

Variabile

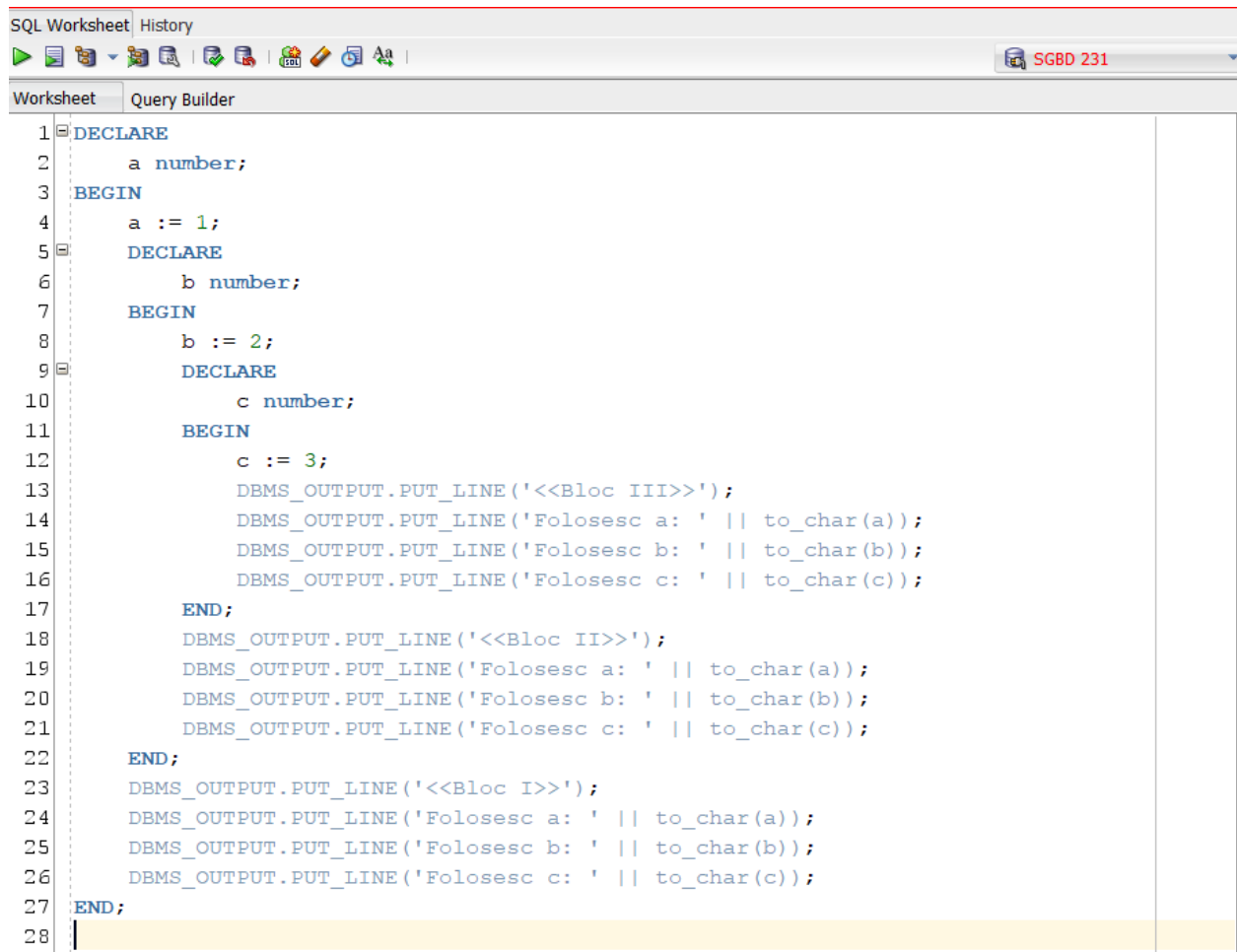
Cursul numărul 3 de SGBD menționează următoarele:

Variabilele

- Stochează datele în același format binar intern ca și baza de date, astfel nefiind necesare conversii suplimentare.
- Pot fi declarate doar în zona declarativă a unui bloc, unde pot fi și inițializate.
 - Li se pot atribui valori noi și pot fi utilizate în zona executabilă a blocului.
 - Pot fi transmise ca parametrii subprogramelor *PL/SQL*.
- Pot fi declarate pentru a menține rezultatul obținut de un subprogram *PL/SQL*.
- Sunt vizibile în blocul în care sunt declarate și în toate subblocurile declarate în acesta.
- Dacă o variabilă nu este declarată local în bloc, atunci este căutată în secțiunea declarativă a blocurilor care includ blocul respectiv.

Așadar, voi da câteva *exemple greșite* de folosire a variabilelor (asemănătoare exemplului 3.3 din curs), urmate de *explicarea erorilor* și *corectarea* acestora:

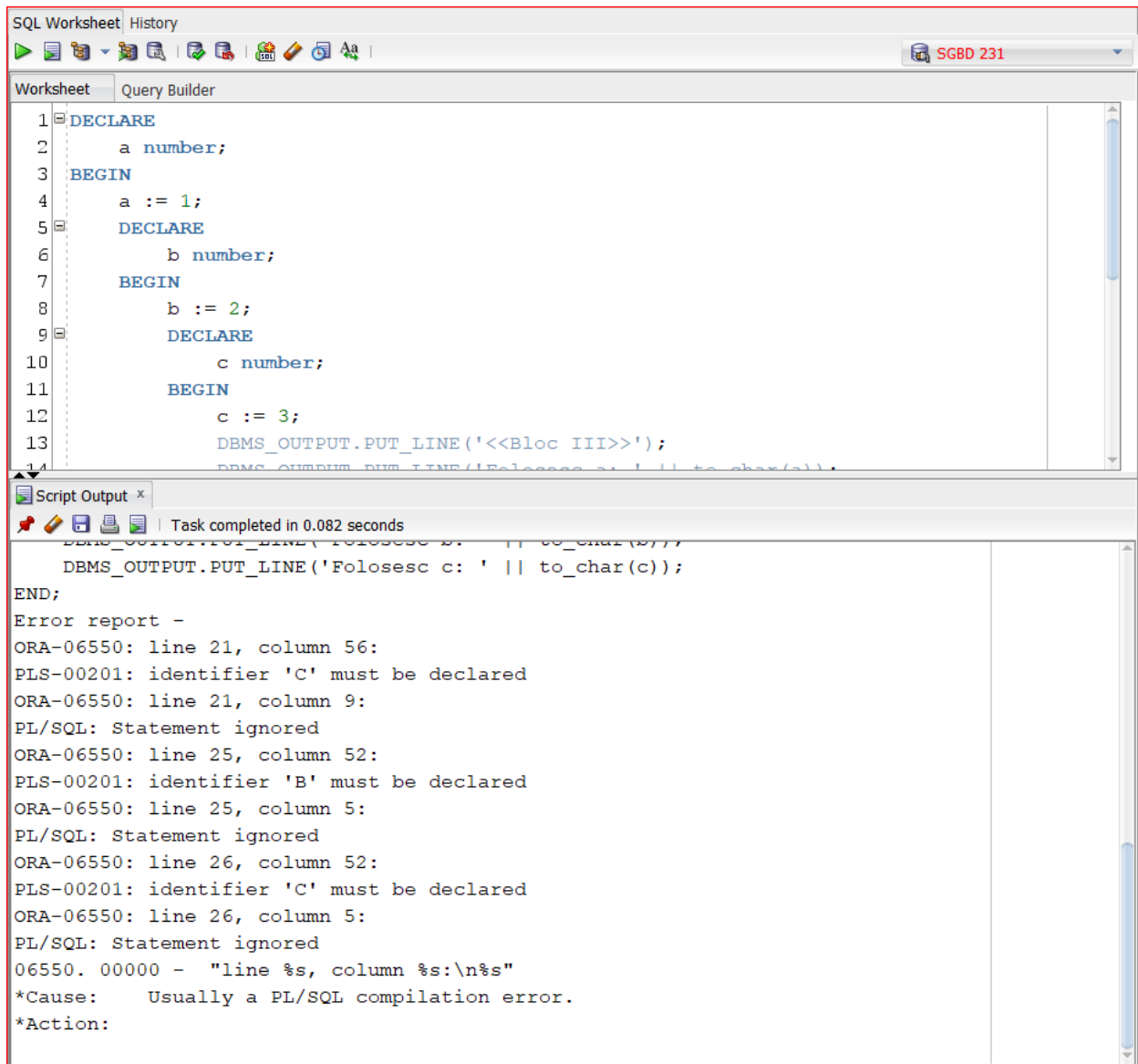
- **Exemplul 1** (trei blocuri)



```
SQL Worksheet History
Worksheet Query Builder
SGBD 231

1 DECLARE
2     a number;
3 BEGIN
4     a := 1;
5 DECLARE
6     b number;
7 BEGIN
8     b := 2;
9 DECLARE
10    c number;
11 BEGIN
12    c := 3;
13    DBMS_OUTPUT.PUT_LINE('<<Bloc III>>');
14    DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
15    DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
16    DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
17 END;
18    DBMS_OUTPUT.PUT_LINE('<<Bloc II>>');
19    DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
20    DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
21    DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
22 END;
23    DBMS_OUTPUT.PUT_LINE('<<Bloc I>>');
24    DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
25    DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
26    DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
27 END;
28
```

- Erori:



The screenshot displays the SQL Worksheet interface with a PL/SQL script and its execution results. The script consists of three nested blocks. The first block (lines 1-14) declares variable 'a' and contains a second block. The second block (lines 6-12) declares variable 'b' and contains a third block. The third block (lines 10-13) declares variable 'c' and outputs its value. The Script Output window shows the execution results, including the output of the third block and an error report.

```
1 DECLARE
2     a number;
3 BEGIN
4     a := 1;
5     DECLARE
6         b number;
7     BEGIN
8         b := 2;
9         DECLARE
10            c number;
11        BEGIN
12            c := 3;
13            DBMS_OUTPUT.PUT_LINE('<<Bloc III>>');
14            DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
15        END;
16    END;
17 END;
```

Script Output x

Task completed in 0.082 seconds

```
DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
END;
```

Error report -

```
ORA-06550: line 21, column 56:
PLS-00201: identifier 'C' must be declared
ORA-06550: line 21, column 9:
PL/SQL: Statement ignored
ORA-06550: line 25, column 52:
PLS-00201: identifier 'B' must be declared
ORA-06550: line 25, column 5:
PL/SQL: Statement ignored
ORA-06550: line 26, column 52:
PLS-00201: identifier 'C' must be declared
ORA-06550: line 26, column 5:
PL/SQL: Statement ignored
06550. 00000 - "line %s, column %s:\n%s"
*Cause:      Usually a PL/SQL compilation error.
*Action:
```

Observăm că erorile au legătură cu liniile 21, 25 și 26. Amintindu-ne ceea ce menționa cursul 3, ne dăm seama că variabilele, dacă nu sunt declarate local în bloc, sunt căutate în secțiunea declarativă a blocurilor care includ blocul respectiv. Astfel, eroarea de la linia 21 ne indică faptul că variabila C nu este găsită deoarece ea nu mai este văzută de blocul II, iar erorile de la liniile 25 și 26 merg pe același principiu: variabilele B și C nu sunt găsite pentru că ele nu pot fi văzute de blocul I.

- Trei blocuri fără erori:

The image shows two side-by-side screenshots of an SQL Worksheet interface, comparing a 'WRONG' script (left) with a 'RIGHT' script (right). A large red arrow points from the 'WRONG' script to the 'RIGHT' script.

WRONG Script (Left):

```
1 DECLARE
2   a number;
3 BEGIN
4   a := 1;
5   DECLARE
6     b number;
7   BEGIN
8     b := 2;
9     DECLARE
10      c number;
11    BEGIN
12      c := 3;
13      DBMS_OUTPUT.PUT_LINE('<<Bloc III>>');
14      DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
15      DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
16      DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
17    END;
18    DBMS_OUTPUT.PUT_LINE('<<Bloc II>>');
19    DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
20    DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
21    DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
22  END;
23  DBMS_OUTPUT.PUT_LINE('<<Bloc I>>');
24  DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
25  DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
26  DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
27 END;
```

RIGHT Script (Right):

```
1 DECLARE
2   a number;
3 BEGIN
4   a := 1;
5   DECLARE
6     b number;
7   BEGIN
8     b := 2;
9     DECLARE
10      c number;
11    BEGIN
12      c := 3;
13      DBMS_OUTPUT.PUT_LINE('<<Bloc III>>');
14      DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
15      DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
16      DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
17    END;
18    DBMS_OUTPUT.PUT_LINE('<<Bloc II>>');
19    DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
20    DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
21    DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
22  END;
23  DBMS_OUTPUT.PUT_LINE('<<Bloc I>>');
24  DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
25  DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
26  DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
27 END;
```

Script Output (Left - WRONG):

Task completed in 0.092 seconds

06550. 00000 - "line %, column %s:\n%s"

*Cause: Usually a PL/SQL compilation error.

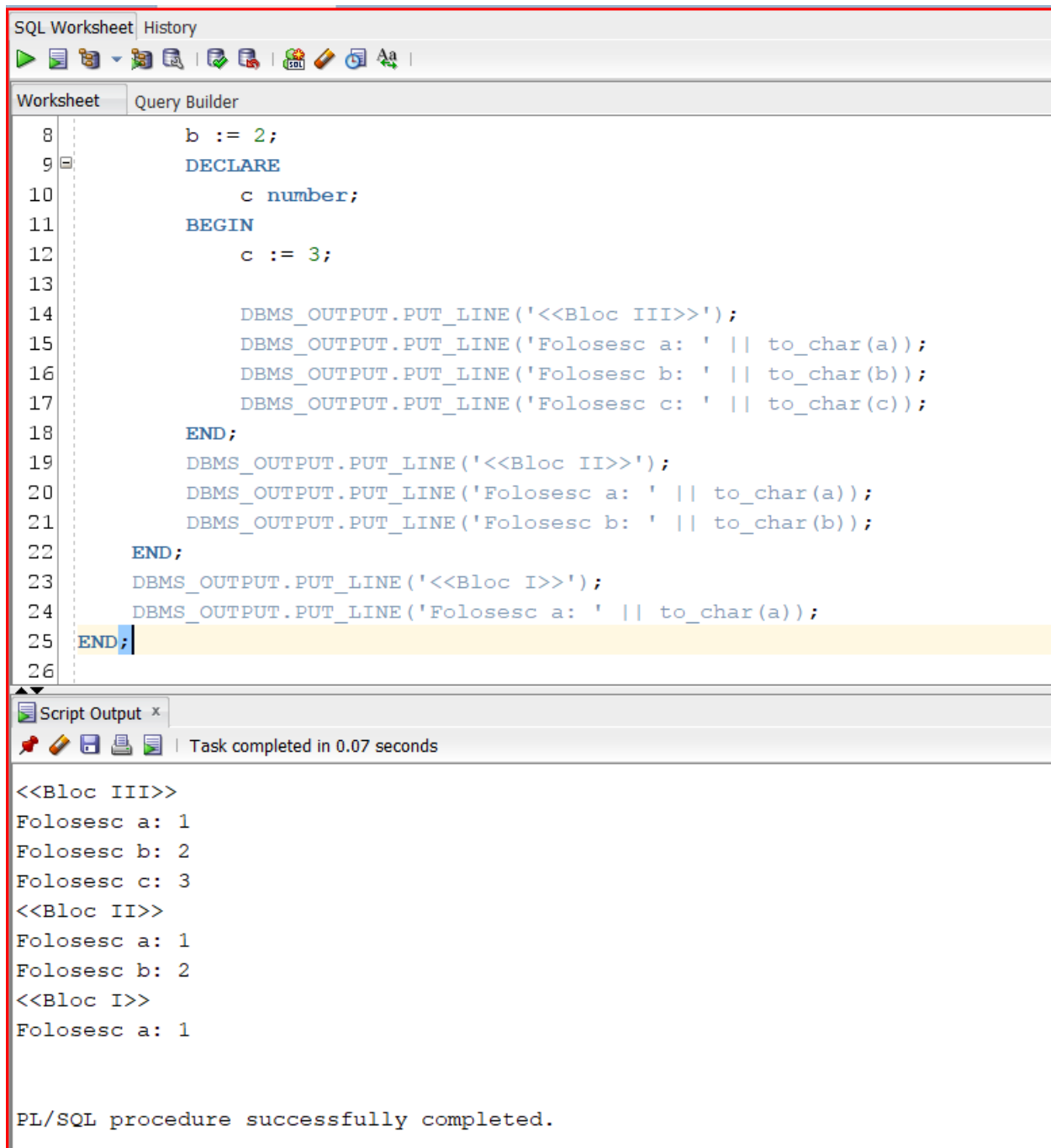
*Action:

Script Output (Right - RIGHT):

Task completed in 0.054 seconds

PL/SQL procedure successfully completed.

- Ce afișează cele 3 blocuri corecte:



The screenshot displays an SQL Worksheet interface with a 'Query Builder' tab. The main area contains a PL/SQL procedure with three blocks of code. The procedure is as follows:

```
8      b := 2;  
9      DECLARE  
10     c number;  
11     BEGIN  
12     c := 3;  
13  
14     DBMS_OUTPUT.PUT_LINE('<<Bloc III>>');  
15     DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));  
16     DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));  
17     DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));  
18     END;  
19     DBMS_OUTPUT.PUT_LINE('<<Bloc II>>');  
20     DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));  
21     DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));  
22     END;  
23     DBMS_OUTPUT.PUT_LINE('<<Bloc I>>');  
24     DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));  
25 END;
```

Below the code editor, the 'Script Output' window shows the results of the execution. It displays the output for each block, indicating that the procedure was successfully completed.

```
<<Bloc III>>  
Folosesc a: 1  
Folosesc b: 2  
Folosesc c: 3  
<<Bloc II>>  
Folosesc a: 1  
Folosesc b: 2  
<<Bloc I>>  
Folosesc a: 1  
  
PL/SQL procedure successfully completed.
```

Vizualizând această soluție corectă și ceea ce afișează ea, ne putem da seama mai bine cum sunt împărțite variabilele pe blocuri și care pot fi văzute de către program la fiecare moment.

- **Exemplul 2** (patru blocuri)

```
SQL Worksheet History
Worksheet Query Builder

1 DECLARE
2     a number;
3 BEGIN
4     a := 1;
5     DECLARE
6         b number;
7     BEGIN
8         b := 2;
9         DECLARE
10            c number;
11        BEGIN
12            c := 3;
13            DECLARE
14                d number;
15            BEGIN
16                d := 4;
17                DBMS_OUTPUT.PUT_LINE('<<Bloc IV>>');
18                DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
19                DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
20                DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
21                DBMS_OUTPUT.PUT_LINE('Folosesc d: ' || to_char(d));
22            END;
23            DBMS_OUTPUT.PUT_LINE('<<Bloc III>>');
24            DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
25            DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
26            DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
27            DBMS_OUTPUT.PUT_LINE('Folosesc d: ' || to_char(d));
28        END;
29        DBMS_OUTPUT.PUT_LINE('<<Bloc II>>');
30        DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
31        DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
32        DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
33        DBMS_OUTPUT.PUT_LINE('Folosesc d: ' || to_char(d));
34    END;
35    DBMS_OUTPUT.PUT_LINE('<<Bloc I>>');
36    DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
37    DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
38    DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
39    DBMS_OUTPUT.PUT_LINE('Folosesc d: ' || to_char(d));
40 END;
```

- Erori:

```
Script Output x
Task completed in 0.099 seconds
DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
DBMS_OUTPUT.PUT_LINE('Folosesc d: ' || to_char(d));
END;
Error report -
ORA-06550: line 27, column 60:
PLS-00201: identifier 'D' must be declared
ORA-06550: line 27, column 13:
PL/SQL: Statement ignored
ORA-06550: line 32, column 56:
PLS-00201: identifier 'C' must be declared
ORA-06550: line 32, column 9:
PL/SQL: Statement ignored
ORA-06550: line 33, column 56:
PLS-00201: identifier 'D' must be declared
ORA-06550: line 33, column 9:
PL/SQL: Statement ignored
ORA-06550: line 37, column 52:
PLS-00201: identifier 'B' must be declared
ORA-06550: line 37, column 5:
PL/SQL: Statement ignored
ORA-06550: line 38, column 52:
PLS-00201: identifier 'C' must be declared
ORA-06550: line 38, column 5:
PL/SQL: Statement ignored
ORA-06550: line 39, column 52:
PLS-00201: identifier 'D' must be declared
ORA-06550: line 39, column 5:
PL/SQL: Statement ignored
06550. 00000 - "line %s, column %s:\n%s"
*Cause:      Usually a PL/SQL compilation error.
*Action:
```

- Patru blocuri fără erori:

The image shows two side-by-side screenshots of a PL/SQL code editor, comparing a 'WRONG' version (left) with a 'RIGHT' version (right). A large red arrow points from the left version to the right version, indicating a correction.

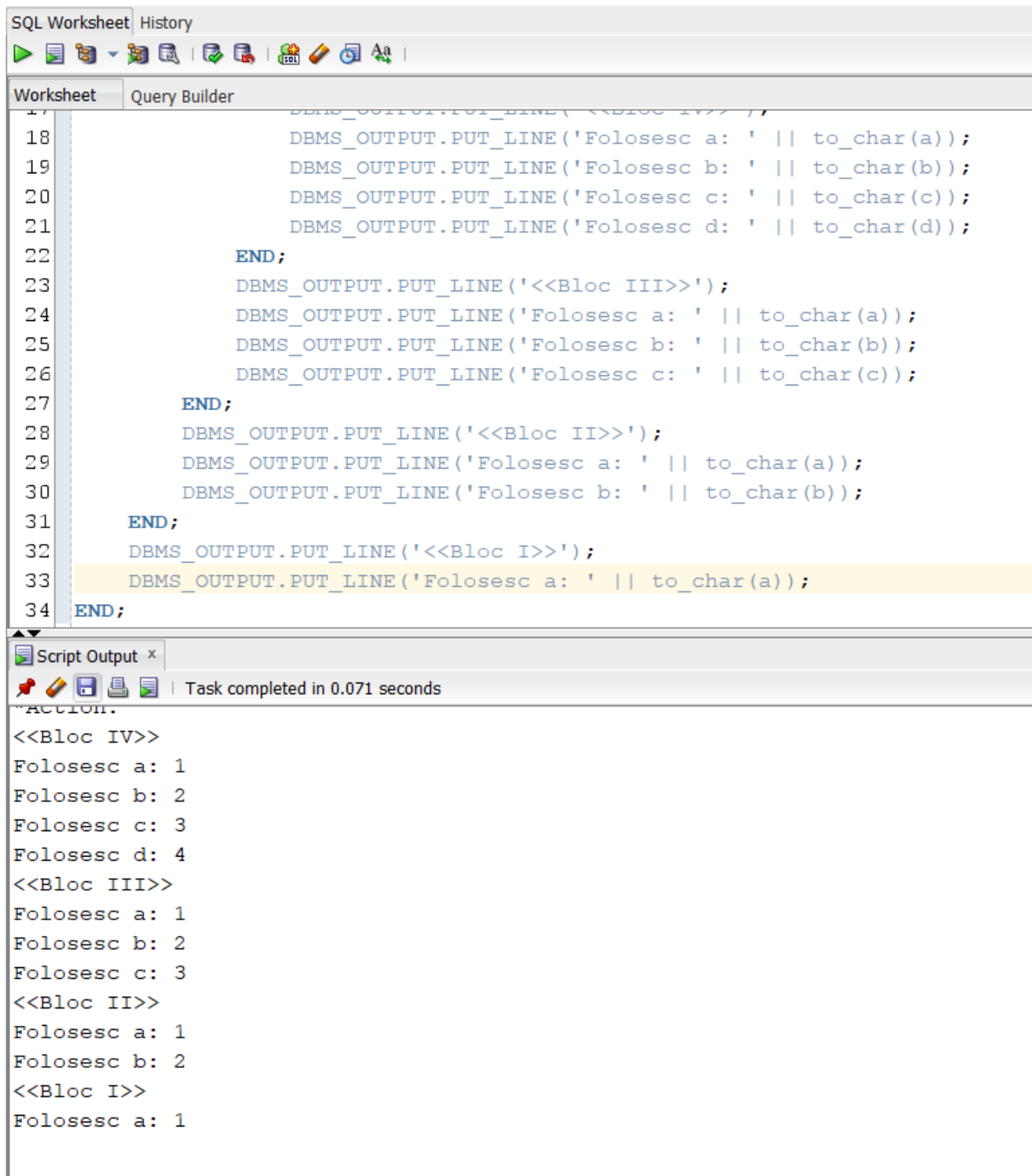
WRONG (Left):

```
1 DECLARE
2   a number;
3 BEGIN
4   a := 1;
5   DECLARE
6     b number;
7   BEGIN
8     b := 2;
9     DECLARE
10      c number;
11    BEGIN
12      c := 3;
13      DECLARE
14        d number;
15      BEGIN
16        d := 4;
17        DBMS_OUTPUT.PUT_LINE('<<Bloc IV>>');
18        DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
19        DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
20        DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
21        DBMS_OUTPUT.PUT_LINE('Folosesc d: ' || to_char(d));
22      END;
23      DBMS_OUTPUT.PUT_LINE('<<Bloc III>>');
24      DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
25      DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
26      DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
27      DBMS_OUTPUT.PUT_LINE('Folosesc d: ' || to_char(d));
28    END;
29    DBMS_OUTPUT.PUT_LINE('<<Bloc II>>');
30    DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
31    DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
32    DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
33    DBMS_OUTPUT.PUT_LINE('Folosesc d: ' || to_char(d));
34  END;
35  DBMS_OUTPUT.PUT_LINE('<<Bloc I>>');
36  DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
37  DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
38  DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
39  DBMS_OUTPUT.PUT_LINE('Folosesc d: ' || to_char(d));
40 END;
```

RIGHT (Right):

```
1 DECLARE
2   a number;
3 BEGIN
4   a := 1;
5   DECLARE
6     b number;
7   BEGIN
8     b := 2;
9     DECLARE
10      c number;
11    BEGIN
12      c := 3;
13      DECLARE
14        d number;
15      BEGIN
16        d := 4;
17        DBMS_OUTPUT.PUT_LINE('<<Bloc IV>>');
18        DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
19        DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
20        DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
21        DBMS_OUTPUT.PUT_LINE('Folosesc d: ' || to_char(d));
22      END;
23      DBMS_OUTPUT.PUT_LINE('<<Bloc III>>');
24      DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
25      DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
26      DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
27    END;
28    DBMS_OUTPUT.PUT_LINE('<<Bloc II>>');
29    DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
30    DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
31  END;
32  DBMS_OUTPUT.PUT_LINE('<<Bloc I>>');
33  DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
34 END;
```


- Ce afișează cele patru blocuri corecte:



The screenshot displays an SQL Worksheet interface with a 'Query Builder' tab. The script contains four blocks of code, each using `DBMS_OUTPUT.PUT_LINE` to print values. The script is as follows:

```
17 DBMS_OUTPUT.PUT_LINE(' <<Bloc IV>> ');
18 DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
19 DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
20 DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
21 DBMS_OUTPUT.PUT_LINE('Folosesc d: ' || to_char(d));
22 END;
23 DBMS_OUTPUT.PUT_LINE(' <<Bloc III>> ');
24 DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
25 DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
26 DBMS_OUTPUT.PUT_LINE('Folosesc c: ' || to_char(c));
27 END;
28 DBMS_OUTPUT.PUT_LINE(' <<Bloc II>> ');
29 DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
30 DBMS_OUTPUT.PUT_LINE('Folosesc b: ' || to_char(b));
31 END;
32 DBMS_OUTPUT.PUT_LINE(' <<Bloc I>> ');
33 DBMS_OUTPUT.PUT_LINE('Folosesc a: ' || to_char(a));
34 END;
```

Below the script, the 'Script Output' window shows the results of the execution. The output is as follows:

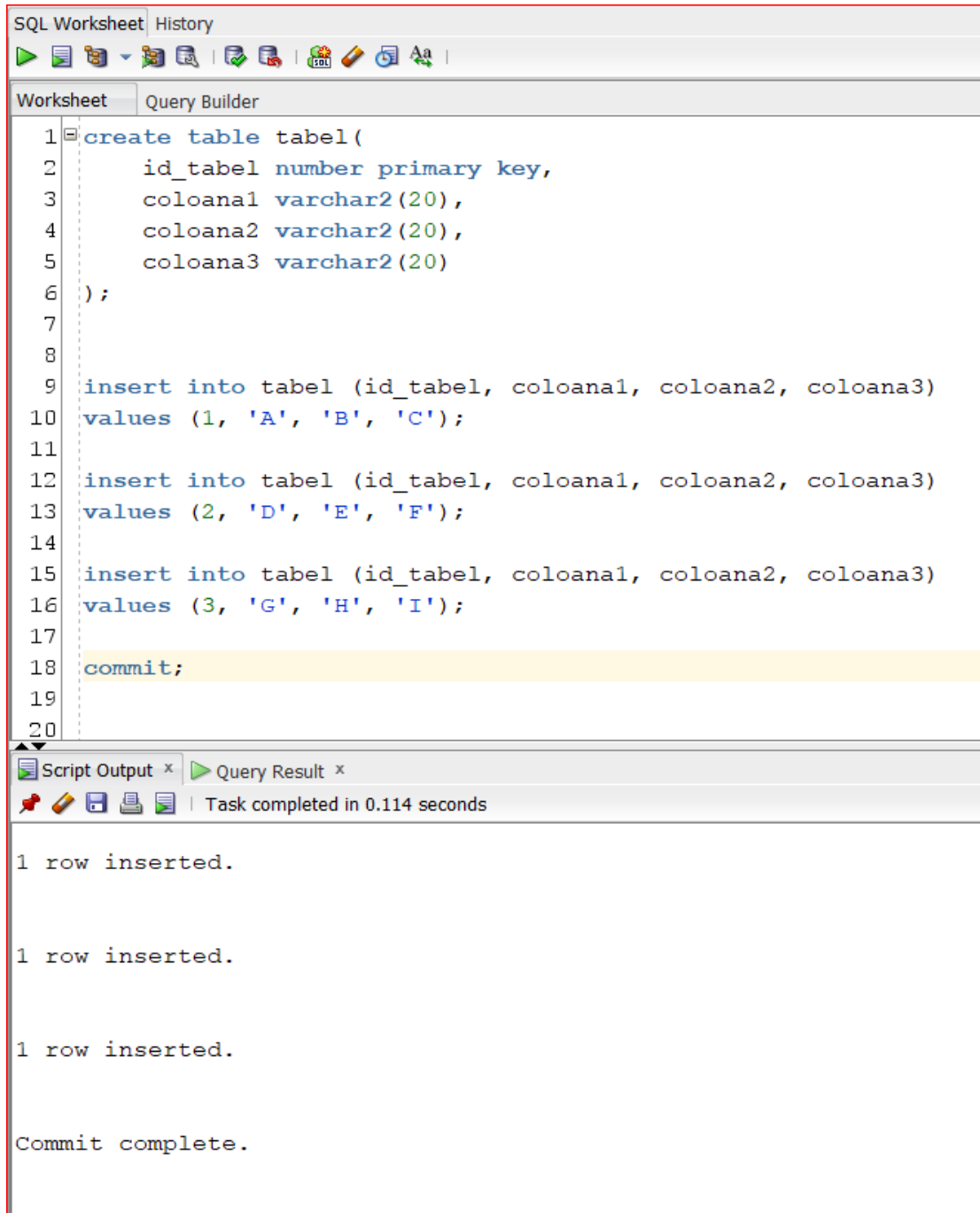
```
Script Output x
Task completed in 0.071 seconds
**ACTION.**
<<Bloc IV>>
Folosesc a: 1
Folosesc b: 2
Folosesc c: 3
Folosesc d: 4
<<Bloc III>>
Folosesc a: 1
Folosesc b: 2
Folosesc c: 3
<<Bloc II>>
Folosesc a: 1
Folosesc b: 2
<<Bloc I>>
Folosesc a: 1
```

În concluzie, datorită acestor exemple am observat cum pot fi declarate variabilele local în blocuri și la ce momente pot fi văzute în program, dar și când nu le mai putem apela.

Instrucțiunea DELETE + RETURNING

Despre instrucțiunea de DELETE cu RETURNING știm că trebuie să returneze exact o linie, dar și că, în cazul în care încercăm să returnăm mai multe linii, ne va fi afișată o eroare (TOO_MANY_ROWS). Dar ce se întâmplă în cazul în care nu returnăm nicio linie? Să vedem:

Creăm tabelul TABEL astfel:



The screenshot displays an SQL Worksheet interface with a toolbar at the top containing icons for execution, saving, and other database operations. The main workspace is divided into two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, showing a SQL script with line numbers 1 through 20. The script defines a table 'tabel' with a primary key 'id_tabel' and three varchar columns. It then inserts three rows of data and commits the transaction. Below the script, the 'Script Output' and 'Query Result' tabs are visible. The 'Script Output' tab shows the execution results: '1 row inserted.' for each of the three insert statements, followed by 'Commit complete.' The status bar at the bottom indicates 'Task completed in 0.114 seconds'.

```
1 create table tabel(  
2     id_tabel number primary key,  
3     coloana1 varchar2(20),  
4     coloana2 varchar2(20),  
5     coloana3 varchar2(20)  
6 );  
7  
8  
9 insert into tabel (id_tabel, coloana1, coloana2, coloana3)  
10 values (1, 'A', 'B', 'C');  
11  
12 insert into tabel (id_tabel, coloana1, coloana2, coloana3)  
13 values (2, 'D', 'E', 'F');  
14  
15 insert into tabel (id_tabel, coloana1, coloana2, coloana3)  
16 values (3, 'G', 'H', 'I');  
17  
18 commit;  
19  
20
```

Script Output x Query Result x
Task completed in 0.114 seconds

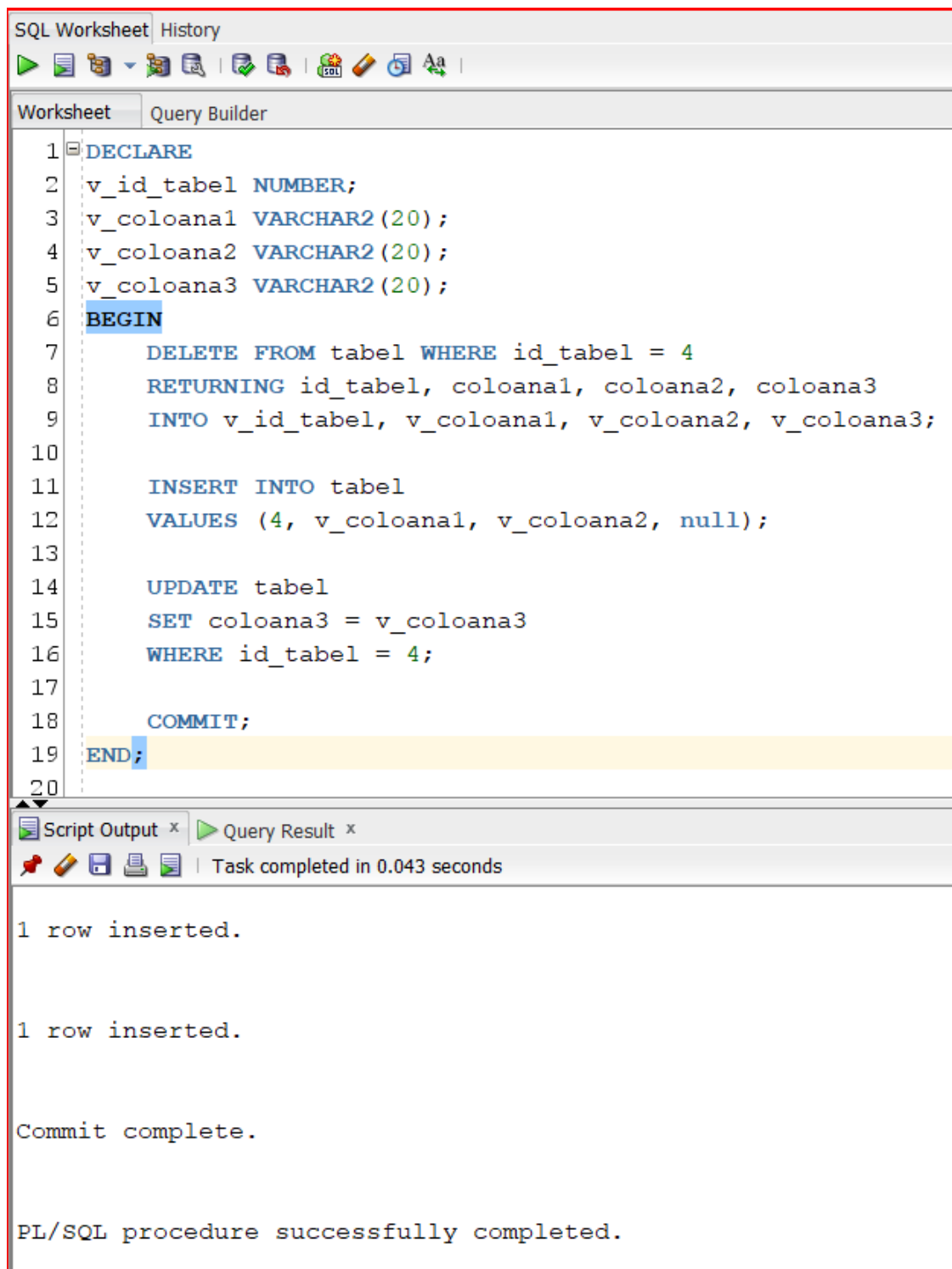
1 row inserted.

1 row inserted.

1 row inserted.

Commit complete.

Observăm că id-urile sunt începând de la 1 până la 3. Astfel, vom încerca să ștergem, cu ajutorul lui DELETE cu RETURNING, înregistrarea cu id-ul 4 (care, de fapt, nu există).



The screenshot displays an SQL Worksheet interface with a toolbar at the top containing icons for running, saving, and other operations. The main area is divided into two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, showing a PL/SQL procedure with the following code:

```
1 DECLARE
2   v_id_tabel NUMBER;
3   v_coloana1 VARCHAR2(20);
4   v_coloana2 VARCHAR2(20);
5   v_coloana3 VARCHAR2(20);
6 BEGIN
7     DELETE FROM tabel WHERE id_tabel = 4
8     RETURNING id_tabel, coloana1, coloana2, coloana3
9     INTO v_id_tabel, v_coloana1, v_coloana2, v_coloana3;
10
11     INSERT INTO tabel
12     VALUES (4, v_coloana1, v_coloana2, null);
13
14     UPDATE tabel
15     SET coloana3 = v_coloana3
16     WHERE id_tabel = 4;
17
18     COMMIT;
19 END;
```

Below the code editor, there is a 'Script Output' tab and a 'Query Result' tab. The 'Script Output' tab is active, showing the execution results of the procedure:

```
1 row inserted.

1 row inserted.

Commit complete.

PL/SQL procedure successfully completed.
```

După ce executăm setul de instrucțiuni, ne apare în consolă “PL/SQL procedure successfully completed”, contrar așteptărilor de a apărea o eroare de tip NO_DATA_FOUND. Practic, neavând înregistrarea cu id-ul 4, instrucțiunea de DELETE cu RETURNING a trecut peste acest lucru și a înlocuit toate coloanele returnate cu NULL.

Remarcăm faptul că înregistrarea a fost adăugată, cu adevărat, în tabelul nostru de test:

The screenshot displays an SQL Worksheet interface. The main editor shows a PL/SQL script with the following code:

```
1 DECLARE
2   v_id_tabel NUMBER;
3   v_coloana1 VARCHAR2(20);
4   v_coloana2 VARCHAR2(20);
5   v_coloana3 VARCHAR2(20);
6 BEGIN
7     DELETE FROM tabel WHERE id_tabel = 4
8     RETURNING id_tabel, coloana1, coloana2, coloana3
9     INTO v_id_tabel, v_coloana1, v_coloana2, v_coloana3;
10
11    INSERT INTO tabel
12    VALUES (4, v_coloana1, v_coloana2, null);
13
14    UPDATE tabel
15    SET coloana3 = v_coloana3
16    WHERE id_tabel = 4;
17
18    COMMIT;
19 END;
20
21 select * from tabel;
```

Below the script editor, the 'Query Result' tab is active, showing the output of the final SELECT statement. The status bar indicates 'All Rows Fetched: 4 in 0.016 seconds'. The result is displayed in a table with 4 rows and 4 columns: ID_TABEL, COLOANA1, COLOANA2, and COLOANA3.

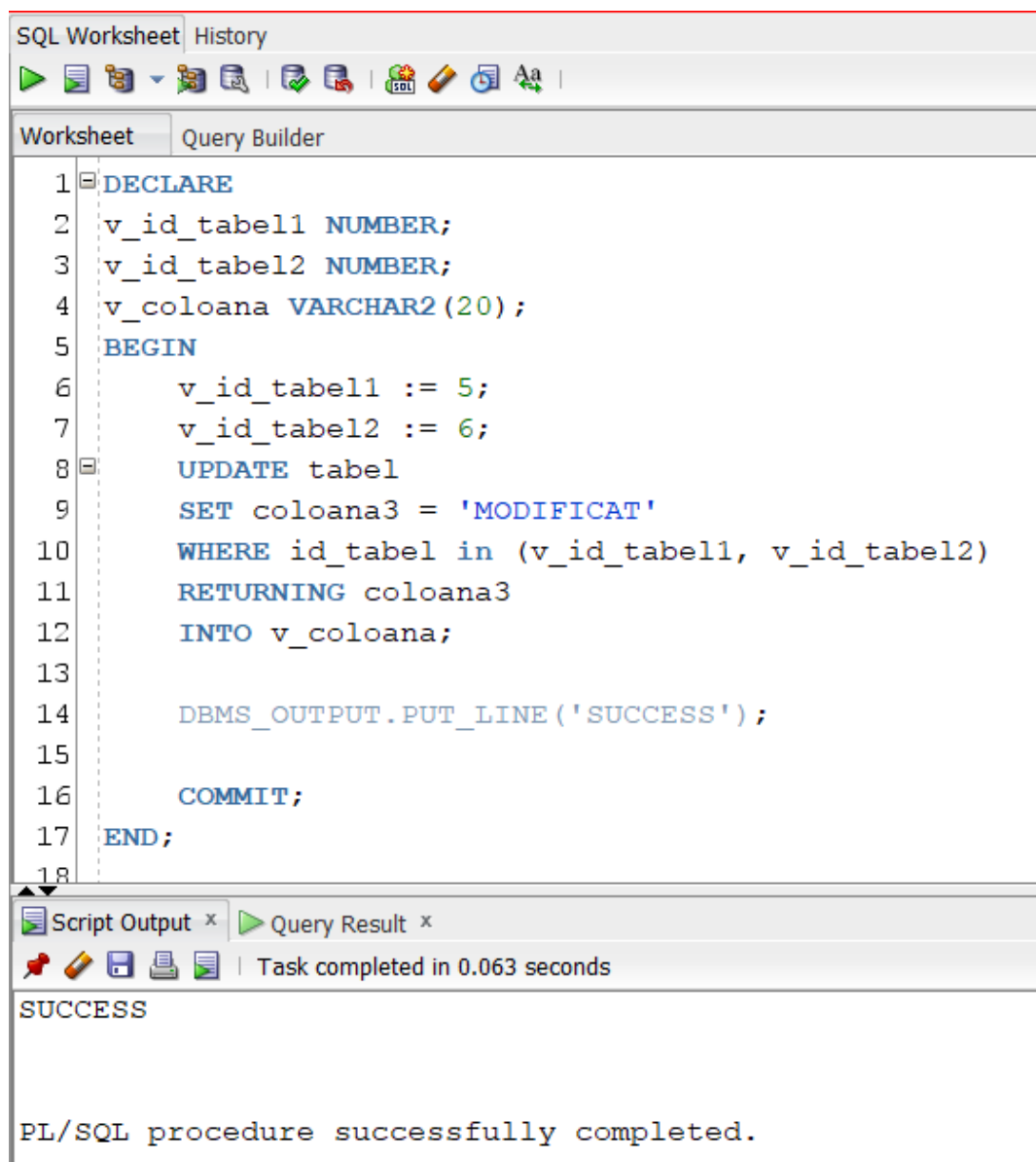
	ID_TABEL	COLOANA1	COLOANA2	COLOANA3
1	1	A	B	C
2	2	D	E	F
3	3	G	H	I
4	4	(null)	(null)	(null)

Instrucțiunea UPDATE + RETURNING

Mai devreme am văzut cum se comportă instrucțiunea DELETE cu RETURNING. Dar cea de UPDATE cu RETURNING? Se va comporta la fel? Să vedem:

Folosim același tabel de test ca în exemplul anterior.

Știm că DELETE cu RETURNING nu va afișa nicio eroare în cazul în care nu avem nicio linie. Remarcăm faptul că UPDATE cu RETURNING face același lucru:



```
1 DECLARE
2   v_id_tabel1 NUMBER;
3   v_id_tabel2 NUMBER;
4   v_coloana VARCHAR2(20);
5 BEGIN
6     v_id_tabel1 := 5;
7     v_id_tabel2 := 6;
8     UPDATE tabel
9     SET coloana3 = 'MODIFICAT'
10    WHERE id_tabel in (v_id_tabel1, v_id_tabel2)
11    RETURNING coloana3
12    INTO v_coloana;
13
14     DBMS_OUTPUT.PUT_LINE('SUCCESS');
15
16     COMMIT;
17 END;
```

Script Output x Query Result x

Task completed in 0.063 seconds

SUCCESS

PL/SQL procedure successfully completed.

Afișăm în consolă tabelul și observăm că nimic nu s-a întâmplat.

The screenshot shows an SQL IDE interface. The top pane, titled 'Worksheet', contains an SQL script. The script defines two variables, `v_id_tabel1` and `v_id_tabel2`, and updates the `coloana3` column of a table named `tabel` for rows where `id_tabel` is in the set of these two variables. The script also includes a `RETURNING` clause to capture the updated values into `v_coloana`. The bottom pane, titled 'Query Result', displays the results of the `select * from tabel;` query. The results show four rows of data, with the last row having null values for the columns `coloana1`, `coloana2`, and `coloana3`.

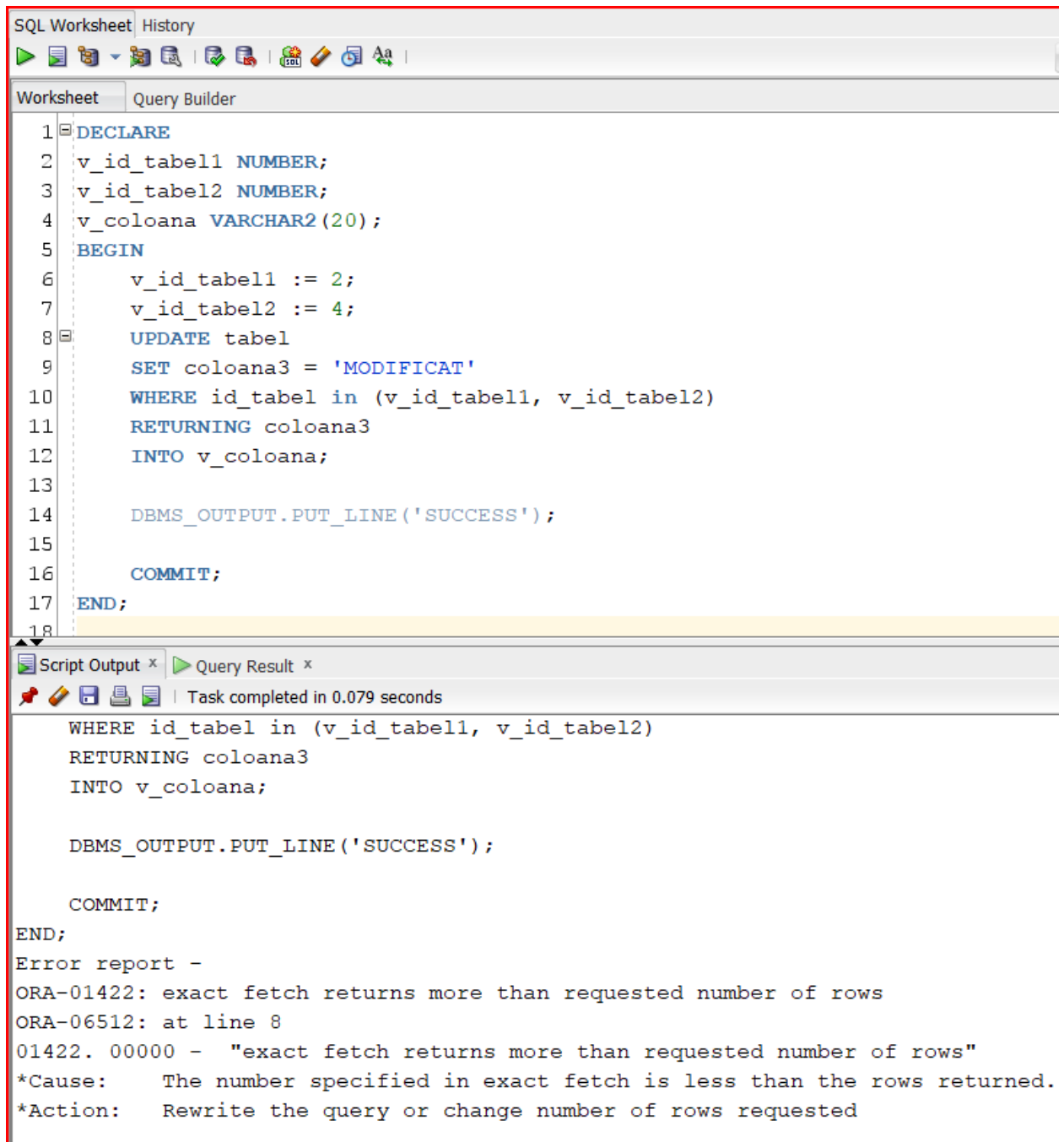
```
3 v_id_tabel2 NUMBER;  
4 v_coloana VARCHAR2(20);  
5 BEGIN  
6     v_id_tabel1 := 5;  
7     v_id_tabel2 := 6;  
8     UPDATE tabel  
9     SET coloana3 = 'MODIFICAT'  
10    WHERE id_tabel in (v_id_tabel1, v_id_tabel2)  
11    RETURNING coloana3  
12    INTO v_coloana;  
13  
14    DBMS_OUTPUT.PUT_LINE('SUCCESS');  
15  
16    COMMIT;  
17 END;  
18  
19 select * from tabel;  
20
```

Script Output x Query Result x

SQL | All Rows Fetched: 4 in 0.015 seconds

	ID_TABEL	COLOANA1	COLOANA2	COLOANA3
1	1	A	B	C
2	2	D	E	F
3	3	G	H	I
4	4	(null)	(null)	(null)

De asemenea, știm că DELETE cu RETURNING va afișa o eroare în cazul mai multor linii. Observăm că UPDATE cu RETURNING face același lucru, returnând eroarea de tip TOO_MANY_ROWS:



```
SQL Worksheet History
Worksheet Query Builder
1 DECLARE
2   v_id_tabel1 NUMBER;
3   v_id_tabel2 NUMBER;
4   v_coloana VARCHAR2(20);
5 BEGIN
6     v_id_tabel1 := 2;
7     v_id_tabel2 := 4;
8     UPDATE tabel
9     SET coloana3 = 'MODIFICAT'
10    WHERE id_tabel in (v_id_tabel1, v_id_tabel2)
11    RETURNING coloana3
12    INTO v_coloana;
13
14    DBMS_OUTPUT.PUT_LINE('SUCCESS');
15
16    COMMIT;
17 END;
18

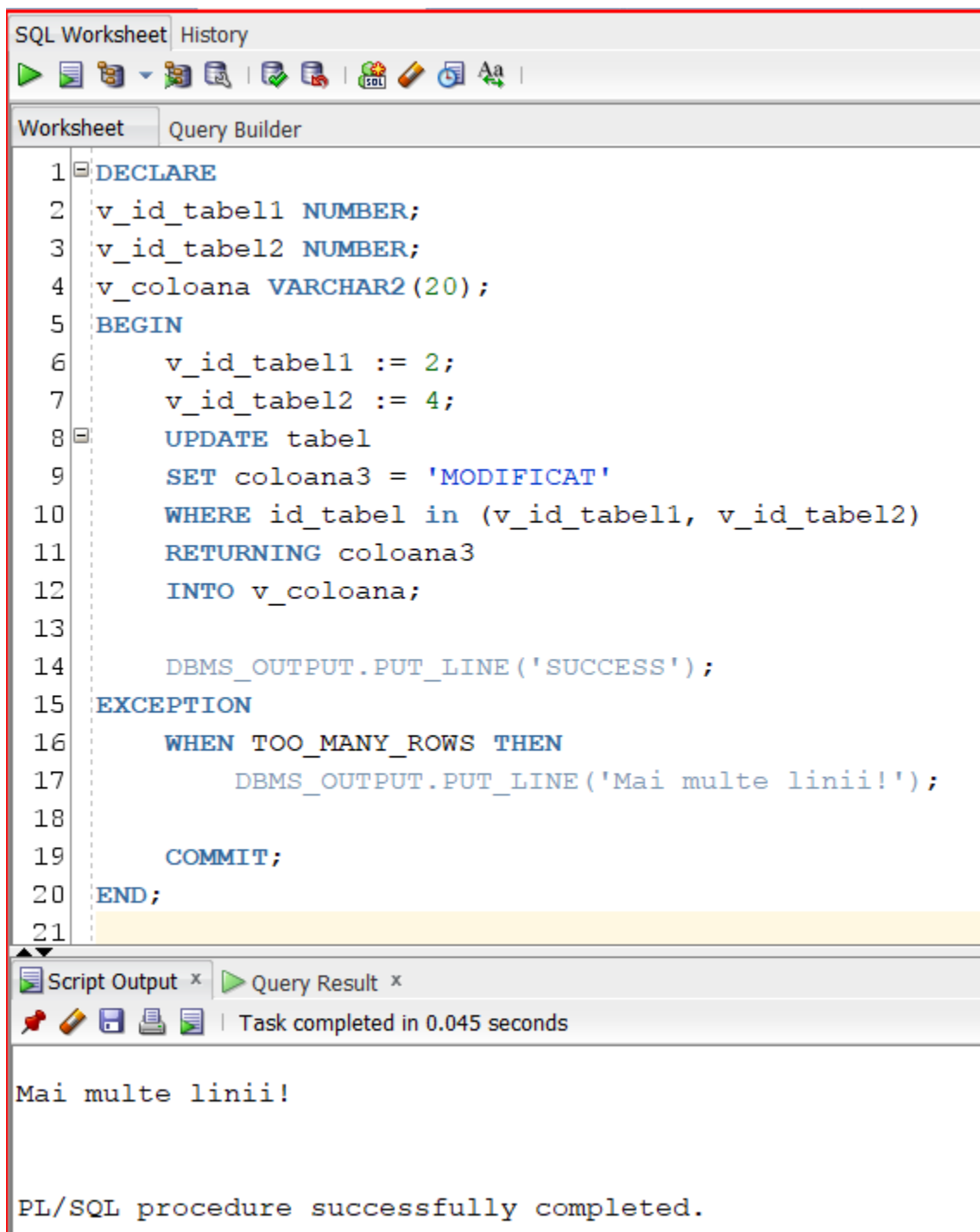
Script Output x Query Result x
Task completed in 0.079 seconds

WHERE id_tabel in (v_id_tabel1, v_id_tabel2)
RETURNING coloana3
INTO v_coloana;

DBMS_OUTPUT.PUT_LINE('SUCCESS');

COMMIT;
END;
Error report -
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 8
01422. 00000 - "exact fetch returns more than requested number of rows"
*Cause:      The number specified in exact fetch is less than the rows returned.
*Action:     Rewrite the query or change number of rows requested
```

Acest lucru poate fi văzut și astfel:



The screenshot displays an SQL Worksheet application window. The top menu bar includes 'SQL Worksheet' and 'History'. Below it is a toolbar with various icons for file operations and execution. The main area is divided into two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, showing a PL/SQL script with line numbers 1 through 21. The script declares two variables, v_id_tabel1 and v_id_tabel2, and a variable v_coloana. It then updates a table named 'tabel' where the id_tabel is in the list of v_id_tabel1 and v_id_tabel2, returning the value of coloana3 into v_coloana. An exception block handles the 'TOO_MANY_ROWS' error by displaying 'Mai multe linii!'. The script ends with a COMMIT and an END statement. Below the script, there are two tabs: 'Script Output' and 'Query Result'. The 'Script Output' tab is active, showing the message 'Mai multe linii!' and 'PL/SQL procedure successfully completed.'.

```
1 DECLARE
2   v_id_tabel1 NUMBER;
3   v_id_tabel2 NUMBER;
4   v_coloana VARCHAR2(20);
5   BEGIN
6       v_id_tabel1 := 2;
7       v_id_tabel2 := 4;
8       UPDATE tabel
9       SET coloana3 = 'MODIFICAT'
10      WHERE id_tabel in (v_id_tabel1, v_id_tabel2)
11      RETURNING coloana3
12      INTO v_coloana;
13
14      DBMS_OUTPUT.PUT_LINE('SUCCESS');
15  EXCEPTION
16      WHEN TOO_MANY_ROWS THEN
17          DBMS_OUTPUT.PUT_LINE('Mai multe linii!');
18
19      COMMIT;
20  END;
21
```

Script Output x Query Result x

Task completed in 0.045 seconds

Mai multe linii!

PL/SQL procedure successfully completed.

Așadar, instrucțiunile DELETE cu RETURNING și UPDATE cu RETURNING se comportă la fel: prezintă erori atunci când sunt mai multe linii, dar nicio eroare atunci când nu sunt date. În cazul din urmă, valorile returnate (RETURNING) sunt înlocuite cu NULL, fapt datorat negăsirii acestora în înregistrările deja existente.