TEMĂ

În această temă voi prezenta două exemple asemănătoare cu exemplul 3.3 din curs, dar și verificarea comenzii delete cu returning.

***Exemplu cu 3 Blocuri.***

Să se efectueze următoarele operații în ordine:

* baza2 = baza2 la puterea baza1.
* baza1 = baza1 la puterea baza1.
* baza2 = baza2 % baza3.
* baza3 = baza3 \* baza1 \* baza2.

***Rezolvare:***

DECLARE

v\_baza1 INTEGER;

BEGIN

v\_baza1 := 12;

DECLARE

v\_baza2 INTEGER := 10;

BEGIN

v\_baza2 := v\_baza2\*\*v\_baza1;

DBMS\_OUTPUT.PUT\_LINE(v\_baza2);

v\_baza1 := v\_baza1\*\*v\_baza1;

DBMS\_OUTPUT.PUT\_LINE(v\_baza1);

DECLARE

v\_baza3 INTEGER := 7;

BEGIN

v\_baza2 := mod(v\_baza2, v\_baza3);

DBMS\_OUTPUT.PUT\_LINE(v\_baza2);

v\_baza3 := v\_baza3 \* v\_baza1 \* v\_baza2;

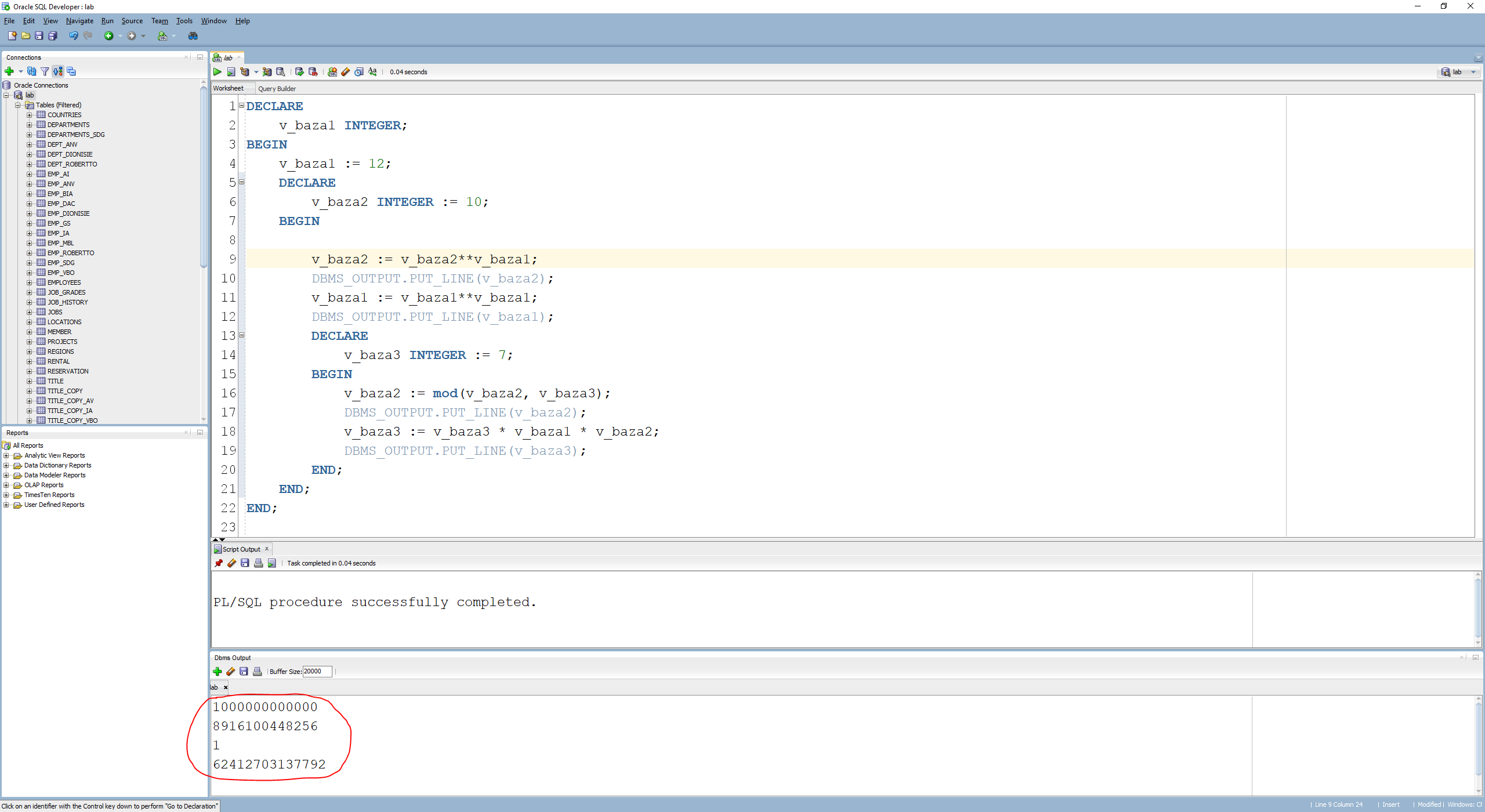
DBMS\_OUTPUT.PUT\_LINE(v\_baza3);

END;

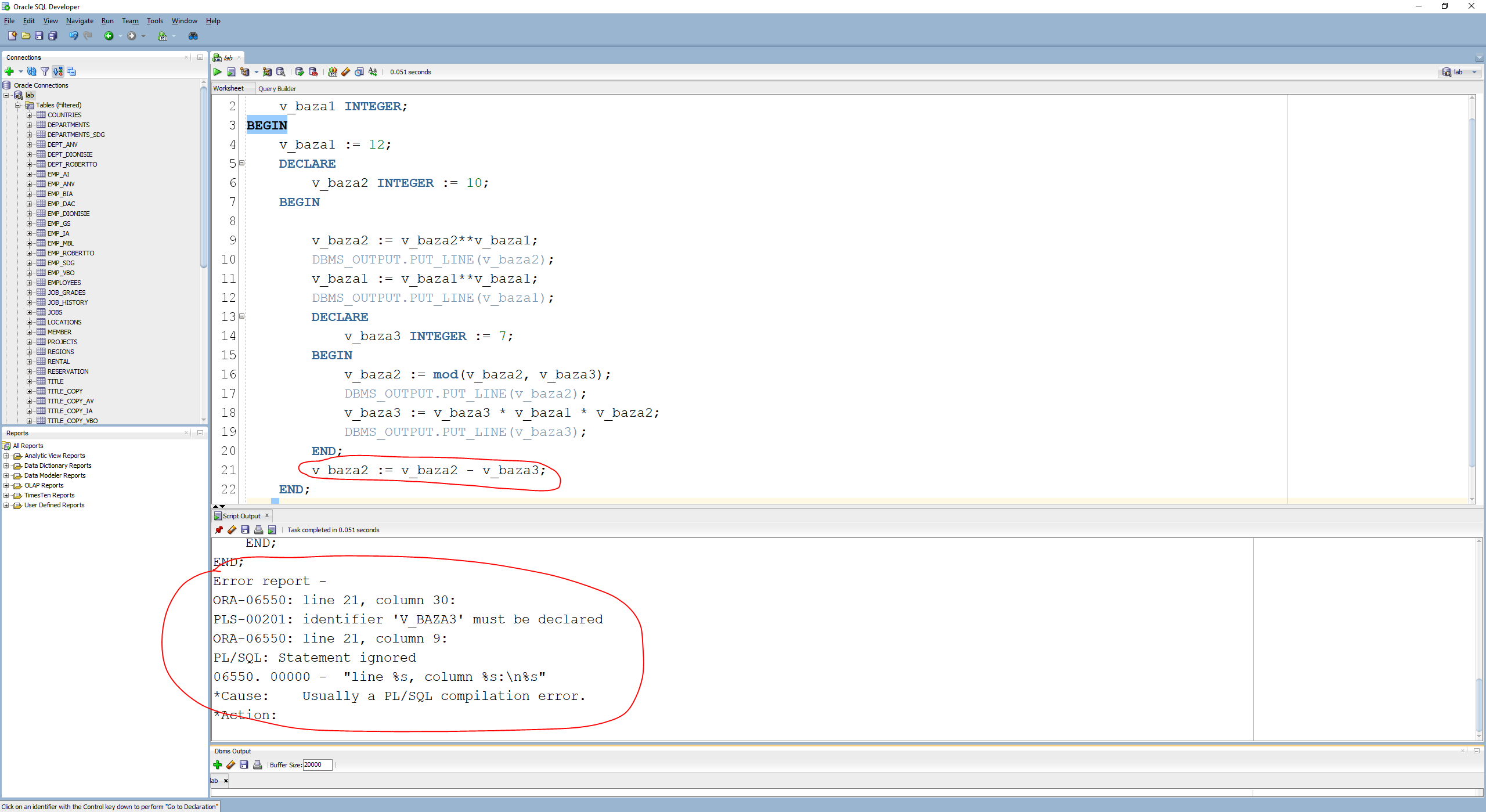
END;

END;

***Print-Screen:***



Încercăm să modificăm baza3 în interiorul blocului baza2 (baza2=baza2-baza3).



Observăm că nu putem modifica variabila baza2 întrucât variabila baza3 nu a fost declarată. Ea a fost declarată local în blocul baza3 deci, odată ce s-a terminat blocul baza3, nu mai poate fi accesibilă (a ieșit din scope, primim eroare de compilare).

***Exemplu cu 4 Blocuri.***

* Să se afișeze variabila din blocul 1 în interiorul blocului 1.
* Să se afișeze variabila din blocul 1 în interiorul blocului 2.
* Să se afișeze variabila din blocul 2 în interiorul blocului 2.
* Să se calculeze și să se afișeze suma dintre blocul 1 si blocul 2, în interiorul blocului 2.
* Să se calculeze și să se afișeze diferența dintre blocul 1 si blocul 2, în interiorul blocului 2.
* Să se afișeze variabila din blocul 1 în interiorul blocului 3.
* Să se afișeze variabila din blocul 2 în interiorul blocului 3.
* Să se afișeze variabila din blocul 3 în interiorul blocului 3.
* Să se calculeze și să se afișeze câtul împărțirii **bloc2/(bloc1/bloc3)**, în interiorul blocului 3.
* Să se afișeze variabila din blocul 1 în interiorul blocului 4.
* Să se afișeze variabila din blocul 2 în interiorul blocului 4.
* Să se afișeze variabila din blocul 3 în interiorul blocului 4.
* Să se afișeze variabila din blocul 4 în interiorul blocului 4.
* Să se calculeze și să se afișeze restul împărțirii **bloc4%(bloc1/bloc3)**, în interiorul blocului 4.
* Să se modifice valoarea din blocul 1 în interiorul blocului 4 cu 121.
* Să se modifice valoarea din blocul 2 în interiorul blocului 4 cu 11.
* Să se modifice valoarea din blocul 2 în interiorul blocului 4 cu 7.
* Să se modifice valoarea din blocul 2 în interiorul blocului 4 cu 3.

***Rezolvare:***

DECLARE

v\_nume\_bloc\_1 VARCHAR2(30);

v\_bloc\_1 NUMBER(10);

v\_suma NUMBER(10) := 0;

v\_diferenta NUMBER(10) := 0;

v\_vizitat VARCHAR2(100);

BEGIN

v\_nume\_bloc\_1 := 'PRINCIPAL';

v\_bloc\_1 := 64;

DBMS\_OUTPUT.PUT\_LINE('!!! AM INTRAT IN BLOCUL ' || v\_nume\_bloc\_1 || ' !!!');

DBMS\_OUTPUT.PUT\_LINE('Accesez variabila din blocul ' || v\_nume\_bloc\_1 || ' in blocul ' || v\_nume\_bloc\_1 || ', care are valoarea ' || v\_bloc\_1 || '.');

DECLARE

v\_nume\_bloc\_2 VARCHAR2(30);

v\_bloc\_2 NUMBER(10);

BEGIN

v\_nume\_bloc\_2 := 'SECUNDAR';

v\_bloc\_2 := 8;

DBMS\_OUTPUT.PUT\_LINE('!!! AM INTRAT IN BLOCUL ' || v\_nume\_bloc\_2 || ' !!!');

DBMS\_OUTPUT.PUT\_LINE('Accesez variabila din blocul ' || v\_nume\_bloc\_1 || ' in blocul ' || v\_nume\_bloc\_2 || ', care are valoarea ' || v\_bloc\_1 || '.');

DBMS\_OUTPUT.PUT\_LINE('Accesez variabila din blocul ' || v\_nume\_bloc\_2 || ' in blocul ' || v\_nume\_bloc\_2 || ', care are valoarea ' || v\_bloc\_2 || '.');

DBMS\_OUTPUT.PUT\_LINE('->Calculez suma dintre blocul ' || v\_nume\_bloc\_1 || ' si cel ' || v\_nume\_bloc\_2 || '.');

v\_suma := v\_bloc\_1 + v\_bloc\_2;

DBMS\_OUTPUT.PUT\_LINE('Suma este: ' || v\_suma || '.');

DBMS\_OUTPUT.PUT\_LINE('->Calculez diferenta dintre blocul ' || v\_nume\_bloc\_1 || ' si cel ' || v\_nume\_bloc\_2 || '.');

v\_diferenta := v\_bloc\_1 - v\_bloc\_2;

DBMS\_OUTPUT.PUT\_LINE('Diferenta este: ' || v\_diferenta || '.');

DECLARE

v\_nume\_bloc\_3 VARCHAR2(30);

v\_bloc\_3 NUMBER(10);

v\_impartire NUMBER(10) := 0;

BEGIN

v\_nume\_bloc\_3 := 'SECUNDAR2';

v\_bloc\_3 := 16;

DBMS\_OUTPUT.PUT\_LINE('!!! AM INTRAT IN BLOCUL ' || v\_nume\_bloc\_3 || ' !!!');

DBMS\_OUTPUT.PUT\_LINE('Accesez variabila din blocul ' || v\_nume\_bloc\_1 || ' in blocul ' || v\_nume\_bloc\_3 || ', care are valoarea ' || v\_bloc\_1 || '.');

DBMS\_OUTPUT.PUT\_LINE('Accesez variabila din blocul ' || v\_nume\_bloc\_2 || ' in blocul ' || v\_nume\_bloc\_3 || ', care are valoarea ' || v\_bloc\_2 || '.');

DBMS\_OUTPUT.PUT\_LINE('Accesez variabila din blocul ' || v\_nume\_bloc\_3 || ' in blocul ' || v\_nume\_bloc\_3 || ', care are valoarea ' || v\_bloc\_3 || '.');

DBMS\_OUTPUT.PUT\_LINE('->Calculez catul impartirii: ' || v\_nume\_bloc\_2 || '/(' || v\_nume\_bloc\_1 || '/' || v\_nume\_bloc\_3 || ').');

v\_impartire := v\_bloc\_2 / (v\_bloc\_1/v\_bloc\_3);

DBMS\_OUTPUT.PUT\_LINE('Catul este: ' || v\_impartire || '.');

DECLARE

v\_nume\_bloc\_4 VARCHAR2(30);

v\_bloc\_4 NUMBER(10);

v\_rest NUMBER(10) := 0;

BEGIN

v\_nume\_bloc\_4 := 'SECUNDAR3';

v\_bloc\_4 := 5;

DBMS\_OUTPUT.PUT\_LINE('!!! AM INTRAT IN BLOCUL ' || v\_nume\_bloc\_4 || ' !!!');

DBMS\_OUTPUT.PUT\_LINE('Accesez variabila din blocul ' || v\_nume\_bloc\_1 || ' in blocul ' || v\_nume\_bloc\_4 || ', care are valoarea ' || v\_bloc\_1 || '.');

DBMS\_OUTPUT.PUT\_LINE('Accesez variabila din blocul ' || v\_nume\_bloc\_2 || ' in blocul ' || v\_nume\_bloc\_4 || ', care are valoarea ' || v\_bloc\_2 || '.');

DBMS\_OUTPUT.PUT\_LINE('Accesez variabila din blocul ' || v\_nume\_bloc\_3 || ' in blocul ' || v\_nume\_bloc\_4 || ', care are valoarea ' || v\_bloc\_3 || '.');

DBMS\_OUTPUT.PUT\_LINE('Accesez variabila din blocul ' || v\_nume\_bloc\_4 || ' in blocul ' || v\_nume\_bloc\_4 || ', care are valoarea ' || v\_bloc\_4 || '.');

DBMS\_OUTPUT.PUT\_LINE('->Calculez restul impartirii: ' || v\_nume\_bloc\_4 || '%(' || v\_nume\_bloc\_1 || '/' || v\_nume\_bloc\_3 || ').');

v\_rest := mod(v\_bloc\_4, v\_bloc\_1/v\_bloc\_3);

DBMS\_OUTPUT.PUT\_LINE('Restul este: ' || v\_rest || '.');

DBMS\_OUTPUT.PUT\_LINE('Incercam sa dam valoarea 121 variabilei din blocul ' || v\_nume\_bloc\_1 || '.');

v\_bloc\_1 := 121;

DBMS\_OUTPUT.PUT\_LINE('Variabila din blocul ' || v\_nume\_bloc\_1 || ' are valoarea acum: ' || v\_bloc\_1 || '.');

DBMS\_OUTPUT.PUT\_LINE('Incercam sa dam valoarea 11 variabilei din blocul ' || v\_nume\_bloc\_2 || '.');

v\_bloc\_2 := 11;

DBMS\_OUTPUT.PUT\_LINE('Variabila din blocul ' || v\_nume\_bloc\_2 || ' are valoarea acum: ' || v\_bloc\_2 || '.');

DBMS\_OUTPUT.PUT\_LINE('Incercam sa dam valoarea 7 variabilei din blocul ' || v\_nume\_bloc\_3 || '.');

v\_bloc\_3 := 7;

DBMS\_OUTPUT.PUT\_LINE('Variabila din blocul ' || v\_nume\_bloc\_3 || ' are valoarea acum: ' || v\_bloc\_3 || '.');

DBMS\_OUTPUT.PUT\_LINE('Incercam sa dam valoarea 3 variabilei din blocul ' || v\_nume\_bloc\_4 || '.');

v\_bloc\_4 := 3;

DBMS\_OUTPUT.PUT\_LINE('Variabila din blocul ' || v\_nume\_bloc\_4 || ' are valoarea acum: ' || v\_bloc\_4 || '.');

DBMS\_OUTPUT.PUT\_LINE('!!! AM IESIT DIN BLOCUL ' || v\_nume\_bloc\_4 || ' !!!');

v\_vizitat := v\_nume\_bloc\_4;

END;

DBMS\_OUTPUT.PUT\_LINE('!!! AM IESIT DIN BLOCUL ' || v\_nume\_bloc\_3 || ' !!!');

v\_vizitat := v\_vizitat || ', ' || v\_nume\_bloc\_3;

END;

DBMS\_OUTPUT.PUT\_LINE('!!! AM IESIT DIN BLOCUL ' || v\_nume\_bloc\_2 || ' !!!');

v\_vizitat := v\_vizitat || ', ' || v\_nume\_bloc\_2;

END;

DBMS\_OUTPUT.PUT\_LINE('!!! AM IESIT DIN BLOCUL ' || v\_nume\_bloc\_1 || ' !!!');

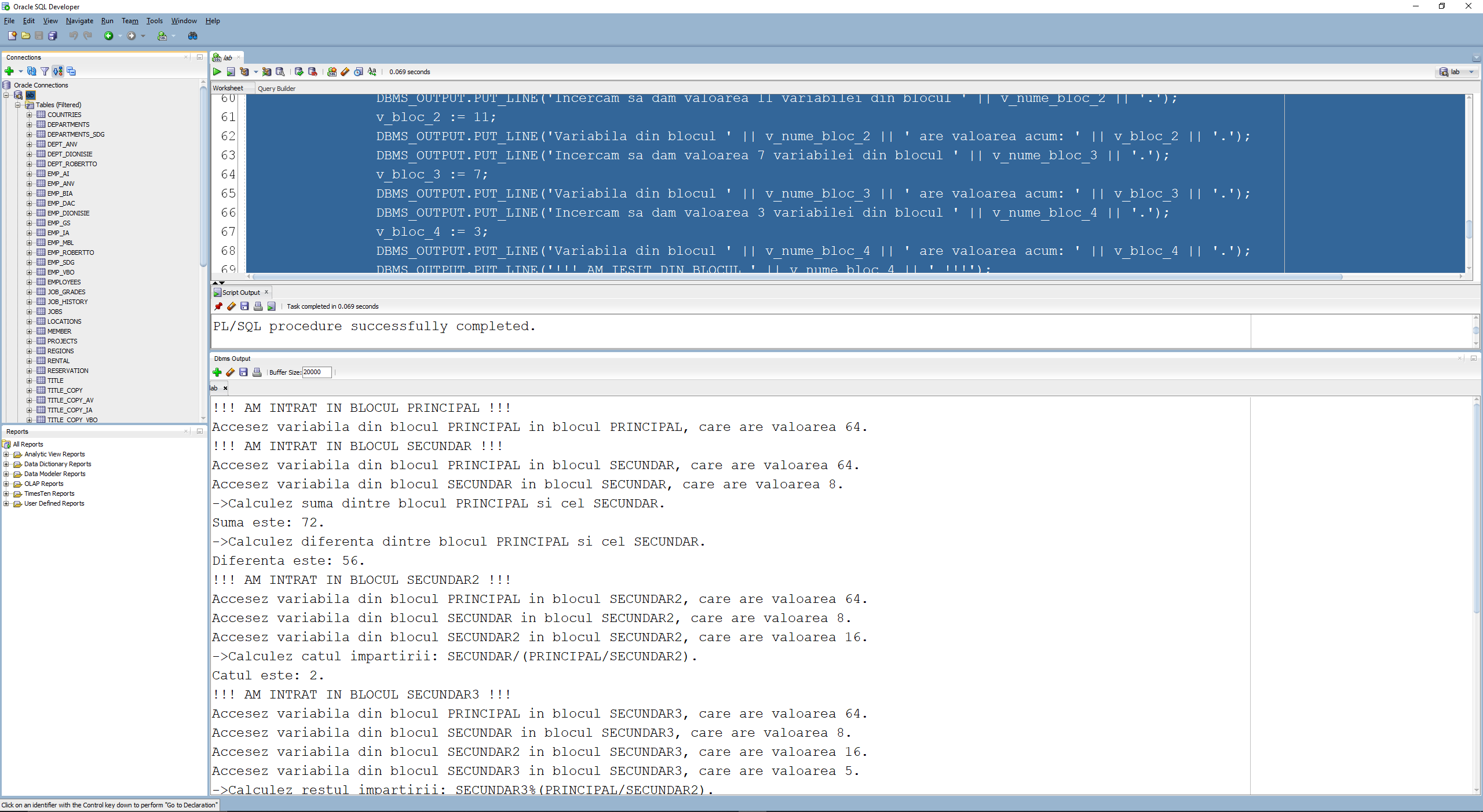
v\_vizitat := v\_vizitat || ', ' || v\_nume\_bloc\_1;

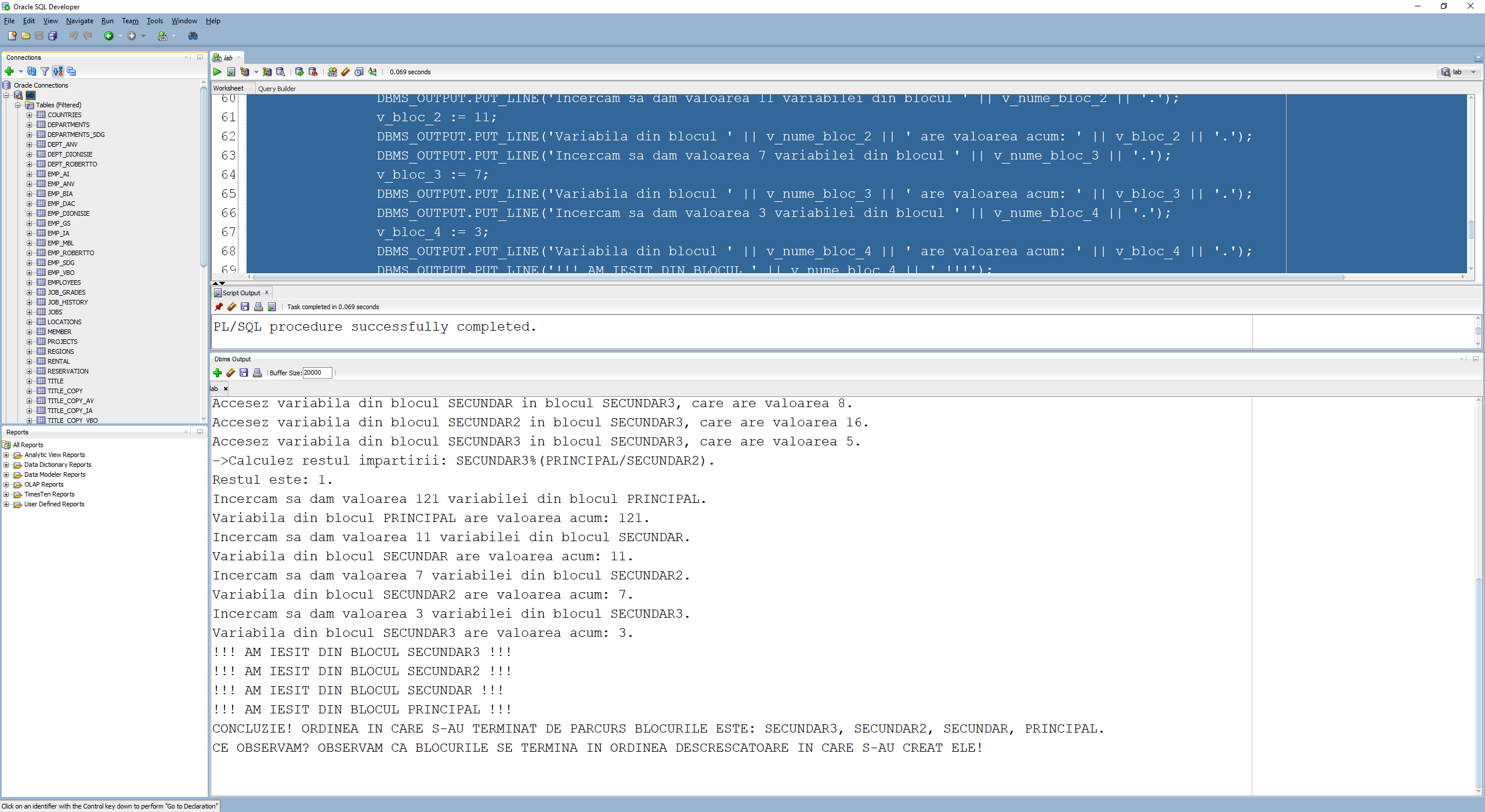
DBMS\_OUTPUT.PUT\_LINE('CONCLUZIE! ORDINEA IN CARE S-AU TERMINAT DE PARCURS BLOCURILE ESTE: ' || v\_vizitat || '.');

DBMS\_OUTPUT.PUT\_LINE('CE OBSERVAM? OBSERVAM CA BLOCURILE SE TERMINA IN ORDINEA DESCRESCATOARE IN CARE S-AU CREAT ELE!');

END;

***Print-Screen:***





* ***Ce observăm?***

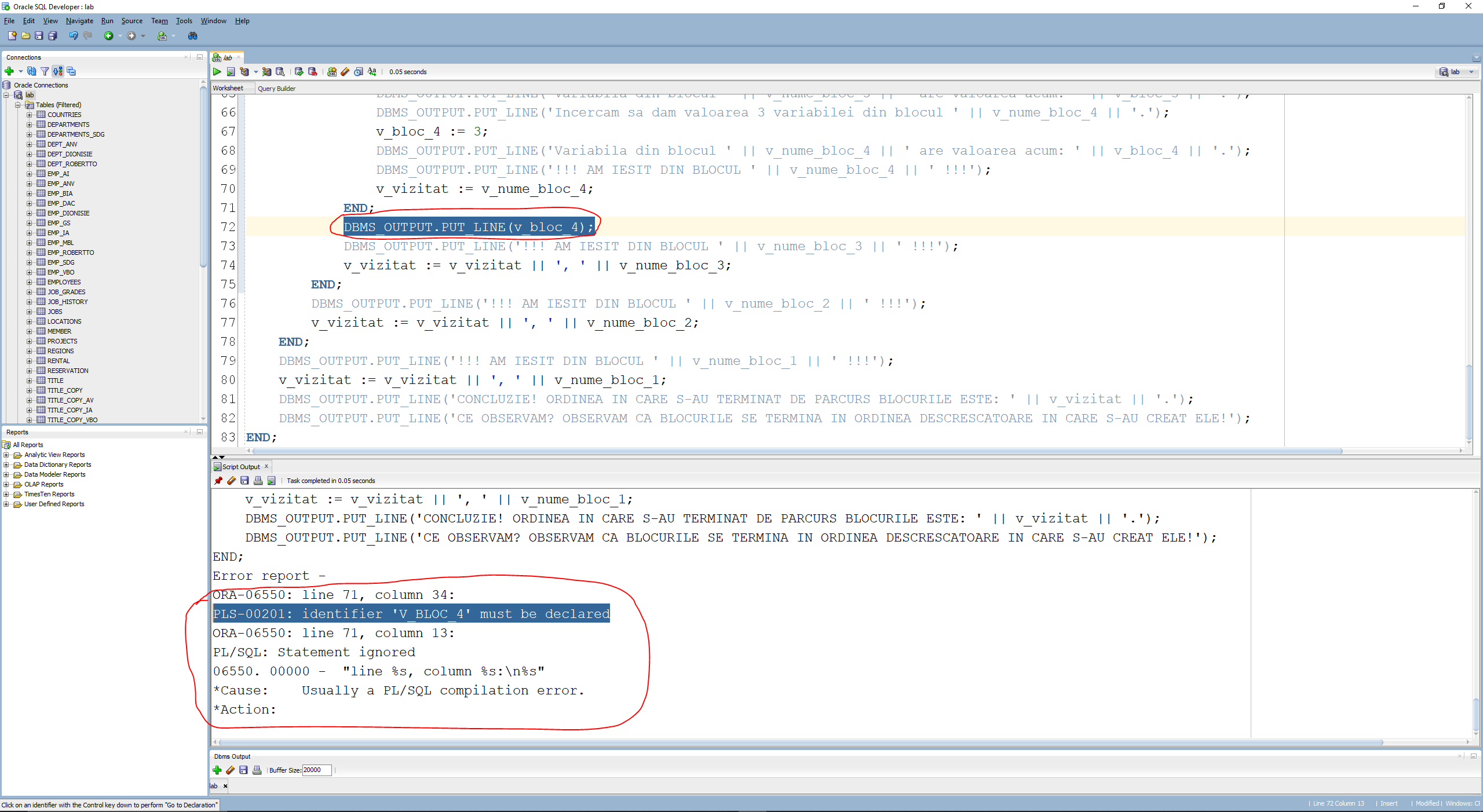
Observăm că variabilele din blocurile părinte sunt accesibile în blocurile de tip fii. Ce înseamnă asta?

Totul funcționează pe un principiu ierarhic. În exemplul nostru blocul 1 este tatăl blocului 2, care la rândul lui tatăl blocului 3, care la rândul lui este tatăl blocului 4.

Deci în interiorul blocului 4 putem accesa toate datele din blocurile mai mari ca el (***bloc3, bloc2, bloc1***), fiind fiul cel mai mic, în interiorul blocului 3 putem accesa toate datele din blocurile mai mari ca el (***bloc2, bloc1***), fiind tatăl blocului 4 și fiul blocului 2. !!! Nu putem accesa datele din blocul fiu (***bloc4***)!!!. În interiorul blocului 2 putem accesa doar datele din ***blocul1***, fiind tatăl său, iar datele din descendenții săi sunt inaccesibile (***bloc3, bloc4***). În interiorul blocului 1 nu mai putem accesa alte date în afară de cele din blocul1.

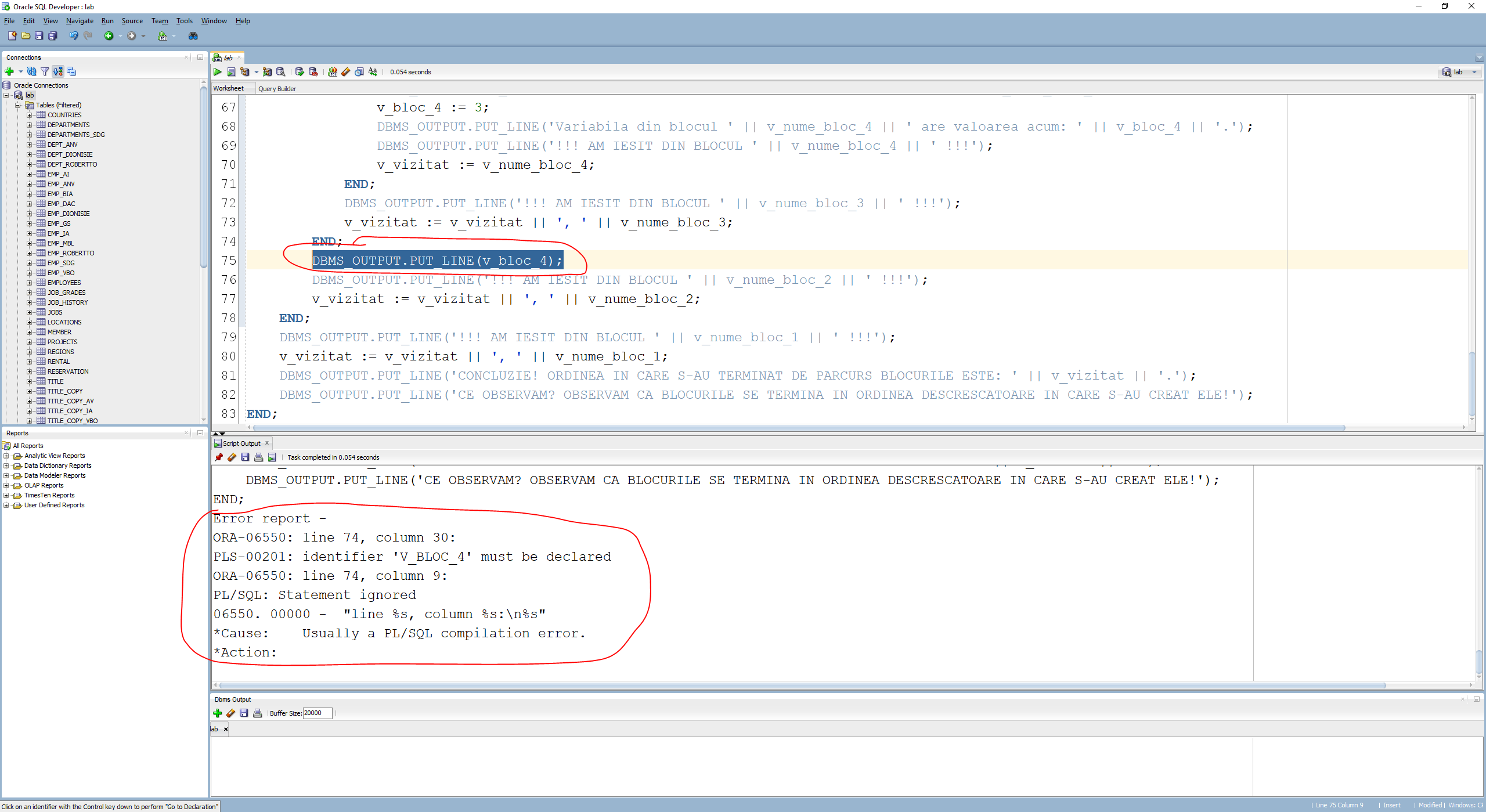
De ce se întâmplă acest lucru? Pentru că în fiecare bloc avem variabile declarate local! Cum am fi putut accesa orice variabile de oriunde? Declaram totul global.

Testăm accesarea variabilei din blocul 4 în blocul 3:



Această eroare ne spune că variabila “***v\_bloc\_4***“ nu a fost declarată, întrucât ea a fost declarată local în interiorul blocului 4. Odată ce am ieșit din blocul 4 (din scope), ea nu mai este accesibilă întrucât ea a fost declarată local. (**EROARE DE COMPILARE!**)

Testăm accesarea variabilei din blocul 4 în blocul 2:



Observăm că se întâmplă același lucru. Variabila nu este accesibilă întrucât nu a fost declarată (a ieșit din scope-ul ei, respectiv **baza4**), ceea ce duce la o eroare de compilare.

***Concluzie:***

* Blocurile se termină în ordinea descrescătoare în care s-au creat (se creează bloc1, bloc2, bloc3 se va termina bloc3, după bloc2, după bloc1)
* Variabilele declarate într-un bloc sunt declarate local și sunt accesibile doar în blocurile sale descendente.
* Dacă încercăm să modificăm/accesăm o variabilă dintr-un bloc descendent, iar noi ne aflăm în afara acestui bloc, vom primi eroare de compilare când vom rula codul în SQL Developer (**PLS-00201: identifier 'nume\_variabila' must be declared**).

***Verficarea comenzii delete cu returning.***

DECLARE

v\_emp\_first\_name VARCHAR2(30);

v\_emp\_last\_name VARCHAR2(30);

v\_emp\_manager NUMBER(10) := 1337;

v\_emp\_salary NUMBER(30) := 1337;

BEGIN

DELETE FROM emp\_Robertto WHERE manager\_id = 101

RETURNING first\_name, last\_name

INTO v\_emp\_first\_name, v\_emp\_last\_name;

DBMS\_OUTPUT.PUT\_LINE ('<<DELETE RETURNING INTO...>>');

DBMS\_OUTPUT.PUT\_LINE ('First Name: '|| v\_emp\_first\_name);

DBMS\_OUTPUT.PUT\_LINE ('Last Name: ' || v\_emp\_last\_name);

UPDATE emp\_Robertto

SET salary = v\_emp\_salary

WHERE employee\_id = 0;

UPDATE emp\_Robertto

SET salary = v\_emp\_salary

WHERE employee\_id = 0

RETURNING first\_name, last\_name, salary

INTO v\_emp\_first\_name, v\_emp\_last\_name, v\_emp\_salary;

DBMS\_OUTPUT.PUT\_LINE ('<<UPDATE RETURNING INTO...>>');

DBMS\_OUTPUT.PUT\_LINE ('First Name: '|| v\_emp\_first\_name);

DBMS\_OUTPUT.PUT\_LINE ('Last Name: ' || v\_emp\_last\_name);

DBMS\_OUTPUT.PUT\_LINE ('Salary: '|| v\_emp\_salary);

COMMIT;

EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('SE RETURNEAZA MAI MULT DE 1 LINIE!');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NU A FOST GASIT UN ASEMENEA ANGAJAT!');

WHEN OTHERS THEN

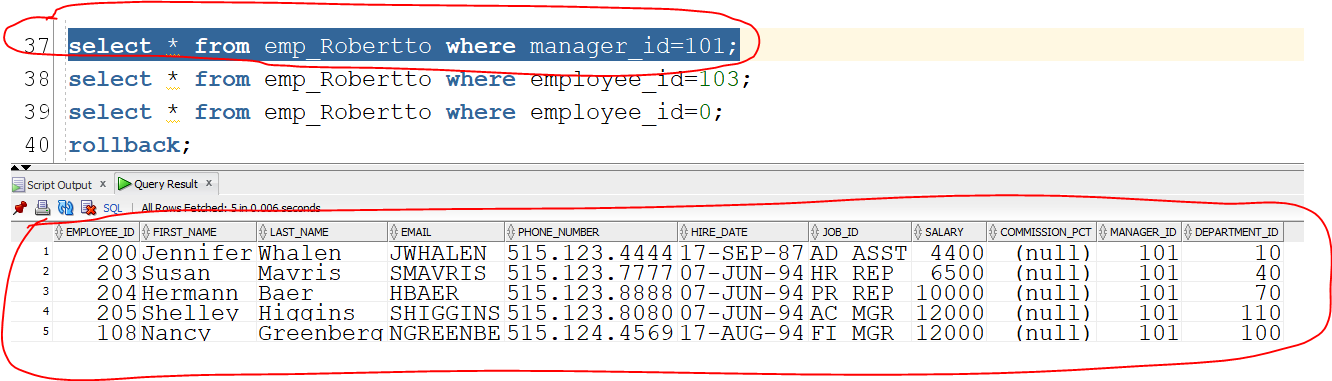
DBMS\_OUTPUT.PUT\_LINE('ALTA EROARE!');

END;

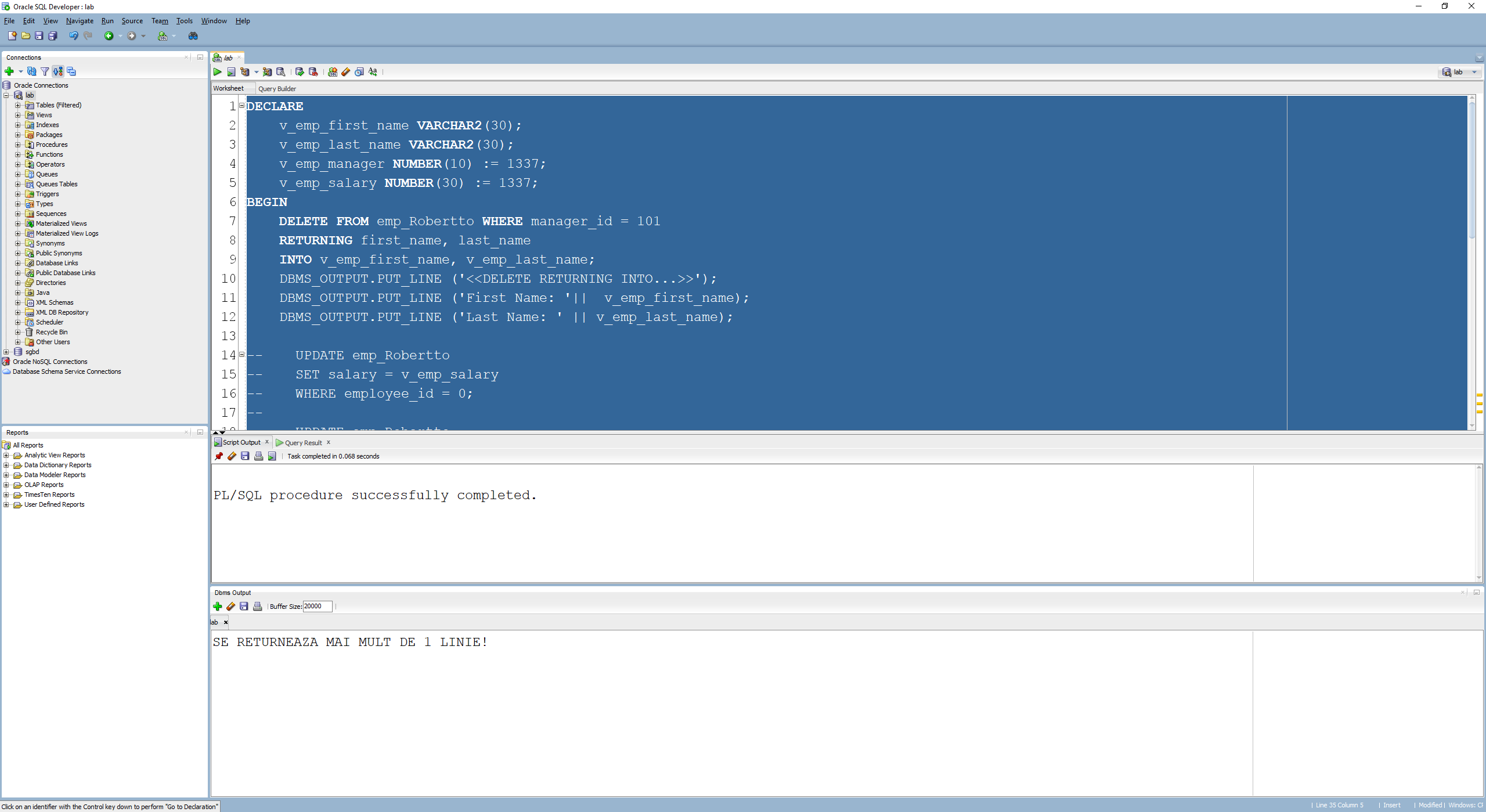
***Print-Screen:***

Vom testa stergerea mai multor linii cu returning into variabile.

În poza de mai jos putem observa ca avem 5 înregistrări pentru care manager\_id = 101;



Încercam să stergem toate înregistrările pentru care manager\_id = 101; și să le reținem datele în variabile (v\_emp\_first\_name și v\_emp\_last\_name).



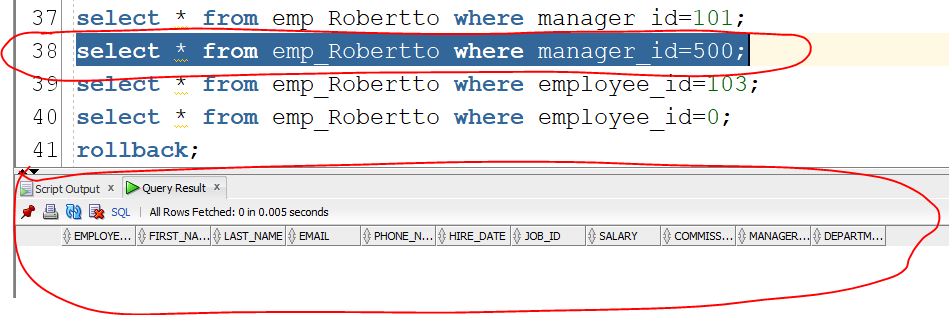
* ***Ce observăm?***

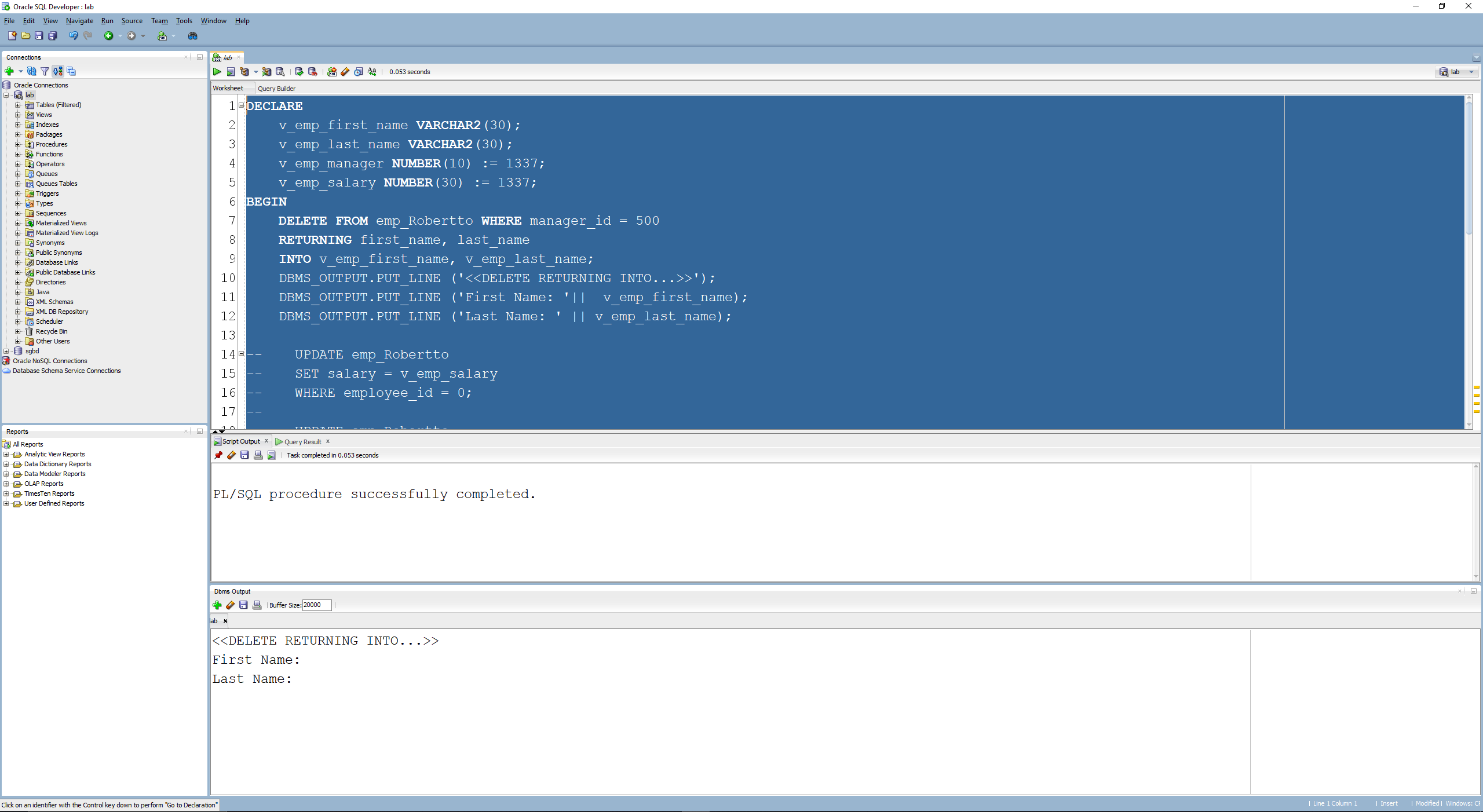
Observăm că intră pe excepția **TOO\_MANY\_ROWS**. De ce? Nu poate stoca mai multe valori într-o variabilă. Noi returnăm 5 inregistrări, și nu le putem stoca pe toate într-o singură variabilă.

Dacă nu am fi vrut să reținem înregistrările pe care vrem să le stergem în variabile totul ar fi mers cu succes:

*DELETE FROM emp\_Robertto WHERE manager\_id = 101;*

Acum încercăm pentru cazul în care sunt 0 înregistrări de șters de salvat în variabile. Știm că nu există niciun angajat cu manager\_id = 500; Deci vom testa pentru acest caz pentru a vedea dacă programul va intra pe excepția **NO\_DATA\_FOUND**.





* ***Ce observăm?***

Observăm că nu intră pe excepția **NO\_DATA\_FOUND**. De ce? Pentru că nu există nicio înregistrare pentru care manager\_id = 500; deci noi nu salvăm nicio înregistrare în variabilele noastre. Practic ele rămân cu valoare default cu care au fost declarate.

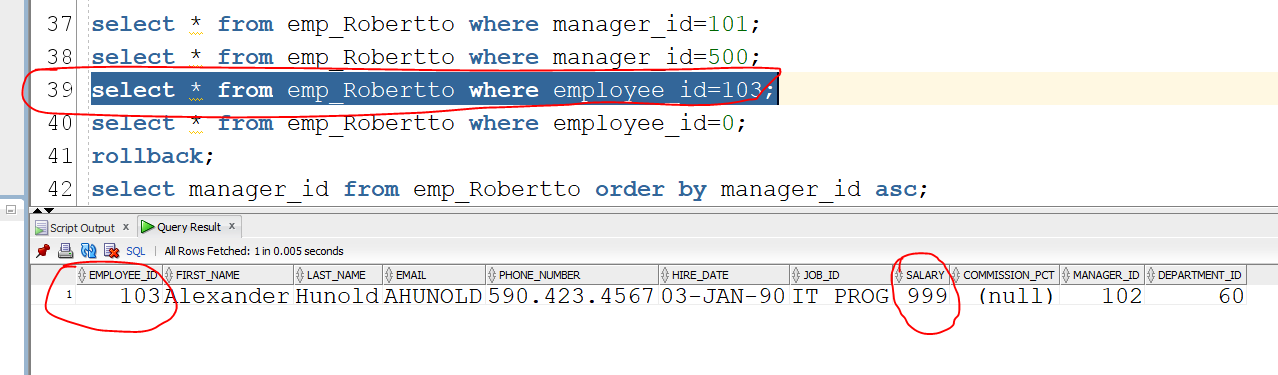
Dacă am fi declarat în zona *DECLARE* variabilele: *v\_emp\_first\_name := ‘Gigel’* și *v\_emp\_last\_name := ‘Cornel’*. Pe ecran s-ar fi afișat:

*<<DELETE RETURNING INTO…>>*

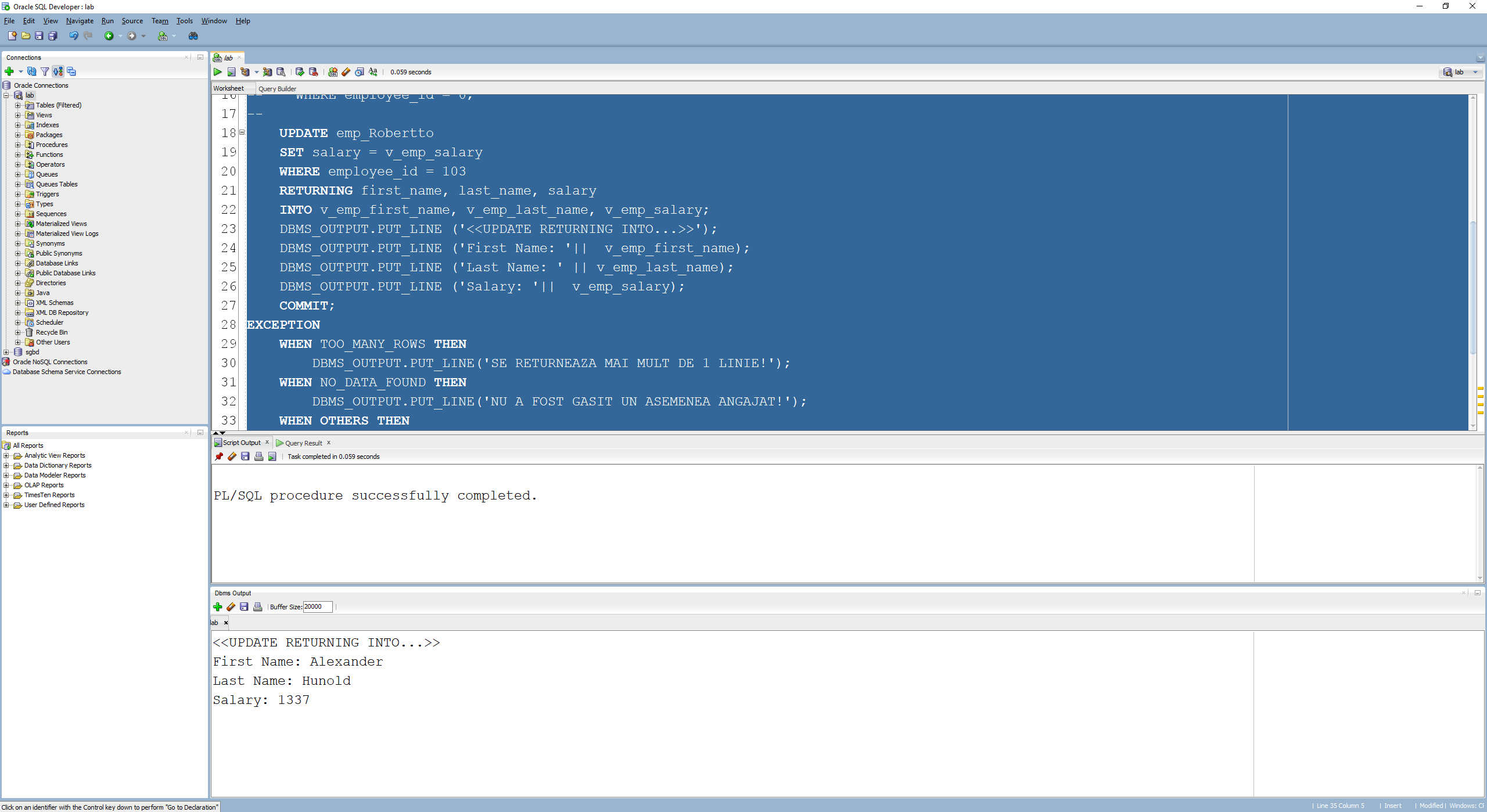
*First Name: Gigel*

*Last Name: Cornel*

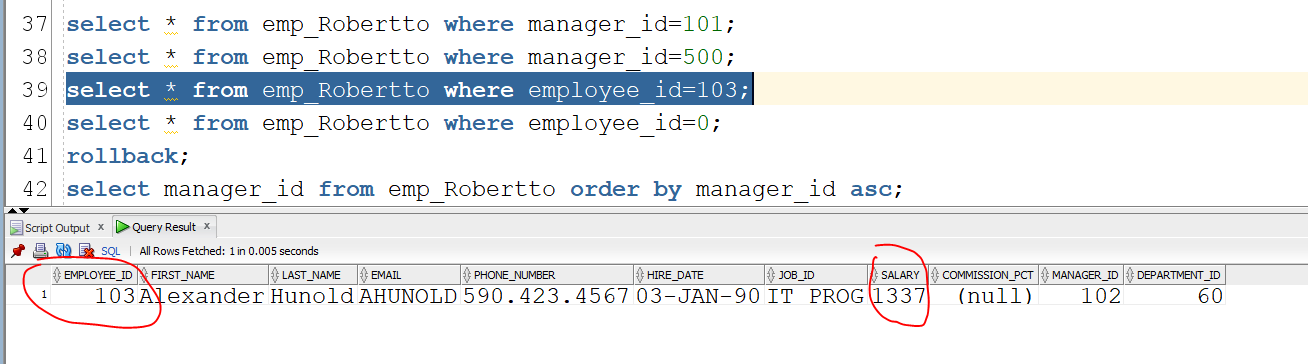
Acum testăm pentru comanda **UPDATE**. Încercăm să updatăm linia pentru care empoyee\_id = 103;



Avem o singură înregistrare, deci totul ar trebui să funcționeze cu succes.

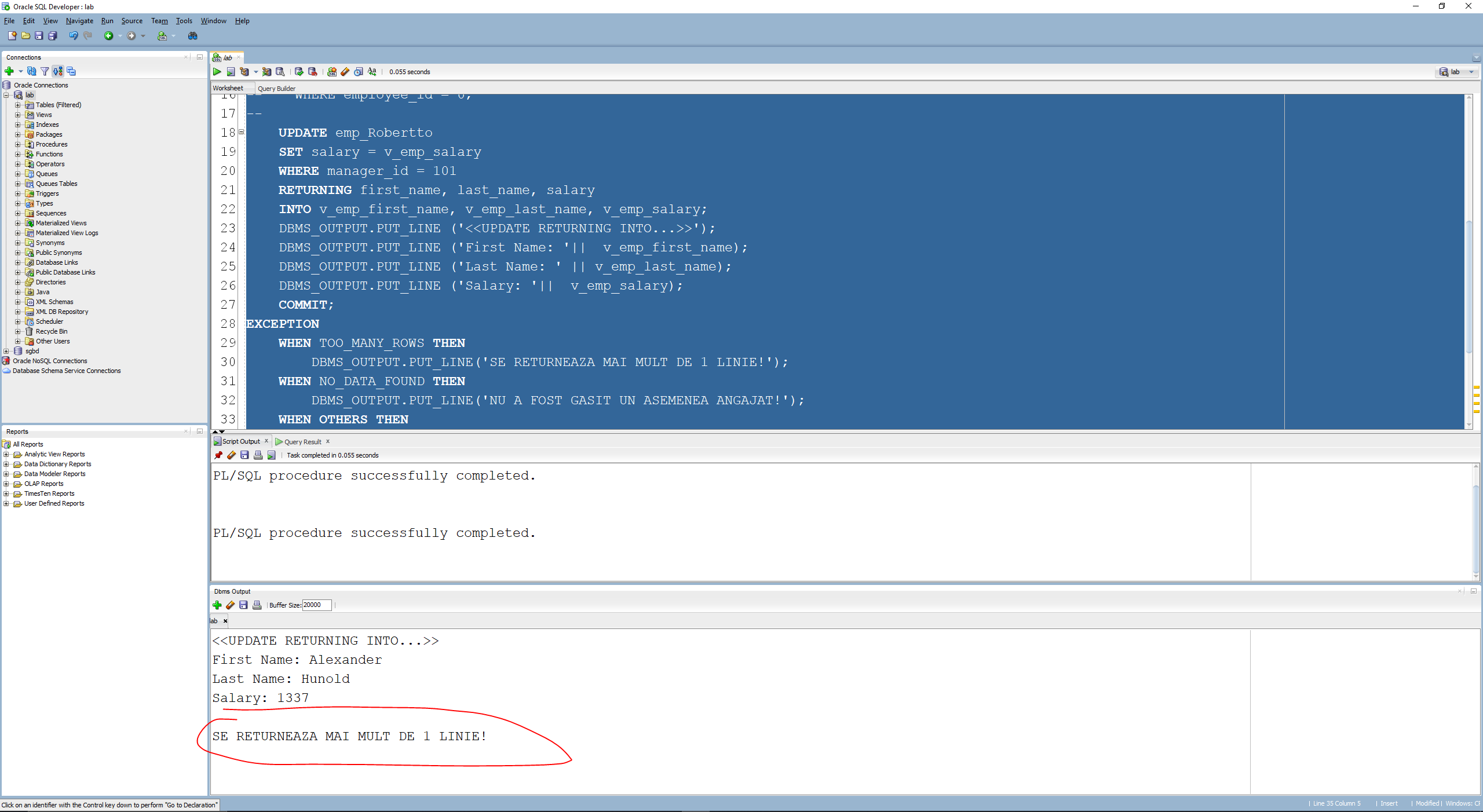


A funcționat, acum verificăm dacă s-a actualizat și în tabel.



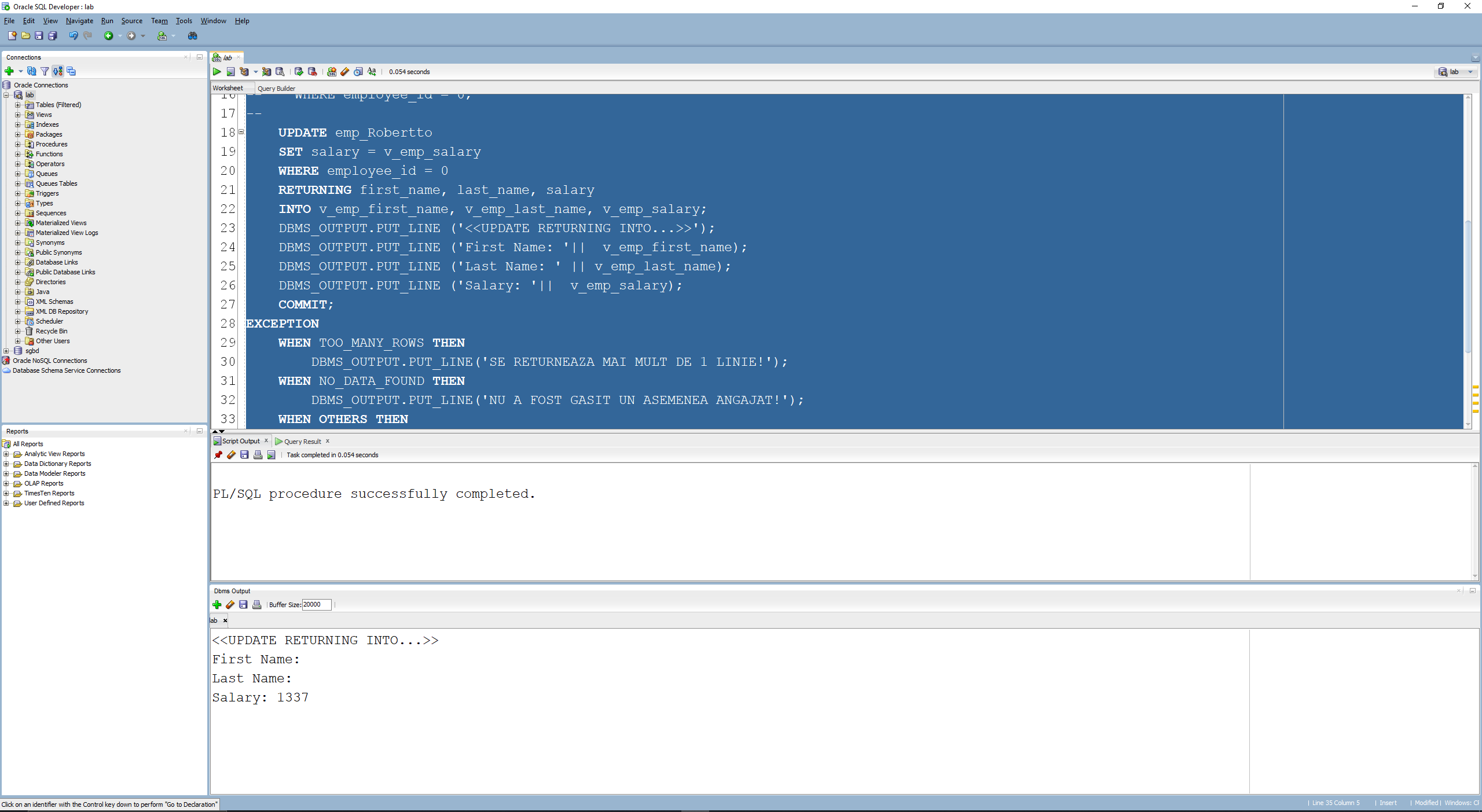
Totul a funcționat cu succes.

Acum verificăm din nou pentru cazul când avem mai multe linii (pentru manager\_id = 101).



Intră de data aceasta pe excepția **TOO\_MANY\_ROWS**.

Încercăm cu 0 înregistrări (pentru employee\_id = 0).



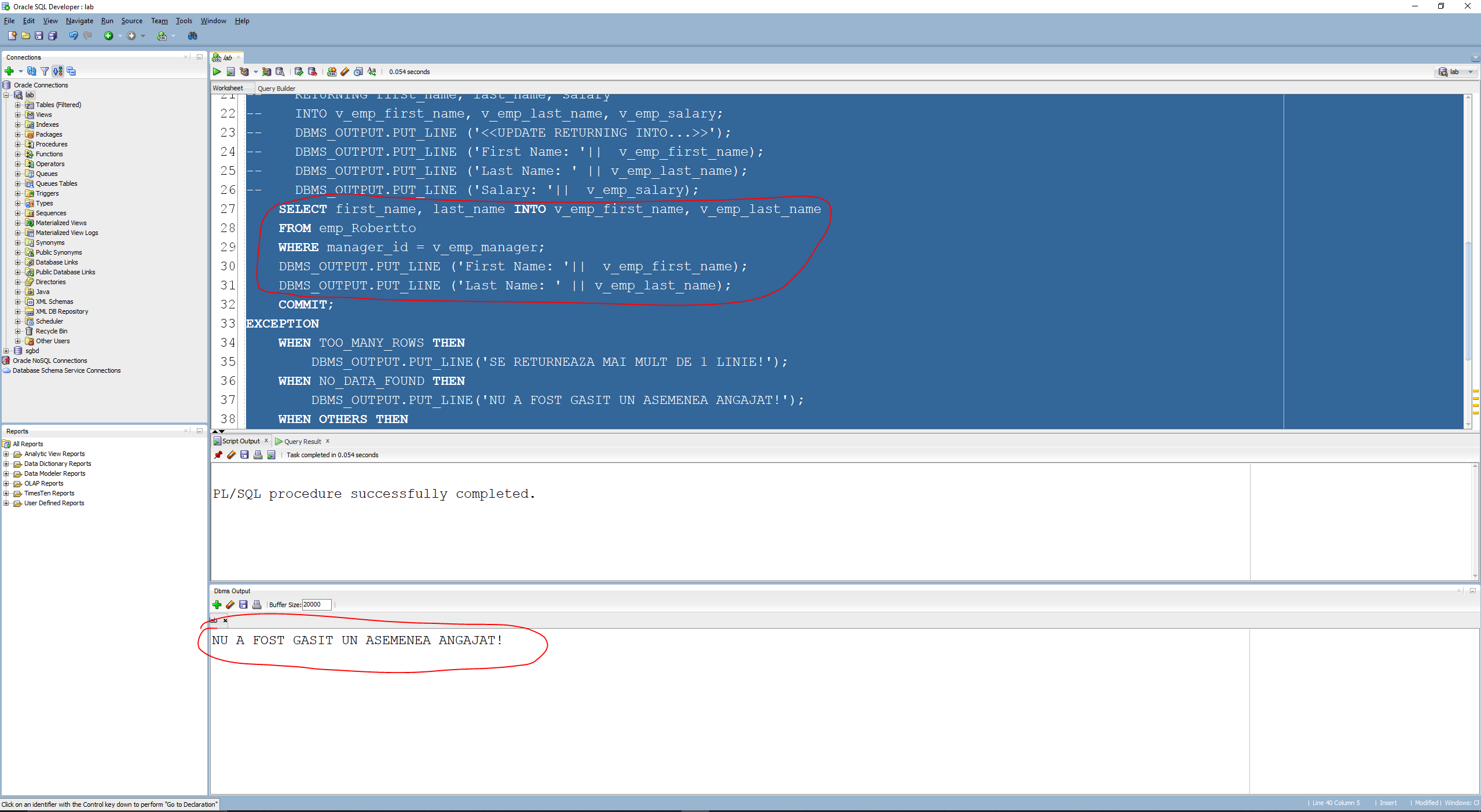
***OBSERVAȚIE! SE COMPORTĂ LA FEL CA DELETE RETURNING INTO...***

***Concluzie:***

* Dacă avem **mai mult de o înregistrare** care satisface un criteriu, atunci când vrem să folosim **RETURNING INTO** pentru a le salva datele în variabile, **VOM PRIMI EROAREA** ***TOO\_MANY\_ROWS***, care poate fi tratată cu o excepție.
* Dacă avem **o singură înregistrare** totul se va desfășura cu succes.
* Dacă nu avem nicio înregistrare care satisface un criteriu, atunci când vrem să folosim **RETURNING INTO** pentru a salva datele în variabile, **NU VOM PRIMI EROAREA *NO\_DATA\_FOUND !***, pentru că practic *“noi salvăm nimic”* în acele variabile, ele rămânând cu valorile default, din zona *DECLARE*.

*Când intrăm pe excepția* ***NO\_DATA\_FOUND*** *?*

Mai jos avem un exemplu când intrăm pe excepția **NO\_DATA\_FOUND**.



***Cauza:*** Nu avem niciun angajat pentru care manager\_id = v\_emp\_manager = 1337, deci nu se poate îndeplini filtrarea după manager\_id. Rezultând astfel eroarea ***NO\_DATA\_FOUND***, care se poate trata cu o excepție (în cazul nostru am afișat pe ecran ‘NU A FOST GASIT UN ASEMENEA ANGAJAT!’).

Popescu Paullo Robertto Karloss

Grupa 231

Temă SGBD #2