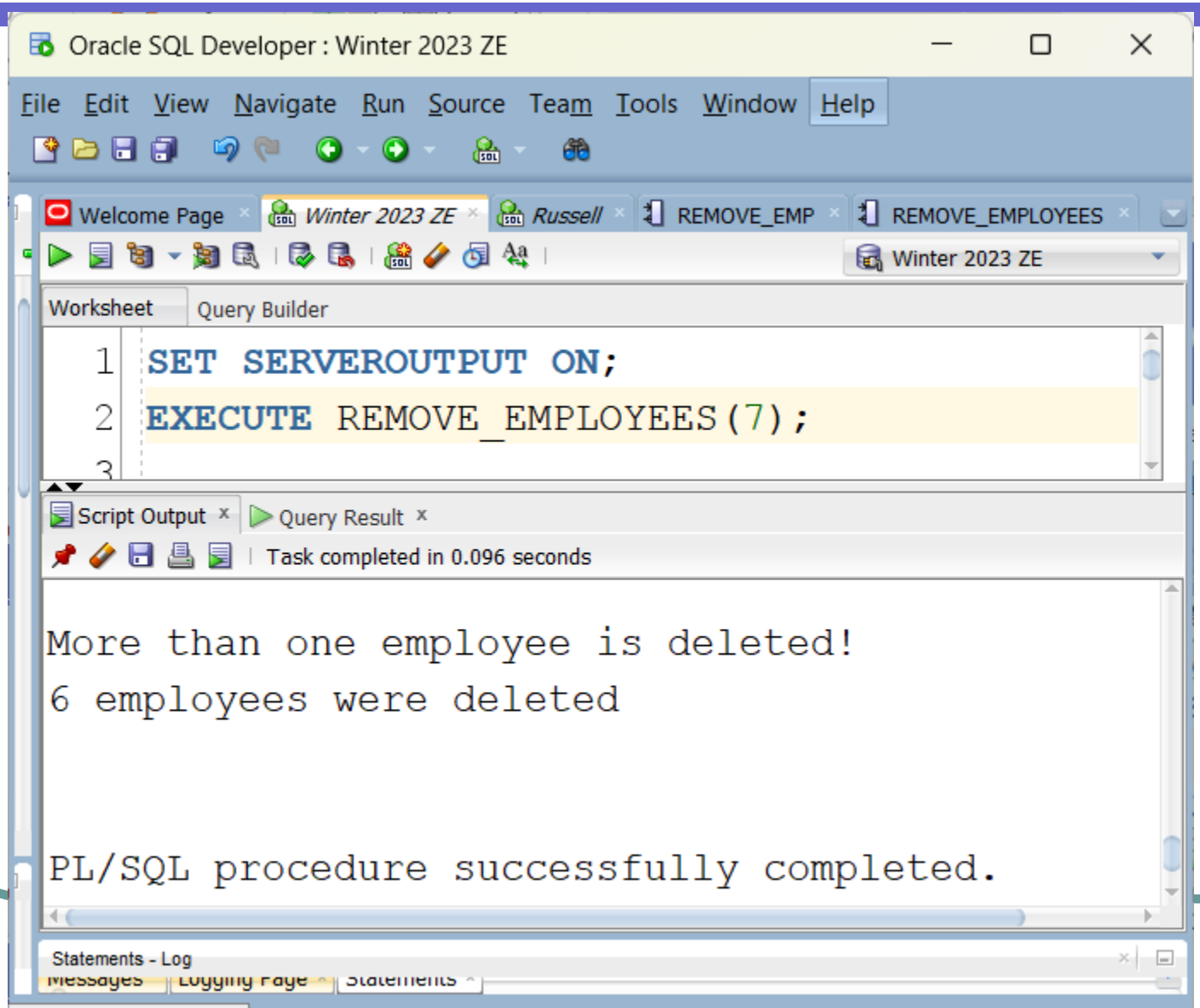
- How is a SELECT used in a procedure?
- SELECT column_list
- INTO variable_list
- FROM table_name
- WHERE condition(s);
- What are the possible exceptions you can monitor for?
- An exception will occur if the SELECT statement returns more than one row.
- TOO_MANY_ROWS exception
- An exception will occur if the SELECT statement returns no data
- NO_DATA_FOUND exception

RemoveEmployees Stored Procedure

- Some staff employees have 7 years of post secondary education

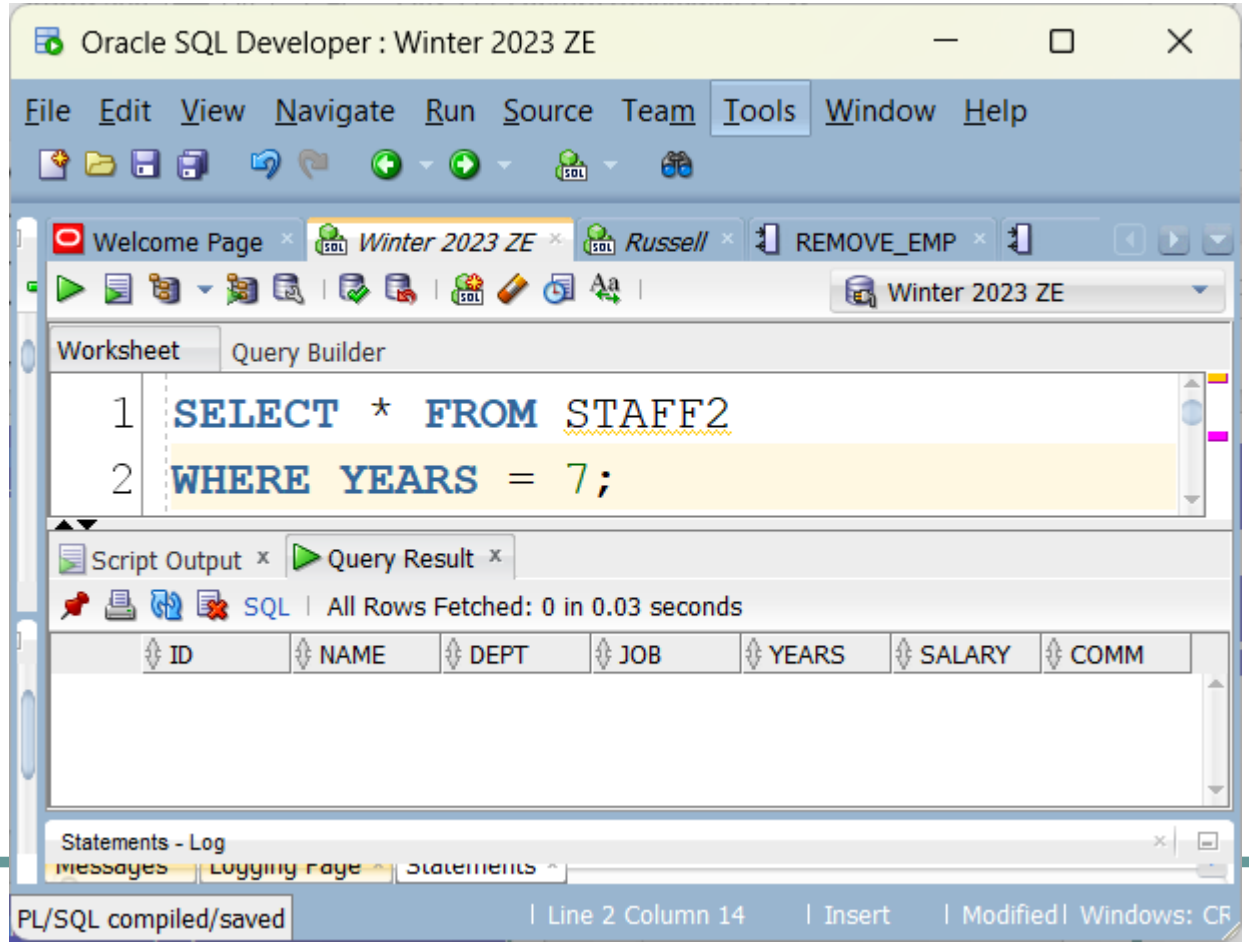


RemoveEmployees Stored Procedure



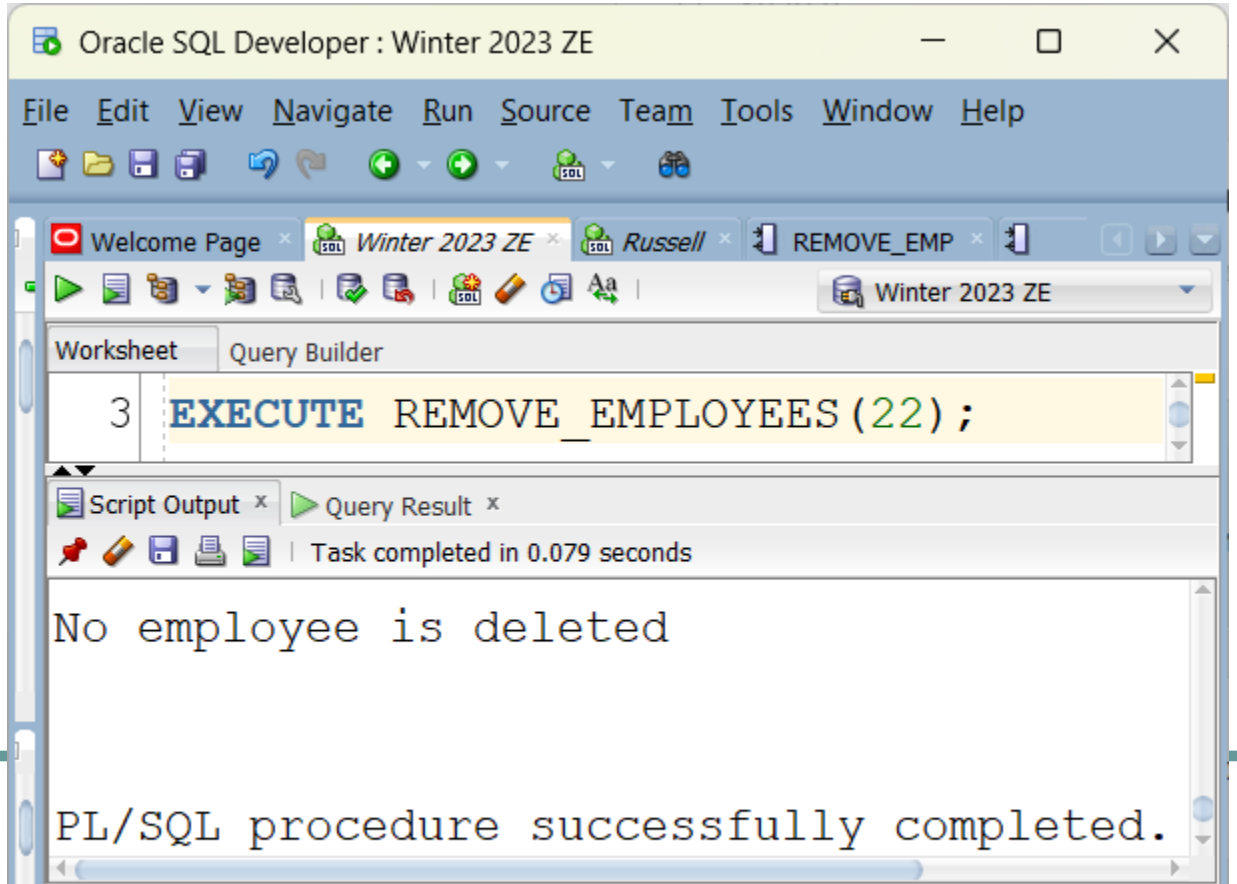
RemoveEmployees Stored Procedure

- After the procedure is run

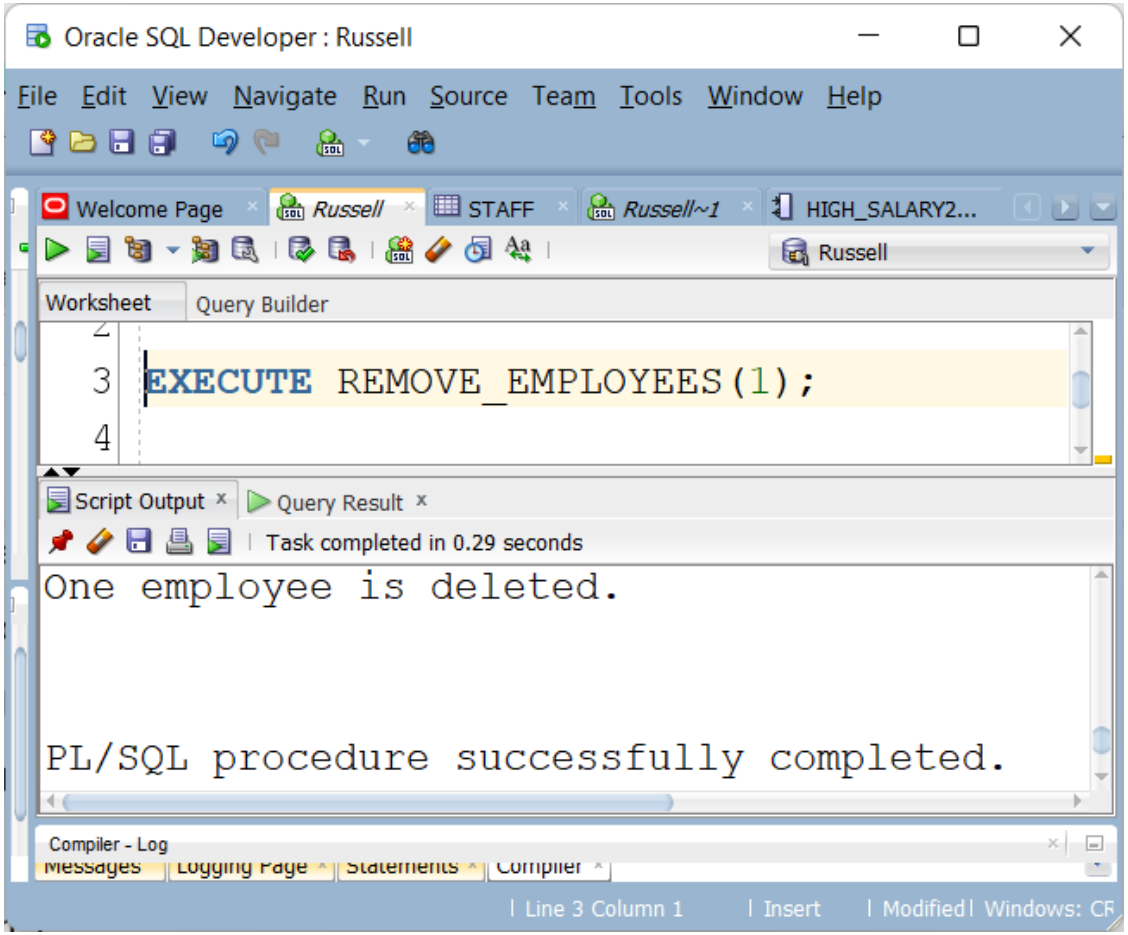


RemoveEmployees Stored Procedure

- No employee has 22 years of post secondary education



RemoveEmployees Stored Procedure



```
create or replace PROCEDURE remove_employeeS
(YEARSIN IN NUMBER)
AS
BEGIN
DELETE FROM STAFF2
WHERE YEARS = YEARSIN;
IF SQL%ROWCOUNT = 0 THEN
DBMS_OUTPUT.PUT_LINE ('No employee is deleted');
ELSIF SQL%ROWCOUNT = 1 THEN
DBMS_OUTPUT.PUT_LINE ('One employee is deleted.');
ELSE
DBMS_OUTPUT.PUT_LINE ('More than one employee is deleted!');
DBMS_OUTPUT.PUT_LINE (SQL%ROWCOUNT || ' employees were deleted');
END IF;
EXCEPTION
WHEN OTHERS
THEN
DBMS_OUTPUT.PUT_LINE ('Error!');
END;;
```

RemoveEmployees (SQL%RowCount)

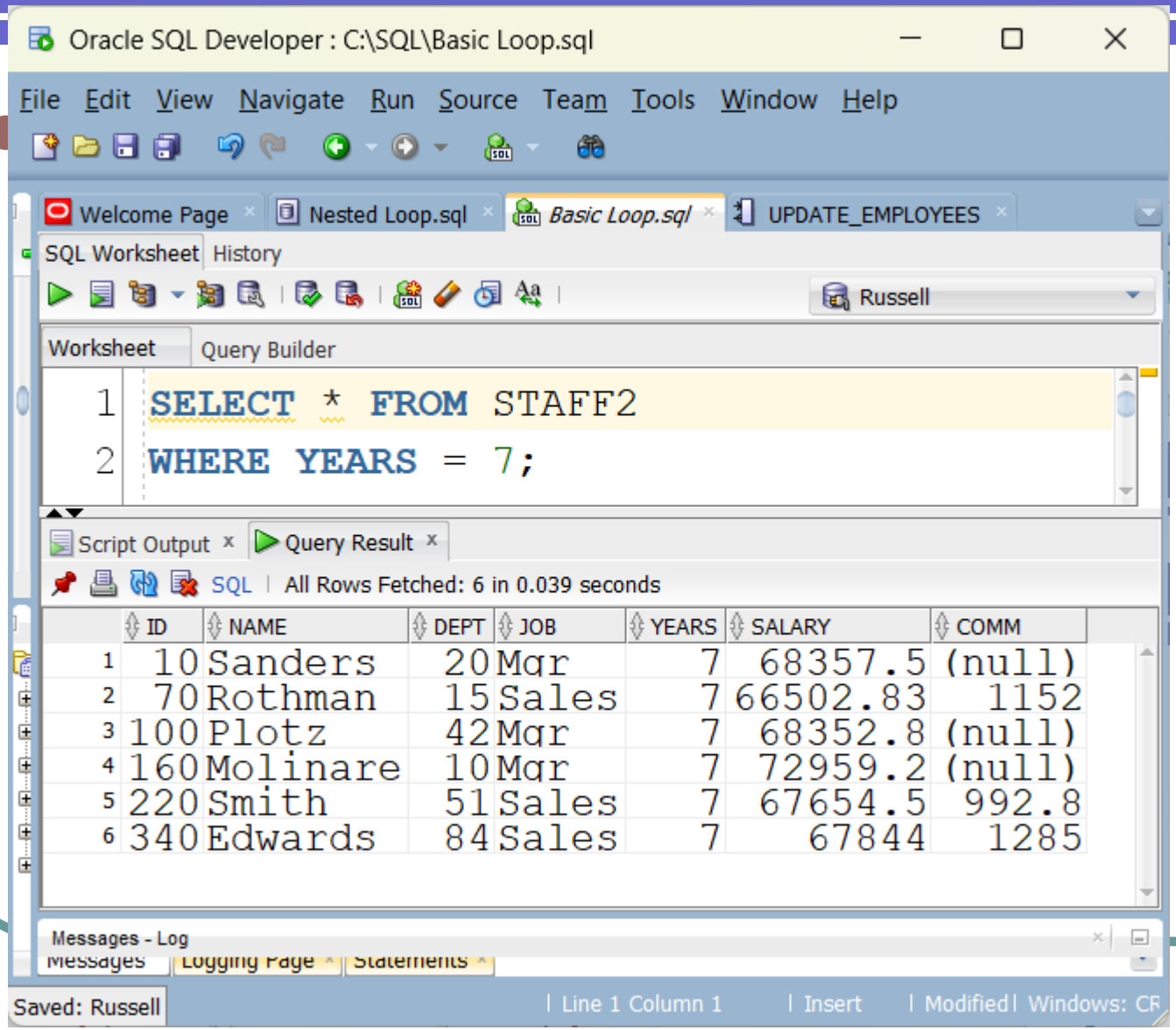
```
create or replace PROCEDURE remove_employeeS
   (YEARSIN IN NUMBER)
AS
BEGIN
   DELETE FROM STAFF2
   WHERE YEARS = YEARSIN;
   IF SQL%ROWCOUNT = 0 THEN
      DBMS_OUTPUT.PUT_LINE ('No employee is deleted');
   ELSIF SQL%ROWCOUNT = 1 THEN
      DBMS_OUTPUT.PUT_LINE ('One employee is deleted.');
   ELSE
      DBMS_OUTPUT.PUT_LINE ('More than one employee is deleted!');
      DBMS_OUTPUT.PUT_LINE (SQL%ROWCOUNT || ' employees were deleted');
   END IF;
   EXCEPTION
      WHEN OTHERS
      THEN
        DBMS_OUTPUT.PUT_LINE ('Error!');
END;
```

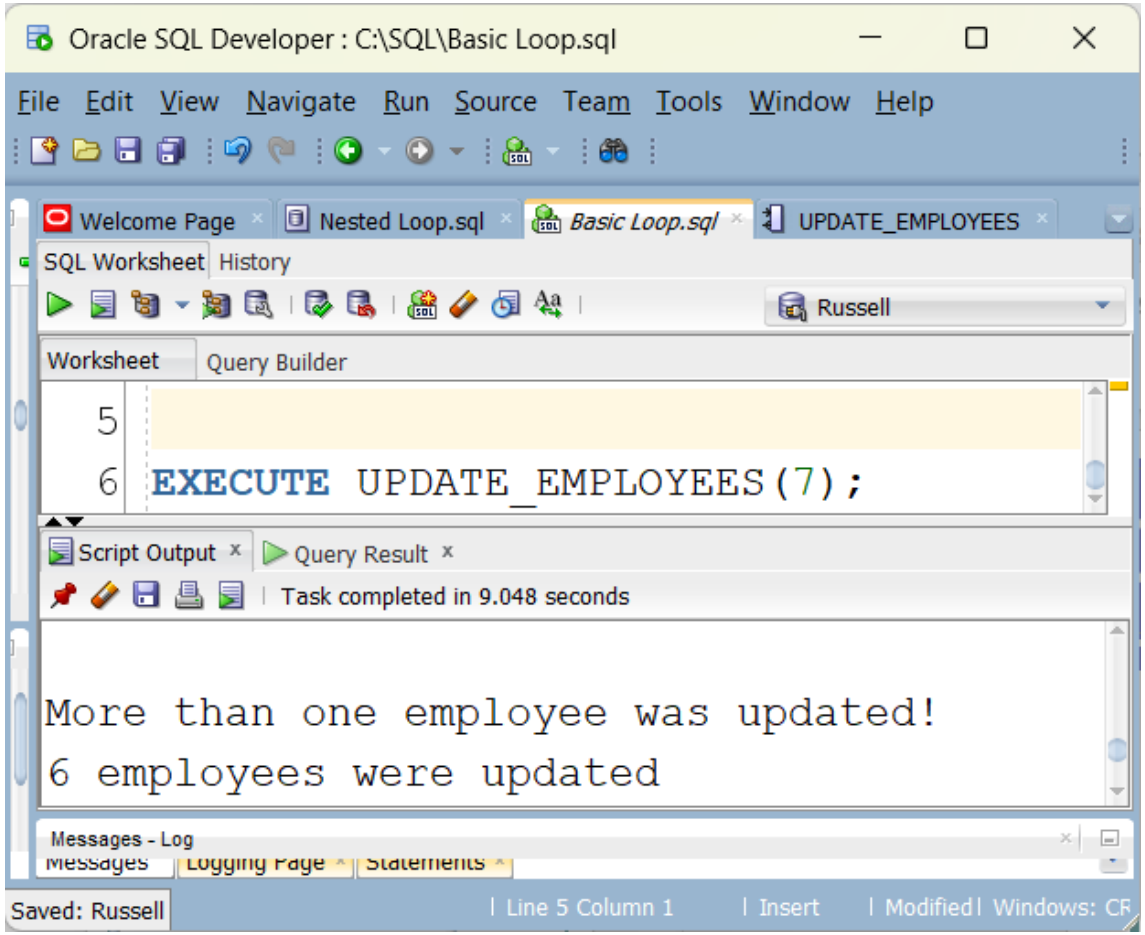
- Can we get all those deleted employees back?
- ROLLBACK;
- Sometimes ROLLBACK does not work for us – Why?
- You ran a DDL statement and that causes an automtic Commit

Resetting data in A2 tables

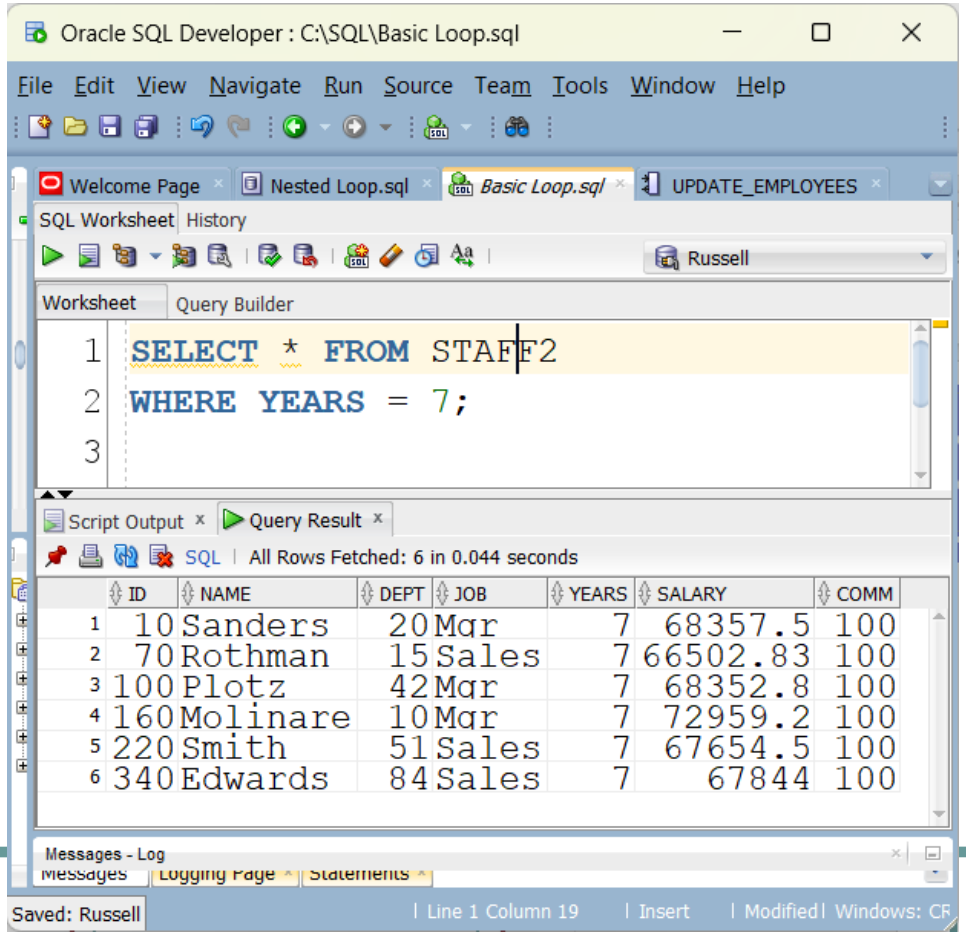
- You can recopy my data or automate the process
- You can work with a copy of your table.
- I am changing STAFF2 and always have the original STAFF table
- EXECUTE RESET;

```
create or replace PROCEDURE RESET
AS
BEGIN
DELETE FROM STAFF2;
INSERT INTO STAFF2
(SELECT * FROM STAFF);
COMMIT;
END;
```


Oracle SQL Developer : C:\SQL\Basic Loop.sql
File Edit View Navigate Run Source Team Tools Window Help
Welcome Page
Nested Loop.sql
Basic Loop.sql
UPDATE_EMPLOYEES
SQL Worksheet
History
Russell
Worksheet
Query Builder
1 SELECT * FROM STAFF2
2 WHERE YEARS = 7;
Script Output
Query Result
SQL | All Rows Fetched: 6 in 0.039 seconds
ID NAME DEPT JOB YEARS SALARY COMM
1 10 Sanders 20 Mgr 7 68357.5 (null)
2 70 Rothman 15 Sales 7 66502.83 1152
3 100 Plotz 42 Mgr 7 68352.8 (null)
4 160 Molinare 10 Mgr 7 72959.2 (null)
5 220 Smith 51 Sales 7 67654.5 992.8
6 340 Edwards 84 Sales 7 67844 1285
Messages - Log
Saved: Russell
Line 1 Column 1
Insert
Modified
Windows: CR


Oracle SQL Developer : C:\SQL\Basic Loop.sql
File Edit View Navigate Run Source Team Tools Window Help
Welcome Page
Nested Loop.sql
Basic Loop.sql
UPDATE_EMPLOYEES
SQL Worksheet
History
Russell
Worksheet
Query Builder
5
6
EXECUTE UPDATE_EMPLOYEES(7);
Script Output
Query Result
Task completed in 9.048 seconds
More than one employee was updated!
6 employees were updated
Messages - Log
Saved: Russell
Line 5 Column 1
Insert
Modified
Windows: CR

- The updated rows



Oracle SQL Developer : Procedure DBS311_241NBB30.UPDATE_EMPLOYEES@DBS Procedures W24

File Edit View Navigate Run Source Team Tools Window Help

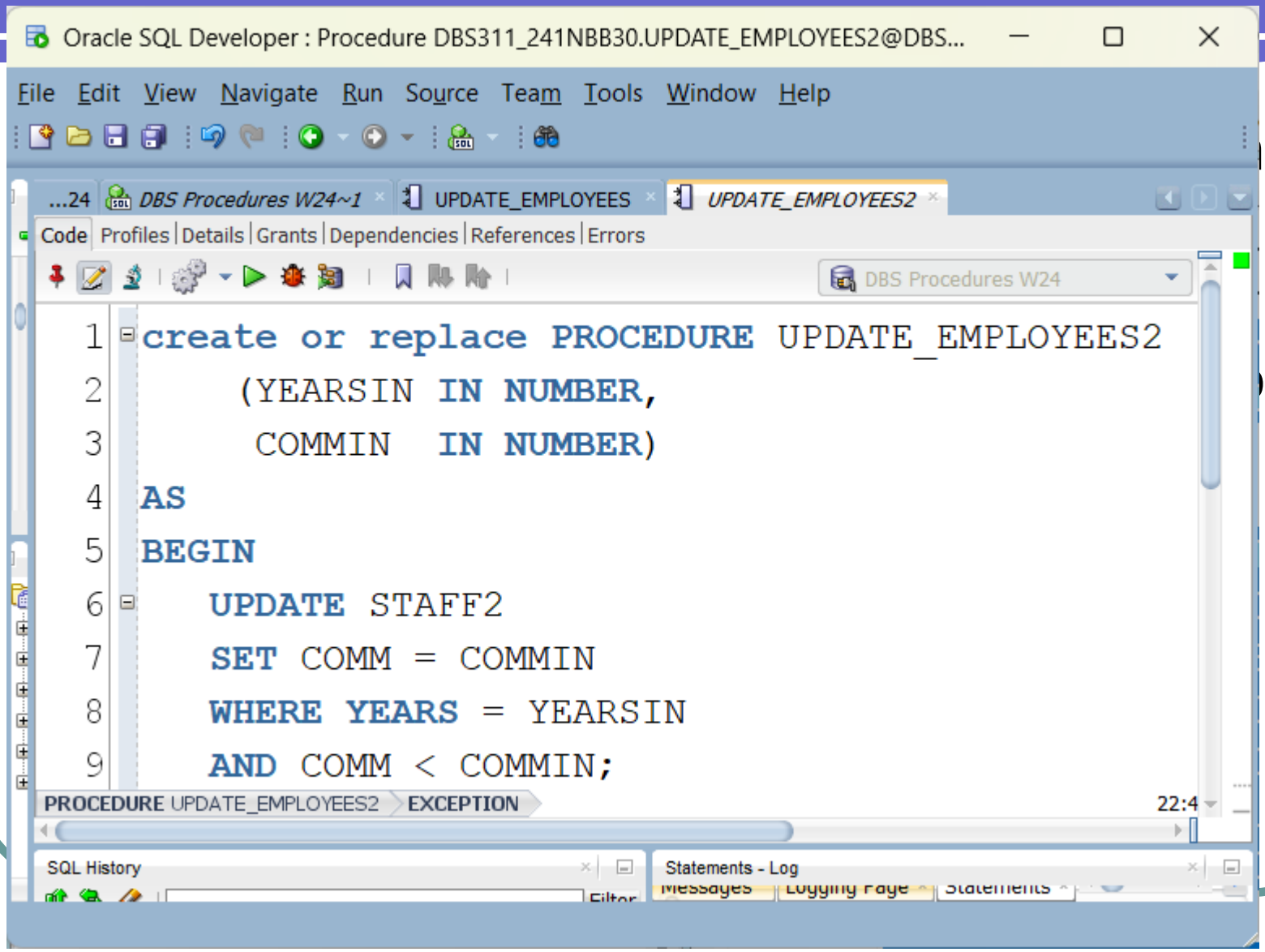
Welcome Page | DBS Procedures W24 | DBS Procedures W24~1 | UPDATE_EMPLOYEES

Code | Profiles | Details | Grants | Dependencies | References | Errors

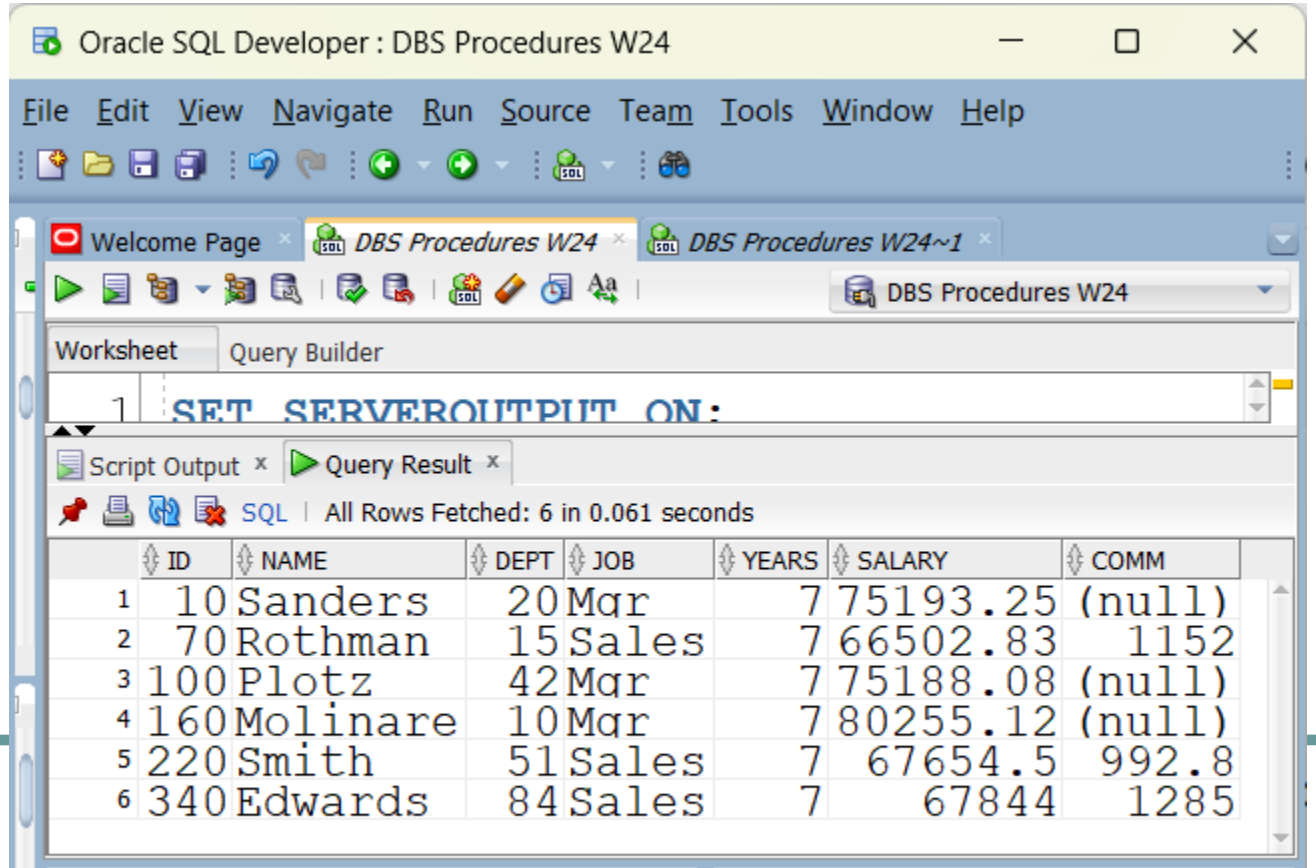
DBS Procedures W24

```
1  create or replace PROCEDURE UPDATE_EMPLOYEES
2          (YEARSIN IN NUMBER)
3     AS
4     BEGIN
5       UPDATE STAFF2
6       SET    COMM = 100
7       WHERE  YEARS = YEARSIN;
8       IF SQL%ROWCOUNT = 0 THEN
9          DBMS_OUTPUT.PUT_LINE ('No employee was updated.');
10      ELSIF SQL%ROWCOUNT = 1 THEN
11         DBMS_OUTPUT.PUT_LINE ('One employee was updated.');
12      ELSE
13         DBMS_OUTPUT.PUT_LINE ('More than one employee was updated!');
14         DBMS_OUTPUT.PUT_LINE (SQL%ROWCOUNT || ' employees were updated');
15      END IF;
16    EXCEPTION
17      WHEN OTHERS
18      THEN
19        DBMS_OUTPUT.PUT_LINE ('Update Error!');
20    END;
```

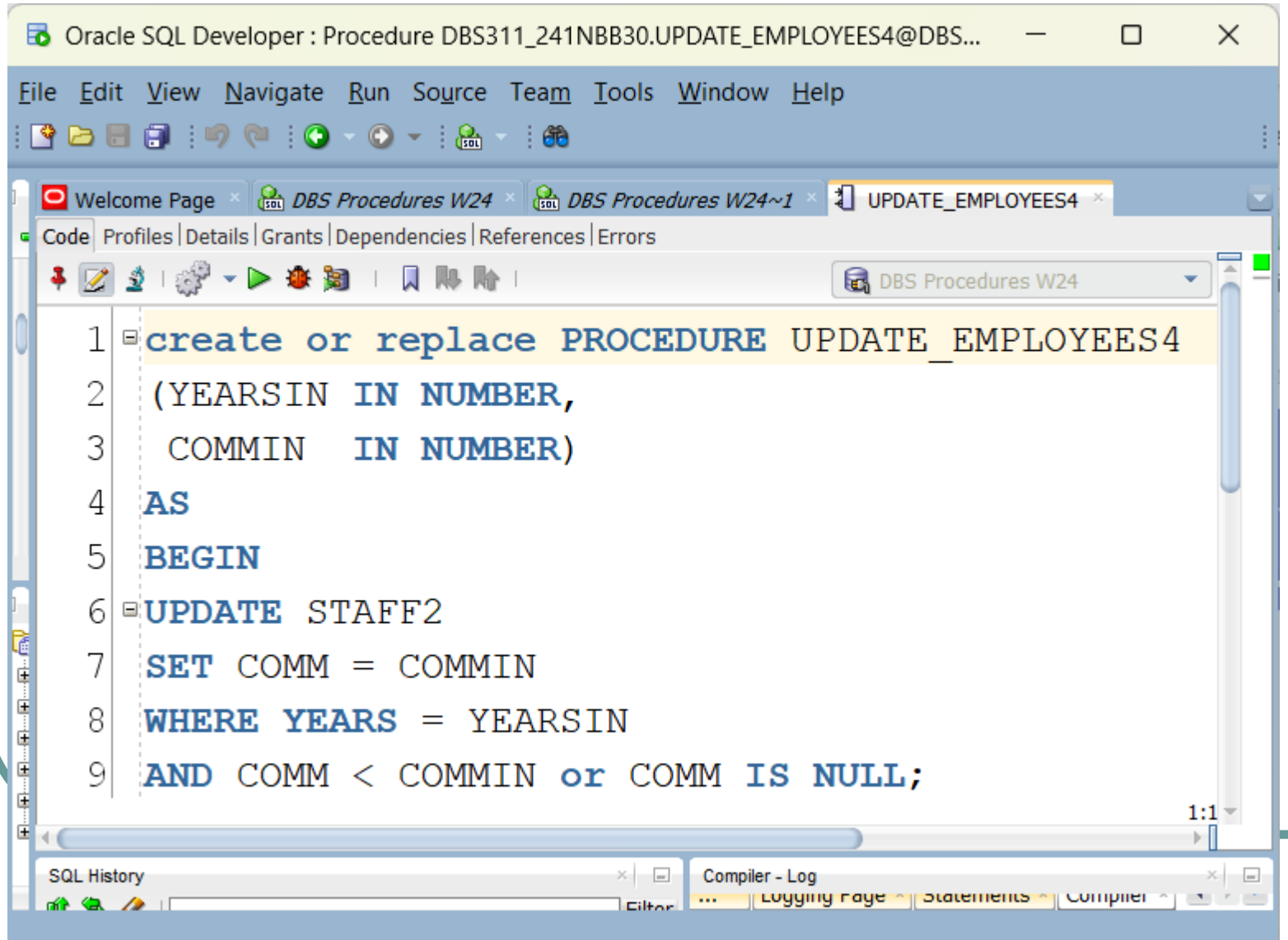
1:1


Oracle SQL Developer : Procedure DBS311_241NBB30.UPDATE_EMPLOYEES2@DBS...
File Edit View Navigate Run Source Team Tools Window Help
...24
DBS Procedures W24~1
UPDATE_EMPLOYEES
UPDATE_EMPLOYEES2
Code Profiles Details Grants Dependencies References Errors
DBS Procedures W24
1 create or replace PROCEDURE UPDATE_EMPLOYEES2
2 (YEARSIN IN NUMBER,
3 COMMIN IN NUMBER)
4 AS
5 BEGIN
6 UPDATE STAFF2
7 SET COMM = COMMIN
8 WHERE YEARS = YEARSIN
9 AND COMM < COMMIN;
PROCEDURE UPDATE_EMPLOYEES2 EXCEPTION
22:4
SQL History
Statements - Log

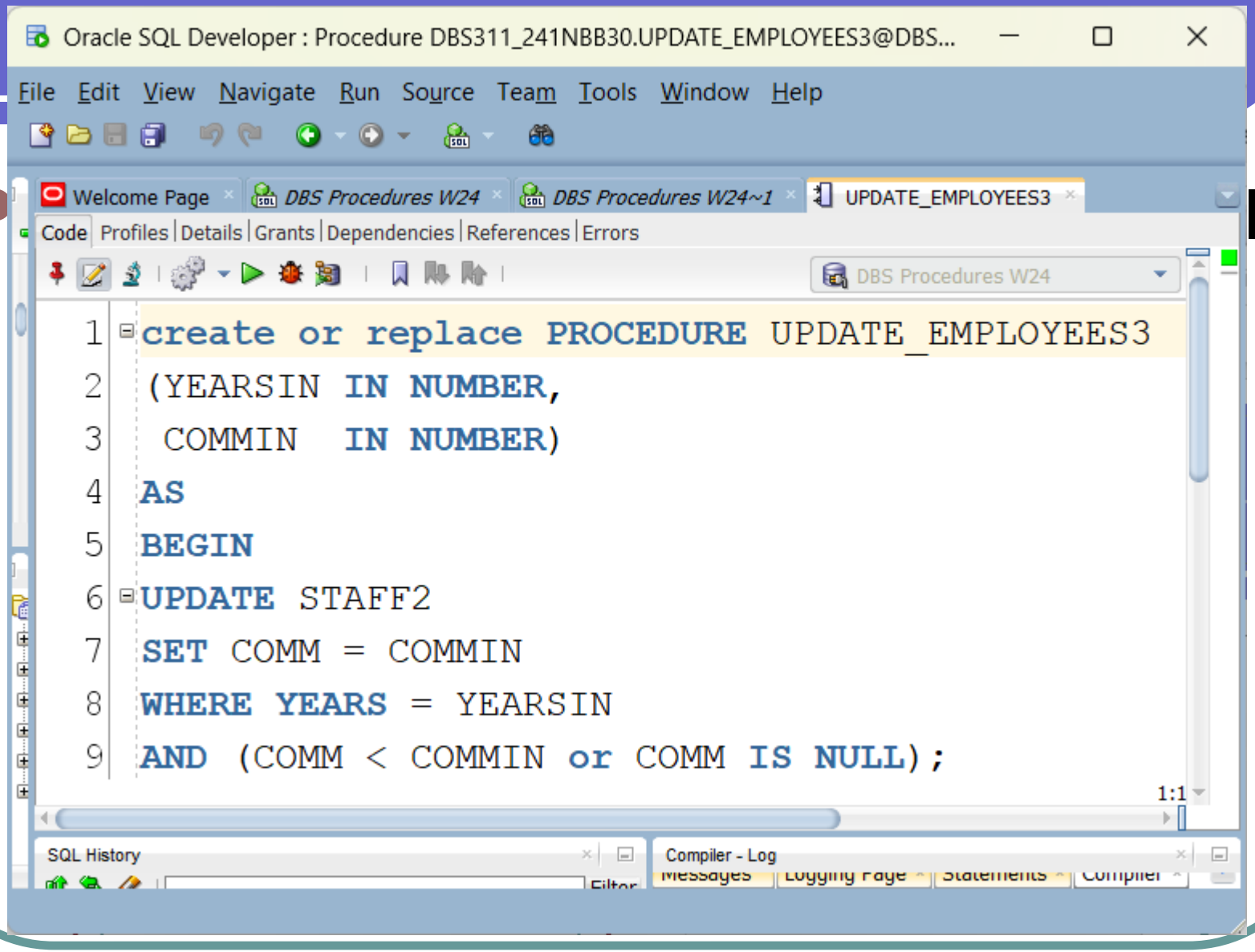
- No employee was updated.
- Why?



	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	10	Sanders	20	Mgr	7	75193.25	(null)
2	70	Rothman	15	Sales	7	66502.83	1152
3	100	Plotz	42	Mgr	7	75188.08	(null)
4	160	Molinare	10	Mgr	7	80255.12	(null)
5	220	Smith	51	Sales	7	67654.5	992.8
6	340	Edwards	84	Sales	7	67844	1285


Oracle SQL Developer : Procedure DBS311_241NBB30.UPDATE_EMPLOYEES4@DBS...
File Edit View Navigate Run Source Team Tools Window Help
Welcome Page
DBS Procedures W24
DBS Procedures W24~1
UPDATE_EMPLOYEES4
Code Profiles Details Grants Dependencies References Errors
DBS Procedures W24
1 create or replace PROCEDURE UPDATE_EMPLOYEES4
2 (YEARSIN IN NUMBER,
3 COMMIN IN NUMBER)
4 AS
5 BEGIN
6 UPDATE STAFF2
7 SET COMM = COMMIN
8 WHERE YEARS = YEARSIN
9 AND COMM < COMMIN or COMM IS NULL;
1:1
SQL History
Compiler - Log

- We get
- More than one employee was updated!
- 11 employees were updated
- We did not have 11 employees with 7 years – what happened?


Oracle SQL Developer : Procedure DBS311_241NBB30.UPDATE_EMPLOYEES3@DBS...
File Edit View Navigate Run Source Team Tools Window Help
Welcome Page
DBS Procedures W24
DBS Procedures W24~1
UPDATE_EMPLOYEES3
Code Profiles Details Grants Dependencies References Errors
DBS Procedures W24
1 create or replace PROCEDURE UPDATE_EMPLOYEES3
2 (YEARSIN IN NUMBER,
3 COMMIN IN NUMBER)
4 AS
5 BEGIN
6 UPDATE STAFF2
7 SET COMM = COMMIN
8 WHERE YEARS = YEARSIN
9 AND (COMM < COMMIN or COMM IS NULL);
1:1
SQL History
Compiler - Log

Convert Days to Weeks & Days

- With DAYS set to 20
- There are 2.857142857142857142857142857142857142860 weeks
- and 6 days

- DECLARE
- DAYS NUMBER(7) := 20;
-
- BEGIN
- DBMS_OUTPUT.PUT_LINE ('With DAYS set to ' || DAYS);
- DBMS_OUTPUT.PUT_LINE ('There are ' || DAYS/7 || ' weeks');
- DBMS_OUTPUT.PUT_LINE ('and ' || Mod(DAYS,7) || ' days');
- END;

- Change 2.8571428571428571428571428571428571428586 weeks
- To 2 weeks
- DBMS_OUTPUT.PUT_LINE
- ('There are ' || trunc(DAYS/7,0) || ' weeks');

Triggers

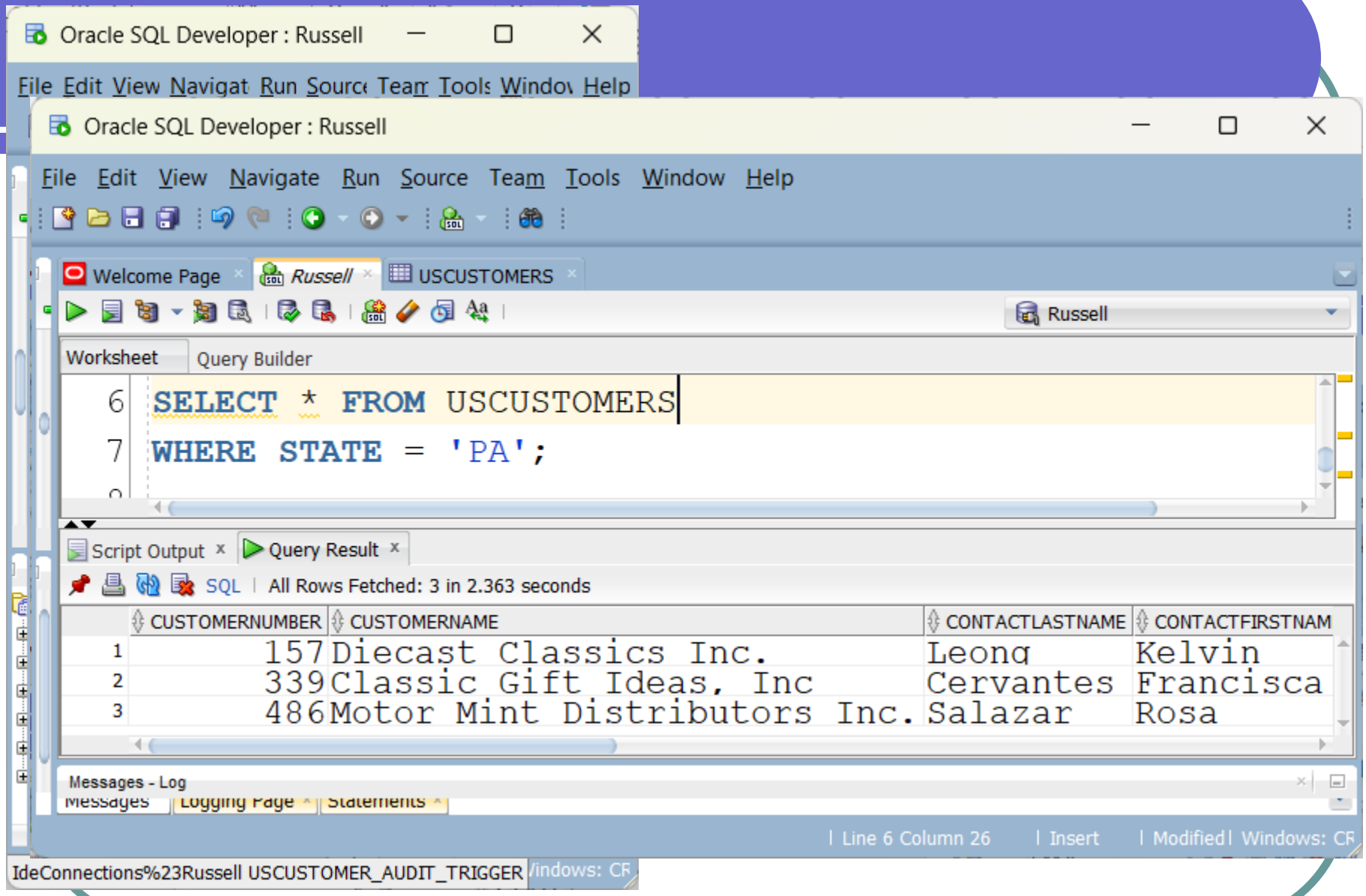
- A trigger is a named PL/SQL block stored in the Oracle Database and executed automatically when a triggering event takes place.
- A DML statement executed against a table can cause the trigger to fire
- A trigger can be set to fire before or after an Update, Insert or Delete and is applied to a table depending what is considered needed for the situation.
- Uses for a Trigger
- for a startup or shutdown of the Oracle Database
- for a user event like a log in or log out.
- Some situations where business rules are more complex then the
- traditional constraints of Unique, Not Null and Check.
- to prevent an invalid transaction
- to Gather statistical information on table access
- to Audit sensitive data

How to Create a Trigger in Oracle

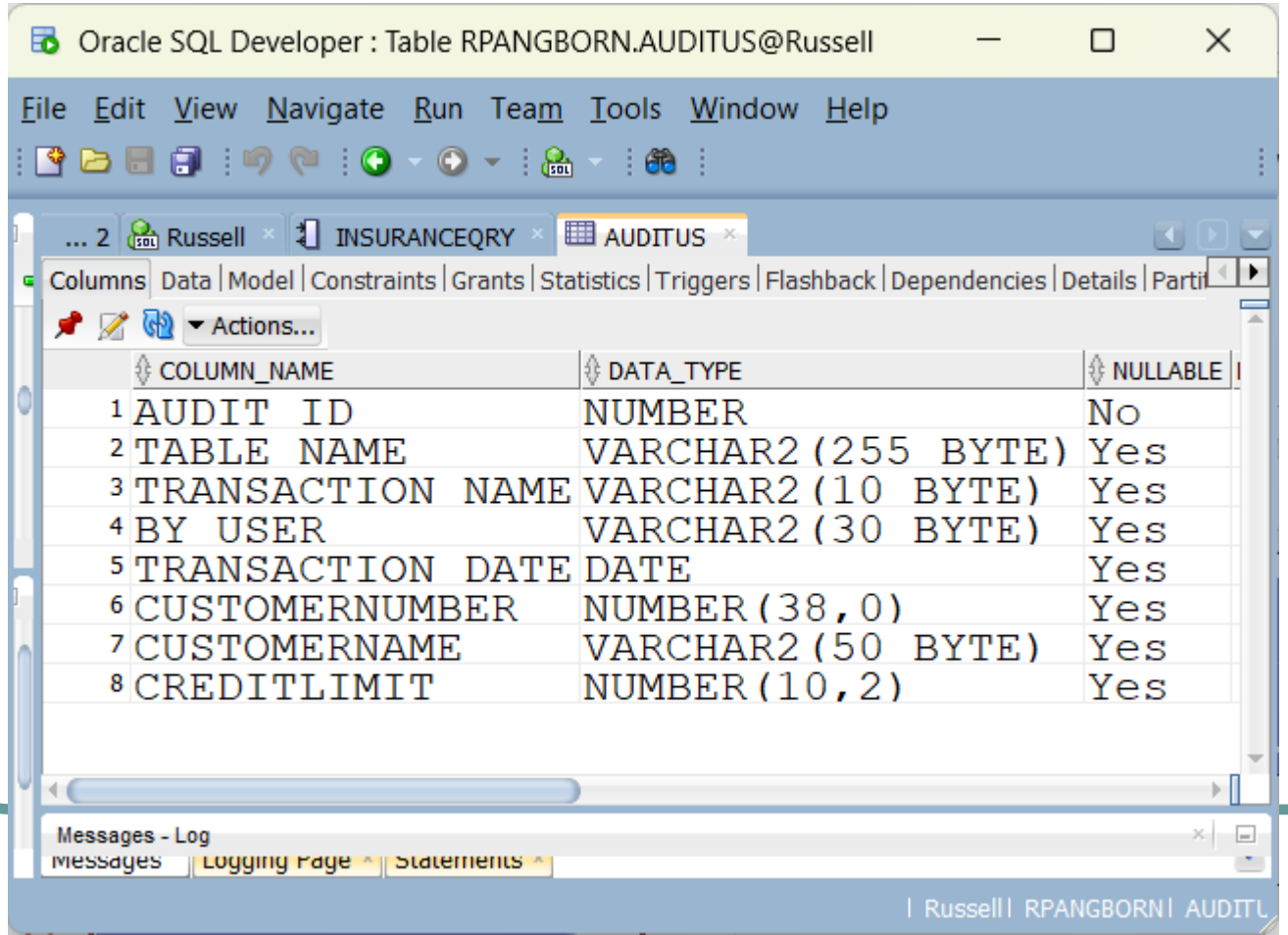
- CREATE [OR REPLACE] TRIGGER trigger_name
- {BEFORE | AFTER} triggering_event ON table_name
- [FOR EACH ROW]
- [FOLLOWS | PRECEDES another_trigger]
- [ENABLE / DISABLE]
- [WHEN condition]
- DECLARE
- declaration_statement
- BEGIN
- executable statements
- EXCEPTION
- exeption_handling statements
- END;

Table to store trigger events

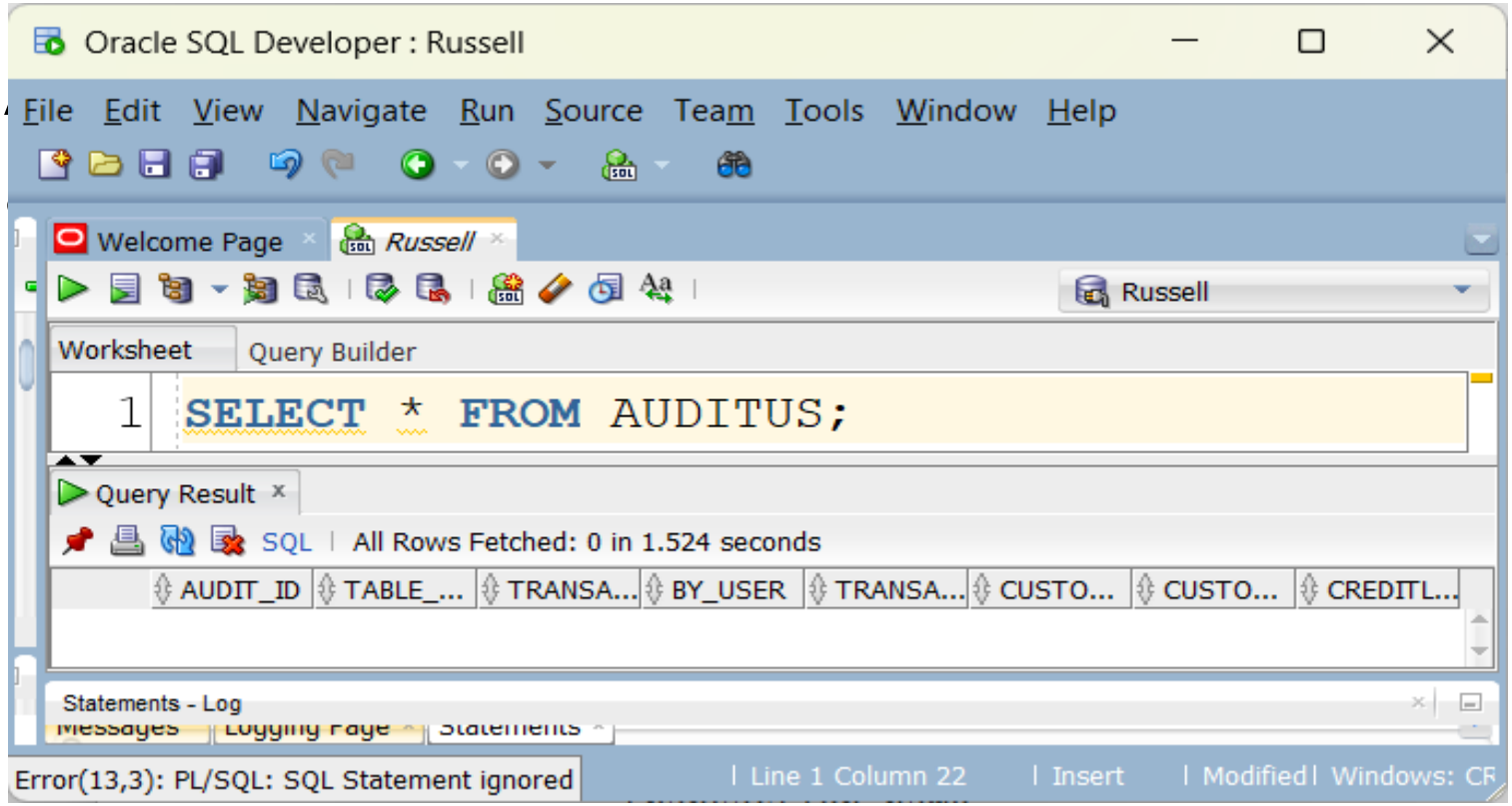
```
CREATE TABLE AUDITS (
   audit_id number generated by default
                    as identity PRIMARY KEY,
   table_name          VARCHAR2(255),
   transaction_name VARCHAR2(10),
   by_user             VARCHAR2(30),
   transaction_date   DATE
);
```


Oracle SQL Developer : Russell
File Edit View Navigat Run Sourc Team Tools Windov Help
Oracle SQL Developer : Russell
File Edit View Navigate Run Source Team Tools Window Help
Welcome Page
Russell
USCUSTOMERS
Russell
Worksheet
Query Builder
6 SELECT * FROM USCUSTOMERS
7 WHERE STATE = 'PA';
Script Output
Query Result
SQL | All Rows Fetched: 3 in 2.363 seconds
CUSTOMERNUMBER CUSTOMERNAME CONTACTLASTNAME CONTACTFIRSTNAM
1 157 Diecast Classics Inc. Leong Kelvin
2 339 Classic Gift Ideas, Inc Cervantes Francisca
3 486 Motor Mint Distributors Inc. Salazar Rosa
Messages - Log
Line 6 Column 26 | Insert | Modified | Windows: CR
IdeConnections%23Russell USCUSTOMER_AUDIT_TRIGGER

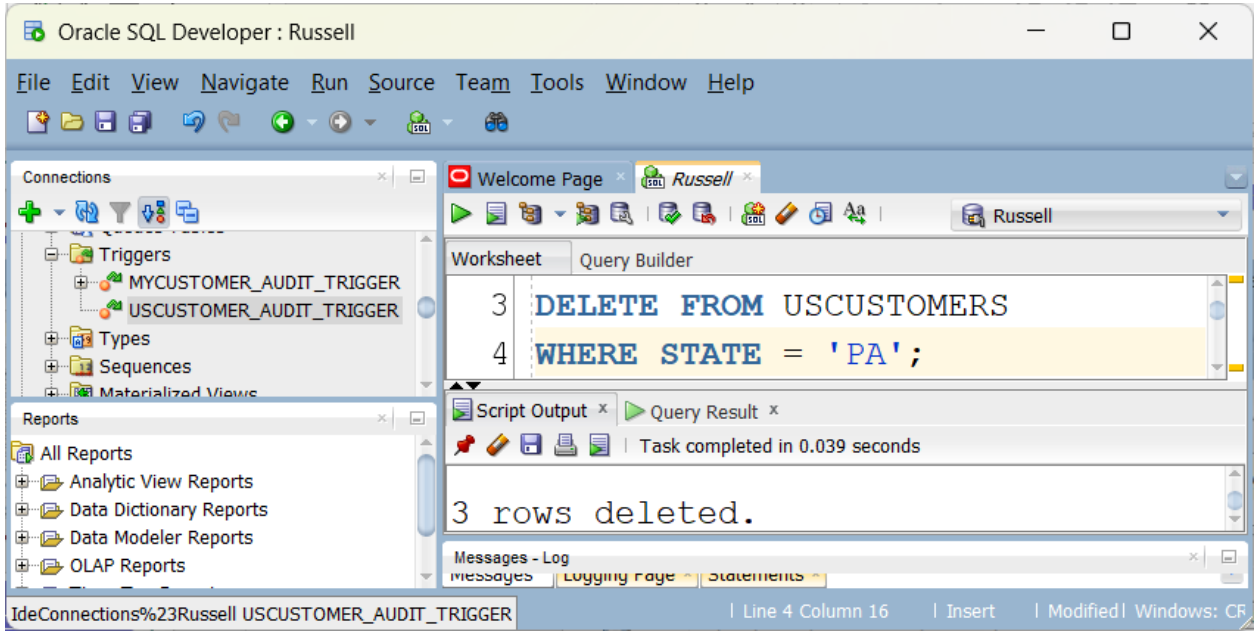
- Columns to monitor added to audit table



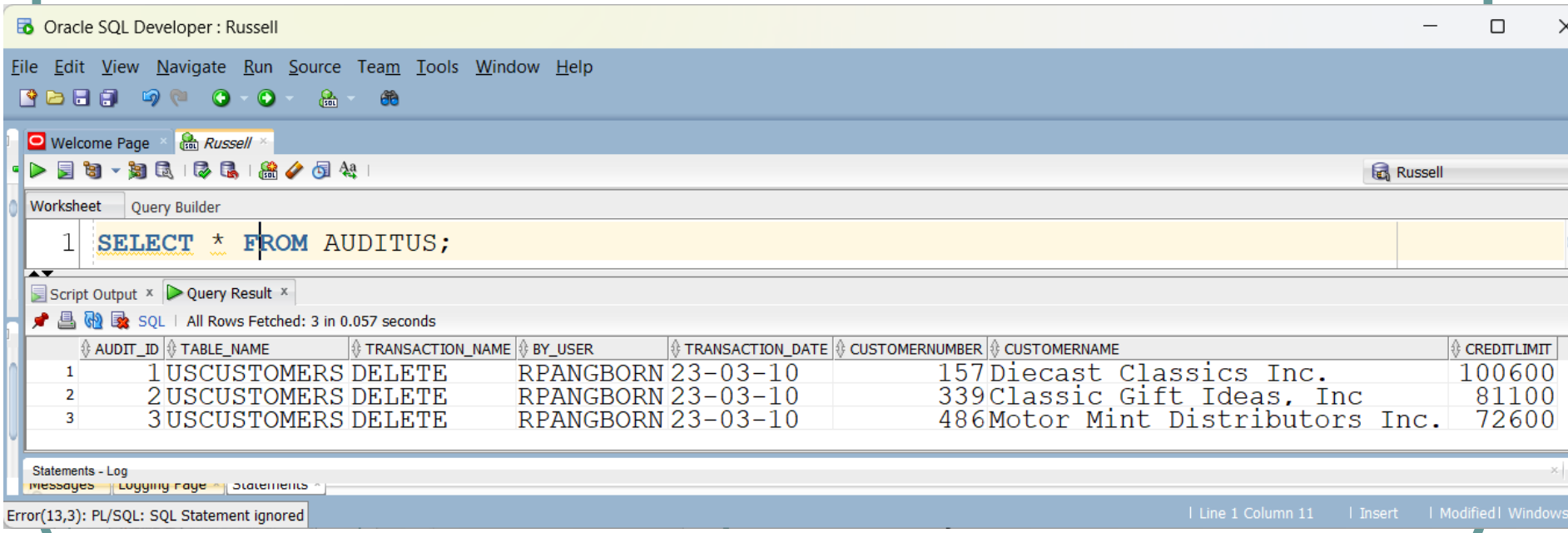
USCUSTOMER TRIGGER



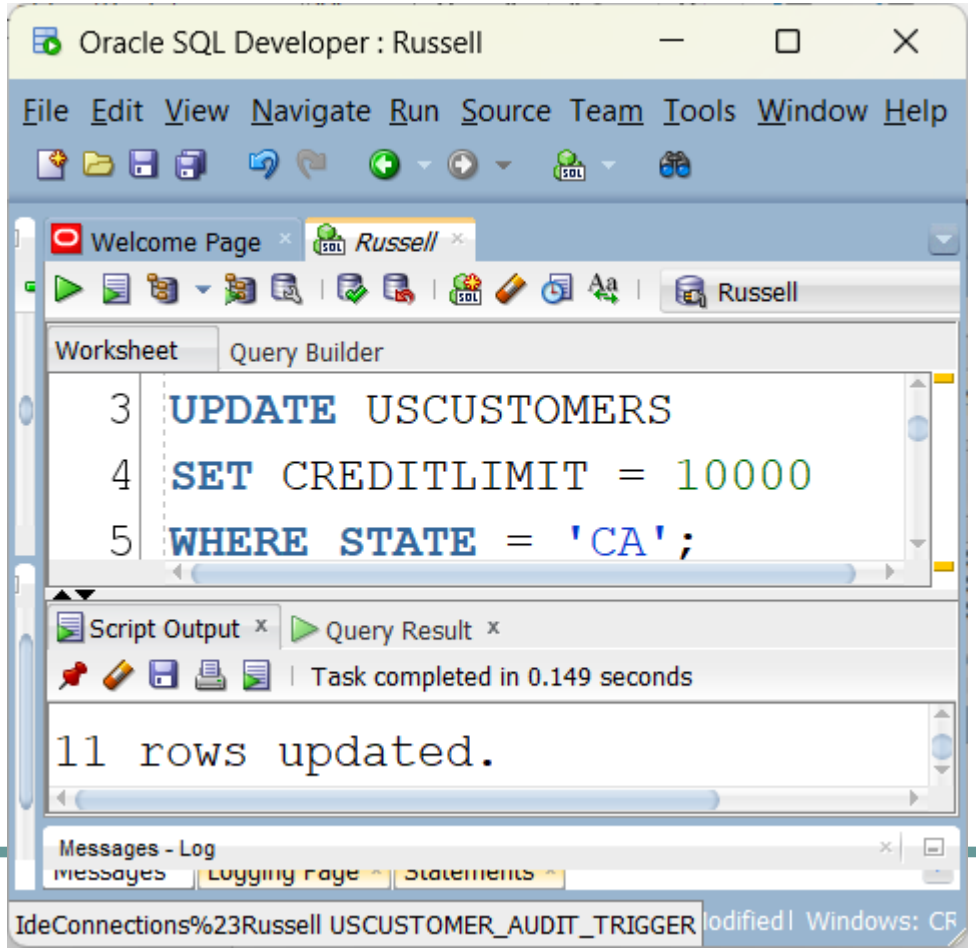
- The trigger fires when rows are deleted from the USCUSTOMER table



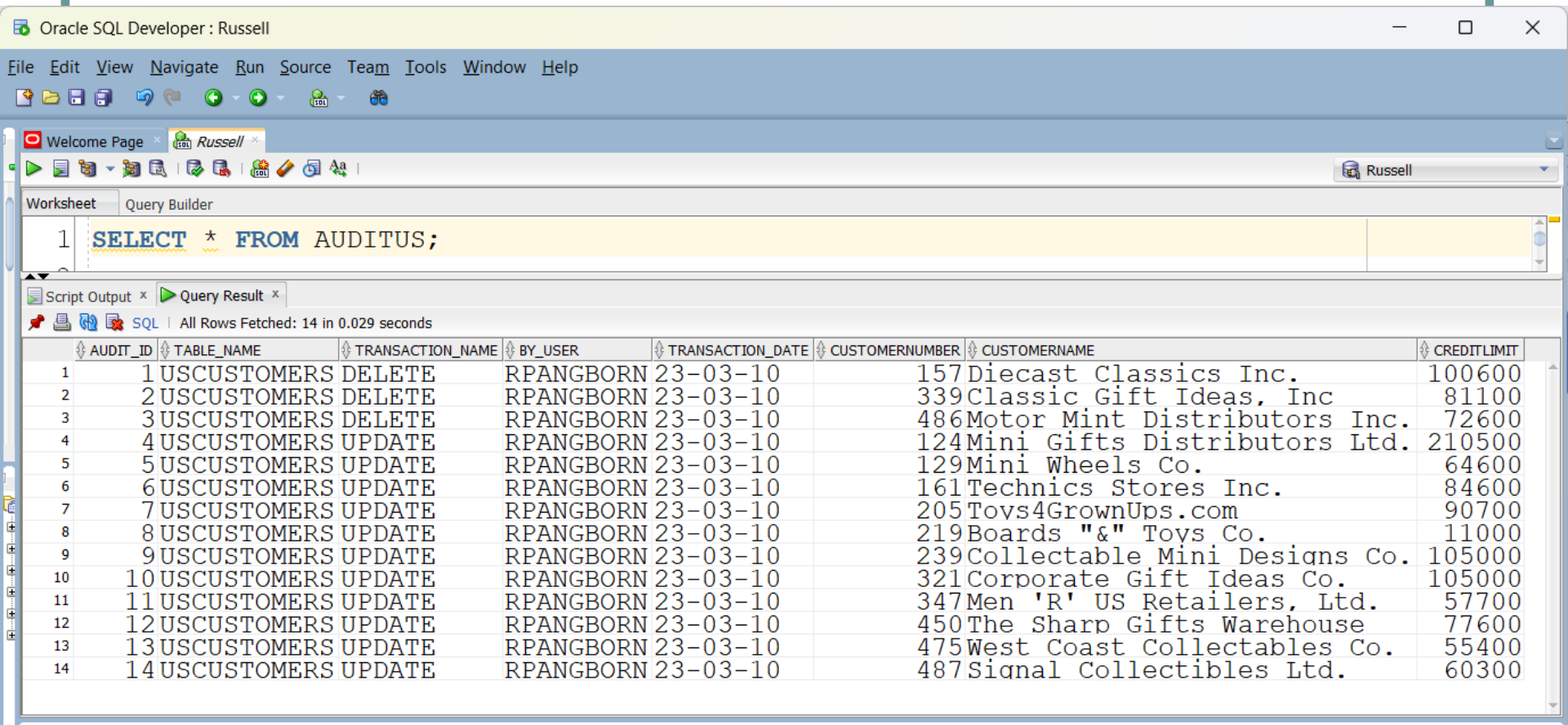
- The audit table reflects the deletions that occurred



- An update will also cause the trigger to fire

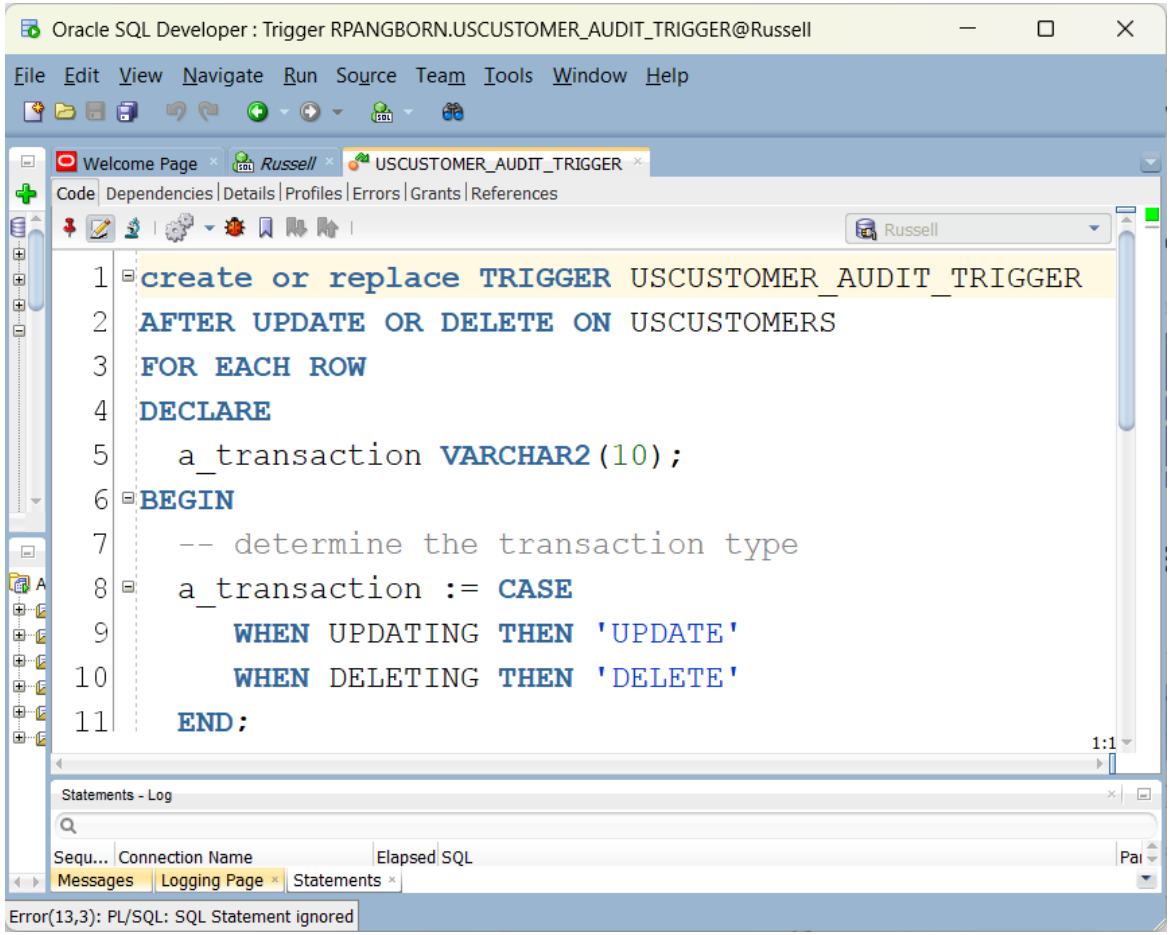


- More entries are inserted into the audit table after the update

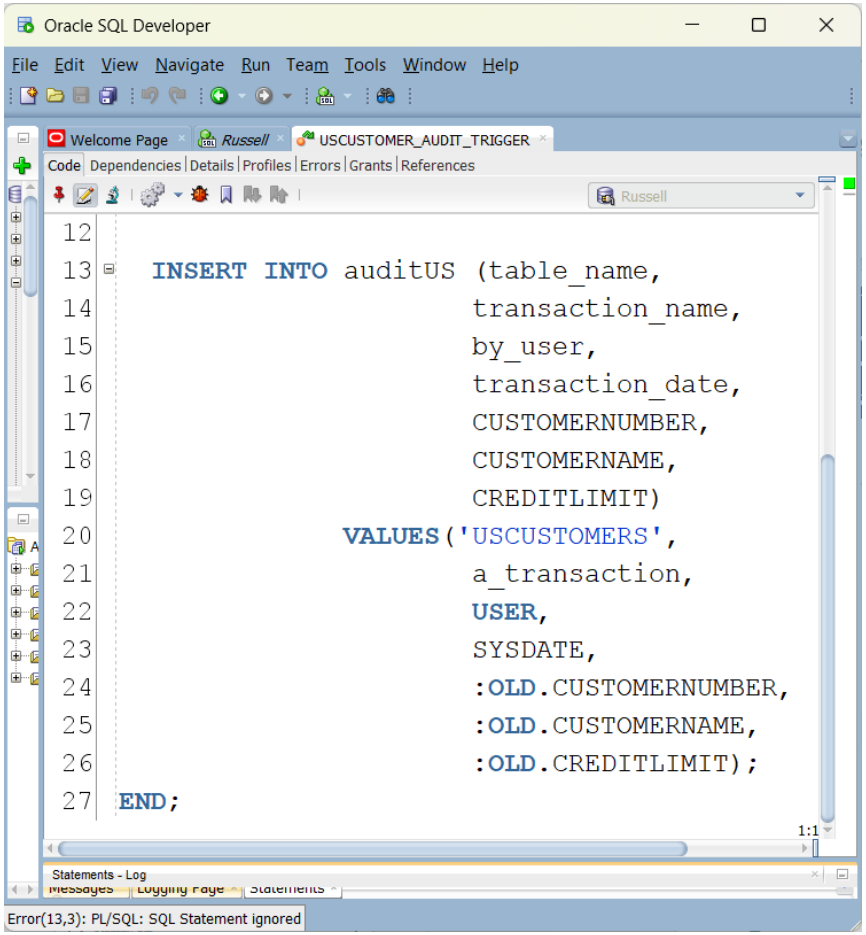


	AUDIT_ID	TABLE_NAME	TRANSACTION_NAME	BY_USER	TRANSACTION_DATE	CUSTOMERNUMBER	CUSTOMERNAME	CREDITLIMIT
1	1	USCUSTOMERS	DELETE	RPANGBORN	23-03-10	157	Diecast Classics Inc.	100600
2	2	USCUSTOMERS	DELETE	RPANGBORN	23-03-10	339	Classic Gift Ideas, Inc	81100
3	3	USCUSTOMERS	DELETE	RPANGBORN	23-03-10	486	Motor Mint Distributors Inc.	72600
4	4	USCUSTOMERS	UPDATE	RPANGBORN	23-03-10	124	Mini Gifts Distributors Ltd.	210500
5	5	USCUSTOMERS	UPDATE	RPANGBORN	23-03-10	129	Mini Wheels Co.	64600
6	6	USCUSTOMERS	UPDATE	RPANGBORN	23-03-10	161	Technics Stores Inc.	84600
7	7	USCUSTOMERS	UPDATE	RPANGBORN	23-03-10	205	Toys4GrownUps.com	90700
8	8	USCUSTOMERS	UPDATE	RPANGBORN	23-03-10	219	Boards "&" Toys Co.	11000
9	9	USCUSTOMERS	UPDATE	RPANGBORN	23-03-10	239	Collectable Mini Designs Co.	105000
10	10	USCUSTOMERS	UPDATE	RPANGBORN	23-03-10	321	Corporate Gift Ideas Co.	105000
11	11	USCUSTOMERS	UPDATE	RPANGBORN	23-03-10	347	Men 'R' US Retailers, Ltd.	57700
12	12	USCUSTOMERS	UPDATE	RPANGBORN	23-03-10	450	The Sharp Gifts Warehouse	77600
13	13	USCUSTOMERS	UPDATE	RPANGBORN	23-03-10	475	West Coast Collectables Co.	55400
14	14	USCUSTOMERS	UPDATE	RPANGBORN	23-03-10	487	Signal Collectibles Ltd.	60300

The Trigger


Oracle SQL Developer : Trigger RPANGBORN.USCUSTOMER_AUDIT_TRIGGER@Russell
File Edit View Navigate Run Source Team Tools Window Help
Welcome Page
Russell
USCUSTOMER_AUDIT_TRIGGER
Code Dependencies Details Profiles Errors Grants References
Russell
1 create or replace TRIGGER USCUSTOMER_AUDIT_TRIGGER
2 AFTER UPDATE OR DELETE ON USCUSTOMERS
3 FOR EACH ROW
4 DECLARE
5 a_transaction VARCHAR2(10);
6 BEGIN
7 -- determine the transaction type
8 a_transaction := CASE
9 WHEN UPDATING THEN 'UPDATE'
10 WHEN DELETING THEN 'DELETE'
11 END;
1:1
Statements - Log
Sequ... Connection Name Elapsed SQL
Messages Logging Page Statements
Error(13,3): PL/SQL: SQL Statement ignored

The Trigger (continued)



Trigger OLD vs NEW

INSERT- :old.value = NULL,
:new.value = post insert value

DELETE- :old.value = Pre Delete value,
:new.value = null

UPDATE- :old.value = Pre update value,
:new.value = Post Update value

Iteration Statements

- A LOOP statement runs a series of statements multiple times.
- Basic LOOP
- FOR LOOP
- Cursor FOR LOOP
- WHILE LOOP
- Statements to exit a loop:
- EXIT
- EXIT WHEN

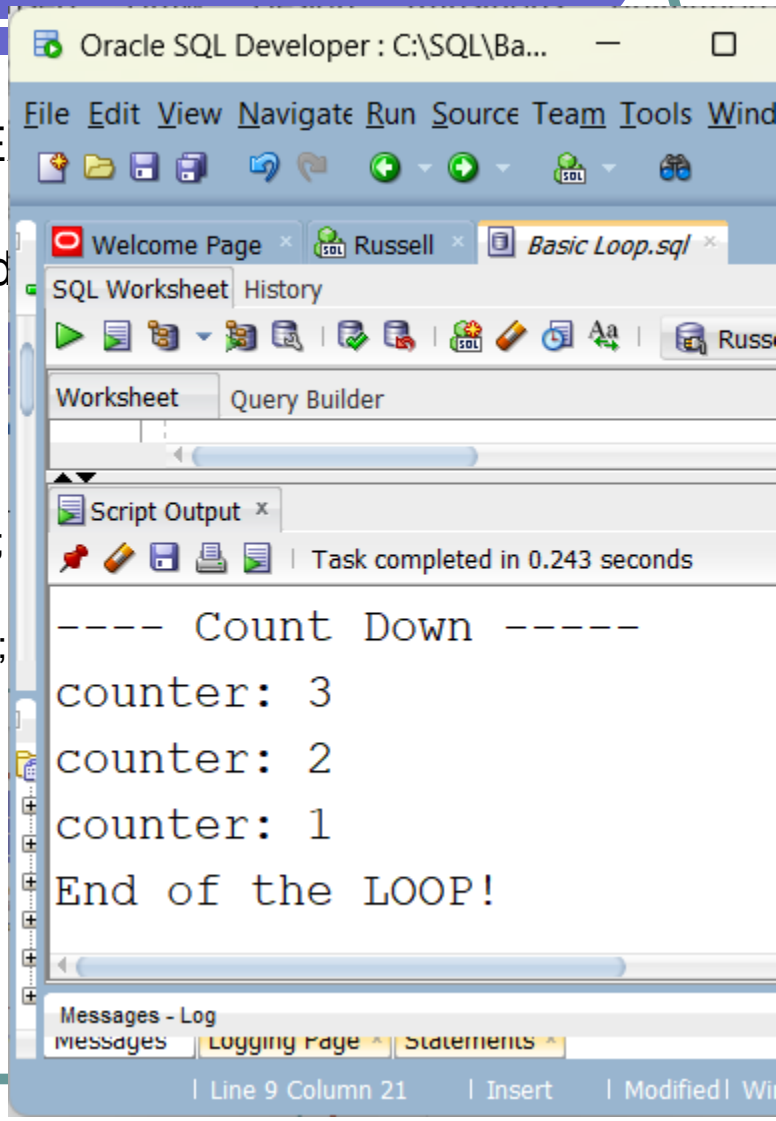
Iteration Statements

- The statements that exit the current iteration of a loop and skip to the next iteration.
- CONTINUE
- CONTINUE WHEN

Basic LOOP Statements

- The loop executes the statements until an E
  loop execution or an exception is raised.
- The EXIT statement terminates the loop and
  end of the current loop.

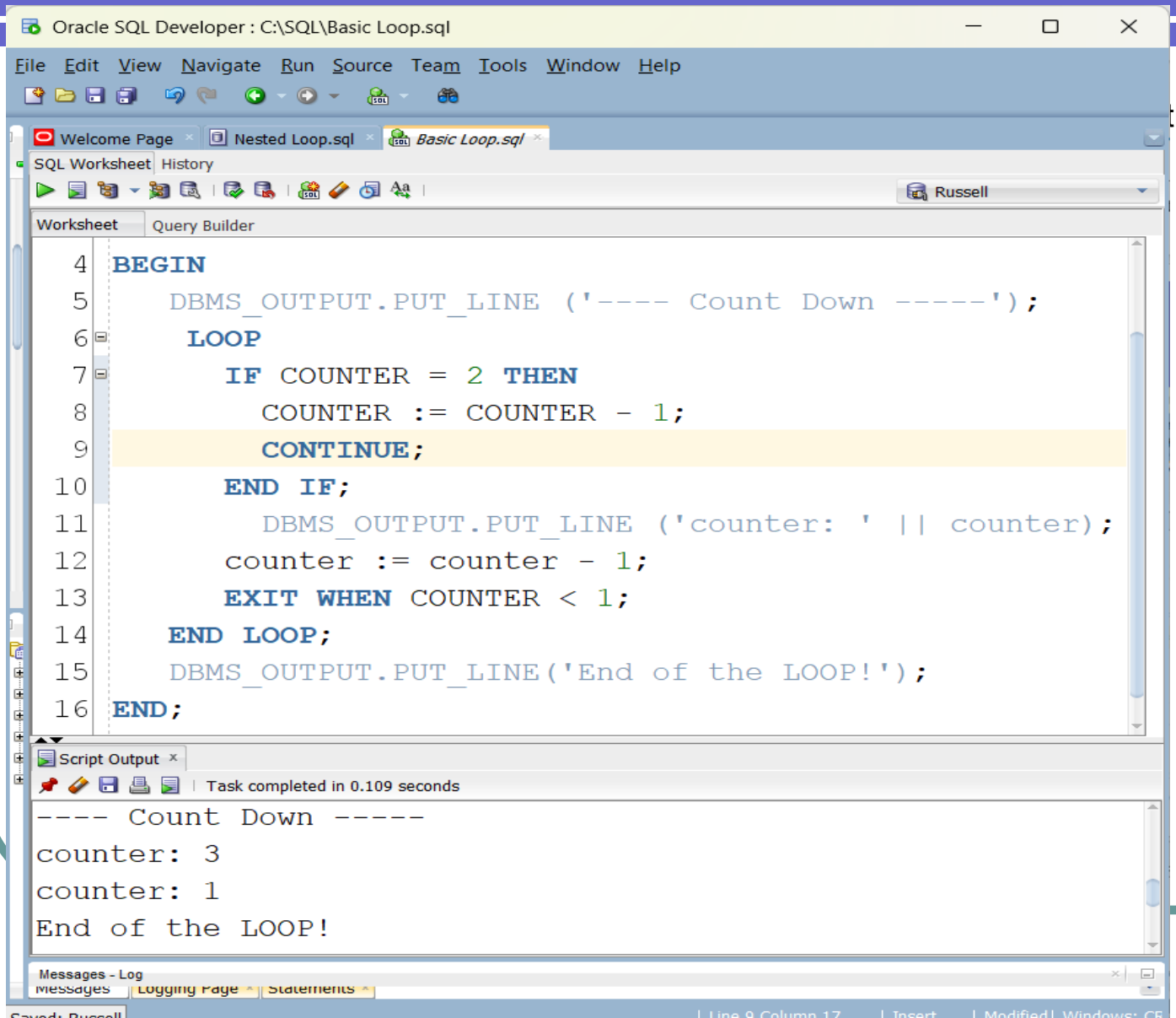
```
DECLARE
counter NUMBER := 3;
  BEGIN
    DBMS_OUTPUT.PUT_LINE ('---- Count Down -----');
    LOOP
      DBMS_OUTPUT.PUT_LINE ('counter: ' || counter);
      counter := counter - 1;
      IF counter < 1 THEN
        EXIT;
      END IF;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('End of the LOOP!');
END;
```



EXIT WHEN Statement

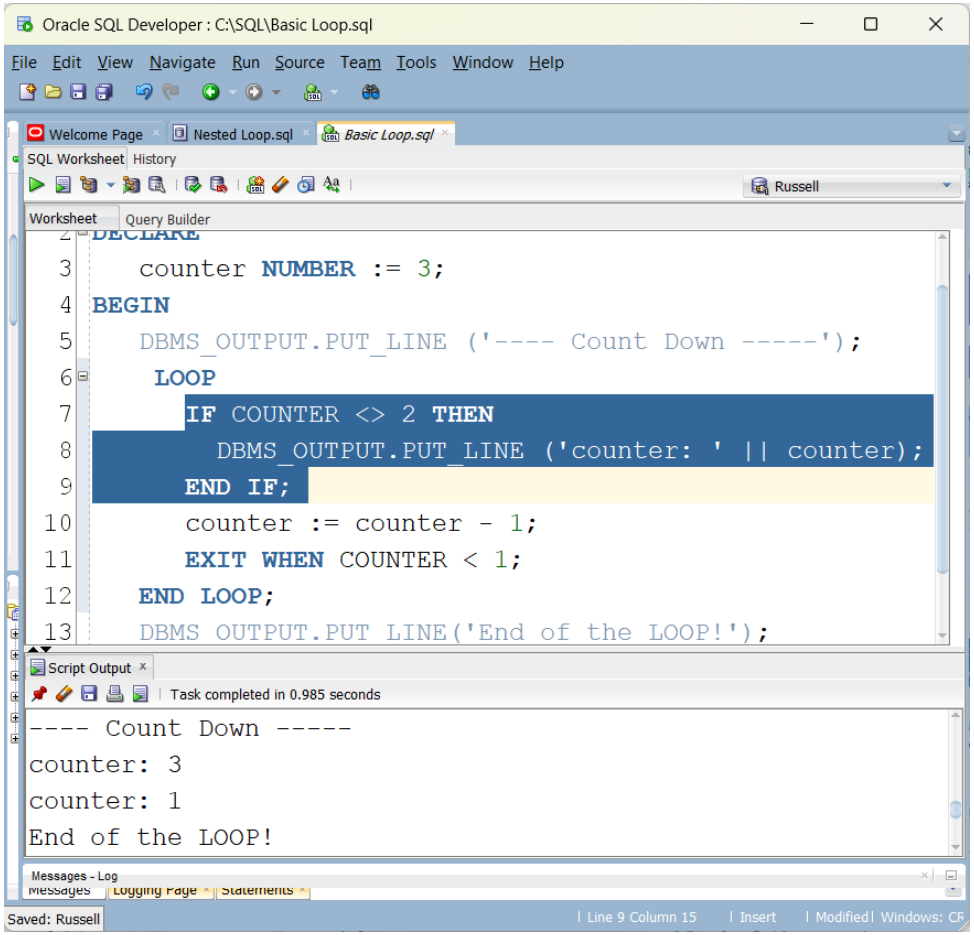
```
DECLARE
   counter NUMBER := 3;
BEGIN
   DBMS_OUTPUT.PUT_LINE ('---- Count Down -----');
   LOOP
      DBMS_OUTPUT.PUT_LINE ('counter: ' || counter);
      counter := counter - 1;
      EXIT WHEN counter < 1;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('End of the LOOP!');
END;
```

CONTINUE Statement



the next

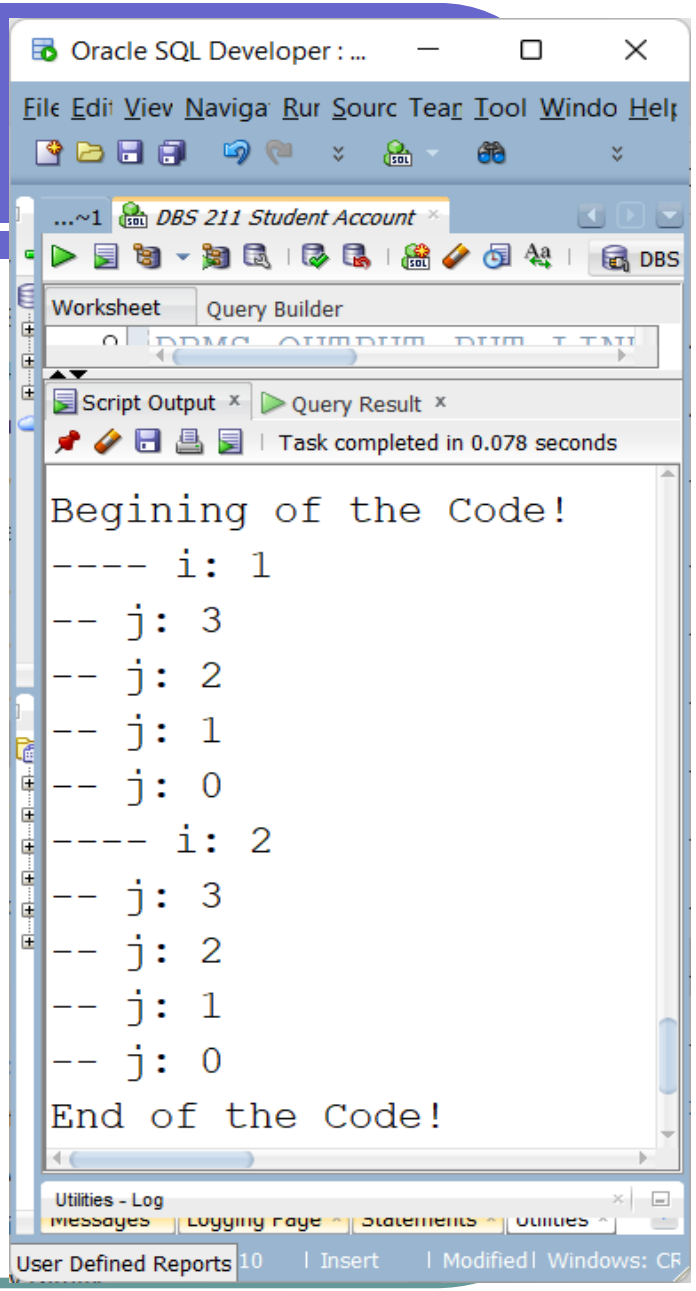
Avoiding use of Continue



- A LOOP statement can be inside another LOOP statement. The EXIT statement inside the inner LOOP exits the inner LOOP and transfers the control to the outer loop.

Nested Loop

```
DECLARE
i NUMBER := 0;
j NUMBER := 2;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Begining of the Code!');
  LOOP
    i := i + 1;
    DBMS_OUTPUT.PUT_LINE ('---- i: ' || i);
    j:= 3;
    LOOP
      DBMS_OUTPUT.PUT_LINE ('-- j: ' || j);
      j := j - 1;
      EXIT WHEN j < 0;
    END LOOP;
    EXIT WHEN i > 1;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('End of the Code!');
END;
```



FOR LOOP Statement

- The FOR LOOP statement executes the statements inside the loop while the value of the loop index is in a given range.

```
FOR index IN [ REVERSE ] lower_bound.. upper_bound LOOP
statements
END LOOP;
```

- By default, the value of the index starts from the lower bound value and increases by one until it becomes equal to the upper bound value.
- IF you include the REVERSE keyword, the value of index starts from the upper bound value and deceases by one until it becomes equal to the lower bound value.
- The upper bound value must be greater than or equal to the lower bound value.
- Index is the local variable of the FOR loop.

FOR LOOP Example

```
BEGIN
   FOR i IN 1..4 LOOP
      IF i < 2 THEN
         DBMS_OUTPUT.PUT_LINE (i || ' is less than 2');
      ELSIF i > 2 THEN
         DBMS_OUTPUT.PUT_LINE (i || ' is greater than 2');
      ELSE
         DBMS_OUTPUT.PUT_LINE (i || ' is equal to 2');
      END IF;
   END LOOP;
END;
```

- Output
- 1 is less than 2
- 2 is equal to 2
- 3 is greater than 2
- 4 is greater than 2

Nested FOR LOOP Statements

- A FOR LOOP (inner loop) statement can be inside another FOR LOOP (outer loop). The inner loop executes until the its index reaches the terminating value or an EXIT statement is executed. The control then will be given to the outer FOR loop.

```
BEGIN
   FOR i IN 1..2 LOOP
      DBMS_OUTPUT.PUT_LINE ('---- i: ' || i);
      FOR j IN REVERSE 1..4 LOOP
         DBMS_OUTPUT.PUT_LINE ('-- j: ' || j);
      END LOOP;
   END LOOP;
END;
```

```
---- i: 1
-- j: 4
-- j: 3
-- j: 2
-- j: 1
---- i: 2
-- j: 4
-- j: 3
-- j: 2
-- j: 1
```

WHILE LOOP Statement

- The WHILE loop executes the statements inside the loop as long as the loop condition is true. If the loop condition is false or an EXIT statement is executed, the control will be transferred to the next statement after the WHILE loop.

```
WHILE condition LOOP
statements
END LOOP;
```

- EXIT, EXIT WHEN, CONTINUE, or CONTINUE WHEN statements can be used inside a WHILE loop to terminate the current loop or the current iteration early.

```
WHILE condition LOOP
   statements
   [CONTINUE WHEN condition;]
   [EXIT WHEN condition;]
END LOOP;
```

WHILE LOOP Example

- DECLARE
- run BOOLEAN := true;
- round NUMBER := 1;

```
BEGIN
   DBMS_OUTPUT.PUT_LINE ('-- First WHILE LOOP --');
   WHILE run LOOP
      DBMS_OUTPUT.PUT_LINE ('round ' || round);
      round := round + 1;
      IF round = 4 THEN
         run := false;
      END IF;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE ('-- Second WHILE LOOP --');
   WHILE NOT run LOOP
      DBMS_OUTPUT.PUT_LINE ('round ' || round);
      round := round - 1;
      IF round = 0 THEN
         run := true;
      END IF;
   END LOOP;
END;
```

```
-- First WHILE LOOP --
round 1
round 2
round 3
-- Second WHILE LOOP --
round 4
round 3
round 2
round 1
```

Upcoming Work

- Lab 6 has been posted as an Oracle lab
- Assignment 2 has been released
- Lab 5 is due after the break week
- No lab this week – a work period, work on Question 1 of the assignment or on lab 5
- Work period in lecture period after the break, work on Question 2 of the assignment or lab 6
- Next lecture is on Functions & Cursors