TEMĂ

În această temă voi prezenta compararea timpilor de execuție pentru toate cele 3 tipuri de colectiții (vector, tabel indexat, respectiv tabel imbricat) precizând după, de ce unul este mai rapid decât celălalalt. De asemenea voi prezenta rezolvarea exercițiului 3 din laboratorul 2 PL/SQL folosind doar operatori multiset în SQL.

***Vector vs Tabel Indexat vs Tabel imbricat (Timpi de execuție).***

Să se inițializeze primele 3.000.000 elemente din fiecare colecție folosind formula **((i\*2)+2)\*10**.

***Rezolvare:***

DECLARE

TYPE tab\_vec IS VARRAY(3000000) OF NUMBER;

t tab\_vec := tab\_vec();

TYPE tab\_ind IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

t2 tab\_ind;

TYPE tab\_imb IS TABLE OF NUMBER;

t3 tab\_imb := tab\_imb();

timp\_executie number(10);

start\_time number(10);

stop\_time number(10);

BEGIN

start\_time := dbms\_utility.get\_time();

for i in 1..3000000 loop

t.extend;

t(i) := ((i\*2)+2)\*10;

end loop;

stop\_time := dbms\_utility.get\_time();

timp\_executie := stop\_time - start\_time;

DBMS\_OUTPUT.PUT\_LINE('Initializarea in vector a durat ' || timp\_executie);

start\_time := dbms\_utility.get\_time();

for i in 1..3000000 loop

t2(i) := ((i\*2)+2)\*10;

end loop;

stop\_time := dbms\_utility.get\_time();

timp\_executie := stop\_time - start\_time;

DBMS\_OUTPUT.PUT\_LINE('Initializarea in tabel indexat a durat ' || timp\_executie);

start\_time := dbms\_utility.get\_time();

for i in 1..3000000 loop

t3.extend;

t3(i) := ((i\*2)+2)\*10;

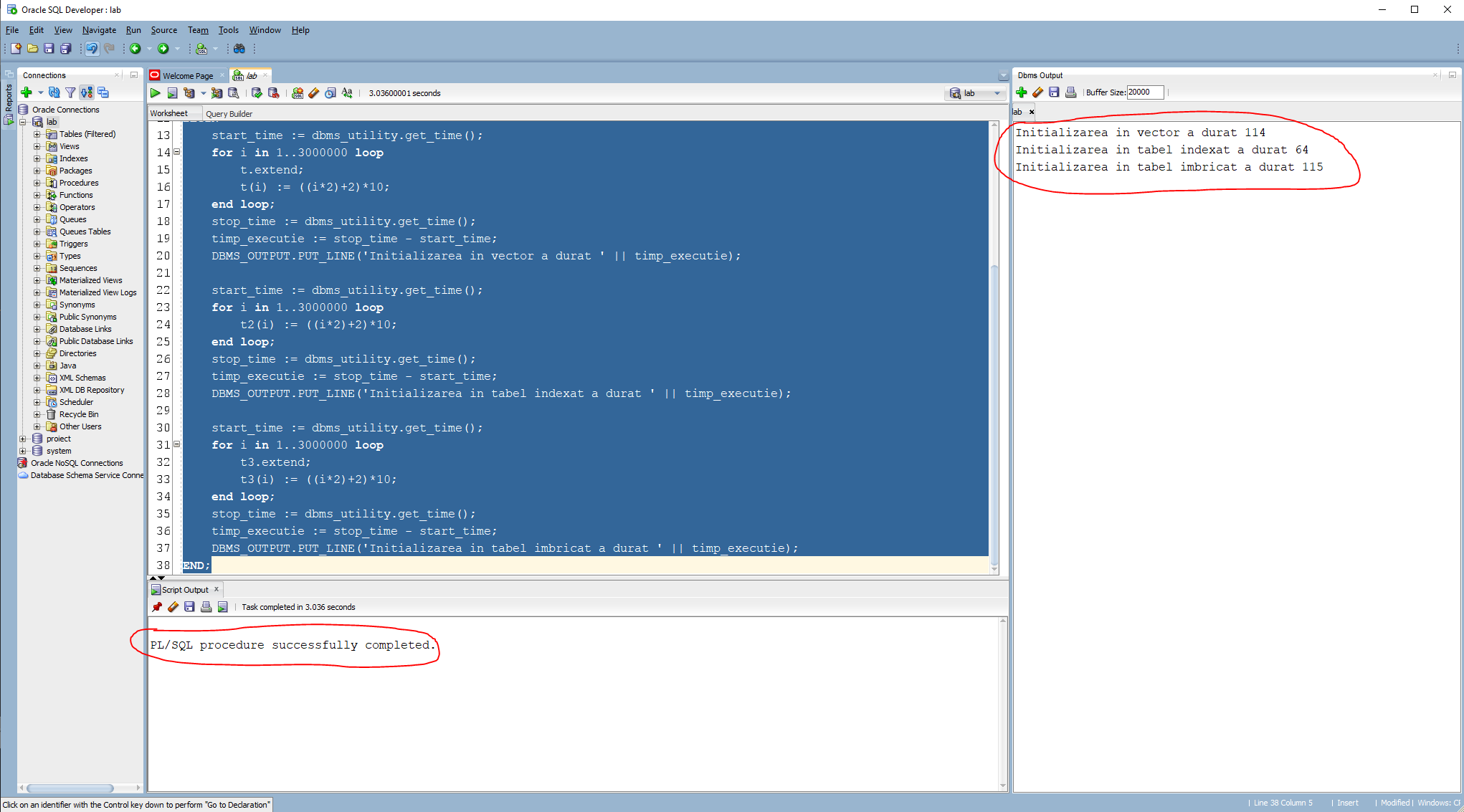
end loop;

stop\_time := dbms\_utility.get\_time();

timp\_executie := stop\_time - start\_time;

DBMS\_OUTPUT.PUT\_LINE('Initializarea in tabel imbricat a durat ' || timp\_executie);

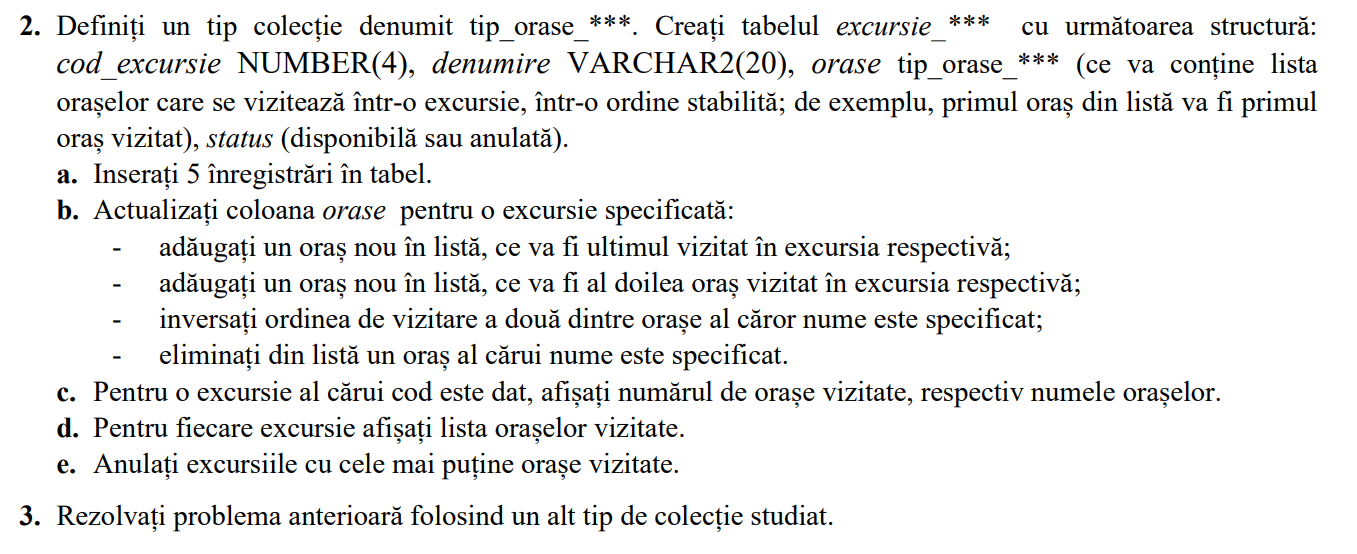
END;

***Print-Screen:*** 

Observăm că inițializarea elementelor folosind formula din cerință a durat 114 unități de timp pentru vector, 64 pentru tabelul indexat și 115 pentru tabelul imbricat.

Inițializarea elementelor în tabelul indexat este mult mai rapidă decât în tabelul imbricat sau în vector deoarece, spre deosebire de acestea nu execută instrucțiunea ***extend*** (o instrucțiune în plus care costă și ea timp pentru a fi executată). Tabelul indexat își alocă dinamic memorie, spre deosebire de vector/tabelul imbricat (necesită comanda ***extend***). De asemenea foloșeste **PLS\_INTEGER** ca tip de date care este mult mai optim, dar și recomandat în practică decât number.

***Rezolvarea exercițiul 3 din laboratorul 2 PL/SQL folosind operatori MULTISET la nivel de SQL.***

***Rezolvare:***

-- Ex3

CREATE OR REPLACE TYPE tip\_orase\_Robertto IS TABLE OF VARCHAR2(100);

/

CREATE TABLE excursie\_Robertto (cod\_excursie NUMBER(4) PRIMARY KEY,

denumire VARCHAR2(20),

orase tip\_orase\_Robertto,

status VARCHAR2(20)

) NESTED TABLE orase STORE AS tabb\_imb\_orase;

select \* from excursie\_Robertto;

--a) Inserați 5 înregistrări în tabel.

INSERT INTO excursie\_Robertto VALUES (

1,

'La arabi',

tip\_orase\_Robertto('Dubai', 'Abu Dhabi', 'Schardscha'),

'OPEN'

);

INSERT INTO excursie\_Robertto VALUES (

2,

'Pe o insula Pustie',

tip\_orase\_robertto('Hawai', 'Ceva frumos', 'Takimi'),

'CLOSED'

);

INSERT INTO excursie\_Robertto VALUES (

3,

'La munte',

tip\_orase\_Robertto('Brasov', 'Tusnad', 'Sinaia'),

'OPEN'

);

INSERT INTO excursie\_Robertto VALUES (

4,

'In Suedia',

tip\_orase\_Robertto('Umea', 'Stockholm', 'Ystad'),

'CLOSED'

);

INSERT INTO excursie\_Robertto VALUES (

5,

'In Italia',

tip\_orase\_Robertto('Milano', 'Roma', 'Bergamo'),

'OPEN'

);

select \* from excursie\_Robertto;

--b) Actualizați coloana orase pentru o excursie specificată:

-- b1) adăugați un oraș nou în listă, ce va fi ultimul vizitat în excursia respectivă;

INSERT INTO TABLE (SELECT orase

FROM excursie\_Robertto

WHERE cod\_excursie = 1)

VALUES('Habibi');

select \* from excursie\_Robertto;

-- b2) adăugați un oraș nou în listă, ce va fi al doilea oraș vizitat în excursia respectivă;

CREATE TABLE aux\_imb (ID NUMBER(10), nume\_oras VARCHAR2(100));

INSERT INTO aux\_imb VALUES (1,(SELECT \* FROM TABLE (SELECT orase

FROM excursie\_Robertto

WHERE cod\_excursie = 2)

WHERE ROWNUM = 1));

INSERT INTO aux\_imb VALUES (2, 'Vulcanos');

UPDATE excursie\_Robertto

SET orase = (SELECT CAST (COLLECT(nume\_oras) AS tip\_orase\_Robertto)

FROM aux\_imb) MULTISET UNION DISTINCT (SELECT orase

FROM excursie\_Robertto

WHERE cod\_excursie = 2)

WHERE cod\_excursie = 2;

/

select \* from excursie\_Robertto;

--b3) - inversați ordinea de vizitare a două dintre orașe al căror nume este specificat;

UPDATE TABLE (SELECT orase

from excursie\_Robertto

where cod\_excursie = 3) a

SET VALUE(a) = CASE

WHEN COLUMN\_VALUE = 'Brasov' THEN 'Tusnad'

WHEN COLUMN\_VALUE = 'Tusnad' THEN 'Brasov'

WHEN COLUMN\_VALUE <> 'Brasov' or COLUMN\_VALUE <> 'Tusnad' THEN COLUMN\_VALUE

END;

select \* from excursie\_Robertto;

--b4) - eliminați din listă un oraș al cărui nume este specificat.

DELETE FROM TABLE (SELECT orase

FROM excursie\_Robertto

WHERE cod\_excursie = 4) a

WHERE COLUMN\_VALUE = 'Umea';

select \* from excursie\_Robertto;

--c) Pentru o excursie al cărui cod este dat, afișați numărul de orașe vizitate, respectiv numele orașelor.

SELECT cardinality(orase) Numar\_orase\_vizitate, orase

FROM excursie\_Robertto

WHERE cod\_excursie = 4;

--d) Pentru fiecare excursie afișați lista orașelor vizitate.

SELECT denumire, orase

FROM excursie\_Robertto;

--e) Anulați excursiile cu cele mai puține orașe vizitate.

UPDATE excursie\_Robertto

SET status = 'CLOSED'

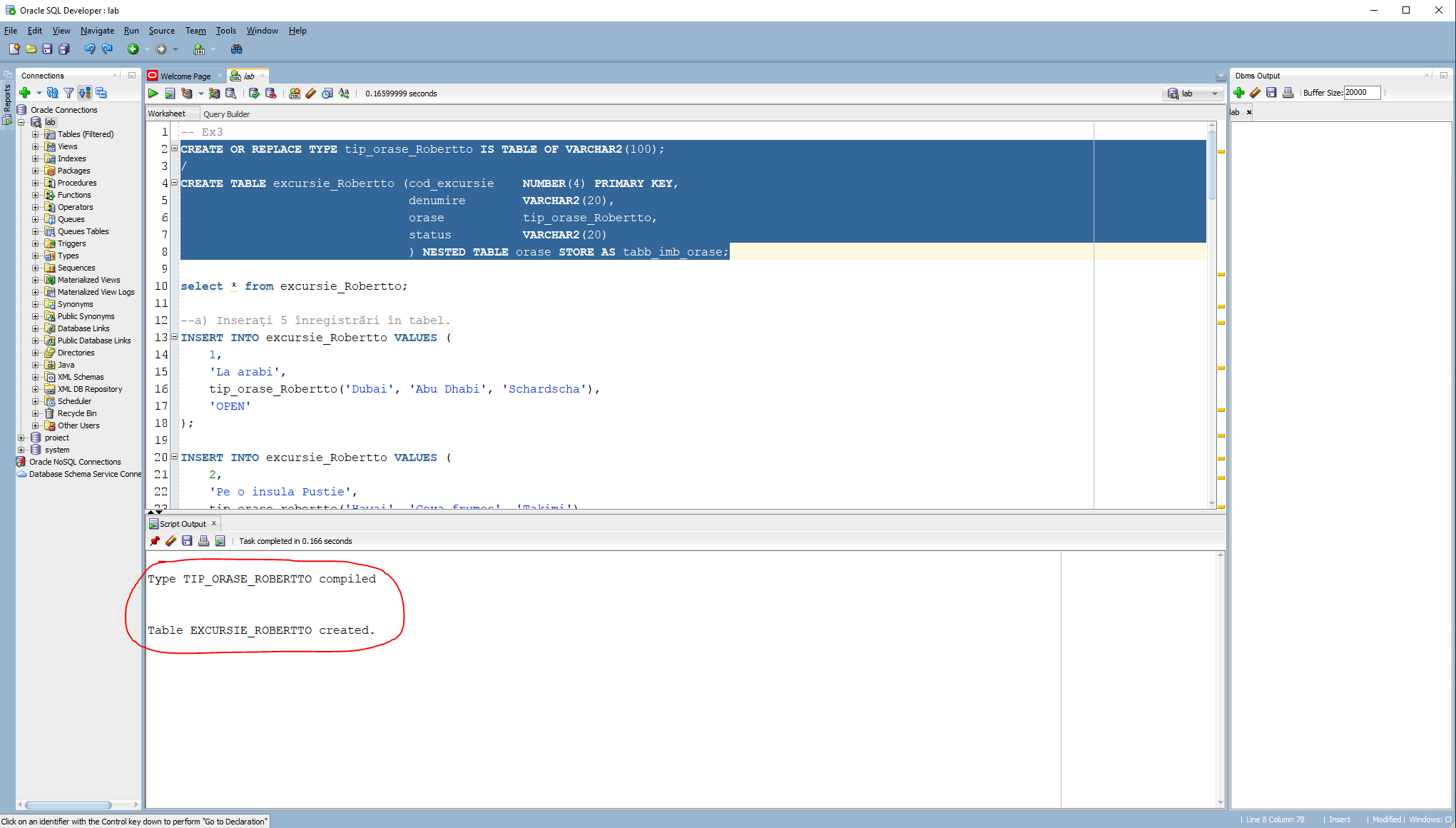
WHERE cardinality(orase) = (SELECT MIN(Numar)

FROM (SELECT cardinality(orase) Numar, orase

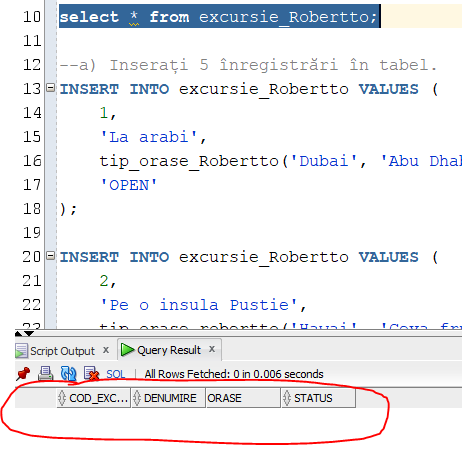
FROM excursie\_Robertto));

select \* from excursie\_Robertto;

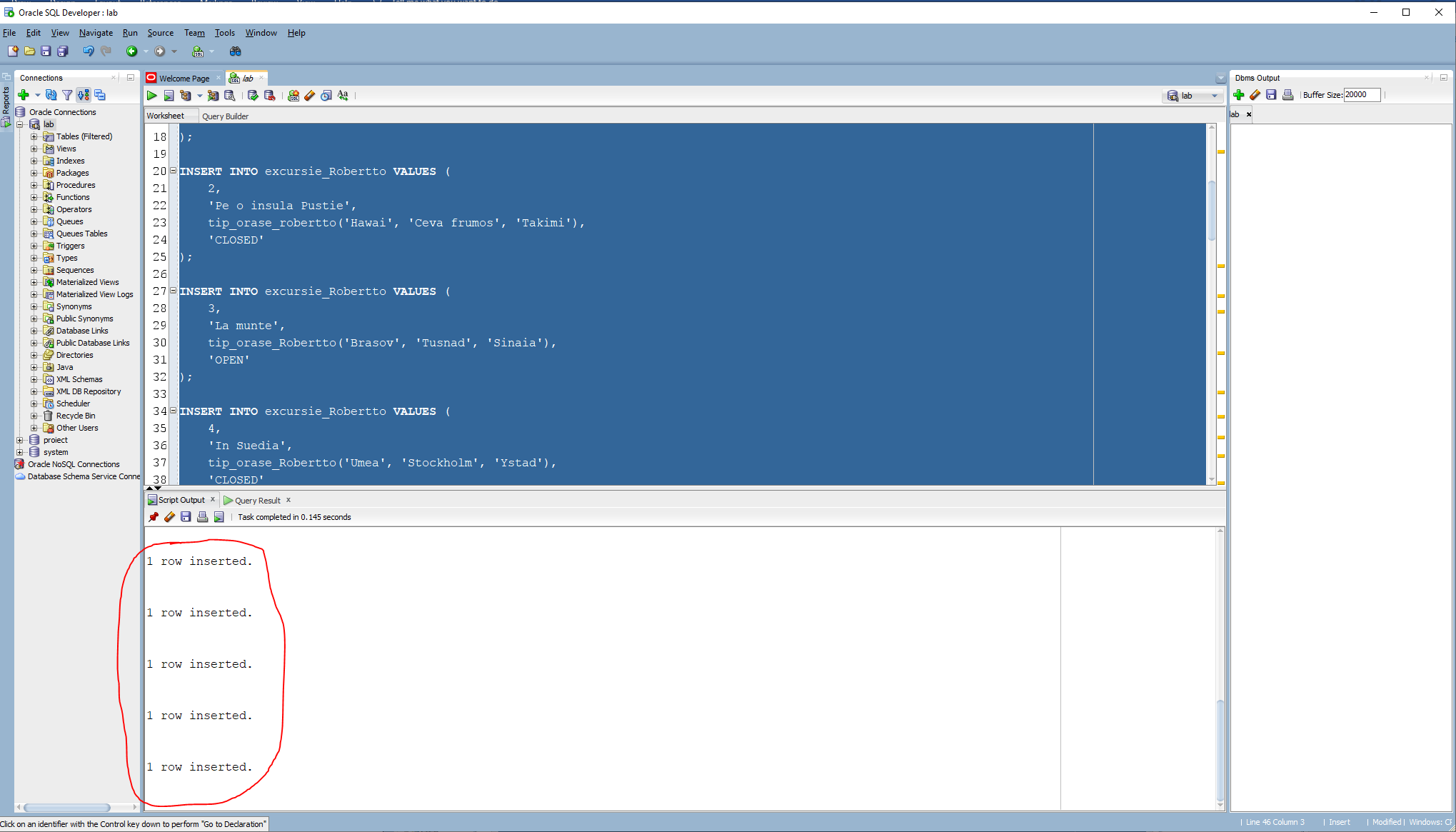
***Print-Screen:***



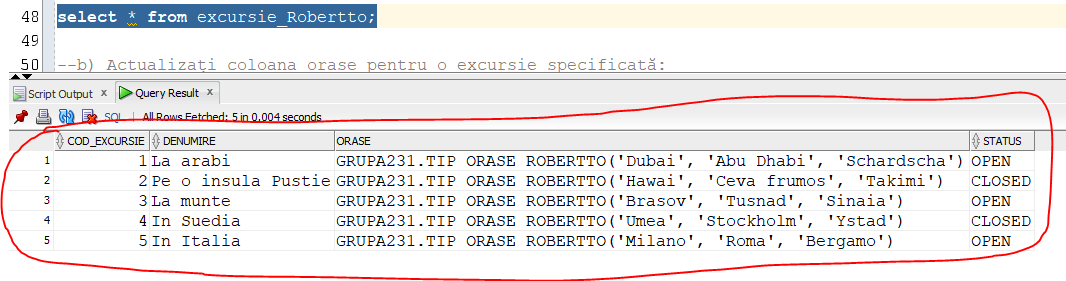
Verifcăm dacă tabelul nostru excursie\_Robertto s-a creeat cu succes.



a) Inserați 5 înregistrări în tabel.

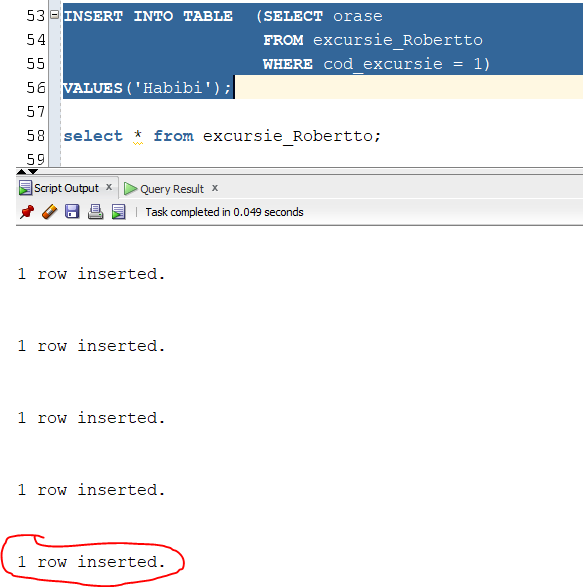


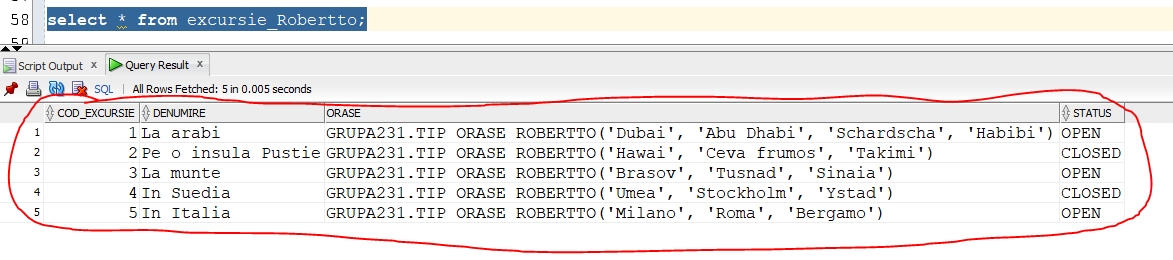
Verifcăm dacă datele noastre au fost inserate cu succes in tabelul nostru excursie\_Robertto.



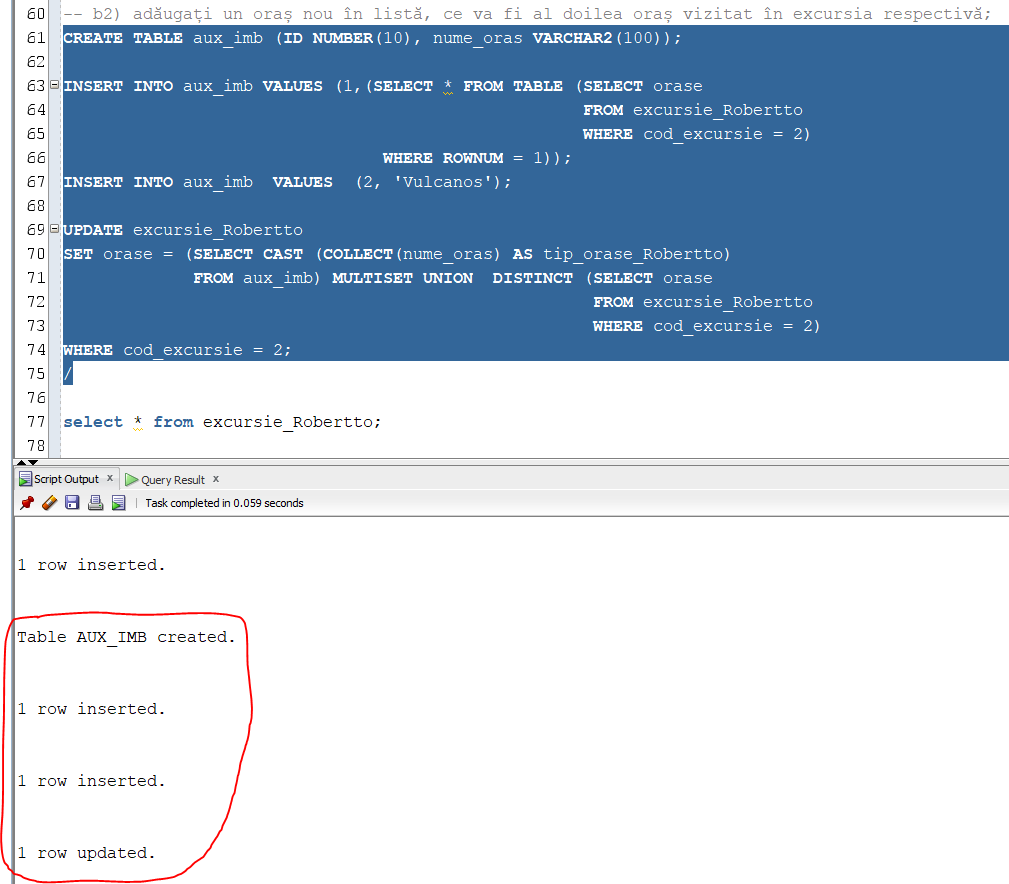
b) Actualizați coloana orase pentru o excursie specificată:

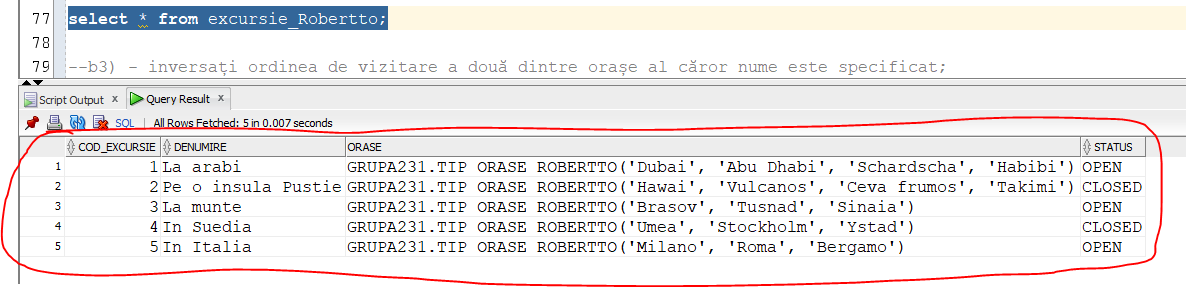
b1) adăugați un oraș nou în listă, ce va fi ultimul vizitat în excursia respectivă;



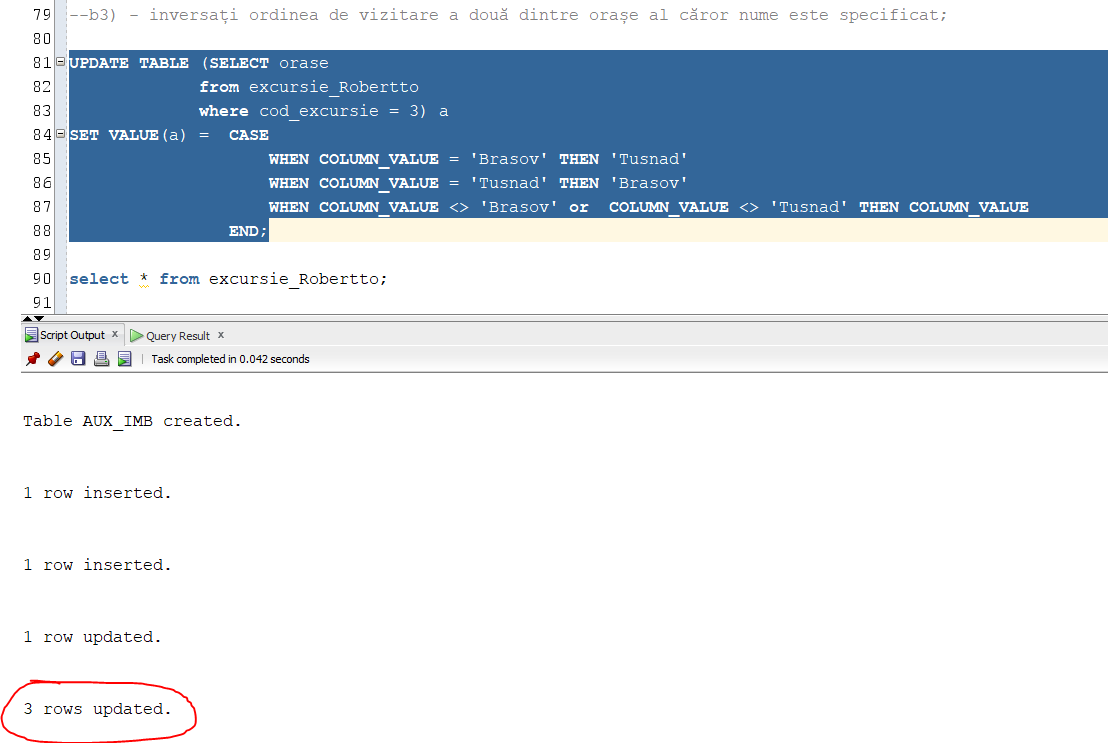


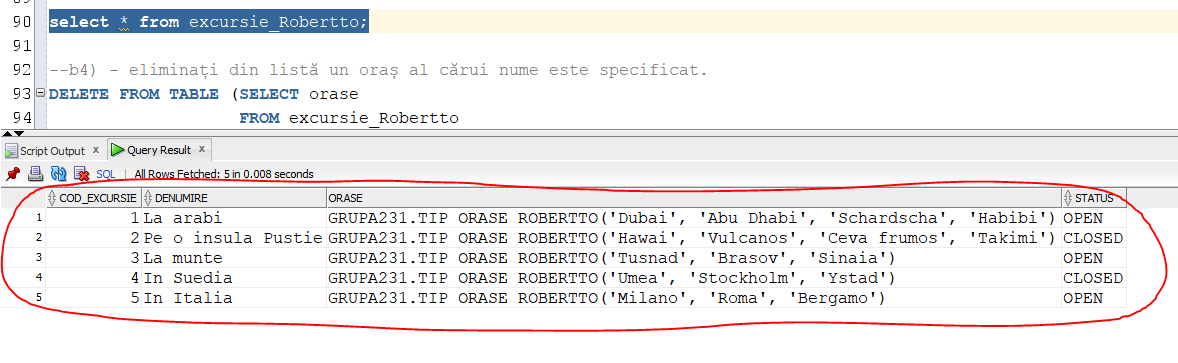
b2) adăugați un oraș nou în listă, ce va fi al doilea oraș vizitat în excursia respectivă;



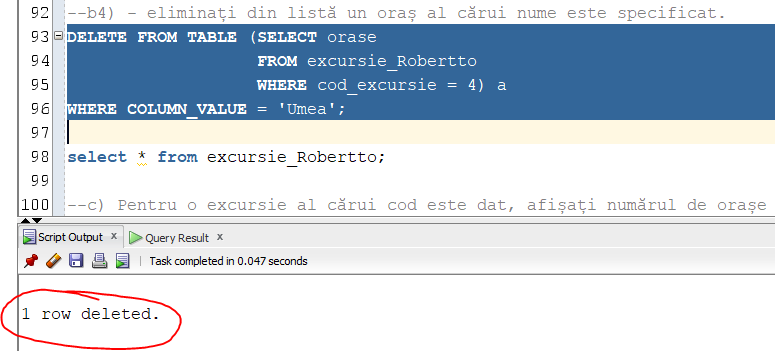


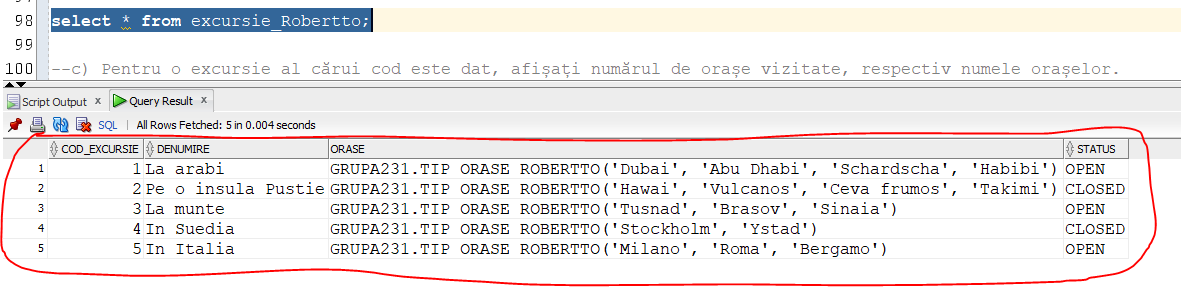
b3) inversați ordinea de vizitare a două dintre orașe al căror nume este specificat;



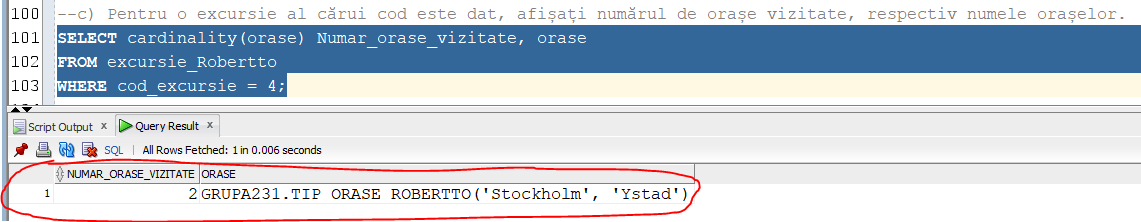


b4) eliminați din listă un oraș al cărui nume este specificat.

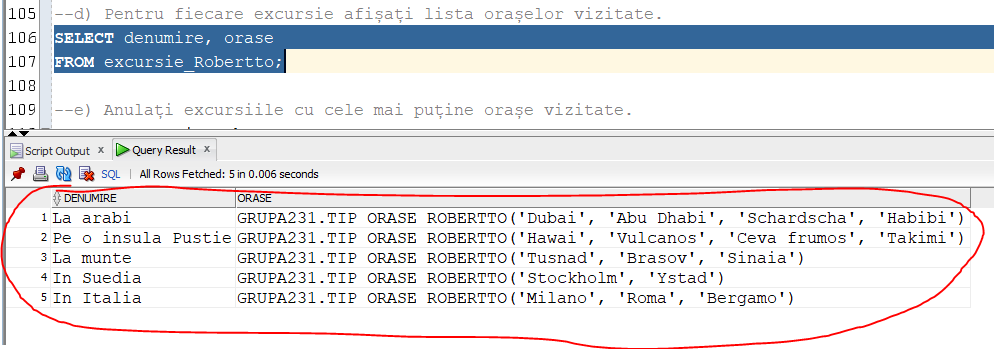




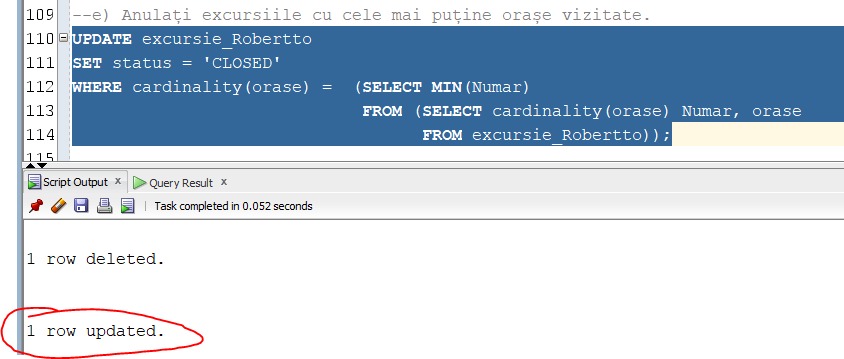
c) Pentru o excursie al cărui cod este dat, afișați numărul de orașe vizitate, respectiv numele orașelor.

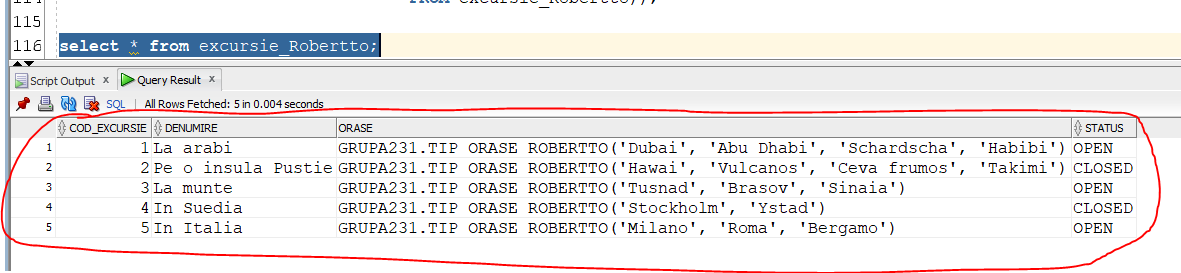


d) Pentru fiecare excursie afișați lista orașelor vizitate.



e) Anulați excursiile cu cele mai puține orașe vizitate.





Popescu Paullo Robertto Karloss

Grupa 231

Temă SGBD #6