

# 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

The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : C:\SQL\DBS311\Lecture 6\Case in Select Statement.sql". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations and SQL navigation. The tabs at the top show "Welcome Page", "DBS 311 W 2025 C Class", "Case in Select Statement.sql", and "Case ...". The main workspace is a "Worksheet" tab, showing the following SQL code:

```
1  SELECT CASE JOB
2      WHEN 'Mgr' then 'Manager'
3      WHEN 'Sales' then 'Sales Person'
4      Else 'Clerical'
5  END JOB,
6      NAME
7  FROM STAFF;
```

The code uses a CASE statement to map job titles ('Mgr', 'Sales') to their descriptions ('Manager', 'Sales Person') and handles other cases ('Else') as 'Clerical'. The "Query Result" tab below displays the resulting data:

	JOB	NAME
1	Manager	Sanders
2	Sales Person	Pernal
3	Manager	Marenghi
4	Sales Person	O'Brien
5	Manager	Hanes
6	Sales Person	Ouigley
7	Sales Person	Rothman
8	Clerical	James
9	Sales Person	Koonitz

The status bar at the bottom indicates "All Rows Fetched: 35 in 1.584 seconds".

# CASE in a SELECT statement

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a small red circular icon with a white exclamation mark.

The main window displays a SQL worksheet with the following code:

```
1
2 SELECT NAME, DEPT, JOB,
3       CASE
4           WHEN YEARS IS NULL THEN 'UNKNOWN'
5           ELSE CAST(YEARS AS VARCHAR (9))
6       END YEARS
7 FROM STAFF;
```

Below the worksheet is a "Query Result" tab showing the output of the query:

NAME	DEPT	JOB	YEARS
Sanders	20	Mar	7
Pernal	20	Sales	8
Marenghi	38	Mar	5
O'Brien	38	Sales	6
Hanes	15	Mar	10
Quigley	38	Sales	UNKNOWN
Rothman	15	Sales	7
James	20	Clerk	UNKNOWN
Koonitz	42	Sales	6
Plotz	42	Mar	7
Ngan	15	Clerk	5
Naughton	38	Clerk	UNKNOWN
Yamauchi	42	Clerk	6
Frave	51	Mar	6
Williams	51	Sales	6
Molinare	10	Mar	7
Kormisch	15	Clerk	1

At the bottom of the interface, there are tabs for "Messages - Log", "Logging Page", and "Statements". There are also status indicators for "Line 1 Column 1", "Insert", "Modified", and "Windows: CR".

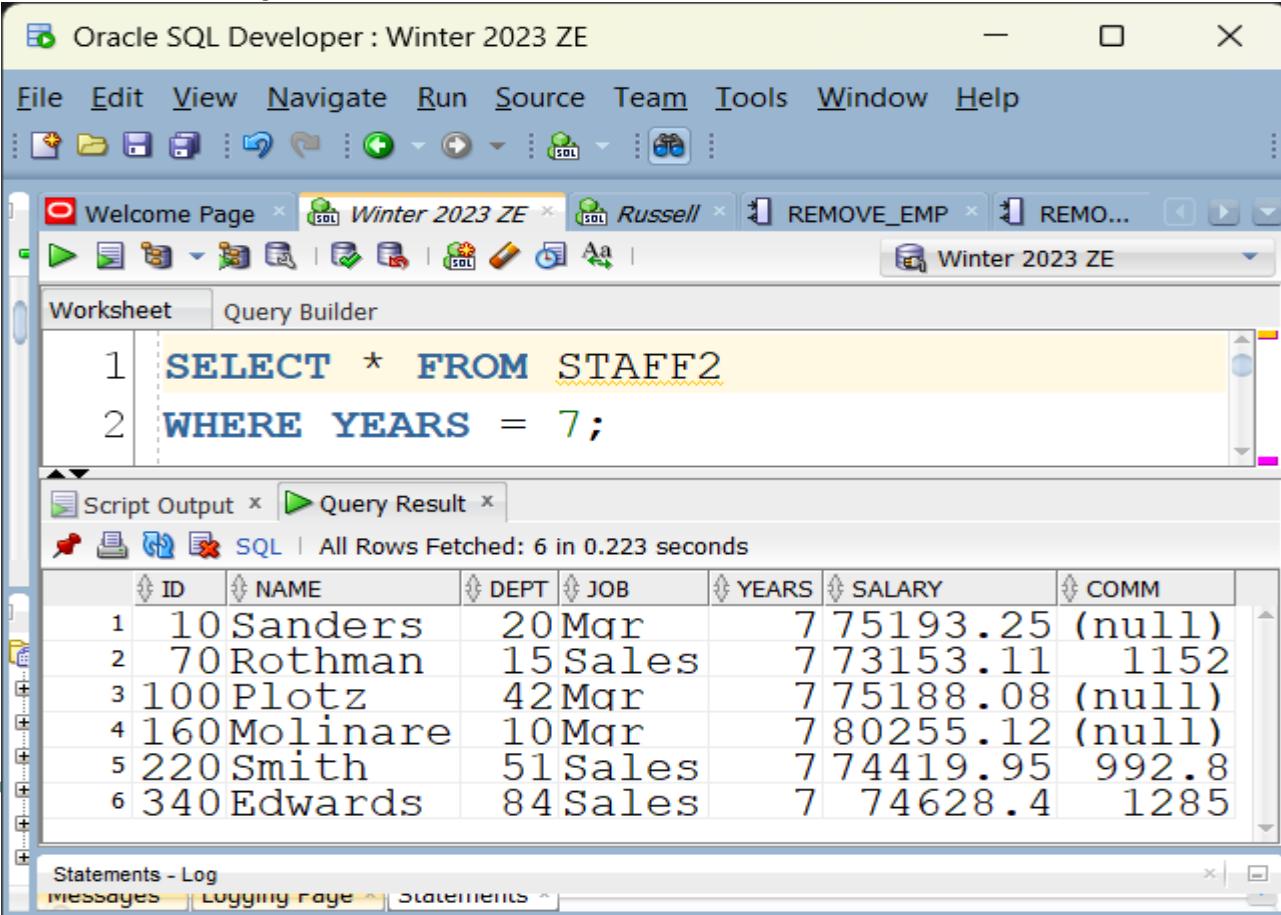
- What are the three possible blocks in a procedure?
- Declarative, Executable, Exception
- What is an anonymous block
- Not saved as a stored procedure on server, no name required, used for testing a stored procedure's arguments
- Where can an anonymous block be saved?
- As a file on your Desktop or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



The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : Winter 2023 ZE". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Print. The main workspace is a "Worksheet" tab, which contains the following SQL code:

```
1 SELECT * FROM STAFF2
2 WHERE YEARS = 7;
```

Below the worksheet, there is a "Script Output" tab showing the results of the query. The output indicates "All Rows Fetched: 6 in 0.223 seconds". The results are displayed in a grid:

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	10 Sanders	20 Mar	7 Sales	7	75193.25	(null)
2	70 Rothman	15 Sales	7 Sales	7	73153.11	1152
3	100 Plotz	42 Mar	7 Sales	7	75188.08	(null)
4	160 Molinare	10 Mar	7 Sales	7	80255.12	(null)
5	220 Smith	51 Sales	7 Sales	7	74419.95	992.8
6	340 Edwards	84 Sales	7 Sales	7	74628.4	1285

# RemoveEmployees Stored Procedure

The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : Winter 2023 ZE". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations like New, Open, Save, and Run. The tab bar shows multiple windows: Welcome Page, Winter 2023 ZE, Russell, REMOVE\_EMP, and REMOVE\_EMPLOYEES. The main area is a Worksheet tab containing the following PL/SQL code:

```
1 SET SERVEROUTPUT ON;
2 EXECUTE REMOVE_EMPLOYEES(7);
```

The code is executed, and the output pane shows:

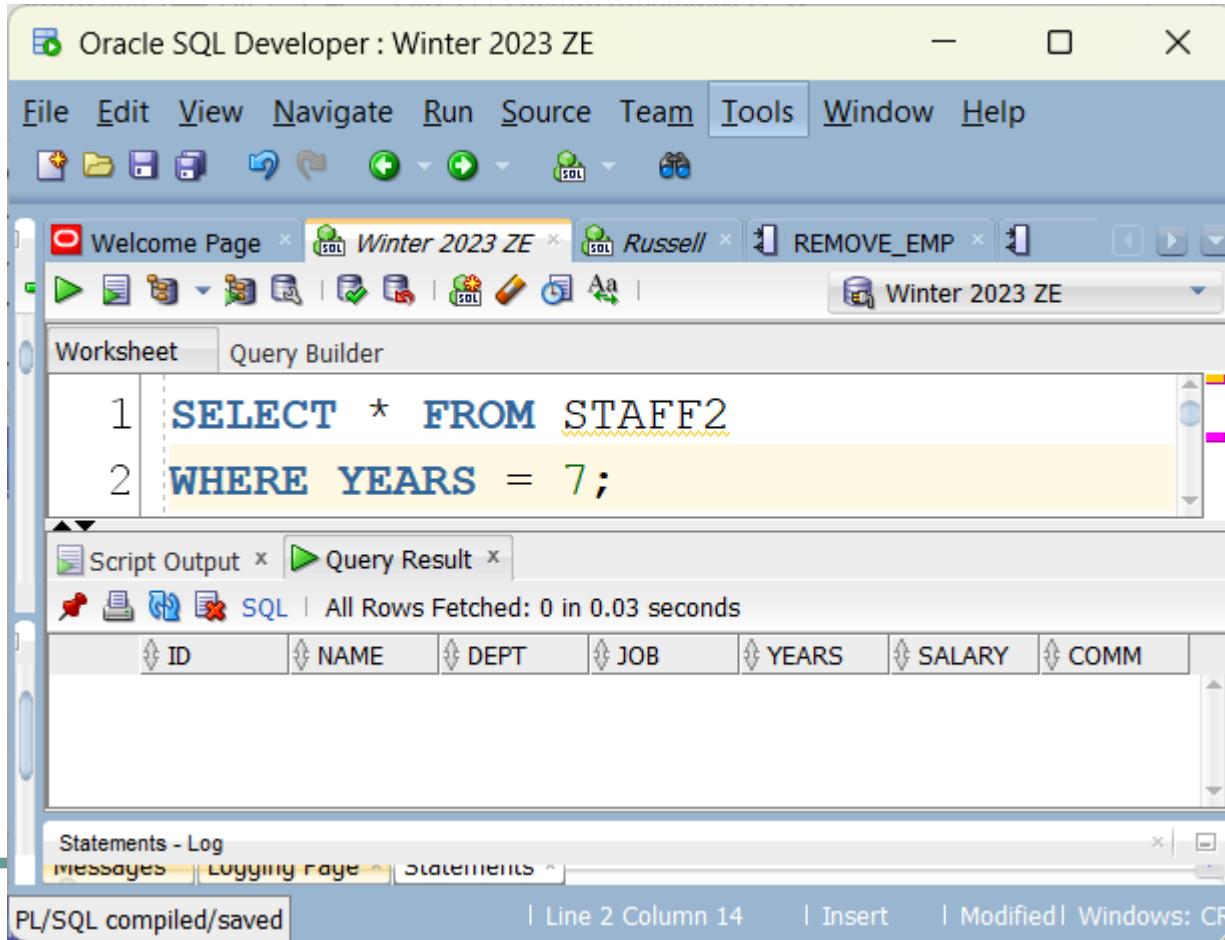
More than one employee is deleted!  
6 employees were deleted

PL/SQL procedure successfully completed.

At the bottom, the status bar shows "Statements - Log" and tabs for "Messages", "Logging Page", and "Statements".

# RemoveEmployees Stored Procedure

- After the procedure is run



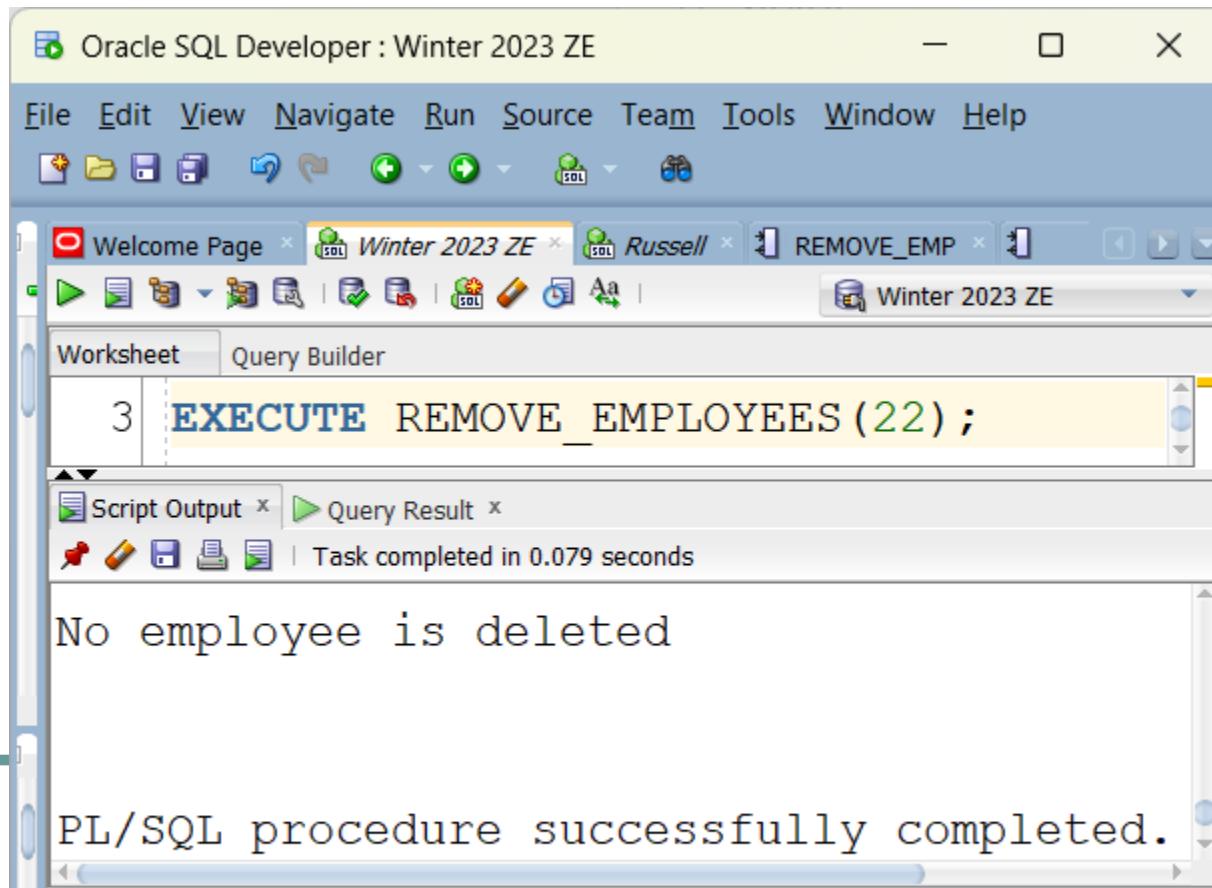
The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : Winter 2023 ZE". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Print. The tab bar shows "Welcome Page", "Winter 2023 ZE", "Russell", and "REMOVE\_EMP". The main area has two tabs: "Worksheet" (selected) and "Query Builder". The Worksheet tab contains the following SQL code:

```
1 | SELECT * FROM STAFF2
2 | WHERE YEARS = 7;
```

Below the code, the "Query Result" tab is open, showing a table with columns ID, NAME, DEPT, JOB, YEARS, SALARY, and COMM. The status bar at the bottom indicates "PL/SQL compiled/saved", "Line 2 Column 14", "Insert", "Modified", and "Windows: CR".

# RemoveEmployees Stored Procedure

- No employee has 22 years of post secondary education



The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : Winter 2023 ZE". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar contains various icons for file operations and navigation. The tab bar shows four tabs: "Welcome Page", "Winter 2023 ZE", "Russell", and "REMOVE\_EMP". The main workspace is a "Worksheet" tab, which contains the following SQL code:

```
3 | EXECUTE REMOVE_EMPLOYEES(22);
```

Below the worksheet, the "Script Output" tab displays the results of the execution:

```
Task completed in 0.079 seconds
```

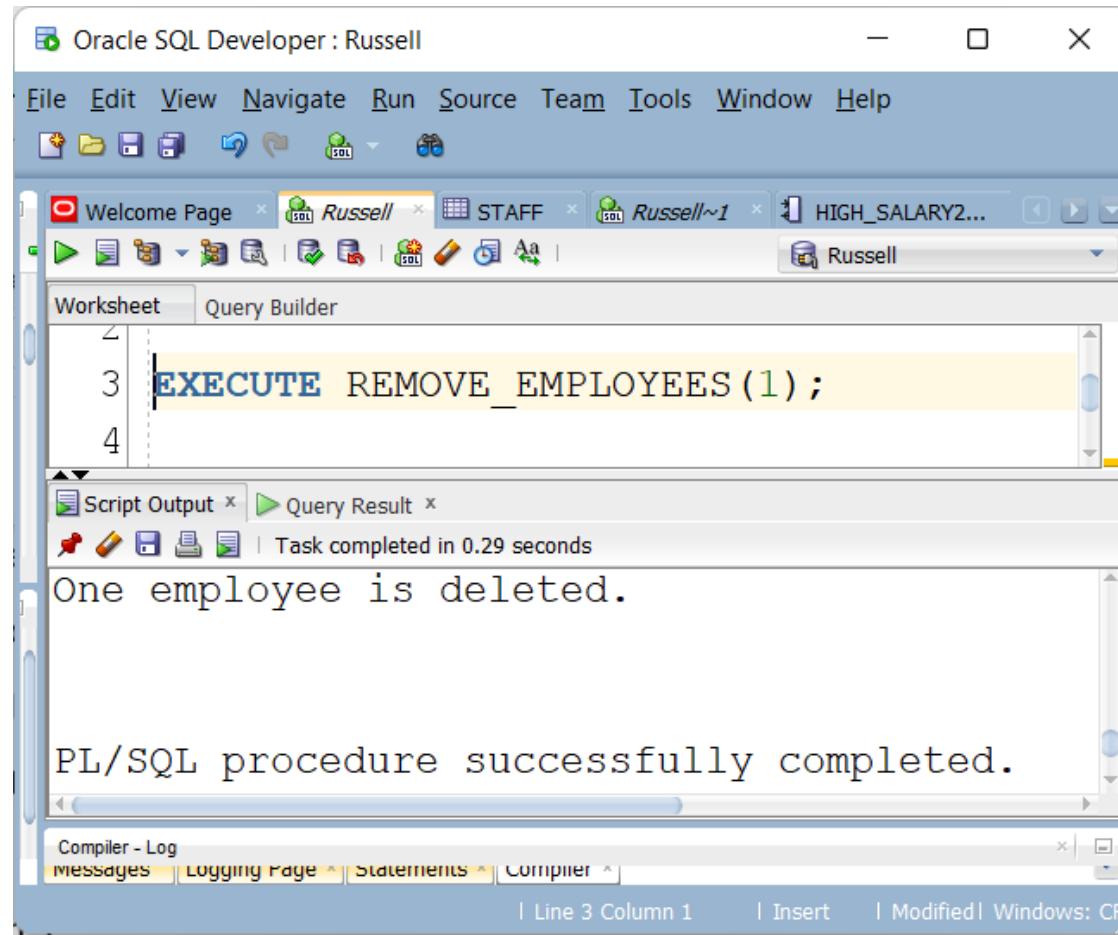
The output pane shows the message:

```
No employee is deleted
```

At the bottom of the output pane, it says:

```
PL/SQL procedure successfully completed.
```

# RemoveEmployees Stored Procedure



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Russell". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Print. The tab bar shows "Welcome Page", "Russell", "STAFF", "Russell~1", and "HIGH\_SALARY2...". The main workspace is titled "Worksheet" and contains the following PL/SQL code:

```
EXECUTE REMOVE_EMPLOYEES(1);
```

The code is highlighted in blue. The output pane below shows the message "One employee is deleted." and "PL/SQL procedure successfully completed." The status bar at the bottom indicates "Line 3 Column 1", "Insert", "Modified", and "Windows: CR".

- 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 automatic 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 x Query Result x

All Rows Fetched: 6 in 0.039 seconds

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	10 Sanders	20	Mar	7	68357.5	(null)
2	70 Rothman	15	Sales	7	66502.83	1152
3	100 Plotz	42	Mar	7	68352.8	(null)
4	160 Molinare	10	Mar	7	72959.2	(null)
5	220 Smith	51	Sales	7	67654.5	992.8
6	340 Edwards	84	Sales	7	67844	1285

Messages - Log

Messages Logging Page Statements

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

Worksheet Query Builder

```
5
6 EXECUTE UPDATE_EMPLOYEES(7);
```

Script Output Query Result

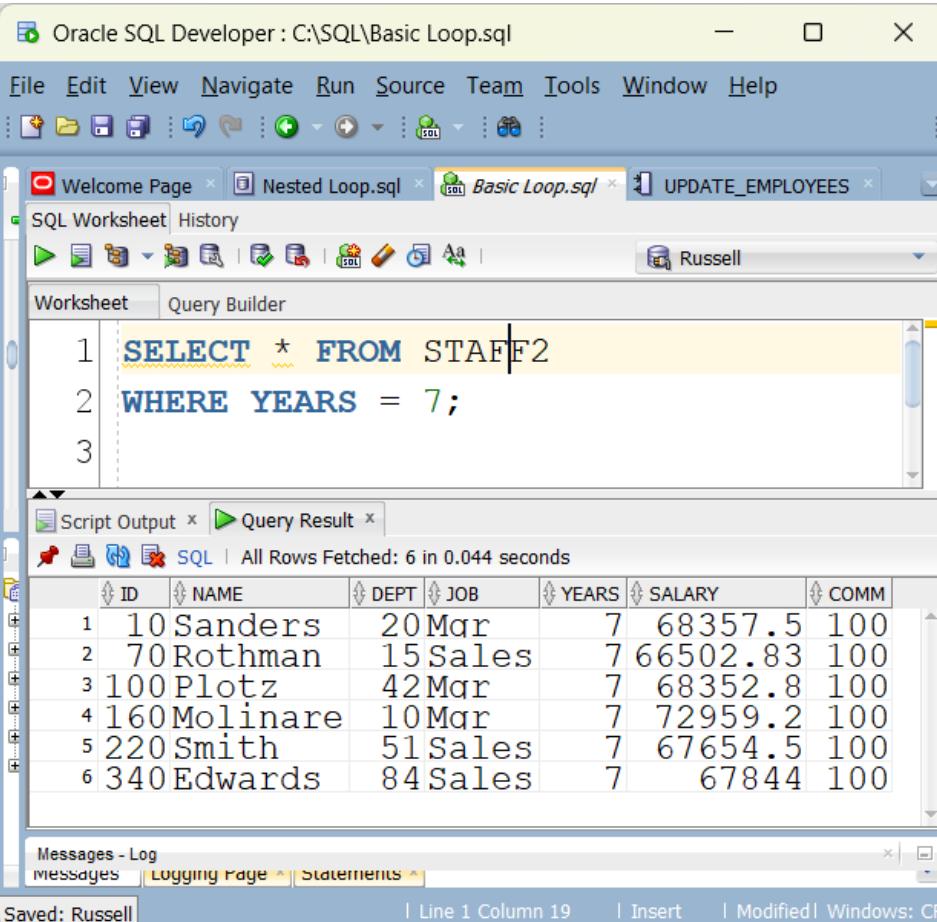
Task completed in 9.048 seconds

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



The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : C:\SQL\Basic Loop.sql". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations and database management. The tabs at the top show "Welcome Page", "Nested Loop.sql", "Basic Loop.sql" (which is selected), and "UPDATE\_EMPLOYEES". The main workspace is a "Worksheet" tab containing the following SQL code:

```
1 SELECT * FROM STAFF2
2 WHERE YEARS = 7;
```

The "Query Result" tab below displays the fetched data from the STAFF2 table:

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	10 Sanders	20Mqr	7	68357.5	100	
2	70 Rothman	15Sales	7	66502.83	100	
3	100 Plotz	42Mqr	7	68352.8	100	
4	160 Molinare	10Mqr	7	72959.2	100	
5	220 Smith	51Sales	7	67654.5	100	
6	340 Edwards	84Sales	7	67844	100	

The status bar at the bottom indicates "Saved: Russell", "Line 1 Column 19", "Insert", "Modified", and "Windows: CR".



```
1  create or replace PROCEDURE UPDATE_EMPLOYEES
2      (YEARSIN IN NUMBER)
3
4      AS
5
6          BEGIN
7
8              UPDATE STAFF2
9              SET COMM = 100
10             WHERE YEARS = YEARSIN;
11
12             IF SQL%ROWCOUNT = 0 THEN
13                 DBMS_OUTPUT.PUT_LINE ('No employee was updated.');
14             ELSIF SQL%ROWCOUNT = 1 THEN
15                 DBMS_OUTPUT.PUT_LINE ('One employee was updated.');
16             ELSE
17                 DBMS_OUTPUT.PUT_LINE ('More than one employee was updated!');
18                 DBMS_OUTPUT.PUT_LINE (SQL%ROWCOUNT || ' employees were updated');
19             END IF;
20
21             EXCEPTION
22                 WHEN OTHERS
23                 THEN
24                     DBMS_OUTPUT.PUT_LINE ('Update Error!');
25
26         END;
```

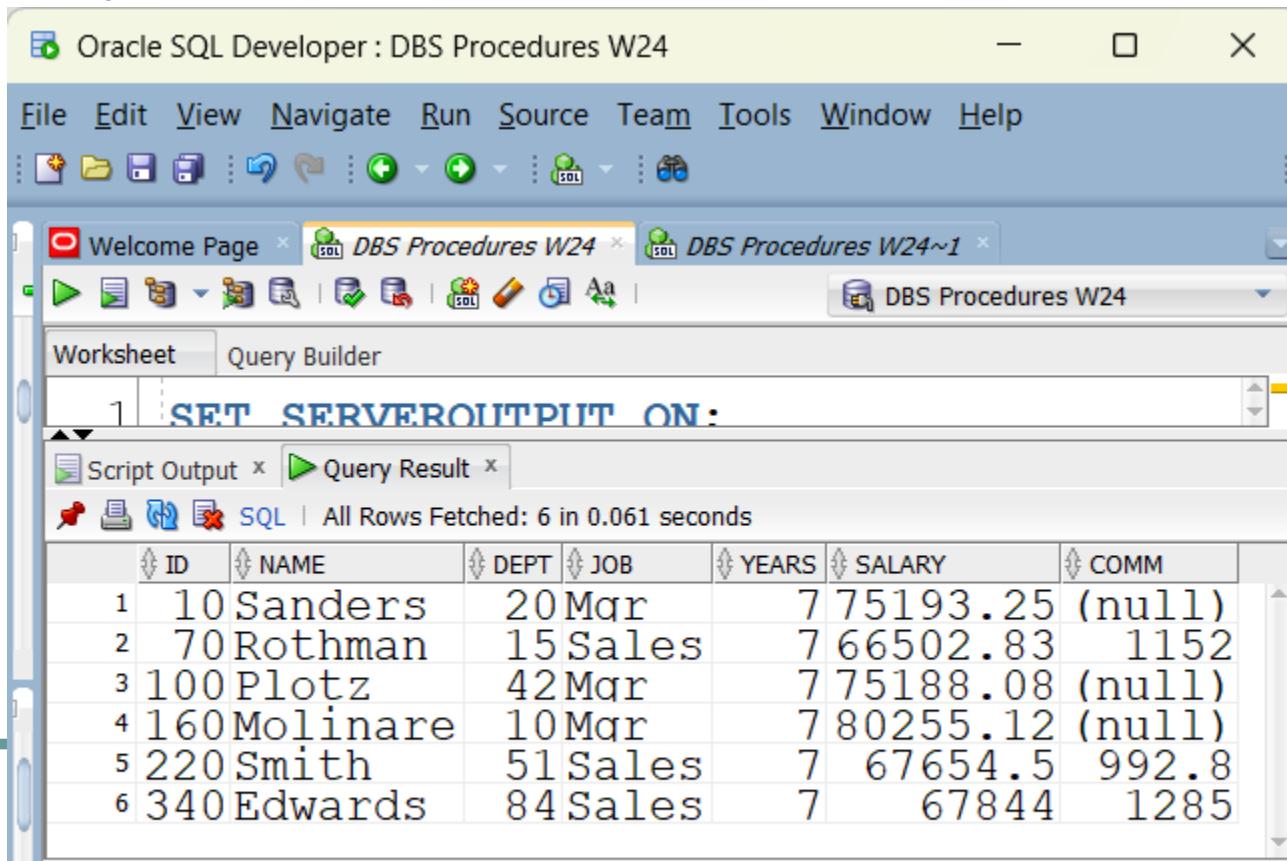
an  
on't

The screenshot shows the Oracle SQL Developer interface with a procedure named `UPDATE_EMPLOYEES2` selected in the tabs. The code editor displays the following PL/SQL procedure:

```
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;
```

The code editor has syntax highlighting and line numbers. Below the code, there is a section labeled `PROCEDURE UPDATE_EMPLOYEES2 EXCEPTION`. At the bottom, there are tabs for `SQL History`, `Statements - Log`, `Messages` (which is selected), `Logging Page`, and `Statements`.

- No employee was updated.
- Why?



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : DBS Procedures W24". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations like New, Open, Save, and Run. The tab bar shows "Welcome Page", "DBS Procedures W24", and "DBS Procedures W24~1". The main area has tabs for Worksheet and Query Builder, with "Worksheet" selected. A code editor window displays the command "SET SERVEROUTPUT ON;". Below it is a results pane titled "Script Output" with a green play button icon. The results show a table of employee data:

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	

The status bar at the bottom indicates "All Rows Fetched: 6 in 0.061 seconds".

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;
```

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);
```

SQL History Compiler - Log  
Messages Logging Page Statements Compiler

# Convert Days to Weeks & Days

- With DAYS set to 20
  - There are 2.85714285714285714285714285714285714286 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 2.85714285714285714285714285714286 weeks
- To 2 weeks
- DBMS\_OUTPUT.PUT\_LINE
- ('There are ' || trunc(DAYS/7,0) || ' weeks');

# Triggers

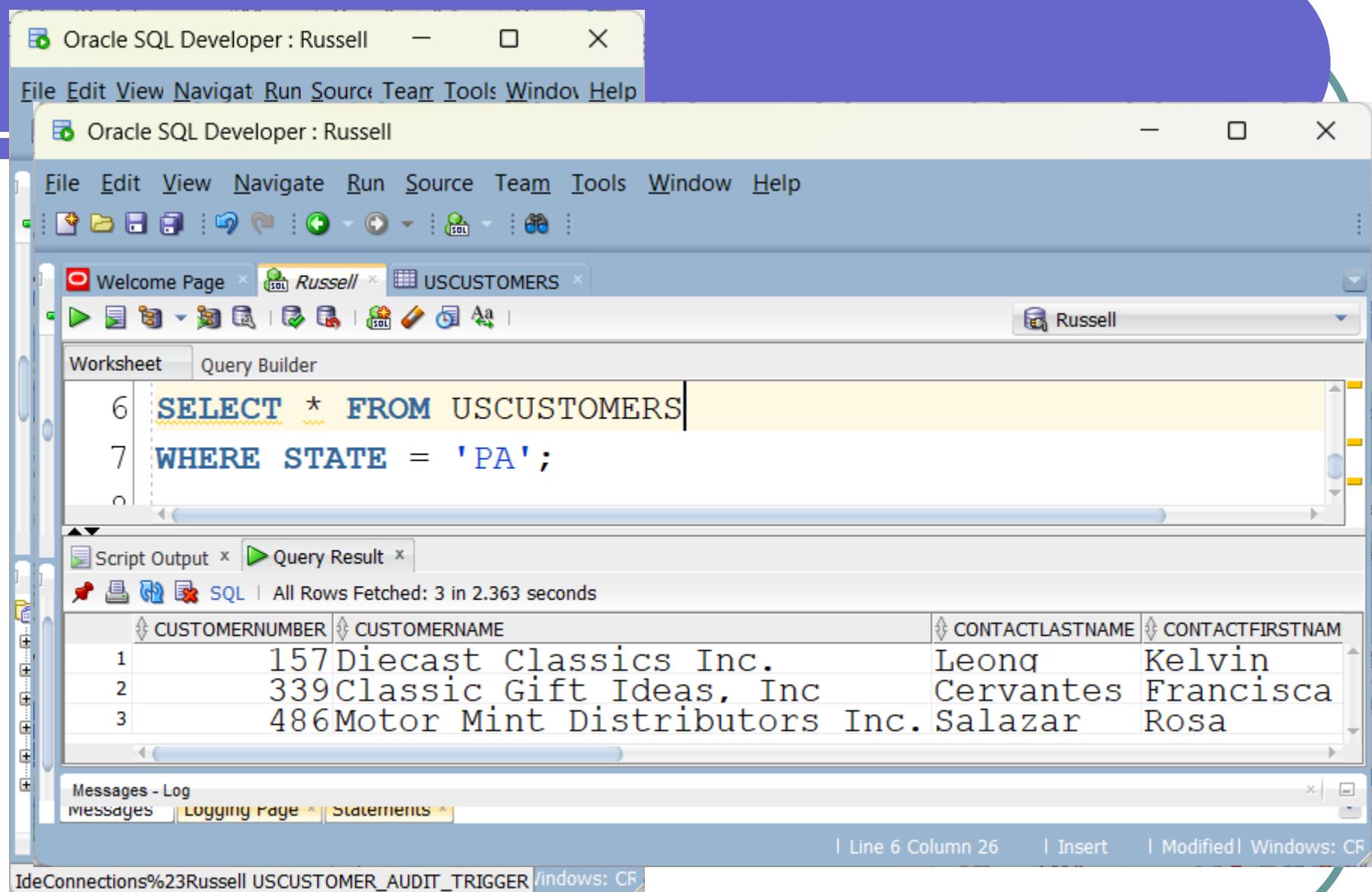
- A trigger is a named PL/SQL block stored in the Oracle Database and executed automatically when a triggering event takes place.
- A DML statement executed against a table can cause the trigger to fire
- A trigger can be set to fire before or after an Update, Insert or Delete and is applied to a table 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 than 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
  - exception\_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
- );



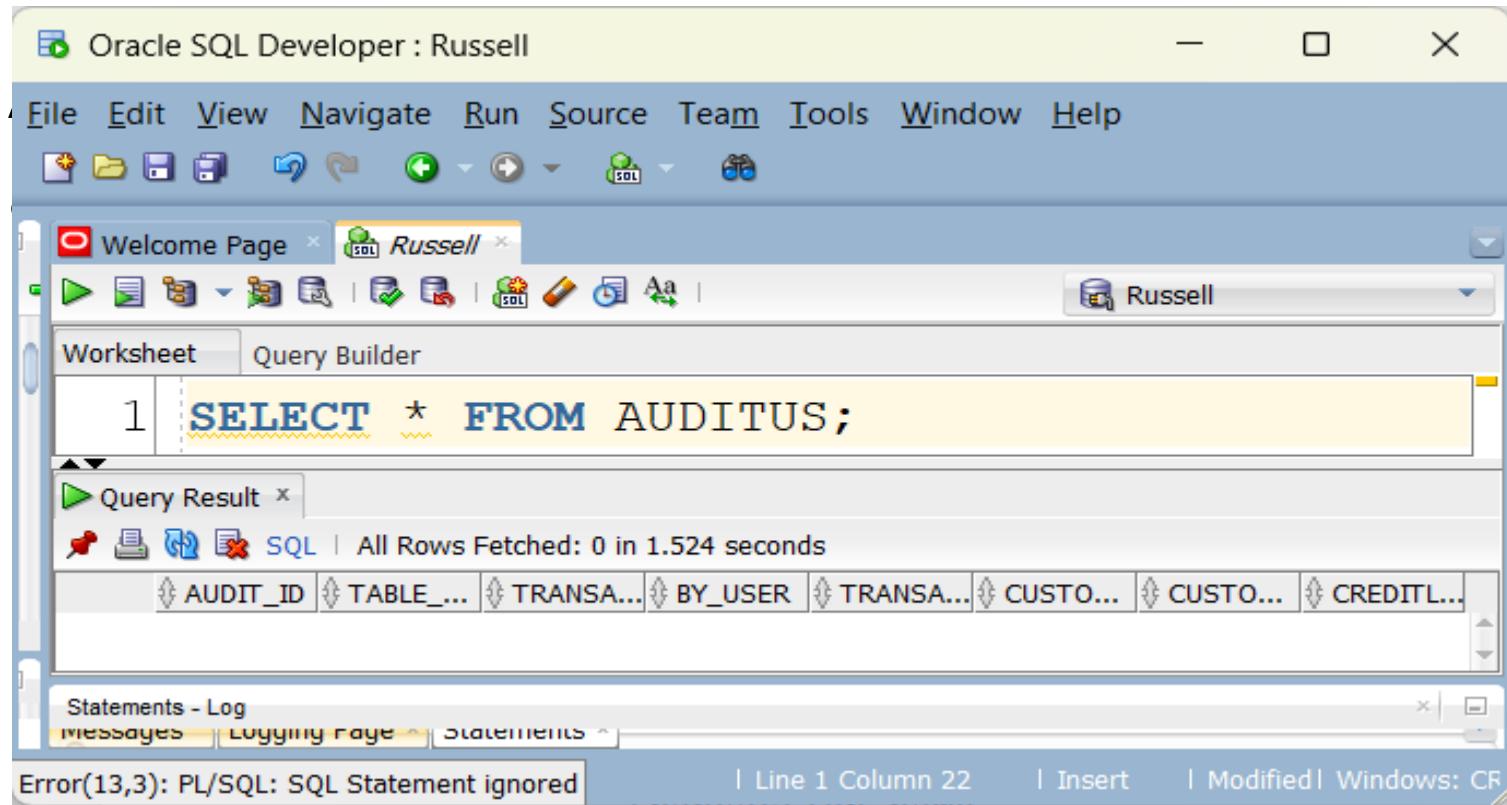
- Columns to monitor added to audit table

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Table RPANGBORN.AUDITUS@Russell". The menu bar includes File, Edit, View, Navigate, Run, Team, Tools, Window, and Help. The toolbar has various icons for file operations. Below the toolbar, there are tabs for "Russell", "INSURANCEQRY", and "AUDITUS", with "AUDITUS" currently selected. A tab bar at the bottom shows "Columns", "Data", "Model", "Constraints", "Grants", "Statistics", "Triggers", "Flashback", "Dependencies", "Details", and "Partitions". The main area displays the table structure:

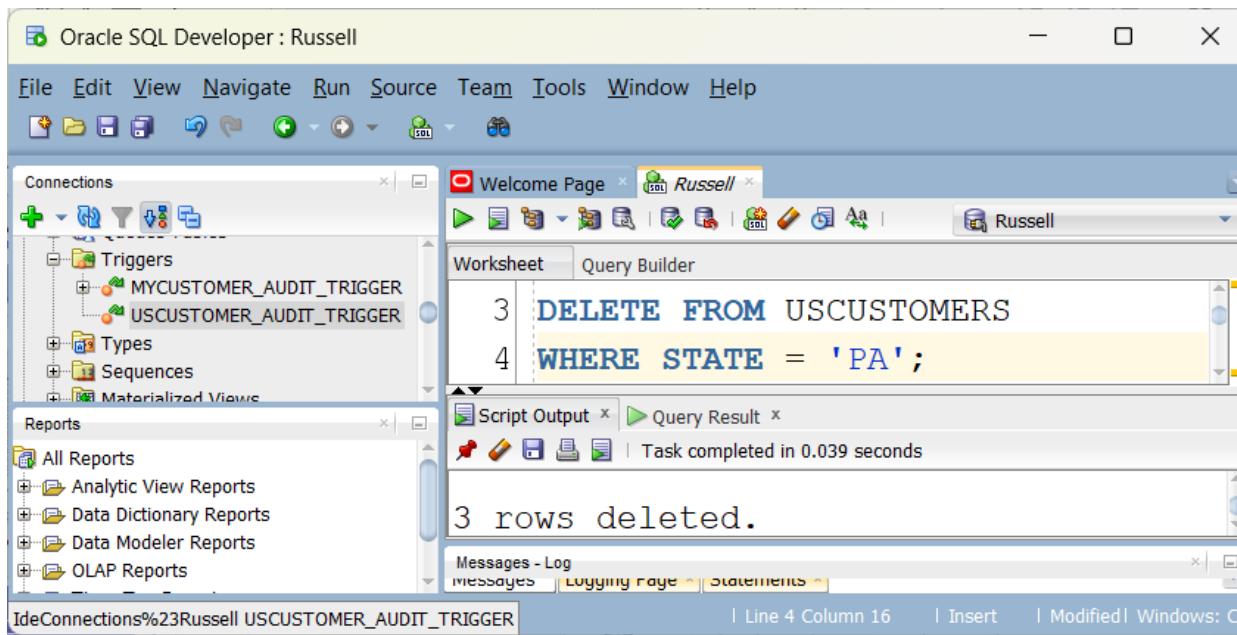
COLUMN_NAME	DATA_TYPE	NULLABLE
1 AUDIT ID	NUMBER	No
2 TABLE NAME	VARCHAR2 (255 BYTE)	Yes
3 TRANSACTION NAME	VARCHAR2 (10 BYTE)	Yes
4 BY USER	VARCHAR2 (30 BYTE)	Yes
5 TRANSACTION DATE	DATE	Yes
6 CUSTOMERNUMBER	NUMBER (38, 0)	Yes
7 CUSTOMERNAME	VARCHAR2 (50 BYTE)	Yes
8 CREDITLIMIT	NUMBER (10, 2)	Yes

At the bottom, there is a "Messages - Log" panel with tabs for "Messages", "Logging Page", and "Statements".

# USCUSTOMER TRIGGER



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



The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : Russell". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for connection management, file operations, and query execution. The Connections sidebar shows a tree structure with Triggers, Types, Sequences, and Materialized Views. Under Triggers, two triggers are listed: "MYCUSTOMER\_AUDIT\_TRIGGER" and "USCUSTOMER\_AUDIT\_TRIGGER". The Reports sidebar shows categories like All Reports, Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, and OLAP Reports. The central workspace is a Worksheet tab where the following SQL code is entered:

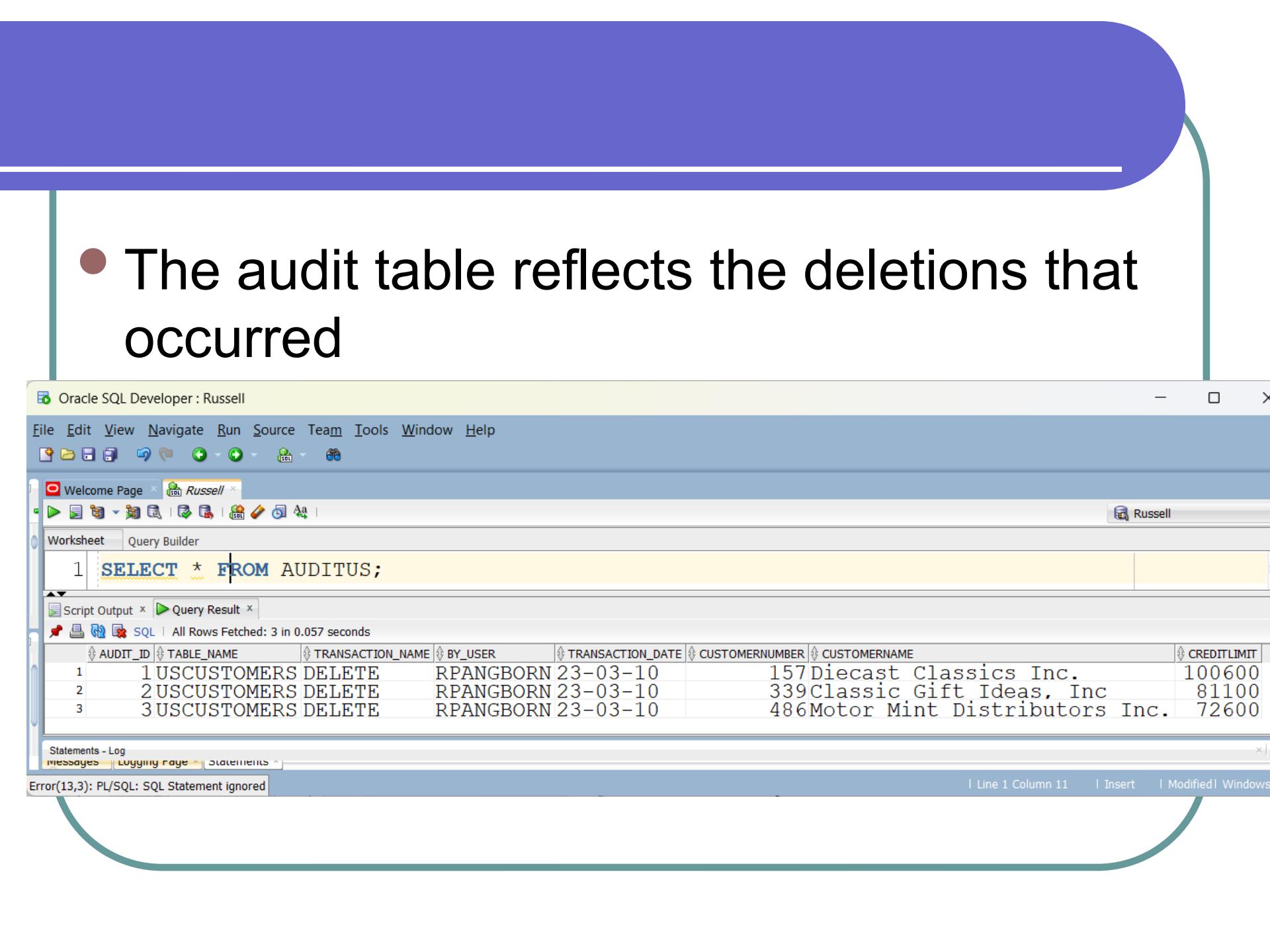
```
3 DELETE FROM USCUSTOMERS
4 WHERE STATE = 'PA';
```

Below the worksheet, the Script Output tab shows the result of the execution:

```
Task completed in 0.039 seconds
3 rows deleted.
```

The status bar at the bottom indicates the current line and column (Line 4 Column 16), and options for Insert, Modified, and Windows: CR.

- The audit table reflects the deletions that occurred



Oracle SQL Developer : Russell

File Edit View Navigate Run Source Team Tools Window Help

Welcome Page Russell

Worksheet Query Builder

```
1 SELECT * FROM AUDITUS;
```

Script Output Query Result

All Rows Fetched: 3 in 0.057 seconds

SQL

AUDIT_ID	TABLE_NAME	TRANSACTION_NAME	BY_USER	TRANSACTION_DATE	CUSTOMERNUMBER	CUSTOMERNAME	CREDITLIMIT
1	USCUSTOMERS	DELETE	RPANGBORN	23-03-10	157	Diecast Classics Inc.	100600
2	USCUSTOMERS	DELETE	RPANGBORN	23-03-10	339	Classic Gift Ideas, Inc	81100
3	USCUSTOMERS	DELETE	RPANGBORN	23-03-10	486	Motor Mint Distributors Inc.	72600

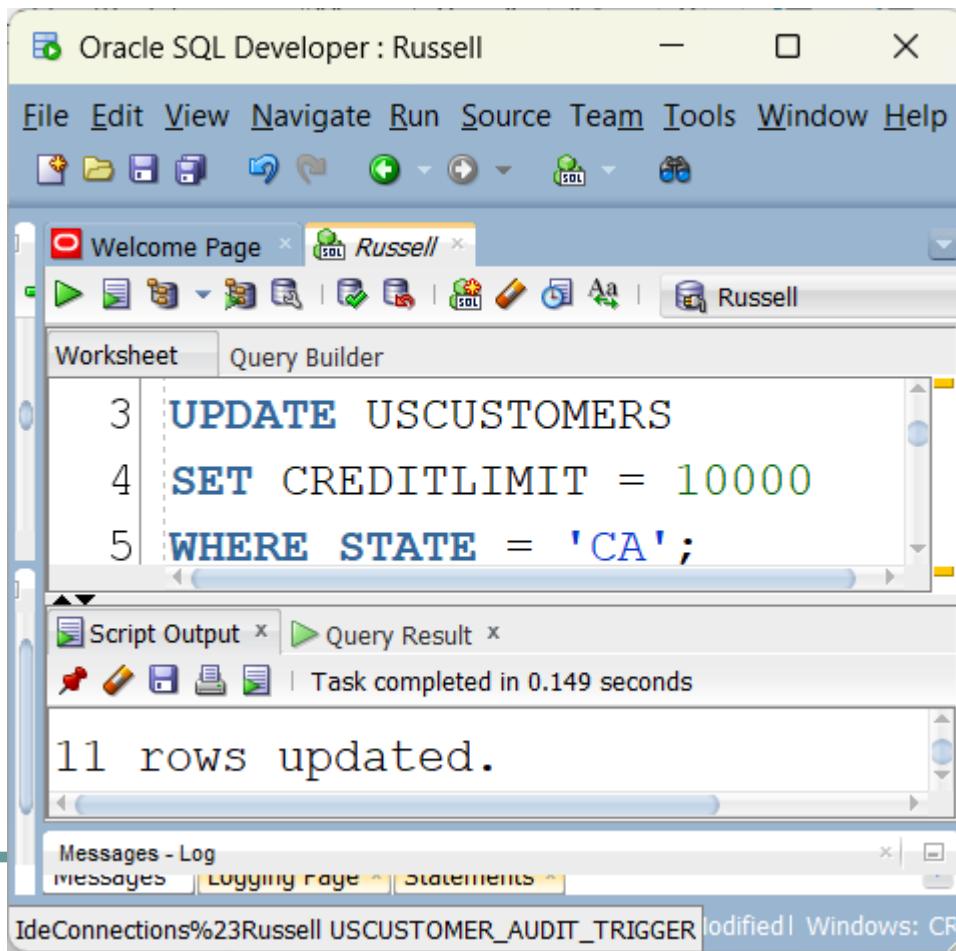
Statements - Log

Messages Logging Page Statements

Error(13,3): PL/SQL: SQL Statement ignored

Line 1 Column 11 Insert Modified Windows

- An update will also cause the trigger to fire



The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : Russell". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar contains various icons for database navigation and management. The main workspace has tabs for "Worksheet" and "Query Builder", with the "Worksheet" tab active. The code area displays the following SQL script:

```
3 | UPDATE USCUSTOMERS
4 | SET CREDITLIMIT = 10000
5 | WHERE STATE = 'CA';
```

Below the code, the "Script Output" pane shows the message "Task completed in 0.149 seconds". The "Query Result" pane displays the output: "11 rows updated.". At the bottom, the "Messages - Log" pane shows "Messages" and "Statements". The status bar at the bottom right indicates "modified" and "Windows: CR".

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

Oracle SQL Developer : Russell

File Edit View Navigate Run Source Team Tools Window Help

Worksheet Query Builder

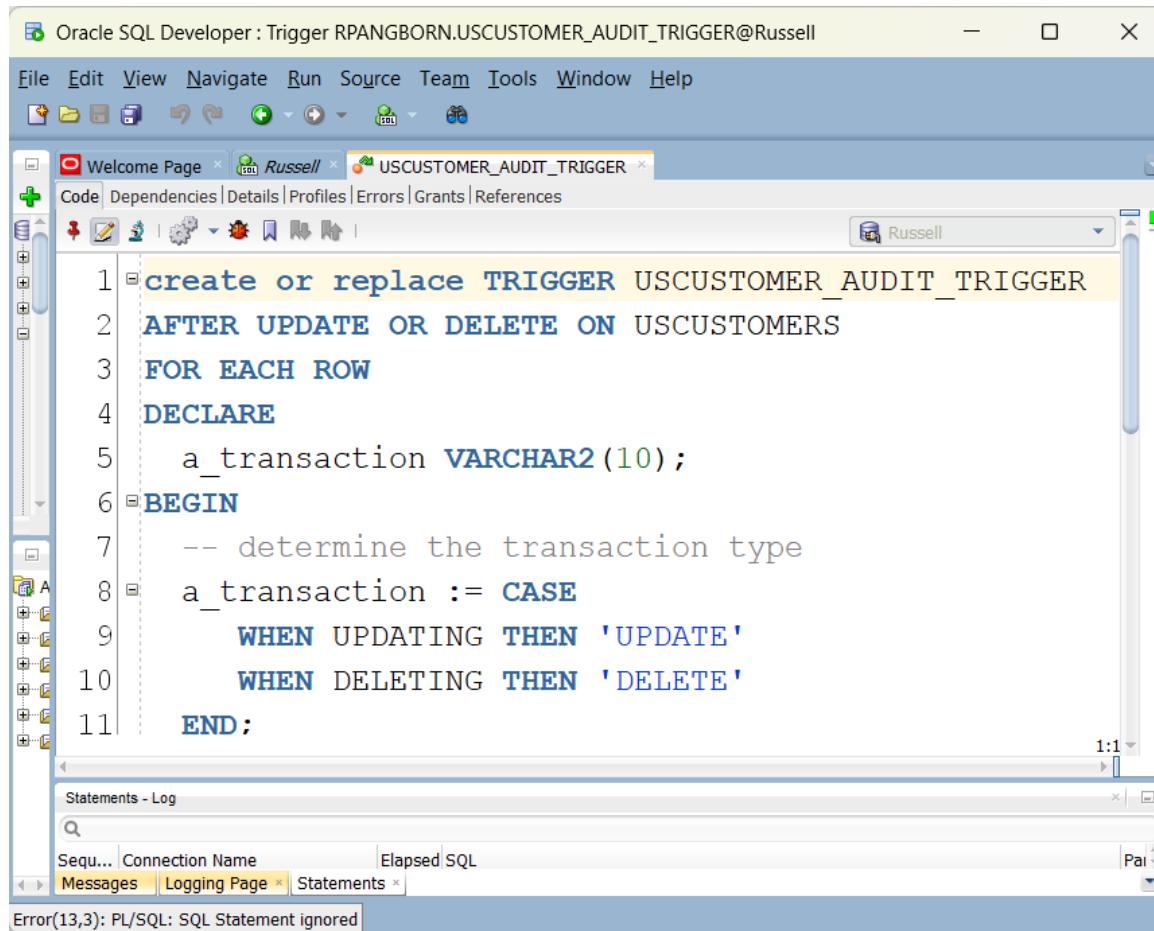
```
1 | SELECT * FROM AUDITUS;
```

Script Output x Query Result x

All Rows Fetched: 14 in 0.029 seconds

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

# The Trigger

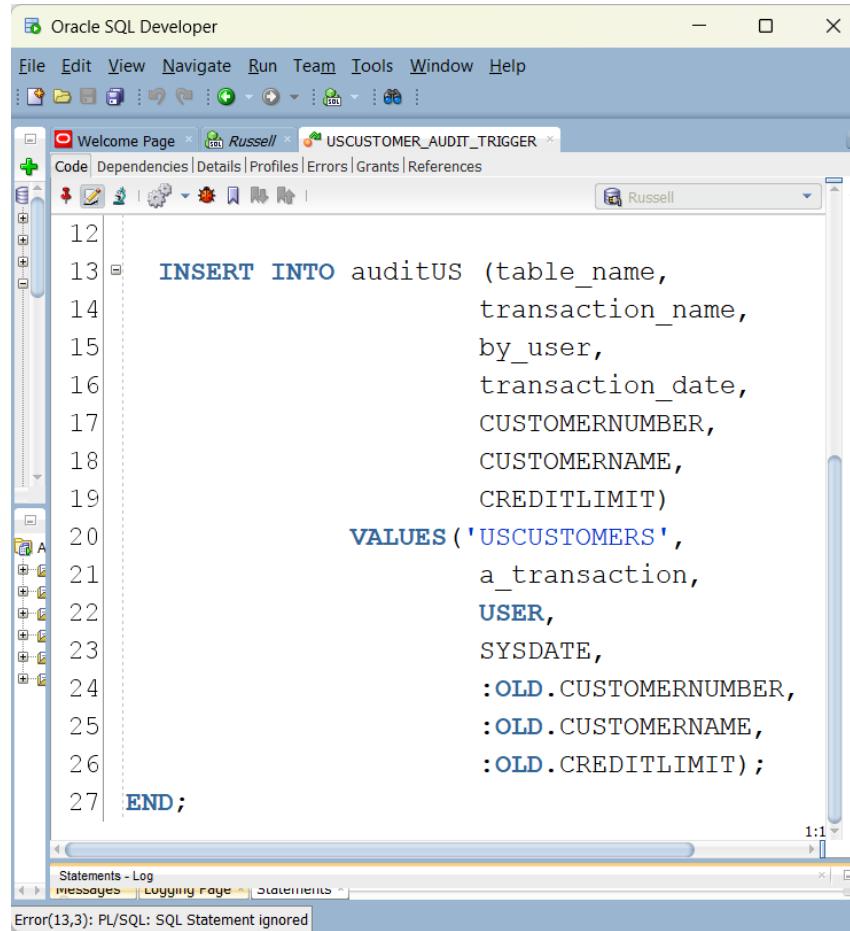


The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Trigger RPANGBORN.USCUSTOMER\_AUDIT\_TRIGGER@Russell". The main window displays the PL/SQL code for a trigger:

```
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;
```

The code editor has syntax highlighting for PL/SQL keywords and data types. The bottom status bar shows the message "Error(13,3): PL/SQL: SQL Statement ignored".

# The Trigger (continued)



The screenshot shows the Oracle SQL Developer interface with a code editor window open. The window title is "USCUSTOMER\_AUDIT\_TRIGGER". The code in the editor is a PL/SQL trigger body:

```
12
13   INSERT INTO auditUS (table_name,
14                         transaction_name,
15                         by_user,
16                         transaction_date,
17                         CUSTOMERNUMBER,
18                         CUSTOMERNAME,
19                         CREDITLIMIT)
20   VALUES ('USCUSTOMERS',
21           a_transaction,
22           USER,
23           SYSDATE,
24           :OLD.CUSTOMERNUMBER,
25           :OLD.CUSTOMERNAME,
26           :OLD.CREDITLIMIT);
27 END;
```

The code editor has syntax highlighting for PL/SQL keywords and identifiers. The status bar at the bottom shows the message "Error(13,3): PL/SQL: SQL Statement ignored".

# 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

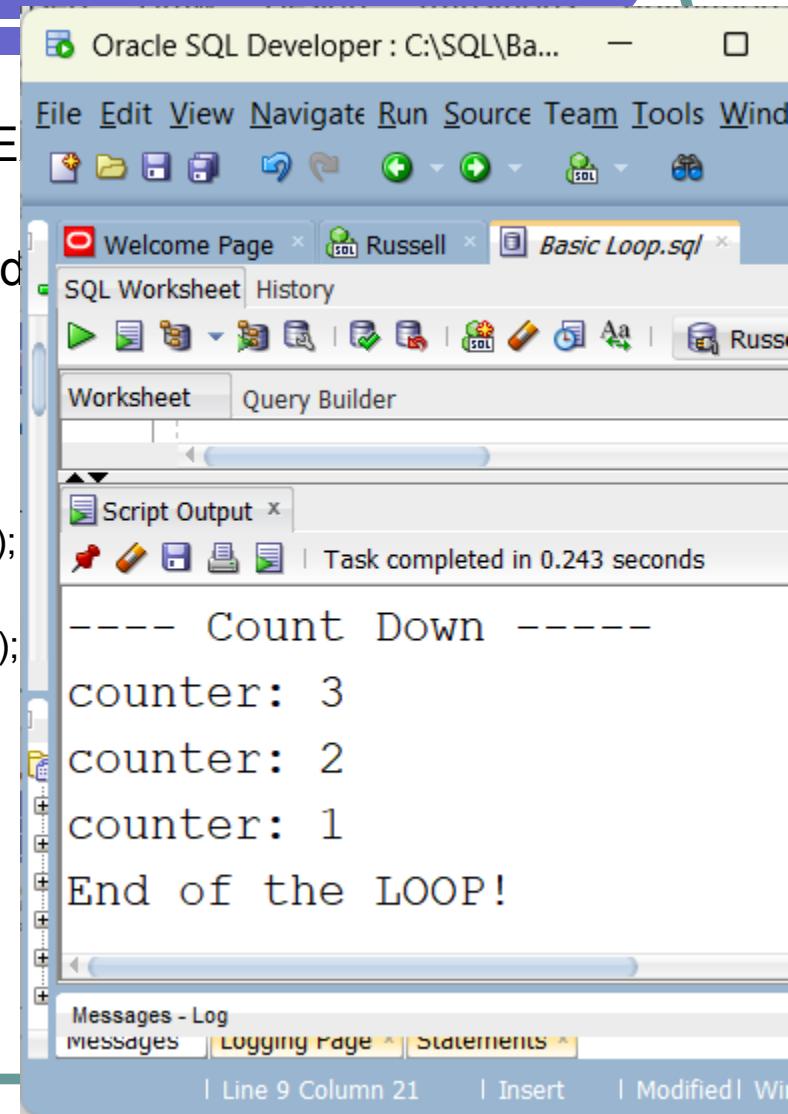
# Iteration Statements

---

- 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 EXIT statement terminates the loop execution or an exception is raised.
- The EXIT statement terminates the loop and exits to the 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;



The screenshot shows the Oracle SQL Developer interface with a script named 'Basic Loop.sql' running. The 'Script Output' window displays the following output:

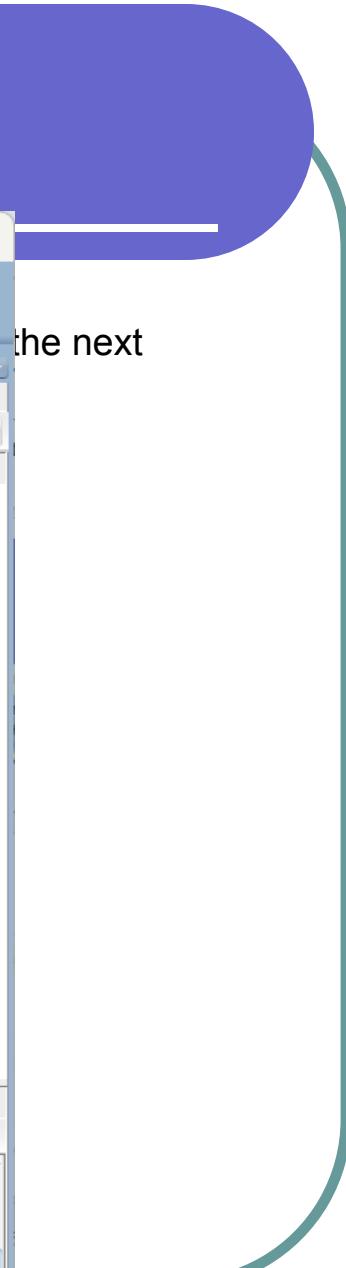
```
---- Count Down ----
counter: 3
counter: 2
counter: 1
End of the LOOP!
```

The status bar at the bottom indicates 'Line 9 Column 21'.

# 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

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

SQL Worksheet History

Russell

Worksheet Query Builder

```
4 BEGIN
5   DBMS_OUTPUT.PUT_LINE ('---- Count Down ----');
6   LOOP
7     IF COUNTER = 2 THEN
8       COUNTER := COUNTER - 1;
9       CONTINUE;
10    END IF;
11    DBMS_OUTPUT.PUT_LINE ('counter: ' || counter);
12    counter := counter - 1;
13    EXIT WHEN COUNTER < 1;
14  END LOOP;
15  DBMS_OUTPUT.PUT_LINE('End of the LOOP!');
16END;
```

Script Output Task completed in 0.109 seconds

```
---- Count Down ----
counter: 3
counter: 1
End of the LOOP!
```

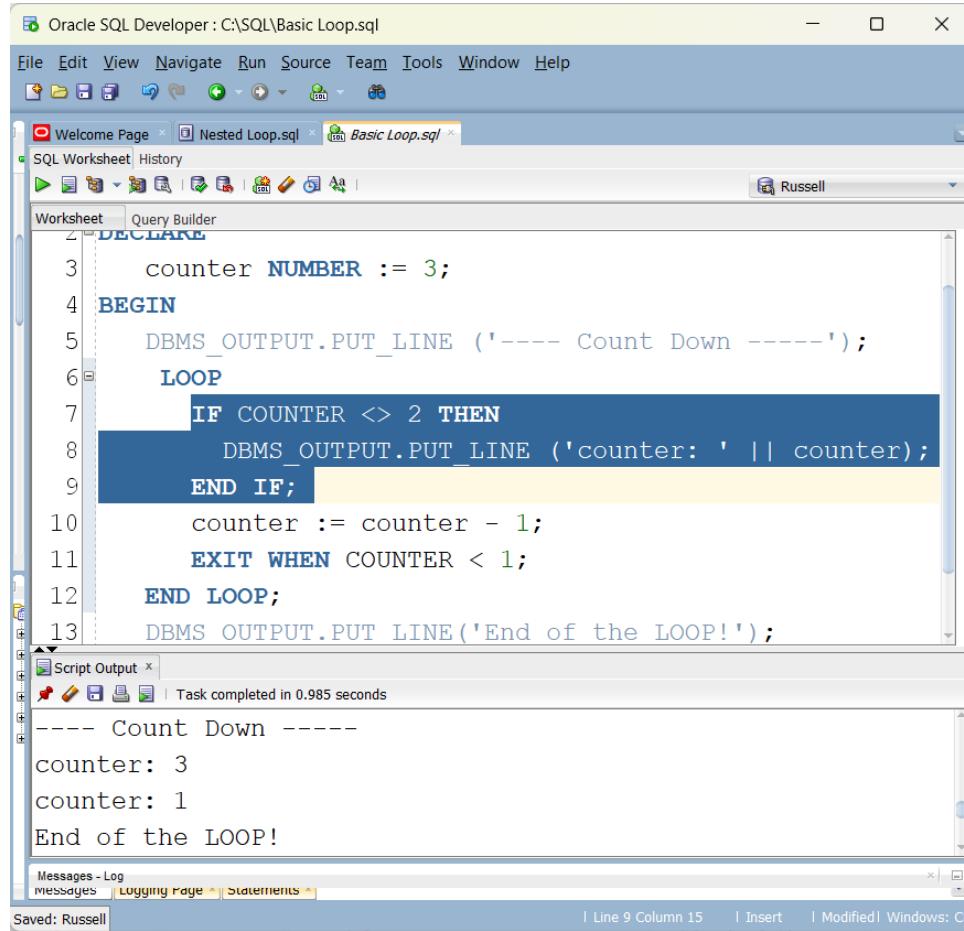
Messages - Log

Messages Logging Page Statements

Line 9 Column 17 Insert Modified Windows: CR

Saved: Russell

# Avoiding use of Continue



The screenshot shows the Oracle SQL Developer interface with a PL/SQL script named "Basic Loop.sql" open in the Worksheet tab. The script contains a loop that counts down from 3 to 1, printing each value to the DBMS\_OUTPUT. The code is as follows:

```
1  DECLARS
2      counter NUMBER := 3;
3
4  BEGIN
5      DBMS_OUTPUT.PUT_LINE ('----- Count Down -----');
6      LOOP
7          IF COUNTER <> 2 THEN
8              DBMS_OUTPUT.PUT_LINE ('counter: ' || counter);
9          END IF;
10         counter := counter - 1;
11         EXIT WHEN COUNTER < 1;
12     END LOOP;
13     DBMS_OUTPUT.PUT_LINE('End of the LOOP!');
```

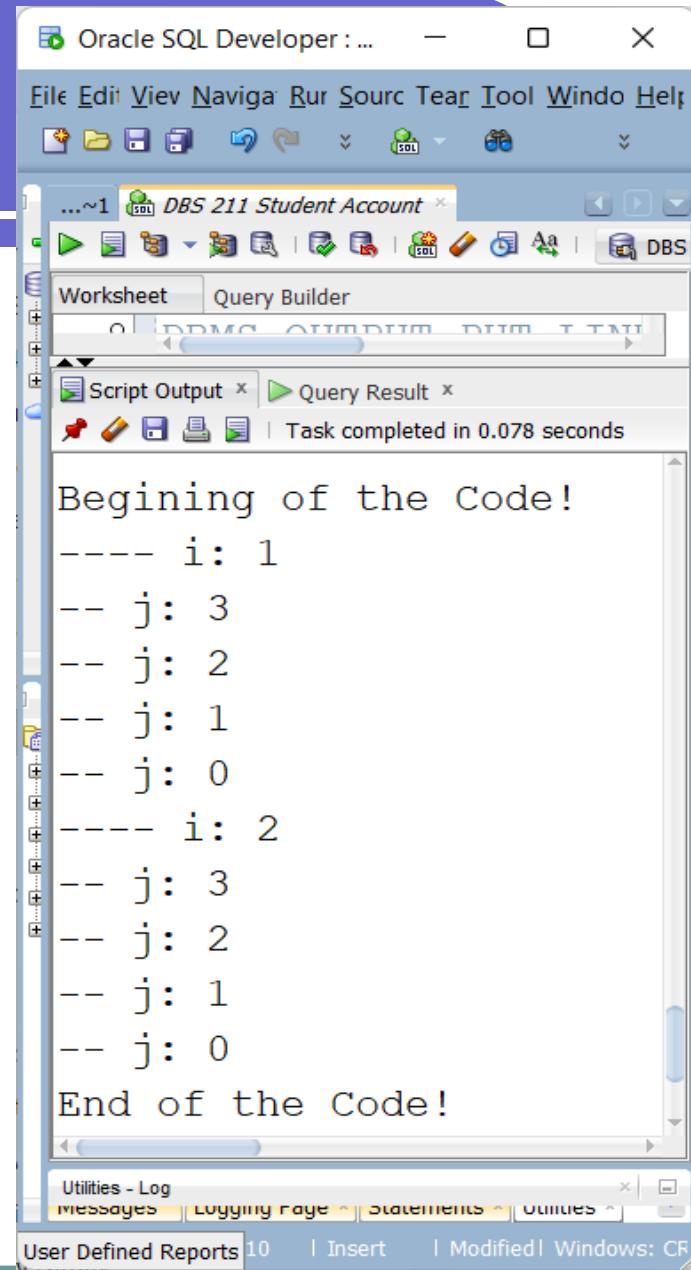
The output window below the worksheet shows the execution results:

```
----- Count Down -----
counter: 3
counter: 1
End of the LOOP!
```

- 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;
```



The screenshot shows the Oracle SQL Developer interface with a worksheet window open. The worksheet contains the PL/SQL code for a nested loop. The output pane displays the results of the execution, showing the output of the DBMS\_OUTPUT.PUT\_LINE statements. The output is as follows:

```
Begining of the Code!
---- i: 1
-- j: 3
-- j: 2
-- j: 1
-- j: 0
---- i: 2
-- j: 3
-- j: 2
-- j: 1
-- j: 0
End of the Code!
```

The interface includes a toolbar, menu bar, and various tool windows like Script Output and Query Result.

# 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 decreases 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 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);      ---- i: 1
        FOR j IN REVERSE 1..4 LOOP
            DBMS_OUTPUT.PUT_LINE ('-- j: ' || j);      -- j: 4
            -- j: 3
            -- j: 2
            -- j: 1
        END LOOP;
    END LOOP;
END;
```

---- 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