ACADEMIA DE STUDII ECONOMICE DIN BUCUREȘTI

FACULTATEA DE CIBERNETICĂ, STATISTICĂ SI INFORMATICĂ ECONOMICĂ

SPECIALIZAREA DE INFORMATICĂ ECONOMICĂ

**PROIECT – SGBD Oracle**

PROFESOR COORDONATOR:

BOTHA IULIANA

STUDENT:

SIMIONESCU CORINA ANDREEA

BUCUREȘTI

2023-2024

Cuprins

[**Descrierea problemei alese** 3](#_Toc167046245)

[**Schema conceptuala a bazei de date** 4](#_Toc167046246)

[**Structuri de control si Cursori** 5](#_Toc167046247)

[**Exceptii** 13](#_Toc167046248)

[**Functii si Proceduri** 24](#_Toc167046249)

[**Pachete** 37](#_Toc167046250)

[**Declansatori** 44](#_Toc167046251)

# **Descrierea problemei alese**

Baza de date este concepută pentru a gestiona informațiile unui site de vânzare de laptopuri.

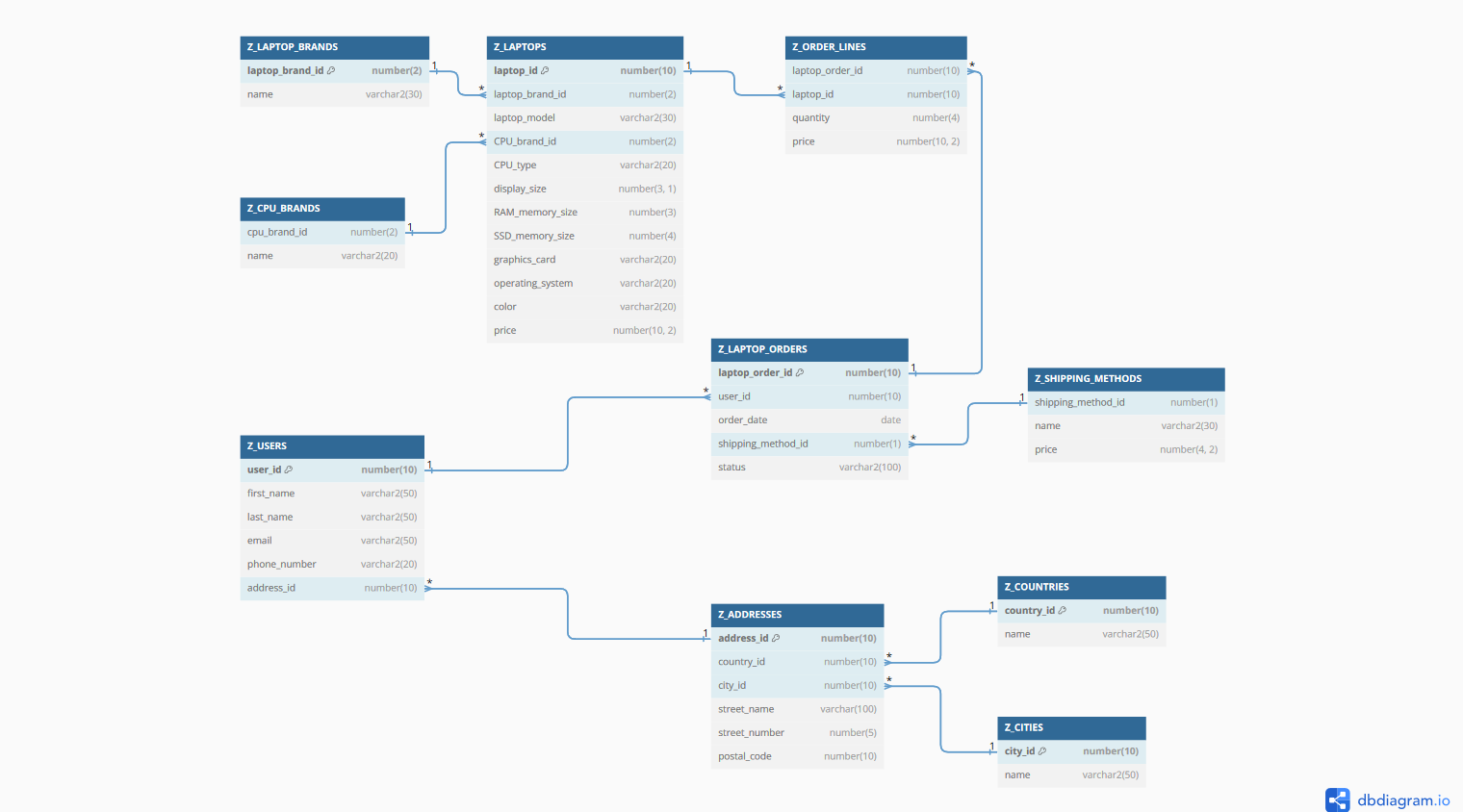
Atunci când un utilizator se loghează pe site, acesta trebuie să introducă informații precum nume, prenume, adresă de e-mail, număr de telefon și adresă. Adresa este asociată cu o țară și un oraș, iar acestea sunt preluate din tabelele Z\_COUNTRIES și Z\_CITIES, asigurând astfel integritatea referențială.

După logare, utilizatorul poate naviga prin catalogul de laptopuri, care sunt stocate în tabela Z\_LAPTOPS. Fiecare laptop are detalii precum brand, model, tip de procesor, dimensiunea ecranului, capacitatea de stocare SSD, placă grafică, sistem de operare și preț.

Atunci când un utilizator decide să plaseze o comandă, este creată o înregistrare în tabela Z\_LAPTOP\_ORDERS, cu detalii precum data comenzii, metoda de livrare și statusul comenzii. Metodele de livrare sunt gestionate în tabela Z\_SHIPPING\_METHODS.

Fiecare linie a comenzii este detaliată în tabela Z\_ORDER\_LINES, indicând laptopul comandat, cantitatea și prețul. Aceste detalii sunt legate de tabelul Z\_LAPTOP\_ORDERS și Z\_LAPTOPS prin foreign keys.

# **Schema conceptuala a bazei de date**



# **Structuri de control si Cursori**

**1)**

-- Se afiseaza populariatea unui laptop in functie de cantitatea comandata.

-- test: laptop\_id = 2 (popularitate mare)

-- test: laptop\_id = 11 (popularitate scazuta)

-- test: laptop\_id = 5 (nepopular)

ACCEPT laptop\_id\_input PROMPT 'Introduceti idul laptopului: ';

DECLARE

    v\_laptop\_id z\_order\_lines.laptop\_id%TYPE := &laptop\_id\_input;

    v\_quantity z\_order\_lines.quantity%TYPE;

    v\_laptop\_model z\_laptops.laptop\_model%TYPE;

BEGIN

    SELECT SUM(quantity) total\_ordered\_quantity

    INTO v\_quantity

    FROM Z\_ORDER\_LINES

    WHERE laptop\_id = v\_laptop\_id;

    SELECT laptop\_model

    INTO v\_laptop\_model

    FROM z\_laptops

    WHERE laptop\_id = v\_laptop\_id;

    IF v\_quantity > 3 THEN

        DBMS\_OUTPUT.PUT\_LINE('Laptopul ' || v\_laptop\_model || ' este un laptop popular, cantitatea comandata fiind: ' || v\_quantity);

    ELSIF v\_quantity BETWEEN 1 AND 3 THEN

        DBMS\_OUTPUT.PUT\_LINE('Laptopul ' || v\_laptop\_model || ' are o populariatate scazuta, cantitatea comandata fiind: ' || v\_quantity);

    ELSE

        DBMS\_OUTPUT.PUT\_LINE('Laptopul ' || v\_laptop\_model || ' nu este un laptop popular, acesta nu fiind comandat de loc ');

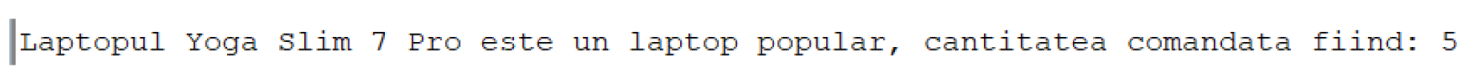
    END IF;

END;

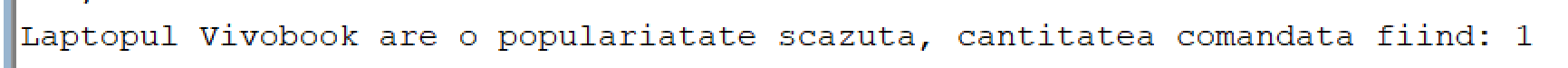
/

Testare

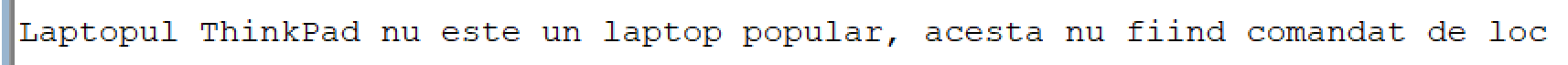
* Input = 2 (popularitate mare)



* Input = 11 (popularitate scazuta)



* Input = 5 (nepopular)



**2)**

-- Se dubleaza capacitatea memoriei RAM a unui laptop in functie de pretul introdus de la tastatura.

-- Daca exista vreun laptop cu pretul mai mare decat pretul introdus, atunci i se dubleaza memoria RAM.

-- Daca nu exista niciun laptop cu pretul mai mare decat pretul introdus, atunci nu se produc modificari.

-- test: pret = 9000 (se produc modificari)

-- test: pret = 52000 (nu se produc modificari)

ACCEPT price\_input PROMPT 'Introduceti pretul: ';

DECLARE

    v\_price z\_laptops.price%TYPE := &price\_input;

BEGIN

    UPDATE z\_laptops

    SET ram\_memory\_size = ram\_memory\_size \* 2

    WHERE price > v\_price;

    IF SQL%NOTFOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('Nu exista laptop cu pretul mai mare ca ' || v\_price);

    ELSE

        DBMS\_OUTPUT.PUT\_LINE('S a dublat capacitatea memoriei RAM pentru ' || SQL%ROWCOUNT || ' laptopuri');

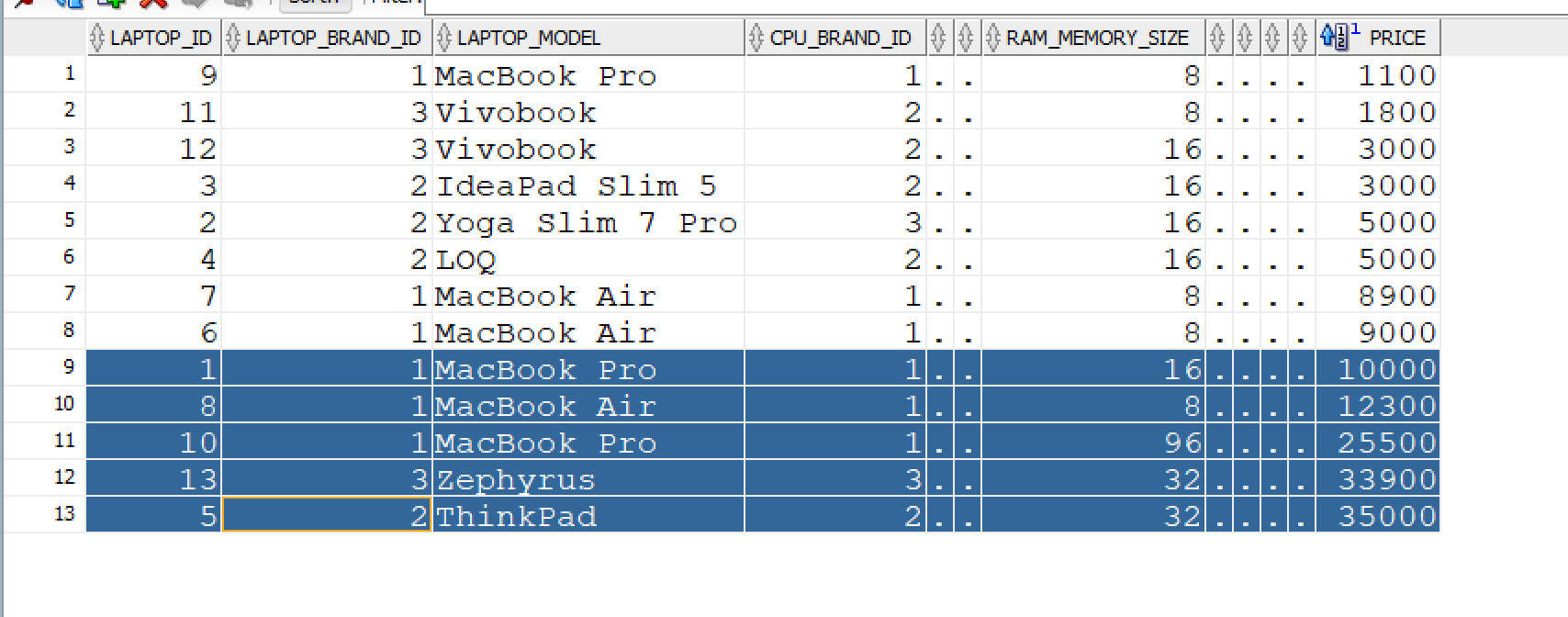
    END IF;

END;

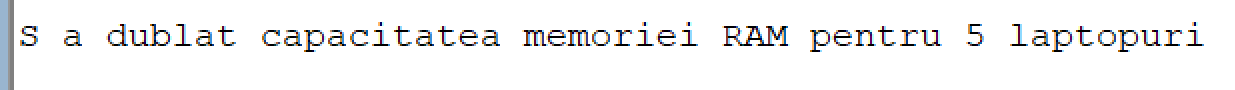
/

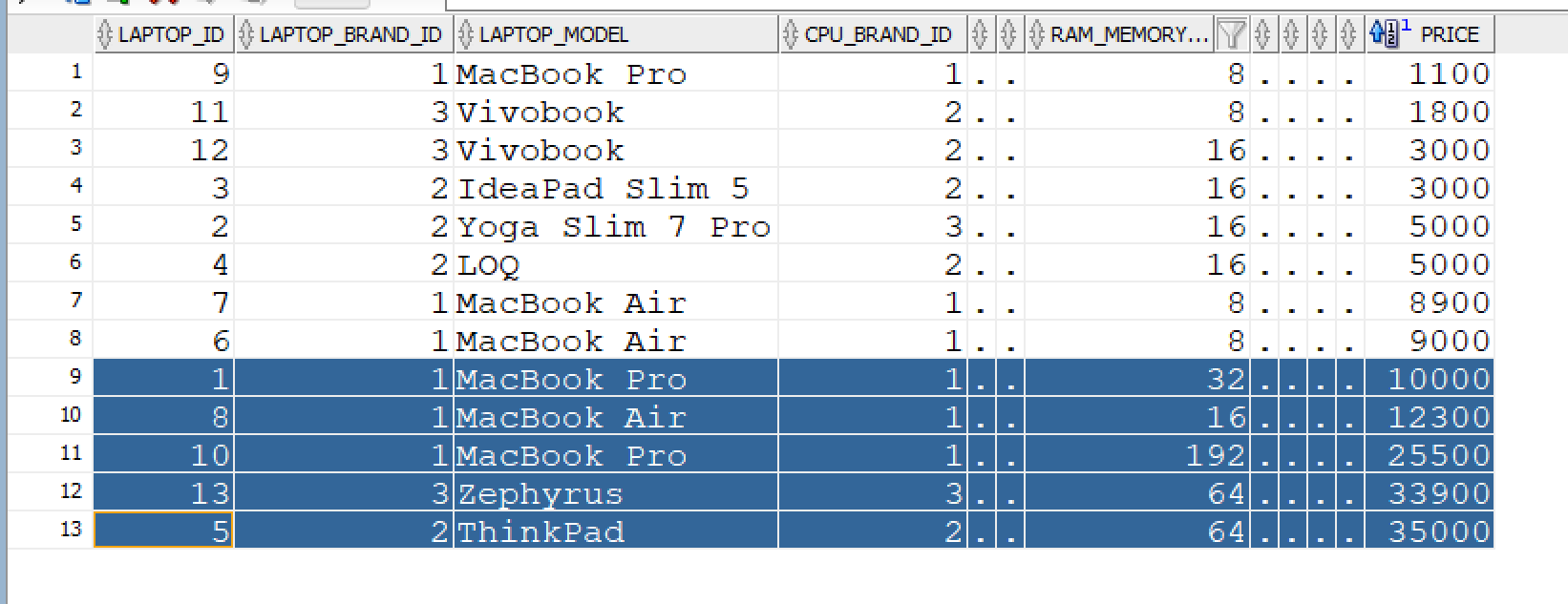
Testare

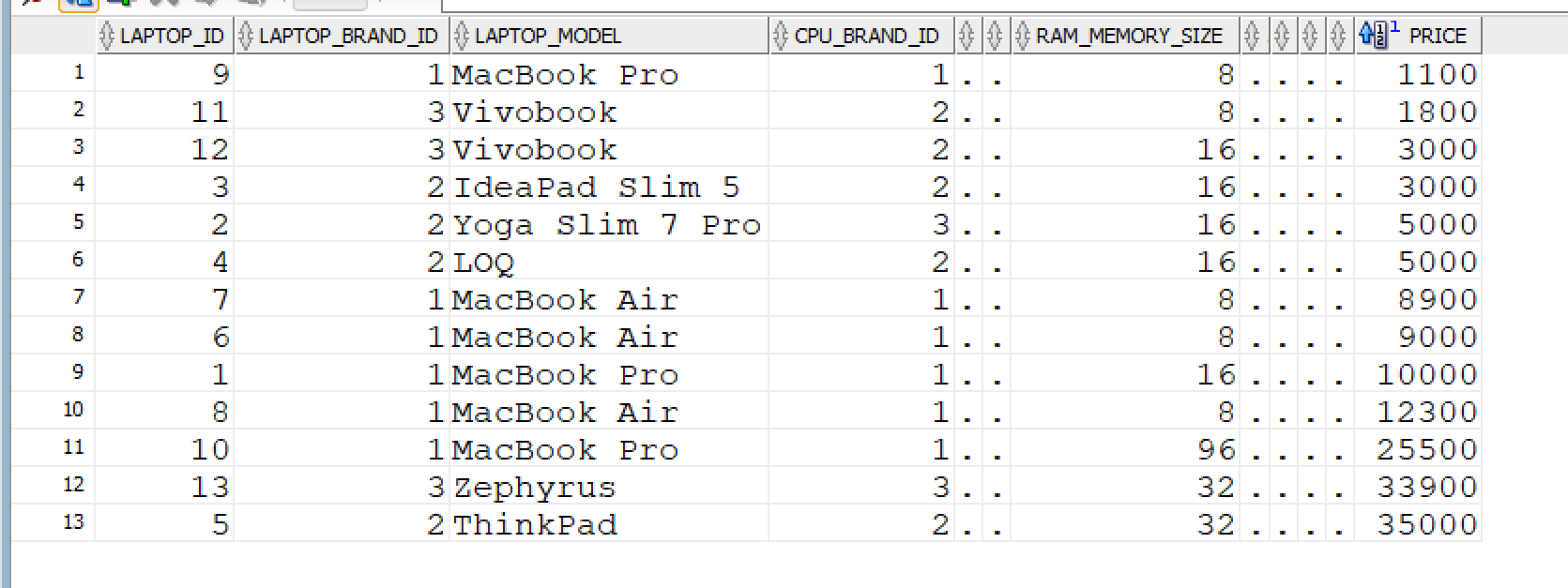
* Input = 9000 (se produc modificari)
  + Inainte de executie

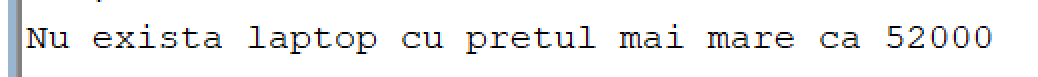


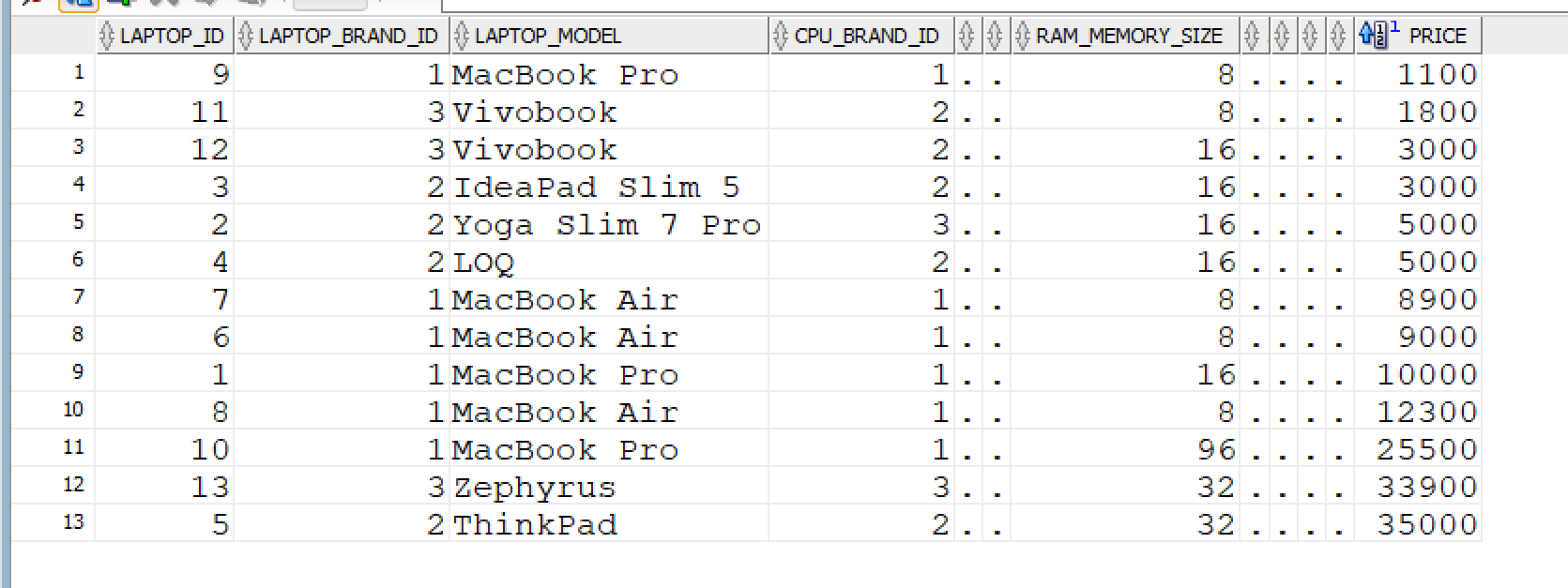
* + Dupa executie





* Input = 52000 (nu se produc modificari)
  + Inainte de executie
  + Dupa executie





**3)**

-- Se afiseaza toate laptopurile si pentru fiecare se precizeaza daca este

-- mic, mediu sau mare in functie de dimensiunea displayului

DECLARE

    CURSOR iterator IS

        SELECT \*

        FROM z\_laptops;

BEGIN

    FOR variable IN iterator LOOP

    DBMS\_OUTPUT.PUT\_LINE(variable.laptop\_id || '. ' || variable.laptop\_model);

        CASE

            WHEN variable.display\_size BETWEEN 13 AND 14 THEN

                DBMS\_OUTPUT.PUT\_LINE('--> ' || variable.display\_size || ' => laptop mic');

            WHEN variable.display\_size BETWEEN 15 AND 16 THEN

--                DBMS\_OUTPUT.PUT\_LINE(variable.laptop\_id || '. ' || variable.laptop\_model);

                DBMS\_OUTPUT.PUT\_LINE('--> ' || variable.display\_size || ' => laptop mediu');

            WHEN variable.display\_size >= 16 THEN

--                DBMS\_OUTPUT.PUT\_LINE(variable.laptop\_id || '. ' || variable.laptop\_model);

                DBMS\_OUTPUT.PUT\_LINE('--> ' || variable.display\_size || ' => laptop mare');

            ELSE

--                DBMS\_OUTPUT.PUT\_LINE(variable.laptop\_id || '. ' || variable.laptop\_model);

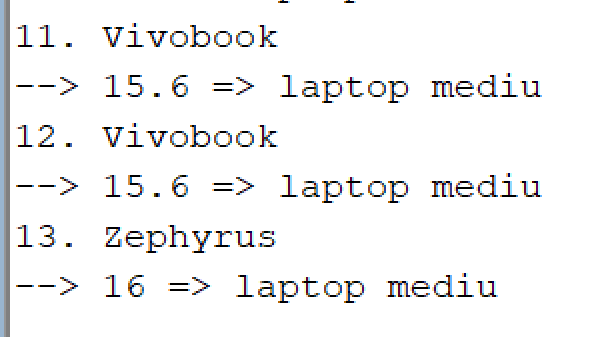
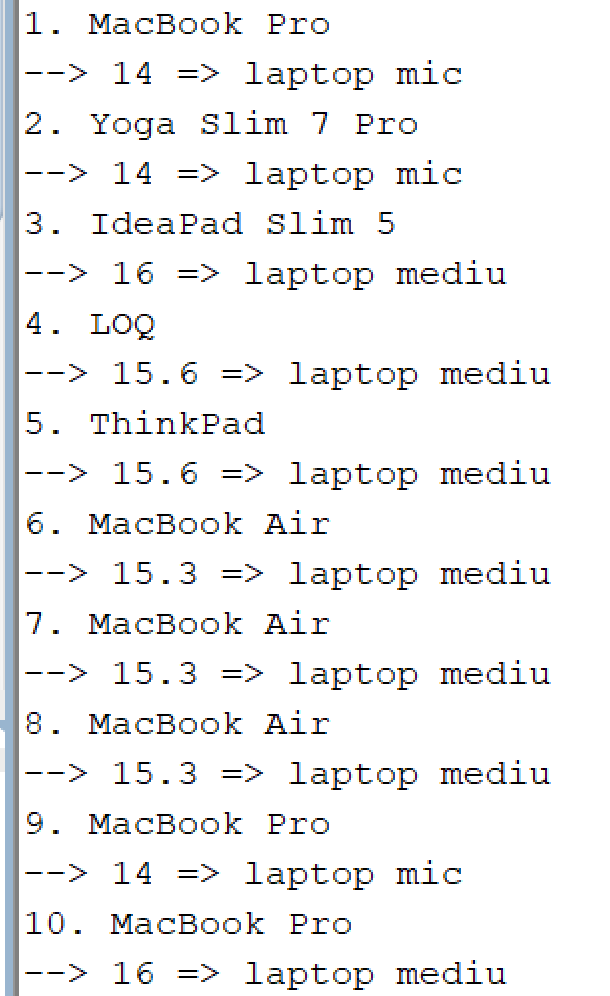
                DBMS\_OUTPUT.PUT\_LINE('--> ' || variable.display\_size || ' => necunoscut');

        END CASE;

    END LOOP;

END;

/



**4)**

-- Se afiseaza toate brandurile. Pentru fiecare brand se afiseaza toate laptopurile si

-- numarul total de laptopuri vandute

DECLARE

    v\_brand\_name z\_laptop\_brands.name%TYPE;

    v\_laptop\_model z\_laptops.laptop\_model%TYPE;

    total\_quantity\_ordered NUMBER;

    CURSOR i\_brand IS

        SELECT \*

        FROM z\_laptop\_brands;

BEGIN

    FOR var\_brand IN i\_brand LOOP

        DBMS\_OUTPUT.PUT\_LINE(var\_brand.laptop\_brand\_id || '. ' || var\_brand.name);

        FOR var\_laptop IN (SELECT \* FROM z\_laptops WHERE laptop\_brand\_id = var\_brand.laptop\_brand\_id) LOOP

            DBMS\_OUTPUT.PUT\_LINE('   ' || var\_laptop.laptop\_id || '. ' || var\_laptop.laptop\_model);

        END LOOP;

        SELECT SUM(ol.quantity) --total\_quantity\_ordered

        INTO total\_quantity\_ordered

        FROM z\_order\_lines ol

        JOIN z\_laptops l ON ol.laptop\_id = l.laptop\_id

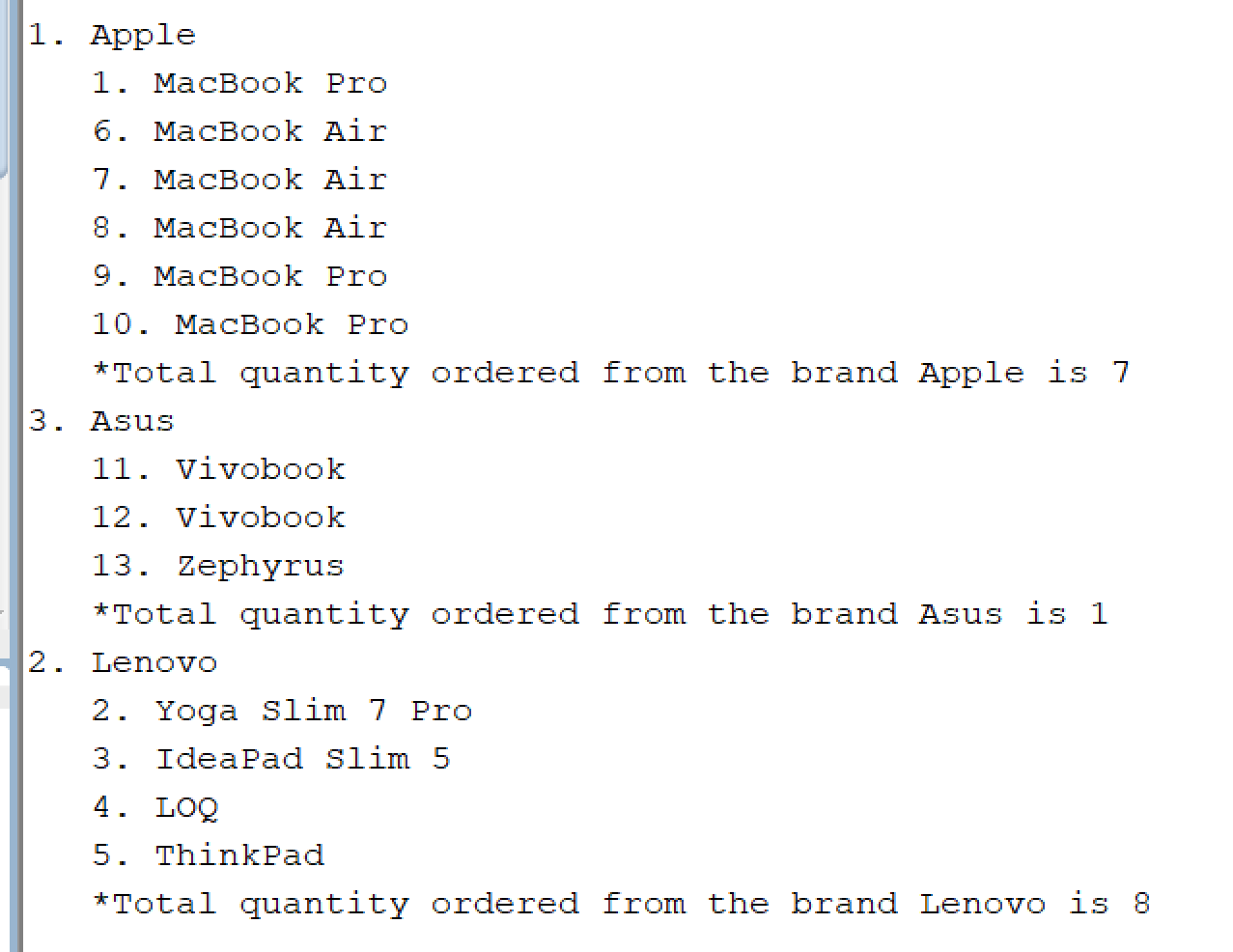
        WHERE laptop\_brand\_id = var\_brand.laptop\_brand\_id;

        DBMS\_OUTPUT.PUT\_LINE('   \*Total quantity ordered from the brand ' || var\_brand.name || ' is ' || total\_quantity\_ordered);

    END LOOP;

END;

/



**5)**

-- Se afiseaza utilizatorii care stau in Romania

DECLARE

    CURSOR i\_country IS

        SELECT \*

        FROM z\_countries;

BEGIN

    FOR var\_country IN i\_country LOOP

        IF(var\_country.name LIKE 'Romania') THEN

        DBMS\_OUTPUT.PUT\_LINE(var\_country.name || ':');

            FOR var\_user IN (SELECT \*

                            FROM z\_users u

                            JOIN z\_addresses a ON u.address\_id = a.address\_id

                            WHERE country\_id = var\_country.country\_id) LOOP

            DBMS\_OUTPUT.PUT\_LINE('  -> ' || var\_user.first\_name || ' ' || var\_user.last\_name);

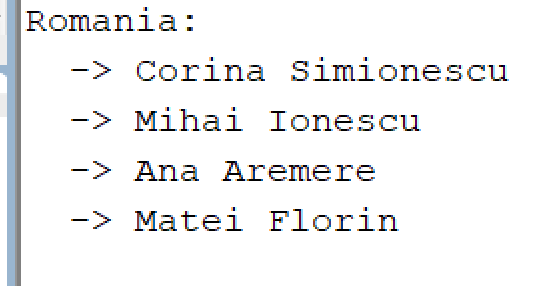
            END LOOP;

        END IF;

    END LOOP;

END;

/



**6)**

-- Se afiseaza tara si orasul in care locuieste fiecare user care a comandat cel mai scump laptop comandat

DECLARE

    CURSOR i\_user (p z\_laptops.laptop\_id%TYPE) IS --idul celui mai scump laptop comandat

        SELECT u.user\_id, u.first\_name, u.last\_name, l.laptop\_model, l.price

        FROM z\_users u

        JOIN z\_laptop\_orders lo ON u.user\_id = lo.user\_id

        JOIN z\_order\_lines ol ON lo.laptop\_order\_id = ol.laptop\_order\_id

        JOIN z\_laptops l ON ol.laptop\_id = l.laptop\_id

        WHERE ol.laptop\_id = p;

    most\_expensive\_ordered\_laptop\_id z\_laptops.laptop\_id%TYPE;

    country\_name z\_countries.name%TYPE;

    city\_name z\_cities.name%TYPE;

BEGIN

    SELECT l.laptop\_id

    INTO most\_expensive\_ordered\_laptop\_id

    FROM z\_laptops l

    JOIN z\_order\_lines ol ON l.laptop\_id = ol.laptop\_id

    ORDER BY l.price DESC

    FETCH FIRST 1 ROW ONLY;

    SELECT co.name

    INTO country\_name

    FROM z\_countries co

    JOIN z\_addresses a ON co.country\_id = a.country\_id

    JOIN z\_users u ON a.address\_id = u.address\_id

    JOIN z\_laptop\_orders lo ON u.user\_id  = lo.user\_id

    JOIN z\_order\_lines ol ON lo.laptop\_order\_id = ol.laptop\_order\_id

    WHERE ol.laptop\_id = most\_expensive\_ordered\_laptop\_id;

    FOR var\_user IN i\_user (most\_expensive\_ordered\_laptop\_id) LOOP

        DBMS\_OUTPUT.PUT\_LINE('Userul ' || var\_user.first\_name || ' ' || var\_user.last\_name || ' este din ' || country\_name || ' si a comandat laptopul ' || var\_user.laptop\_model || ', ');

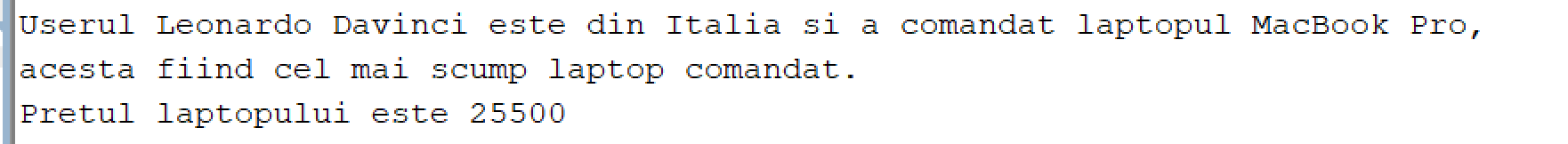
        DBMS\_OUTPUT.PUT\_LINE('acesta fiind cel mai scump laptop comandat. ');

        DBMS\_OUTPUT.PUT\_LINE('Pretul laptopului este ' || var\_user.price);

    END LOOP;

END;

/



# **Exceptii**

**1)**

-- Se dubleaza pretul metodei de livrare al carui id este introdus de la tastatura

-- test: shipping\_method\_id = 1, pretul s a modificat in 30

-- test: shipping\_method\_id = 4, nu exista metoda de livrare cu idul introdus

ACCEPT shipping\_method\_id\_input PROMPT 'Introduceti idul metodei de livrare careia vreti sa ii dublati pretul: ';

DECLARE

    v\_shipping\_method\_id z\_shipping\_methods.shipping\_method\_id%TYPE := &shipping\_method\_id\_input;

    v\_price z\_shipping\_methods.price%TYPE;

    v\_name z\_shipping\_methods.name%TYPE;

BEGIN

    SELECT price, name

    INTO v\_price, v\_name

    FROM z\_shipping\_methods

    WHERE shipping\_method\_id = v\_shipping\_method\_id;

    v\_price := v\_price \* 2;

    UPDATE z\_shipping\_methods

    SET price = v\_price

    WHERE shipping\_method\_id = v\_shipping\_method\_id;

    SELECT price

    INTO v\_price

    FROM z\_shipping\_methods

    WHERE shipping\_method\_id = v\_shipping\_method\_id;

    DBMS\_OUTPUT.PUT\_LINE('Pretul metodei de livrare '|| v\_name || ' s a modificat. Noul pret este: ' || v\_price);

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

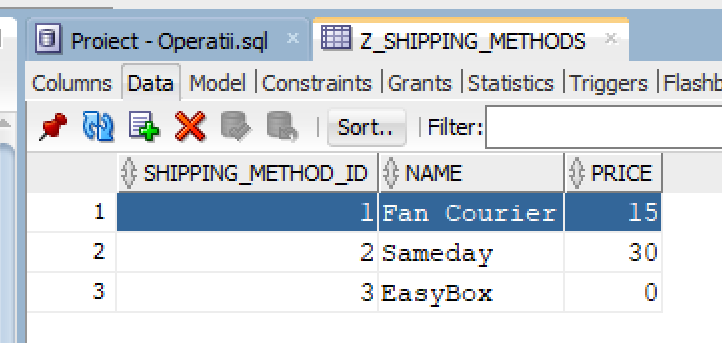
        DBMS\_OUTPUT.PUT\_LINE('Nu exista nicio meotda de livrare cu idul introdus');

END;

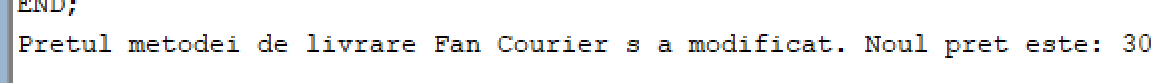
/

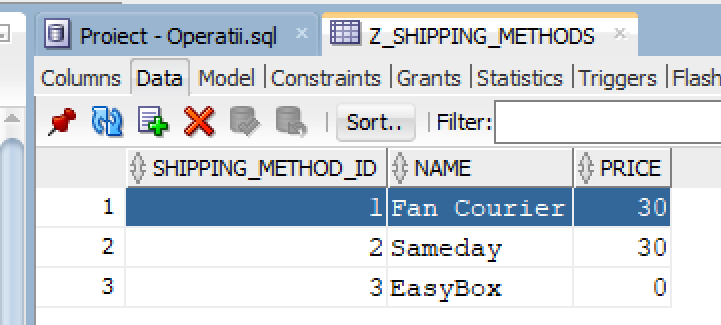
Testare

* Input = 1 (pretul se modifica in 30)
  + Inainte de executie

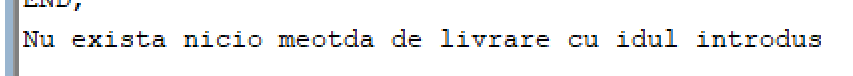


* + Dupa executie





* Input = 4 (nu exista metoda de livrare cu idul introdus)



**2)**

-- Se dubleaza pretul pentru un singur laptop cu capacitatea RAM introdusa de utilizator

-- test: input = 96 (exista un singur laptop cu capacitatea RAM = 96 => se modifica pretul)

-- test: input = 8 (mai multe laptopuri au capacitatea RAM = 8 => TOO\_MANY\_ROWS)

-- test: input = 100 (nu exista niciun laptop cu capacitatea RAM = 100 => NO\_DATA\_FOUND)

ACCEPT v\_ram\_memory\_size\_input PROMPT 'Introduceti capacitatea RAM: ';

DECLARE

    v\_ram\_memory\_size z\_laptops.ram\_memory\_size%TYPE := &v\_ram\_memory\_size\_input;

    v\_price z\_laptops.price%TYPE;

    v\_laptop\_id z\_laptops.laptop\_id%TYPE;

BEGIN

    SELECT laptop\_id, price

    INTO v\_laptop\_id, v\_price

    FROM z\_laptops

    WHERE ram\_memory\_size = v\_ram\_memory\_size;

    DBMS\_OUTPUT.PUT\_LINE('Modificare laptop cu id: ' || v\_laptop\_id);

    DBMS\_OUTPUT.PUT\_LINE('Pret vechi: ' || v\_price);

    UPDATE z\_laptops

    SET price = price\*2

    WHERE laptop\_id = v\_laptop\_id;

    SELECT price

    INTO v\_price

    FROM z\_laptops

    WHERE ram\_memory\_size = v\_ram\_memory\_size;

    DBMS\_OUTPUT.PUT\_LINE('Pret nou: ' || v\_price);

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('Niciun laptop cu capacitatea RAM introdusa');

    WHEN TOO\_MANY\_ROWS THEN

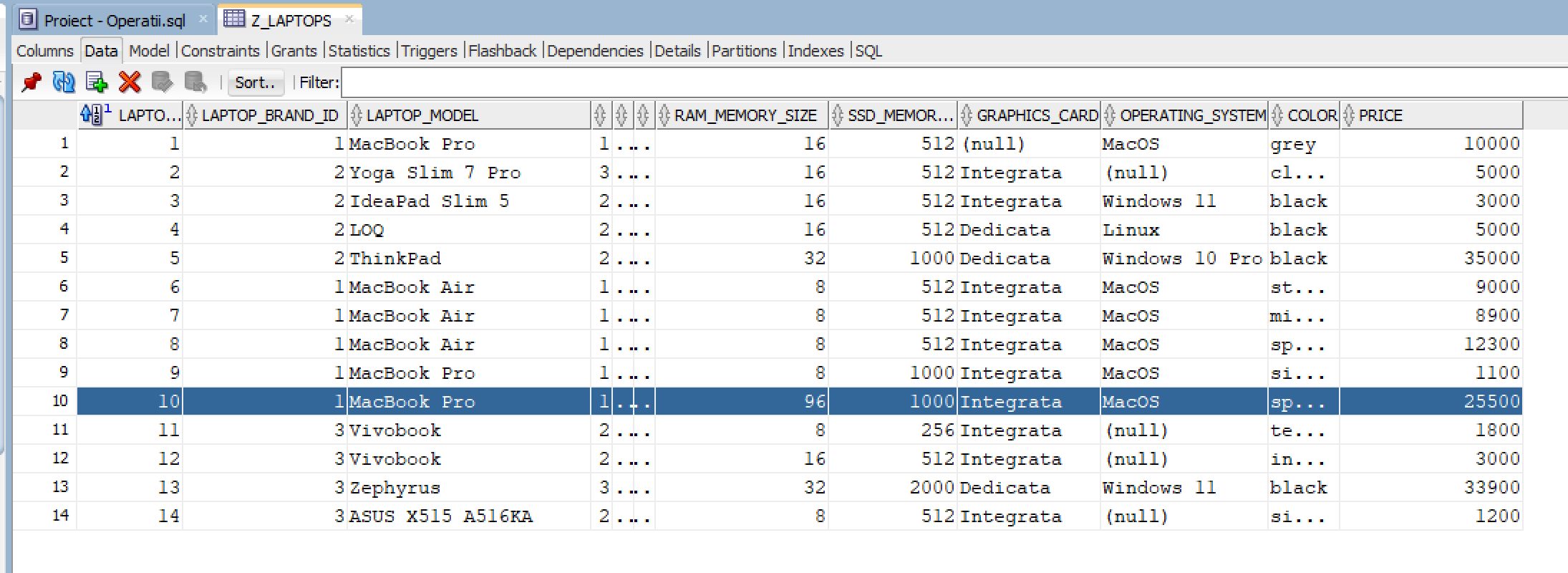
        DBMS\_OUTPUT.PUT\_LINE('Prea multe randuri');

END;

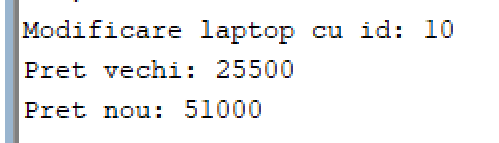
/

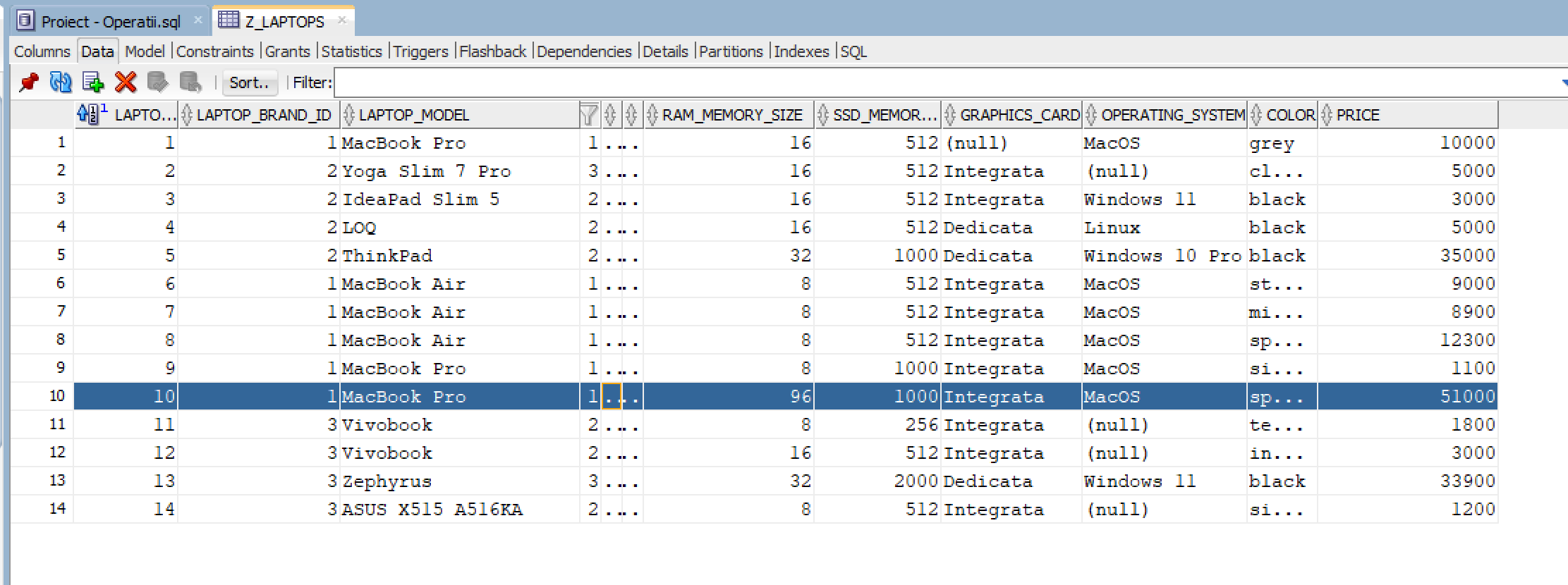
Testare

* Input = 96 (exista un singur laptop cu capacitatea RAM = 96 => se modifica pretul)
  + Inainte de executie

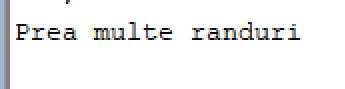


* + Dupa executie

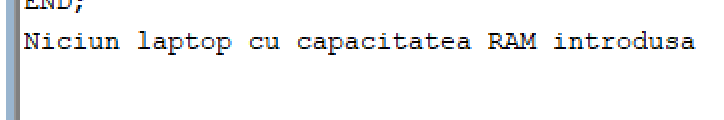




* Input = 8 (mai multe laptopuri au capacitatea RAM = 8 => TOO\_MANY\_ROWS)



* Input = 100 (nu exista niciun laptop cu capacitatea RAM = 100 => NO\_DATA\_FOUND)



**3)**

-- Se afiseaza userii din tara introdusa de la tastatura

-- test: input = 1 (tara exista si are useri)

-- test: input = 4 (tara exista, dar niciun user nu este din aceasta tara)

-- test: input = 100 (tara nu exista)

ACCEPT country\_id\_input PROMPT 'Introduceti idul tarii: ';

DECLARE

    v\_country\_id z\_countries.country\_id%TYPE := &country\_id\_input;

    v\_country\_name z\_countries.name%TYPE;

    v\_user\_count NUMBER := 0;

    NO\_USERS\_IN\_COUNTRY EXCEPTION;

BEGIN

    SELECT name INTO v\_country\_name

    FROM z\_countries

    WHERE country\_id = v\_country\_id;

    IF v\_country\_name IS NOT NULL THEN

        DBMS\_OUTPUT.PUT\_LINE(v\_country\_name || ':');

        SELECT COUNT(\*)

        INTO v\_user\_count

        FROM z\_users u

        JOIN z\_addresses a ON u.address\_id = a.address\_id

        WHERE a.country\_id = v\_country\_id;

        IF v\_user\_count <> 0 THEN

            FOR var\_user IN (SELECT first\_name, last\_name

                              FROM z\_users u

                              JOIN z\_addresses a ON u.address\_id = a.address\_id

                              WHERE a.country\_id = v\_country\_id) LOOP

                DBMS\_OUTPUT.PUT\_LINE('  -> ' || var\_user.first\_name || ' ' || var\_user.last\_name);

            END LOOP;

        ELSE

            RAISE NO\_USERS\_IN\_COUNTRY;

        END IF;

    END IF;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('Nu exista nicio tara cu idul introdus');

    WHEN NO\_USERS\_IN\_COUNTRY THEN

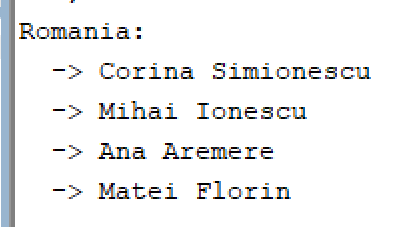
        DBMS\_OUTPUT.PUT\_LINE('Nu exista useri din tara introdusa');

END;

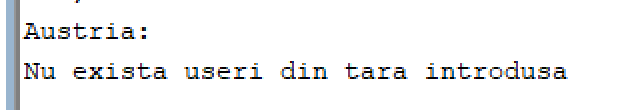
/

Testare

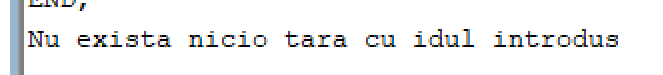
* Input = 1 (tara exista si are useri)



* Input = 4 (tara exista, dar niciun user nu este din aceasta tara)



* Input = 100 (tara nu exista)



**4)**

-- Inserarea unui user in tabela z\_users cu datele introduse de la tastatura

-- test: Se face inserarea cu succes

--       -> user\_id\_input = 100

--       -> first\_name\_input = a

--       -> last\_name\_input = a

--       -> email\_input = a

-- test: Se apeleaza exceptia ID\_GRESIT: Idul trebuie sa contina doar numere

--       -> user\_id\_input = abc

--       -> first\_name\_input = a

--       -> last\_name\_input = a

--       -> email\_input = a

-- test: Se apeleaza exceptia ID\_NU\_E\_UNIC: Idul introdus nu este unic

--       -> user\_id\_input = 1

--       -> first\_name\_input = a

--       -> last\_name\_input = a

--       -> email\_input = a

ACCEPT user\_id\_input PROMPT 'user id: ';

ACCEPT first\_name\_input PROMPT 'first name: ';

ACCEPT last\_name\_input PROMPT 'last name: ';

ACCEPT email\_input PROMPT 'email : ';

DECLARE

    v\_user\_id z\_users.user\_id%TYPE;

    v\_first\_name z\_users.first\_name%TYPE := LOWER('&first\_name\_input');

    v\_last\_name z\_users.last\_name%TYPE := LOWER('&last\_name\_input');

    v\_email z\_users.email%TYPE := LOWER('&email\_input');

    ID\_GRESIT EXCEPTION;

    PRAGMA EXCEPTION\_INIT(ID\_GRESIT, -06502);

BEGIN

    v\_user\_id := TO\_NUMBER('&user\_id\_input');

    DBMS\_OUTPUT.PUT\_LINE(v\_user\_id || ' - ' || v\_first\_name || ' - ' || v\_last\_name || ' - ' || v\_email);

    INSERT INTO z\_users(user\_id, first\_name, last\_name, email)

    VALUES(v\_user\_id, v\_first\_name, v\_last\_name, v\_email);

    DBMS\_OUTPUT.PUT\_LINE('User inserat cu succes!');

EXCEPTION

    WHEN ID\_GRESIT THEN

        DBMS\_OUTPUT.PUT\_LINE('Userul nu a putut fi introdus. Idul trebuie sa contina doar numere');

    WHEN DUP\_VAL\_ON\_INDEX THEN

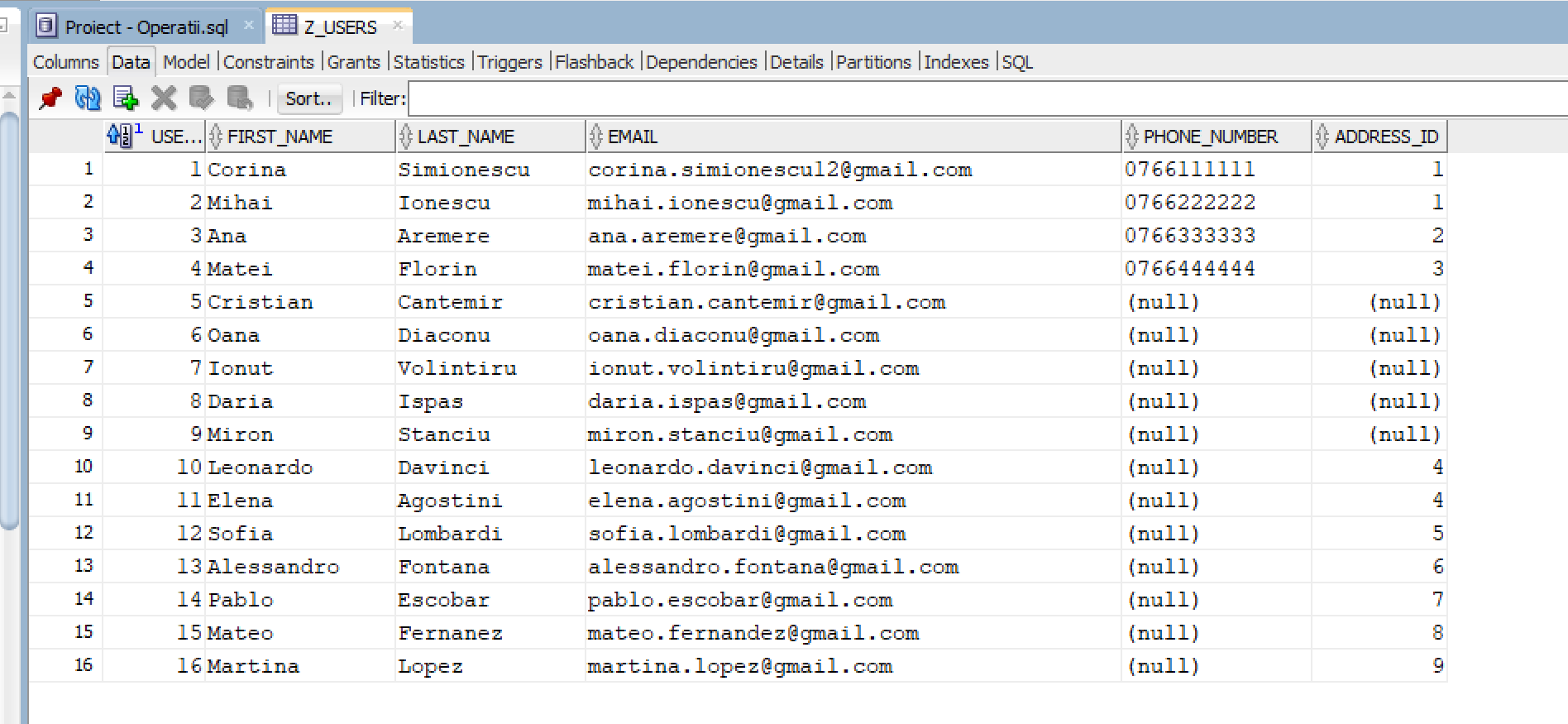
        DBMS\_OUTPUT.PUT\_LINE('Userul nu a putut fi introdus. Idul introdus nu este unic');

END;

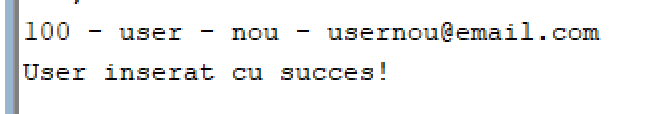
/

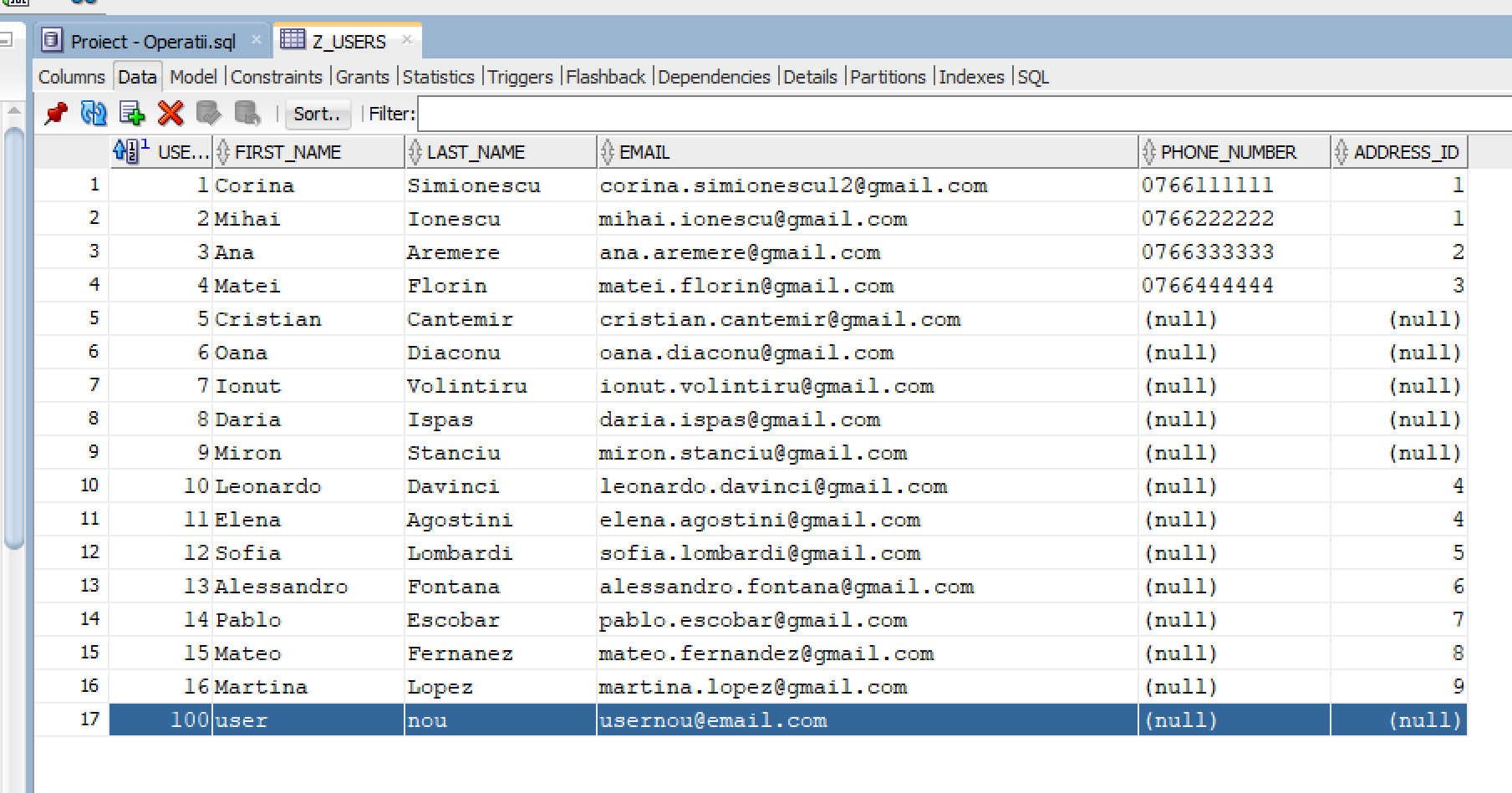
Testare

* Test 1: Se face inserarea cu succes
  + user\_id\_input = 100
  + first\_name\_input = User
  + last\_name\_input = Nou
  + email\_input = usernou@email.com
* Inainte de executie

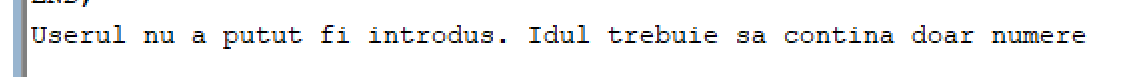


* Dupa executie

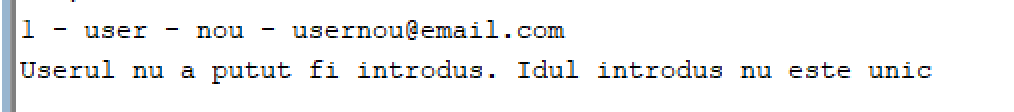




* Test 2: Se apeleaza exceptia ID\_GRESIT.
  + user\_id\_input = abc
  + first\_name\_input = User
  + last\_name\_input = Nou
  + email\_input = [usernou@email.com](mailto:usernou@email.com)



* Test 3: Se apeleaza exceptia ID\_NU\_E\_UNIC.
  + user\_id\_input = 1
  + first\_name\_input = User
  + last\_name\_input = Nou
  + email\_input = [usernou@email.com](mailto:usernou@email.com)



**5)**

-- Stergerea unei adrese pe baza idului introdus de la tastatura. Daca adresa este atribuita

-- macar unui user, atunci nu poate fi stearsa.

-- test: address\_id\_input = 1 (Adresa nu se sterge)

-- test:

--      -> Rulati comanda de INSERT a unei adrese noi (este dupa blocul pl/sql)

--      -> address\_id\_input = 100 (Adresa se sterge)

ACCEPT address\_id\_input PROMPT 'Introduceti idul adresei pe care vreti sa o stergeti:';

DECLARE

    v\_address\_id Z\_addresses.address\_id%TYPE := &address\_id\_input;

    ADRESA\_ESTE\_FOLOSITA EXCEPTION;

    PRAGMA EXCEPTION\_INIT(ADRESA\_ESTE\_FOLOSITA, -2292);

BEGIN

    DELETE FROM z\_addresses

    WHERE address\_id = v\_address\_id;

    DBMS\_OUTPUT.PUT\_LINE('Adresa stearsa cu succes!');

EXCEPTION

    WHEN ADRESA\_ESTE\_FOLOSITA THEN

        DBMS\_OUTPUT.PUT\_LINE('Adresa nu poate fi stearsa deoarece este atribuita unuia sau mai multor useri.');

        DBMS\_OUTPUT.PUT\_LINE('Stergeti userii cu address\_id = ' || v\_address\_id || ', dupa care stergeti adresa.');

END;

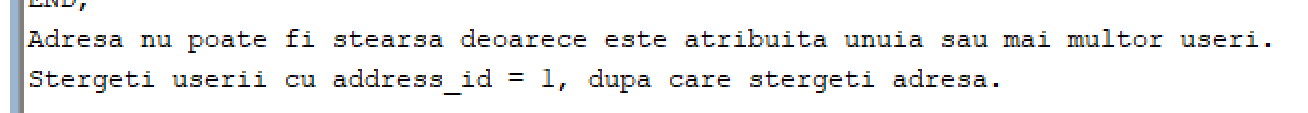
/

INSERT INTO Z\_ADDRESSES (address\_id, country\_id, city\_id)

VALUES (100, 1, 1);

Testare

* Input = 1 (adresa nu se sterge)



* Test
  + Rulati comanda de INSERT a unei adrese noi (este dupa blocul pl/sql)
  + Input = 100 (Adresa se sterge)



# **Functii si Proceduri**

**1) Procedura + Functie**

-- Creez coloana 'discount' in tabela z\_users.

-- Procedura atribuie discount clientilor care au cumparat 3 sau mai multe laptopuri.

-- Pentru calcularea numarului de laptopuri comandate, am facut o functie.

-- Functia creata o folosesc in procedura pentru acordarea discountului.

ALTER TABLE z\_users

ADD discount NUMBER DEFAULT 0;

/

CREATE OR REPLACE FUNCTION get\_no\_ordered\_laptops (p\_user\_id z\_users.user\_id%TYPE)

RETURN NUMBER

IS

    no\_ordered\_laptops NUMBER;

BEGIN

    SELECT SUM(ol.quantity)

    INTO no\_ordered\_laptops

    FROM z\_users u

    JOIN z\_laptop\_orders lo ON u.user\_id = lo.user\_id

    JOIN z\_order\_lines ol ON lo.laptop\_order\_id = ol.laptop\_order\_id

    WHERE u.user\_id = p\_user\_id;

    RETURN no\_ordered\_laptops;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        RETURN NULL;

END;

/

/

CREATE OR REPLACE PROCEDURE give\_discount

IS

    CURSOR c IS

        SELECT \*

        FROM z\_users

        FOR UPDATE OF discount;

    var c%ROWTYPE;

BEGIN

    OPEN c;

    LOOP

        FETCH c INTO var;

        EXIT WHEN c%NOTFOUND;

        IF get\_no\_ordered\_laptops(var.user\_id) >= 3 THEN

            UPDATE z\_users

            SET discount = 1000

            WHERE CURRENT OF c;

        END IF;

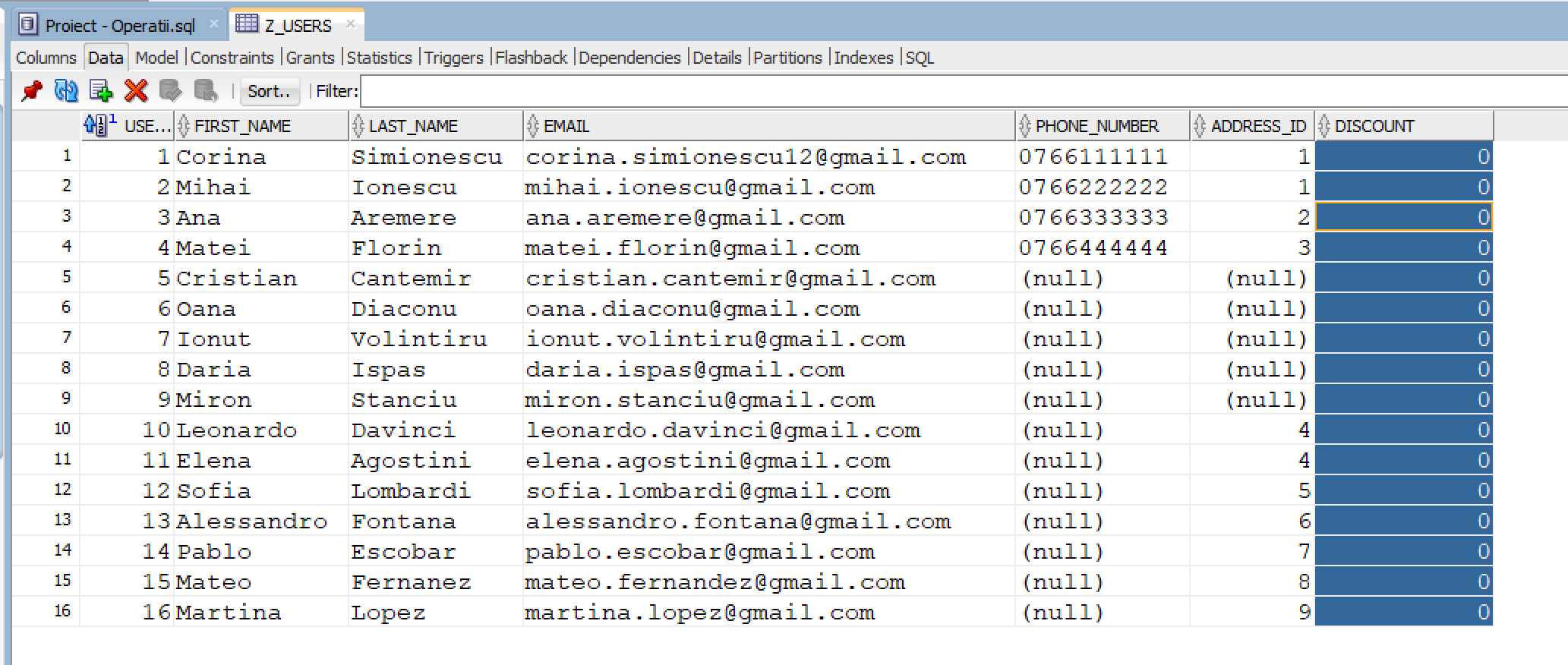
    END LOOP;

END;

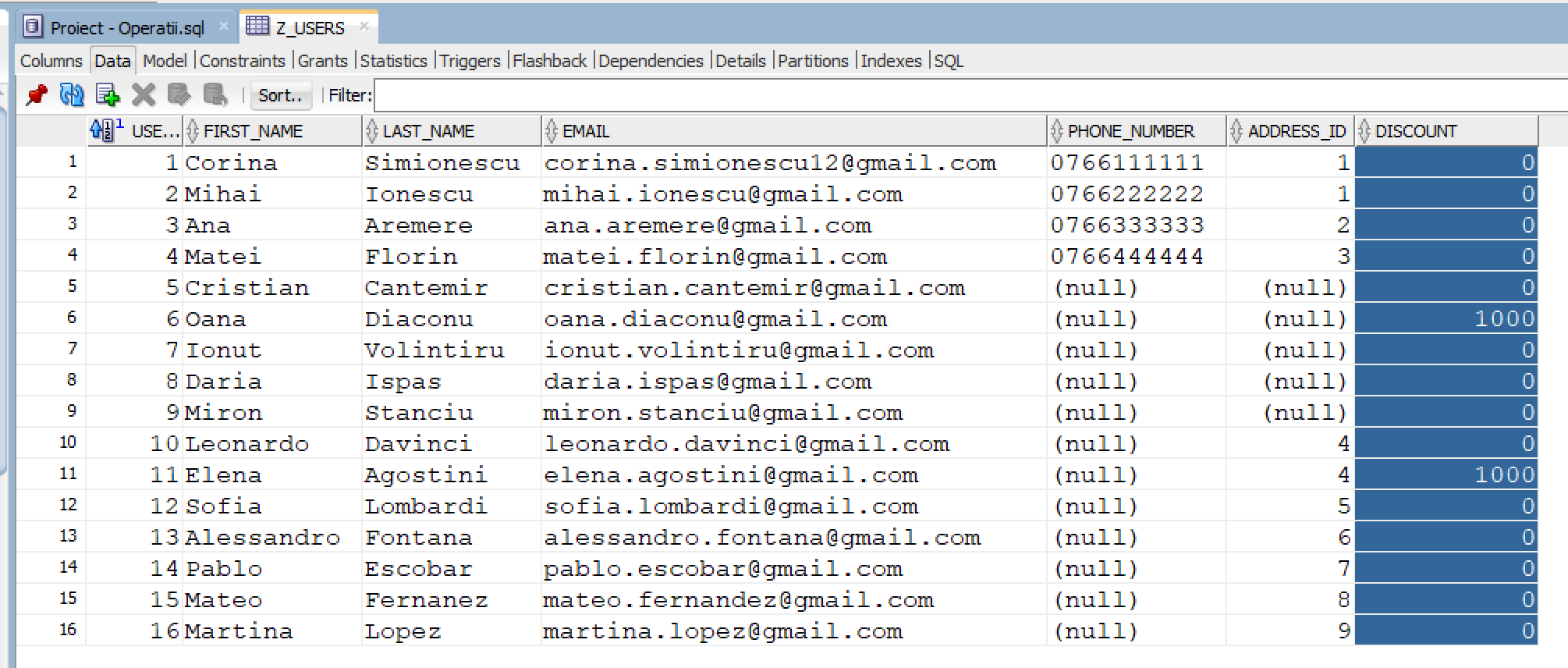
/

EXECUTE give\_discount;

* Inainte de executie



* Dupa executie



**2) Procedura**

-- Procedura care creeaza si afiseaza o tabela indexata cu userii care au cumparat laptopuri cu brandul

-- dat ca parametru.

/

CREATE OR REPLACE PROCEDURE create\_table\_users\_laptop\_brand (p\_brand\_name z\_laptop\_brands.name%TYPE)

IS

    TYPE rec IS RECORD

    (

        user\_id z\_users.user\_id%TYPE,

        user\_full\_name VARCHAR2(100)

    );

    TYPE table\_users\_laptop\_brand IS TABLE OF rec;

    t table\_users\_laptop\_brand;

BEGIN

    SELECT u.user\_id, u.first\_name ||' '|| u.last\_name

    BULK COLLECT INTO t

    FROM z\_users u

    JOIN z\_laptop\_orders lo ON u.user\_id = lo.user\_id

    JOIN z\_order\_lines ol ON lo.laptop\_order\_id = ol.laptop\_order\_id

    JOIN z\_laptops l ON ol.laptop\_id = l.laptop\_id

    JOIN z\_laptop\_brands lb ON l.laptop\_brand\_id = lb.laptop\_brand\_id

    WHERE UPPER(lb.name) LIKE UPPER(p\_brand\_name);

    DBMS\_OUTPUT.PUT\_LINE('\*\*\*\*\*\*\*\*\*\*\*'||' Utilizatori care au comandat de la: '||UPPER(p\_brand\_name)||' \*\*\*\*\*\*\*\*\*\*\*');

    FOR i IN 1.. t.COUNT LOOP

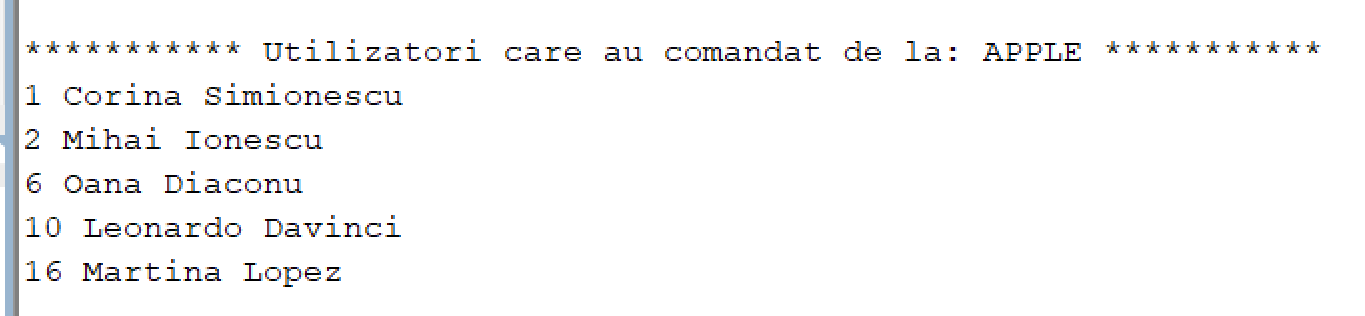
        DBMS\_OUTPUT.PUT\_LINE(t(i).user\_id ||' '|| t(i).user\_full\_name);

    END LOOP;

END;

/

EXECUTE create\_table\_users\_laptop\_brand('apple');



**3) Functie**

-- Functie care insereaza un laptop in tabela z\_laptops. Datele ce trebuie inserate se dau ca parametrii si sunt

-- verificate. Idul laptopului este calculat automat in interiorul functiei.

/

CREATE OR REPLACE FUNCTION insert\_new\_laptop (

    p\_laptop\_brand\_id z\_laptops.laptop\_brand\_id%TYPE,

    p\_laptop\_model z\_laptops.laptop\_model%TYPE,

    p\_cpu\_brand\_id z\_laptops.cpu\_brand\_id%TYPE,

    p\_price z\_laptops.price%TYPE

)

RETURN VARCHAR2

IS

    CURSOR c\_laptop\_brand\_ids IS

        SELECT laptop\_brand\_id

        FROM z\_laptop\_brands;

    CURSOR c\_cpu\_brand\_ids IS

        SELECT cpu\_brand\_id

        FROM z\_cpu\_brands;

    correct\_data BOOLEAN := FALSE;

    p\_laptop\_id z\_laptops.laptop\_id%TYPE;

    message VARCHAR2(100);

    WRONG\_BRAND EXCEPTION;

    WRONG\_CPU EXCEPTION;

BEGIN

    FOR i IN c\_laptop\_brand\_ids LOOP

        IF p\_laptop\_brand\_id = i.laptop\_brand\_id THEN

            correct\_data := TRUE;

        END IF;

    END LOOP;

    IF correct\_data = FALSE THEN

        RAISE WRONG\_BRAND;

    END IF;

    correct\_data := FALSE;

    FOR i IN c\_cpu\_brand\_ids LOOP

        IF p\_cpu\_brand\_id = i.cpu\_brand\_id THEN

            correct\_data := TRUE;

        END IF;

    END LOOP;

    IF correct\_data = FALSE THEN

        RAISE WRONG\_CPU;

    END IF;

    SELECT MAX(laptop\_id) + 1

    INTO p\_laptop\_id

    FROM z\_laptops;

    INSERT INTO z\_laptops (laptop\_id, laptop\_brand\_id, laptop\_model, cpu\_brand\_id, price)

    VALUES(p\_laptop\_id, p\_laptop\_brand\_id, p\_laptop\_model, p\_cpu\_brand\_id, p\_price);

    message := 'Laptopul a fost inserat cu succes!';

    RETURN message;

EXCEPTION

    WHEN WRONG\_BRAND THEN

        message := 'Error: Idul brandului este gresit';

        RETURN message;

    WHEN WRONG\_CPU THEN

        message := 'Error: Idul CPU este gresit';

        RETURN message;

END;

/

/

BEGIN

    -- Test 1: date corecte

--    DBMS\_OUTPUT.PUT\_LINE(insert\_new\_laptop(2, 'Laptop Gaming Lenovo Legion 7', 3, 11067));

    -- Test 2: brand id gresit

--    DBMS\_OUTPUT.PUT\_LINE(insert\_new\_laptop(0, 'Laptop Gaming Lenovo Legion 7', 3, 11067));

    -- Test 3: cpu id gresit

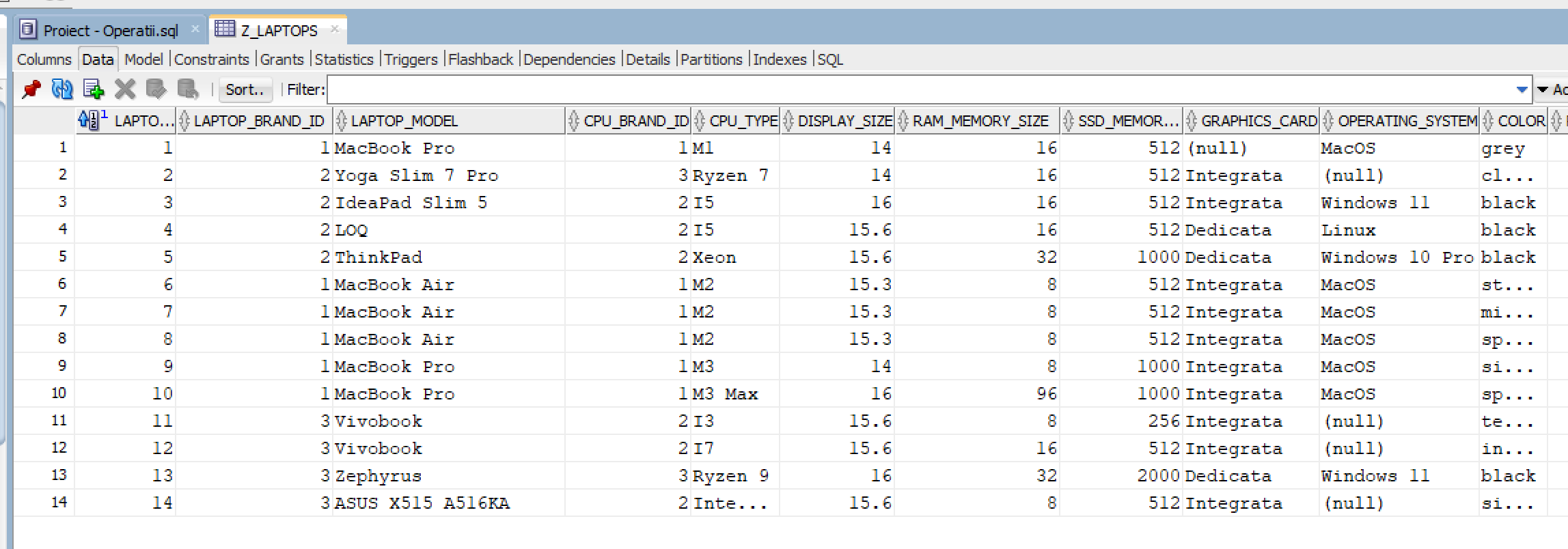
    DBMS\_OUTPUT.PUT\_LINE(insert\_new\_laptop(3, 'Laptop Gaming Lenovo Legion 7', 100, 11067));

END;

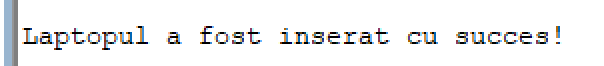
/

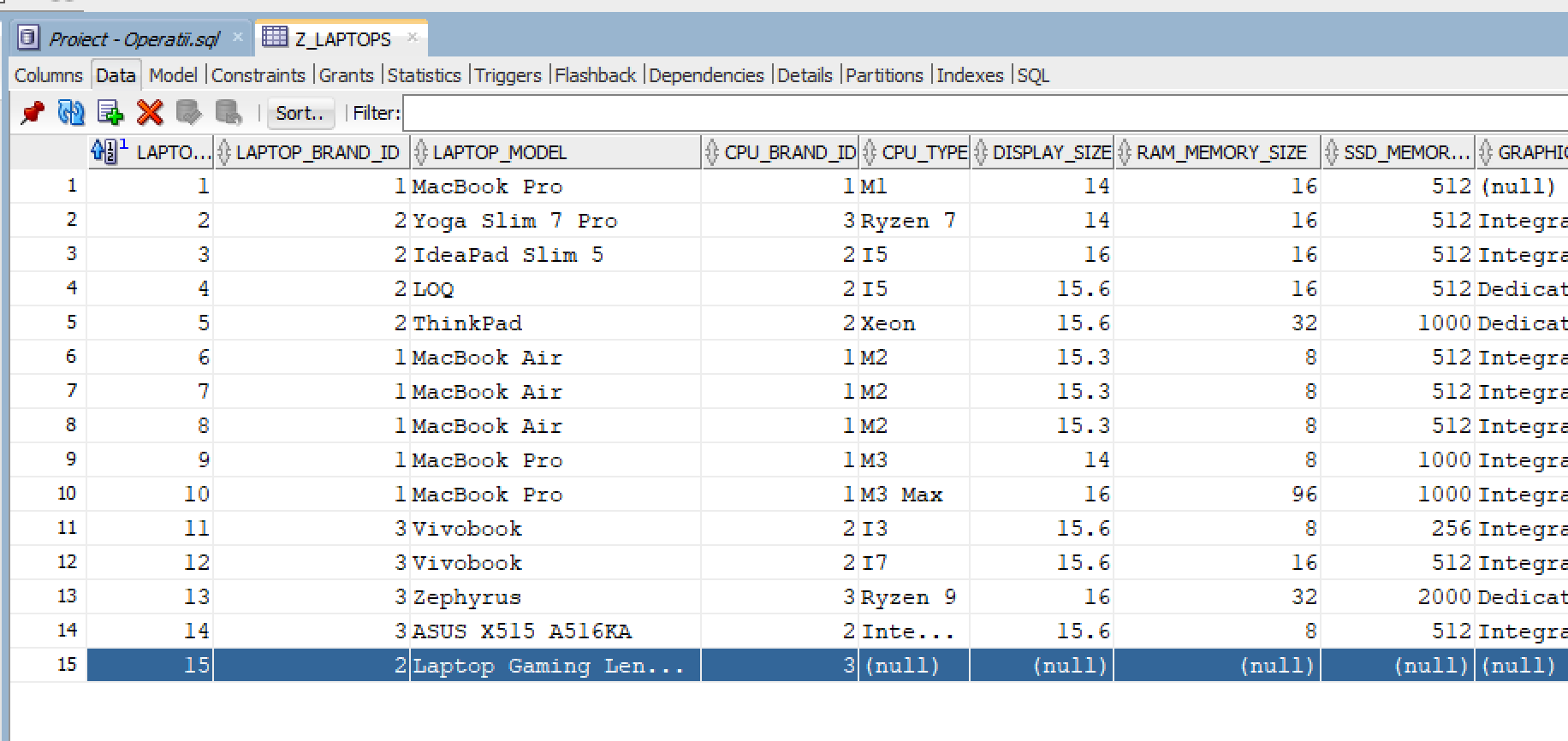
Testare

* Inainte de orice inserare

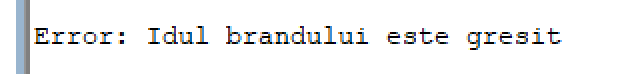


* Test 1 (date corecte)

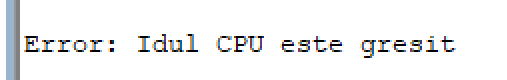




* Test 2 (brand id gresit)



* Test 3 (id cpu gresit)



**4) Functie**

-- Functie care returneaza cantitatea comandata a unui laptop. Idul laptopului este dat ca parametru

-- Test: laptop\_id\_input = 1 (laptopul exista si a fost comandat)

-- Test: laptop\_id\_input = 100 (laptopul nu exista)

/

CREATE OR REPLACE FUNCTION get\_ordered\_quantity (p\_laptop\_id z\_laptops.laptop\_id%TYPE)

RETURN NUMBER

IS

    ordered\_quantity NUMBER;

BEGIN

    SELECT SUM(quantity)

    INTO ordered\_quantity

    FROM z\_order\_lines ol

    WHERE laptop\_id = p\_laptop\_id;

    RETURN ordered\_quantity;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        RETURN NULL;

END;

/

/

ACCEPT laptop\_id\_input PROMPT 'laptop id:'

DECLARE

    v\_laptop\_id z\_laptops.laptop\_id%TYPE := &laptop\_id\_input;

BEGIN

    IF get\_ordered\_quantity(v\_laptop\_id) IS NULL THEN

        DBMS\_OUTPUT.PUT\_LINE('Laptopul introdus nu exista sau nu a fost comandat');

    ELSE

        DBMS\_OUTPUT.PUT\_LINE('Cantitate comandata ' || get\_ordered\_quantity(v\_laptop\_id));

    END IF;

END;

/

Testare

* Input = 1 (laptopul exista si a fost comandat)



* Input = 100 (laptopul nu exista)



**5) Procedura**

-- Procedura: primeste ca parametru numele unui brand si afiseaza toate laptopurile din acest brand, cantitatea comandata si valoarea totala adusa.

/

CREATE OR REPLACE PROCEDURE laptops\_details (p\_brand\_name z\_laptop\_brands.name%TYPE)

IS

    CURSOR c IS

        SELECT

            l.laptop\_id,

            l.laptop\_model,

            SUM(ol.quantity) as ordered\_quantity,

            SUM(ol.price) as total\_value

        FROM z\_laptops l

        JOIN z\_order\_lines ol ON l.laptop\_id = ol.laptop\_id

        JOIN z\_laptop\_brands lb ON l.laptop\_brand\_id = lb.laptop\_brand\_id

        WHERE UPPER(lb.name) LIKE UPPER(p\_brand\_name)

        GROUP BY l.laptop\_id, l.laptop\_model;

    check\_brand\_name z\_laptop\_brands.name%TYPE;

BEGIN

    SELECT name

    INTO check\_brand\_name

    FROM z\_laptop\_brands

    WHERE UPPER(name) LIKE UPPER(p\_brand\_name);

    DBMS\_OUTPUT.PUT\_LINE('\*\*\*\*\*\*\*\* '||UPPER(p\_brand\_name) || ' laptops \*\*\*\*\*\*\*\*');

    DBMS\_OUTPUT.PUT\_LINE('laptop id - laptop model - ordered quantity - total value');

    FOR var IN c LOOP

        DBMS\_OUTPUT.PUT\_LINE(var.laptop\_id ||' - '|| var.laptop\_model ||' - '|| var.ordered\_quantity ||' - '|| var.total\_value);

    END LOOP;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: Nume brand gresit!');

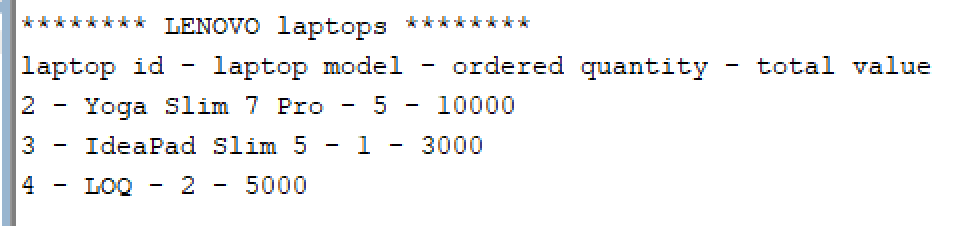
END;

/

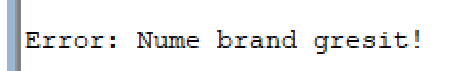
EXECUTE laptops\_details('lenovo');

Testare

* EXECUTE laptops\_details(‘lenovo’);

****

* EXECUTE laptops\_details(‘aaa’);

****

**6) Procedura**

-- Procedura care primeste ca parametrii filtrele: firma si dimensiune

-- si afiseaza laptopurile corespunzatoare

/

CREATE OR REPLACE PROCEDURE brand\_and\_size\_filter (p\_brand\_name z\_laptop\_brands.name%TYPE, p\_display\_size z\_laptops.display\_size%TYPE)

IS

    CURSOR c IS

        SELECT

            l.laptop\_id,

            l.laptop\_model,

            lb.name,

            l.display\_size,

            l.color,

            l.price

        FROM z\_laptops l

        JOIN z\_laptop\_brands lb ON l.laptop\_brand\_id = lb.laptop\_brand\_id

        WHERE

            UPPER(lb.name) LIKE UPPER(p\_brand\_name)

            AND

            l.display\_size = p\_display\_size;

    check\_filters NUMBER;

    NO\_DATA\_FOUND\_FOR\_SPECIFIED\_FILTERS EXCEPTION;

BEGIN

    SELECT COUNT(l.laptop\_id)

    INTO check\_filters

    FROM z\_laptop\_brands lb

    JOIN z\_laptops l ON lb.laptop\_brand\_id = l.laptop\_brand\_id

    WHERE

        UPPER(lb.name) LIKE UPPER(p\_brand\_name)

        AND

        l.display\_size = p\_display\_size;

    IF check\_filters = 0 THEN

        RAISE NO\_DATA\_FOUND\_FOR\_SPECIFIED\_FILTERS;

    END IF;

    DBMS\_OUTPUT.PUT\_LINE(UPPER(p\_brand\_name) || ' laptopuri cu dimensiunea displayului ' || p\_display\_size || ' inci :');

    FOR var IN c LOOP

        DBMS\_OUTPUT.PUT\_LINE(

            var.laptop\_id || ' - ' ||

            var.laptop\_model || ' - ' ||

            var.name || ' - ' ||

            var.display\_size || ' - ' ||

            var.color || ' - ' ||

            var.price

        );

    END LOOP;

EXCEPTION

    WHEN NO\_DATA\_FOUND\_FOR\_SPECIFIED\_FILTERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: Nu s a gasit niciun laptop cu filtrele introduse!');

END;

/

EXECUTE brand\_and\_size\_filter('apple', 15.3);

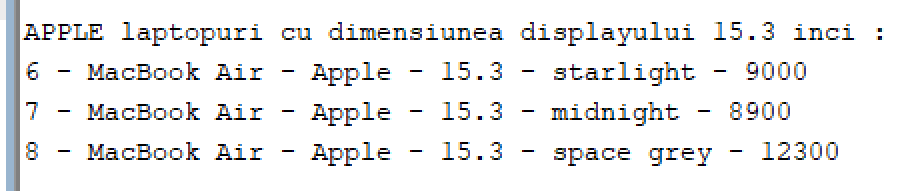
EXECUTE brand\_and\_size\_filter('lenovo', 14);

EXECUTE brand\_and\_size\_filter('aaa', 15.3);

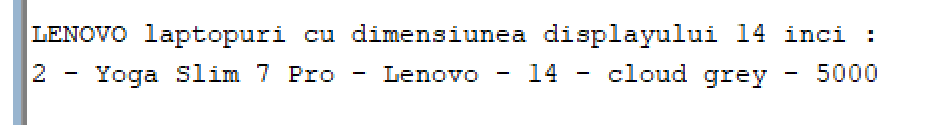
EXECUTE brand\_and\_size\_filter('apple', 100);

Testare

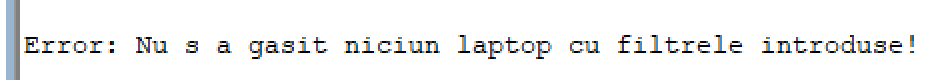
* EXECUTE brand\_and\_size\_filter('apple', 15.3);



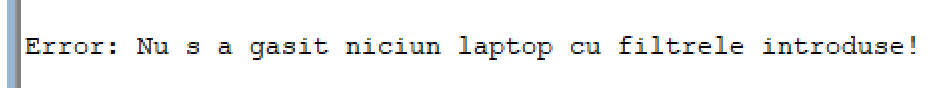
* EXECUTE brand\_and\_size\_filter('lenovo', 14);



* EXECUTE brand\_and\_size\_filter('aaa', 15.3);



* EXECUTE brand\_and\_size\_filter('apple', 100);brand\_and\_size\_filter('apple', 100);



# **Pachete**

**1)**

-- Pachet user\_management care contine:

--    - Functia check\_user\_id. Primeste ca parametru user\_id si verifica daca exista un anumit user,

--      returnand TRUE sau FALSE

--    - Procedura user\_details. Primeste ca parametru user\_id si afiseaza detalii despre user: nume,

--      tara, orasul si laptopurile comandate de acesta (daca exista). Aceasta procedura foloseste

--      functia check\_user\_id pentru a verifica existenta userului.

-- test: input = 1 (userul exista si a comandat laptopuri)

-- test: input = 4 (userul exista, dar nu are comenzi)

-- test: input = 100 (userul nu exista)

/

CREATE OR REPLACE PACKAGE user\_management IS

    FUNCTION check\_user\_id (p\_user\_id z\_users.user\_id%TYPE) RETURN BOOLEAN;

    PROCEDURE user\_details (p\_user\_id z\_users.user\_id%TYPE);

END;

/

/

CREATE OR REPLACE PACKAGE BODY user\_management IS

    FUNCTION check\_user\_id (p\_user\_id z\_users.user\_id%TYPE) RETURN BOOLEAN

    IS

        test z\_users.user\_id%TYPE;

    BEGIN

        SELECT user\_id

        INTO test

        FROM z\_users

        WHERE user\_id = p\_user\_id;

        RETURN TRUE;

    EXCEPTION

        WHEN NO\_DATA\_FOUND THEN

            RETURN FALSE;

        WHEN OTHERS THEN

            RETURN FALSE;

    END;

    PROCEDURE user\_details (p\_user\_id z\_users.user\_id%TYPE)

    IS

        test NUMBER;

        contor NUMBER := 0;

        user\_full\_name VARCHAR2(50);

        country\_name z\_countries.name%TYPE;

        city\_name z\_cities.name%TYPE;

        CURSOR laptops IS

            SELECT

                l.laptop\_model,

                lb.name as brand\_name,

                cb.name as cpu\_name

            FROM z\_laptops l

            JOIN z\_laptop\_brands lb ON l.laptop\_brand\_id = lb.laptop\_brand\_id

            JOIN z\_cpu\_brands cb ON l.cpu\_brand\_id = cb.cpu\_brand\_id

            JOIN z\_order\_lines ol ON l.laptop\_id = ol.laptop\_id

            JOIN z\_laptop\_orders lo ON ol.laptop\_order\_id = lo.laptop\_order\_id

            WHERE lo.user\_id = p\_user\_id;

        WRONG\_USER EXCEPTION;

        USER\_WITH\_NO\_ORDERS EXCEPTION;

    BEGIN

        -- test daca exista userul

        IF check\_user\_id (p\_user\_id) = FALSE THEN

            RAISE WRONG\_USER;

        END IF;

        SELECT

            u.first\_name || ' ' || u.last\_name,

            co.name,

            ci.name

        INTO user\_full\_name, country\_name, city\_name

        FROM z\_users u

        JOIN z\_addresses a ON u.address\_id = a.address\_id

        JOIN z\_countries co ON a.country\_id = co.country\_id

        JOIN z\_cities ci ON a.city\_id = ci.city\_id

        WHERE u.user\_id = p\_user\_id;

        DBMS\_OUTPUT.PUT\_LINE('User: '||user\_full\_name||' din tara '||country\_name||' si orasul '||city\_name);

        -- test daca userul a dat comenzi

        SELECT COUNT(user\_id)

        INTO test

        FROM z\_laptop\_orders

        WHERE user\_id = p\_user\_id;

        IF test = 0 THEN

            RAISE USER\_WITH\_NO\_ORDERS;

        END IF;

        DBMS\_OUTPUT.PUT\_LINE('      a comandat urmatoarele laptopuri:');

        FOR i IN laptops LOOP

            contor := contor + 1;

            DBMS\_OUTPUT.PUT\_LINE('  '||contor||'. '||i.laptop\_model||' - '||i.brand\_name||' - '||i.cpu\_name);

        END LOOP;

    EXCEPTION

        WHEN WRONG\_USER THEN

            DBMS\_OUTPUT.PUT\_LINE('Nu exista userul introdus.');

        WHEN USER\_WITH\_NO\_ORDERS THEN

            DBMS\_OUTPUT.PUT\_LINE('Userul nu a comandat niciun laptop.');

    END;

END;

/

/

ACCEPT input\_user\_id PROMPT 'id user: '

DECLARE

    p\_user\_id z\_users.user\_id%TYPE := &input\_user\_id;

    test BOOLEAN;

BEGIN

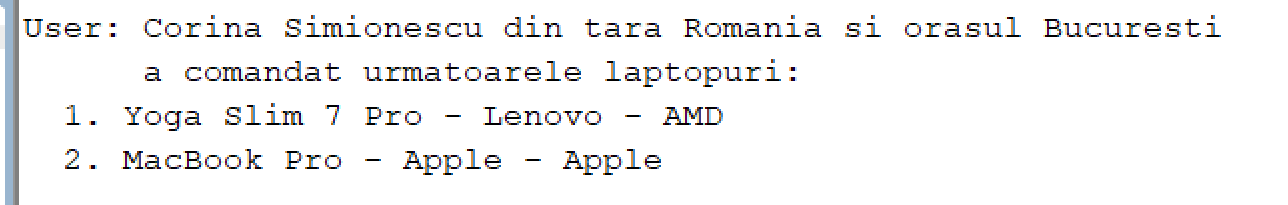
    user\_management.user\_details(p\_user\_id);

END;

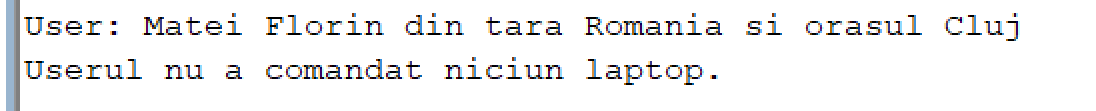
/

Testare

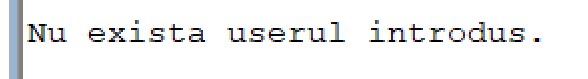
* Input = 1 (userul exista si a comandat laptopuri)



* Input = 4 (userul exista, dar nu are comenzi)



* Input = 1 (userul nu exista)



**2)**

-- 2. Pachet order\_management:

--    - Functia order\_total\_price. Primeste ca parametru order\_id si returneaza pretul total

--      al unei comenzi

--    - Procedura user\_orders. Primeste ca parametru user\_id si afiseaza detalii despre toate

--      comenzile unui user. Se foloseste de functia order\_total\_price pentru a afisa valoarea

--      unei comenzi.

-- test: input = 10 (userul exista si a dat comenzi)

-- test: input = 8 (userul exista, dar nu a dat comenzi)

-- test  input = 100 (userul nu exista)

/

CREATE OR REPLACE PACKAGE order\_management IS

    FUNCTION order\_total\_price (p\_order\_id z\_order\_lines.laptop\_order\_id%TYPE) RETURN NUMBER;

    PROCEDURE user\_orders (p\_user\_id z\_users.user\_id%TYPE);

END;

/

/

CREATE OR REPLACE PACKAGE BODY order\_management IS

    FUNCTION order\_total\_price (p\_order\_id z\_order\_lines.laptop\_order\_id%TYPE) RETURN NUMBER

    IS

        total\_price NUMBER;

    BEGIN

        SELECT SUM(ol.price \* ol.quantity)

        INTO total\_price

        FROM z\_order\_lines ol

        JOIN z\_laptop\_orders lo ON ol.laptop\_order\_id = lo.laptop\_order\_id

        WHERE lo.laptop\_order\_id = p\_order\_id;

        RETURN total\_price;

    END;

    PROCEDURE user\_orders (p\_user\_id z\_users.user\_id%TYPE)

    IS

        CURSOR c IS

            SELECT

                laptop\_order\_id,

                order\_date,

                sm.name AS shipping\_method\_name

            FROM z\_laptop\_orders lo

            JOIN z\_shipping\_methods sm ON lo.shipping\_method\_id = sm.shipping\_method\_id

            WHERE lo.user\_id = p\_user\_id;

        contor NUMBER := 0;

        test1 z\_users.user\_id%TYPE;

        test2 NUMBER;

        USER\_WITH\_NO\_ORDERS EXCEPTION;

    BEGIN

        SELECT user\_id

        INTO test1

        FROM z\_users

        WHERE user\_id = p\_user\_id;

        SELECT COUNT(user\_id)

        INTO test2

        FROM z\_laptop\_orders

        WHERE user\_id = p\_user\_id;

        IF test2 = 0 THEN

            RAISE USER\_WITH\_NO\_ORDERS;

        END IF;

        DBMS\_OUTPUT.PUT\_LINE('Comenzile userului cu idul '||p\_user\_id||':');

        FOR i IN c LOOP

            contor := contor + 1;

            DBMS\_OUTPUT.PUT\_LINE(contor||'. '||'Id comanda: '||i.laptop\_order\_id||', Data comenzii: '||i.order\_date||', Metoda de livrare: '||i.shipping\_method\_name||'.');

            DBMS\_OUTPUT.PUT\_LINE('  -> Valoarea comenzii: '||order\_total\_price(i.laptop\_order\_id));

        END LOOP;

    EXCEPTION

        WHEN NO\_DATA\_FOUND THEN

            DBMS\_OUTPUT.PUT\_LINE('Userul nu exista.');

        WHEN USER\_WITH\_NO\_ORDERS THEN

            DBMS\_OUTPUT.PUT\_LINE('Userul nu a dat comenzi.');

    END;

END;

/

/

ACCEPT input PROMPT 'user id: ';

DECLARE

    v\_user\_id z\_users.user\_id%TYPE := &input;

BEGIN

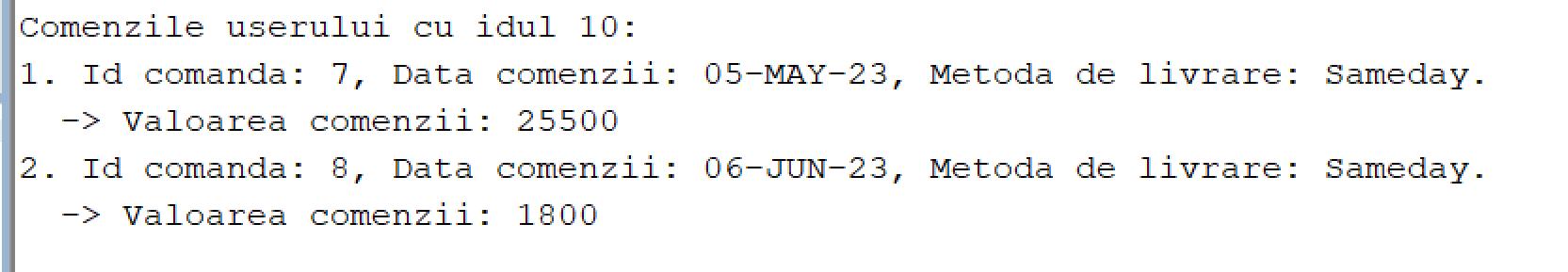
    order\_management.user\_orders(v\_user\_id);

END;

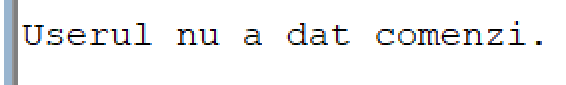
/

Testare

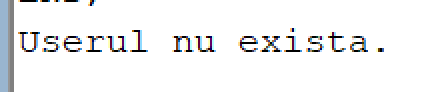
* Input = 10 (userul exista si a dat comenzi)



* Input = 8 (userul exista, dar nu a dat comenzi)



* Input = 100 (userul nu exista)



# **Declansatori**

**1)**

-- In z\_users adaug coloana 'bought\_quantity' in care tin cont de numarul de laptopuri cumparate de fiecare user.

-- Calculez numarul de laptopuri comandate pana acum de fiecare user si il pun in coloana creata.

-- Trigger: atunci cand un user cumpara o anumita cantitate de laptopuri, se modifica valoarea coloanei 'bought\_quantity'.

ALTER TABLE z\_users

ADD bought\_quantity NUMBER DEFAULT 0;

/

DECLARE

    CURSOR c IS

        SELECT u.user\_id, SUM(ol.quantity) AS bought\_quantity\_per\_user

        FROM z\_order\_lines ol

        JOIN z\_laptop\_orders lo ON ol.laptop\_order\_id = lo.laptop\_order\_id

        JOIN z\_users u ON lo.user\_id = u.user\_id

        GROUP BY u.user\_id;

BEGIN

    FOR var IN c LOOP

        UPDATE z\_users

        SET bought\_quantity = var.bought\_quantity\_per\_user

        WHERE user\_id = var.user\_id;

    END LOOP;

END;

/

/

CREATE OR REPLACE TRIGGER trg\_update\_bought\_quantity

BEFORE INSERT ON z\_order\_lines

FOR EACH ROW

DECLARE

    v\_user\_id z\_users.user\_id%TYPE;

BEGIN

    SELECT user\_id

    INTO v\_user\_id

    FROM z\_laptop\_orders

    WHERE laptop\_order\_id = :NEW.laptop\_order\_id;

    UPDATE z\_users

    SET bought\_quantity = bought\_quantity + :NEW.quantity

    WHERE user\_id= v\_user\_id;

END;

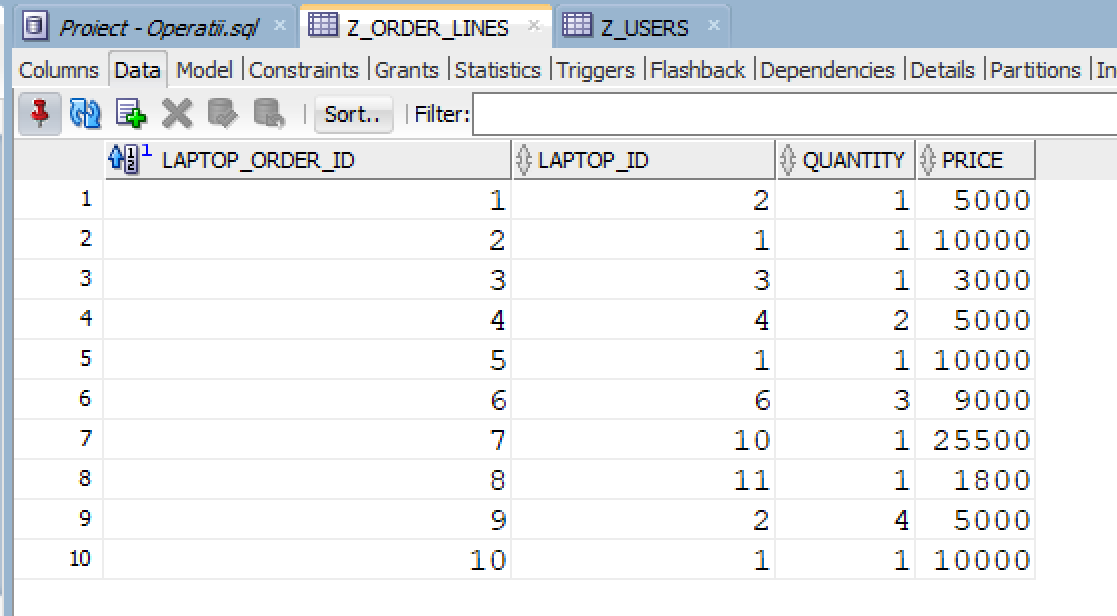
/

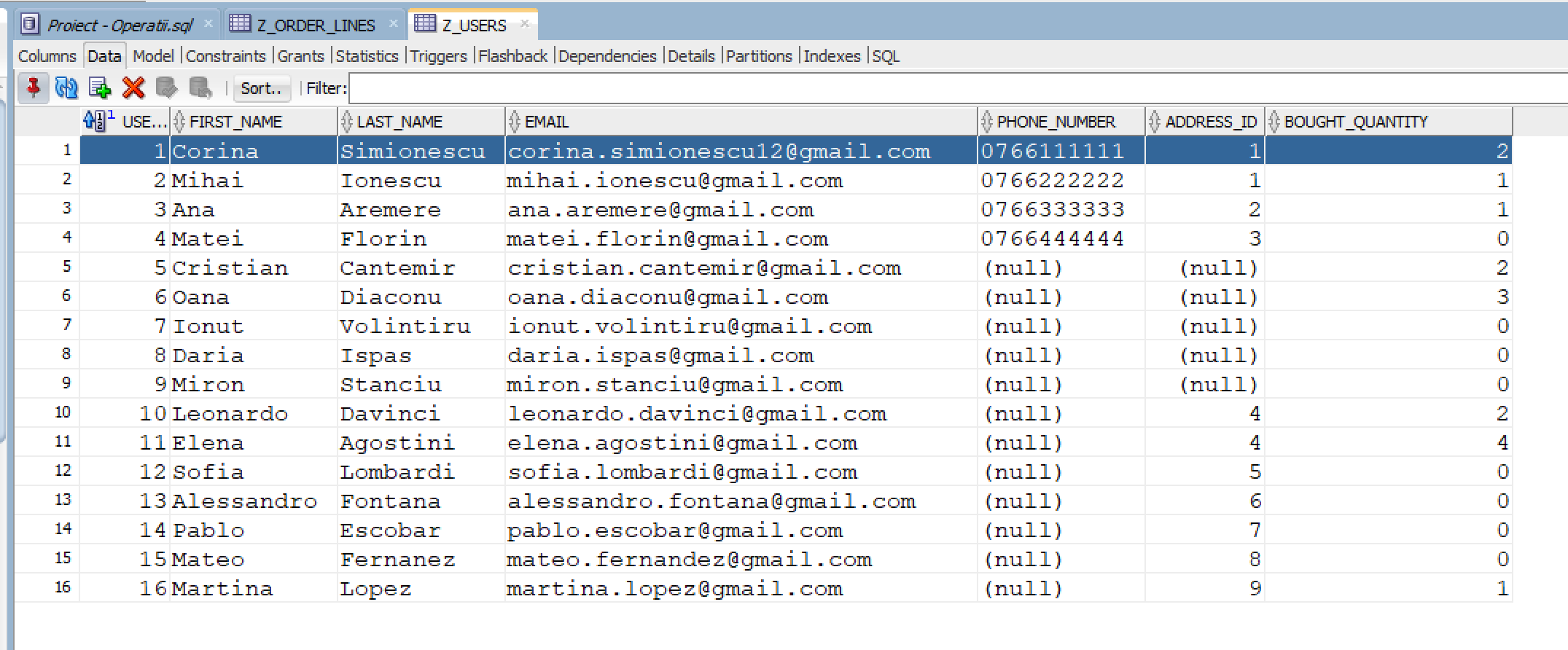
-- Comanda declansatoare

INSERT INTO z\_order\_lines (laptop\_order\_id, laptop\_id, quantity, price)

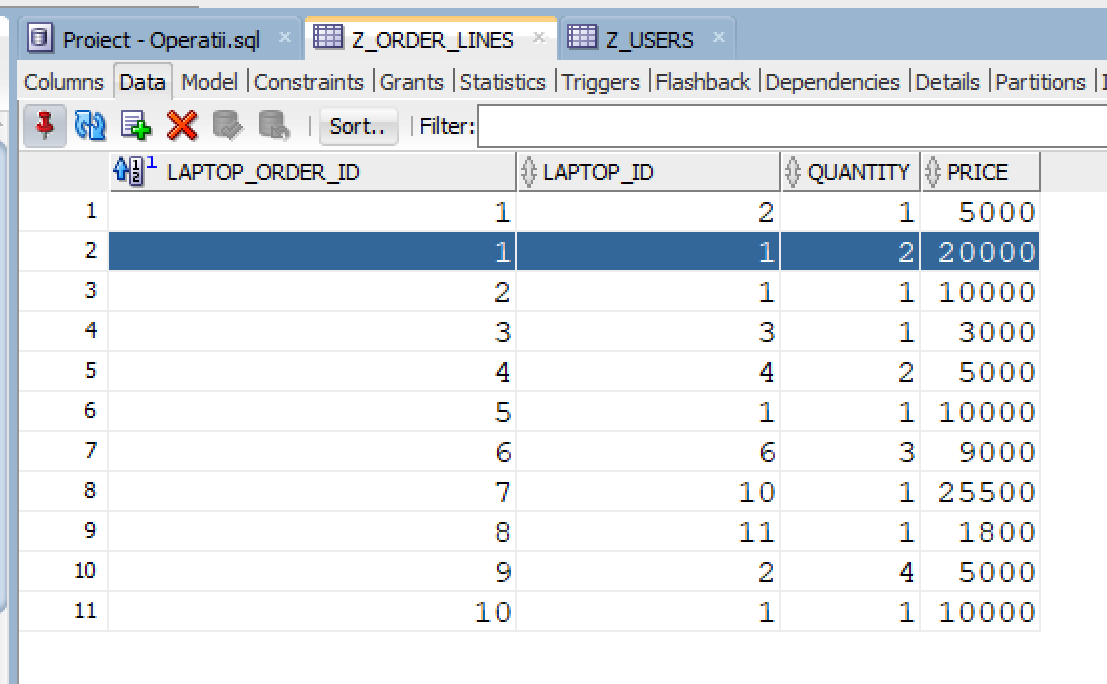
VALUES (1, 1, 2, 20000);

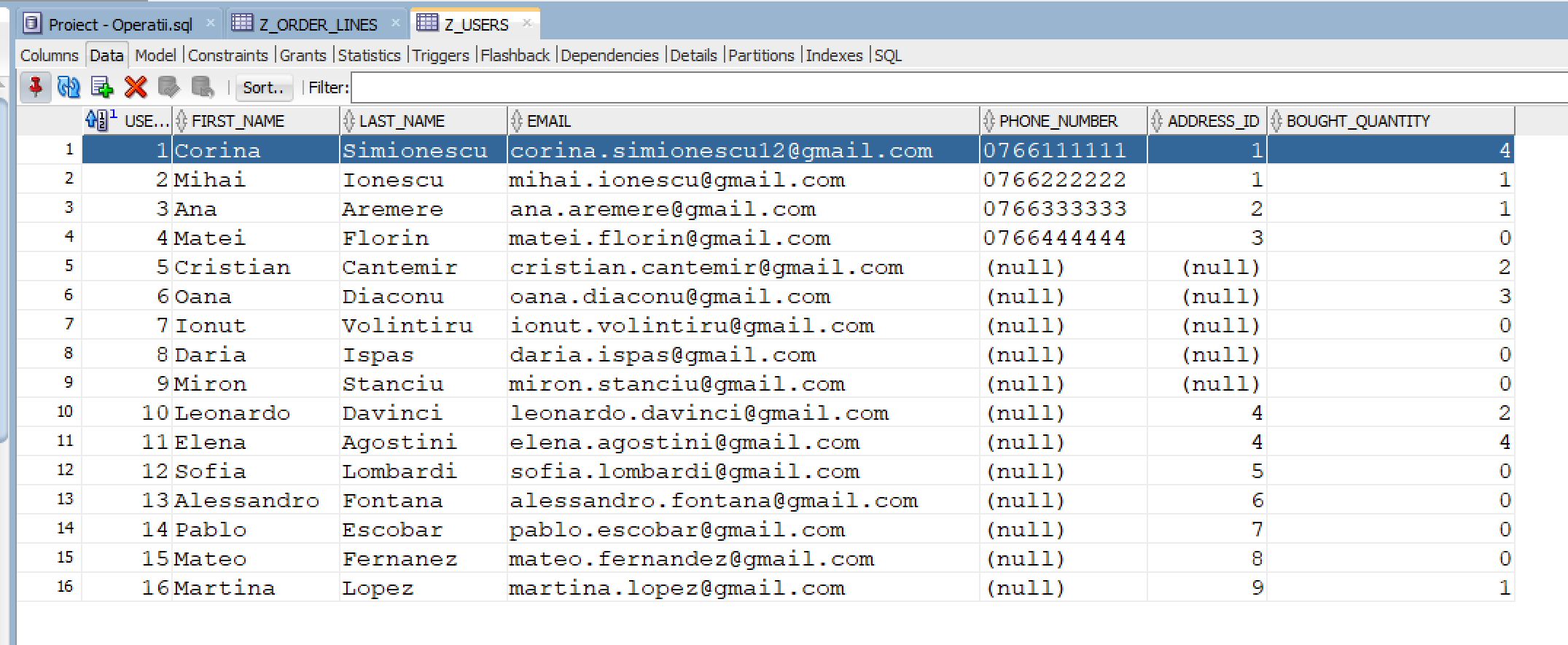
* Inainte de inserare





* Dupa inserare





**2)**

-- In z\_users adaug coloana 'discount' si o initializez cu 0.

-- Trigger: Atunci cand un user are comandate mai mult de 5 laptopuri, i se acorda un discount in valoare de 1000

--          de lei, iar daca userul a comandat mai mult de 10 laptopuri, atunci discontul este 2000 de lei.

ALTER TABLE z\_users

ADD discount NUMBER DEFAULT 0;

/

CREATE OR REPLACE TRIGGER trg\_give\_discount

BEFORE INSERT ON z\_order\_lines

FOR EACH ROW

DECLARE

    v\_user\_id z\_users.user\_id%TYPE;

    total\_bought\_quantity NUMBER;

BEGIN

    SELECT user\_id

    INTO v\_user\_id

    FROM z\_laptop\_orders

    WHERE laptop\_order\_id = :NEW.laptop\_order\_id;

    SELECT SUM(quantity) + :NEW.quantity

    INTO total\_bought\_quantity

    FROM z\_order\_lines

    WHERE laptop\_order\_id IN (

        SELECT laptop\_order\_id

        FROM z\_laptop\_orders

        WHERE user\_id = v\_user\_id

    );

    IF total\_bought\_quantity >= 10 THEN

        UPDATE z\_users

        SET discount = 2000

        WHERE user\_id = v\_user\_id;

    ELSIF total\_bought\_quantity >= 5 THEN

        UPDATE z\_users

        SET discount = 1000

        WHERE user\_id = v\_user\_id;

    END IF;

END;

/

-- Test pentru discount = 1000

INSERT INTO z\_laptop\_orders (laptop\_order\_id, user\_id, order\_date, shipping\_method\_id, status)

VALUES (11, 1, SYSDATE, 1, 'Comanda este in procesare.');

--Comanda declansatoare

INSERT INTO z\_order\_lines (laptop\_order\_id, laptop\_id, quantity, price)

VALUES (11, 1, 3, 30000);

--Test pentru discount = 2000

INSERT INTO z\_laptop\_orders (laptop\_order\_id, user\_id, order\_date, shipping\_method\_id, status)

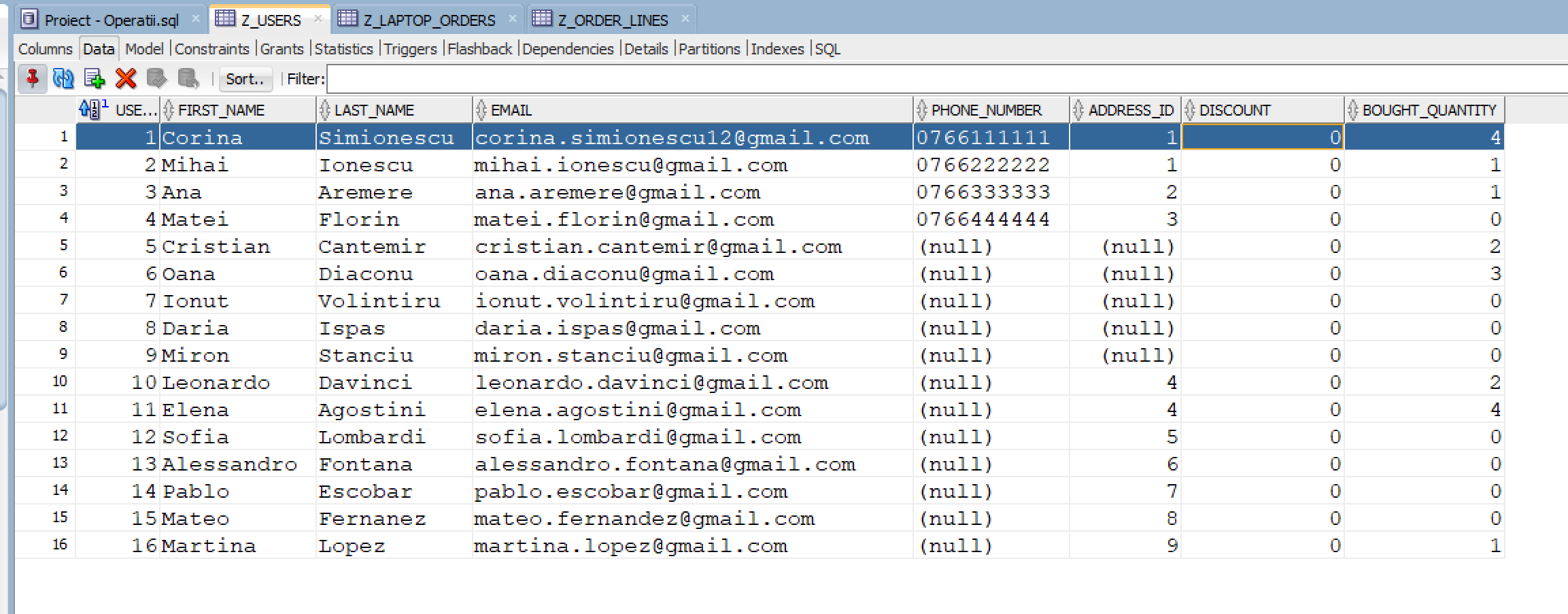
VALUES (12, 3, SYSDATE, 1, 'Comanda este in procesare.');

--Comanda declansatoare

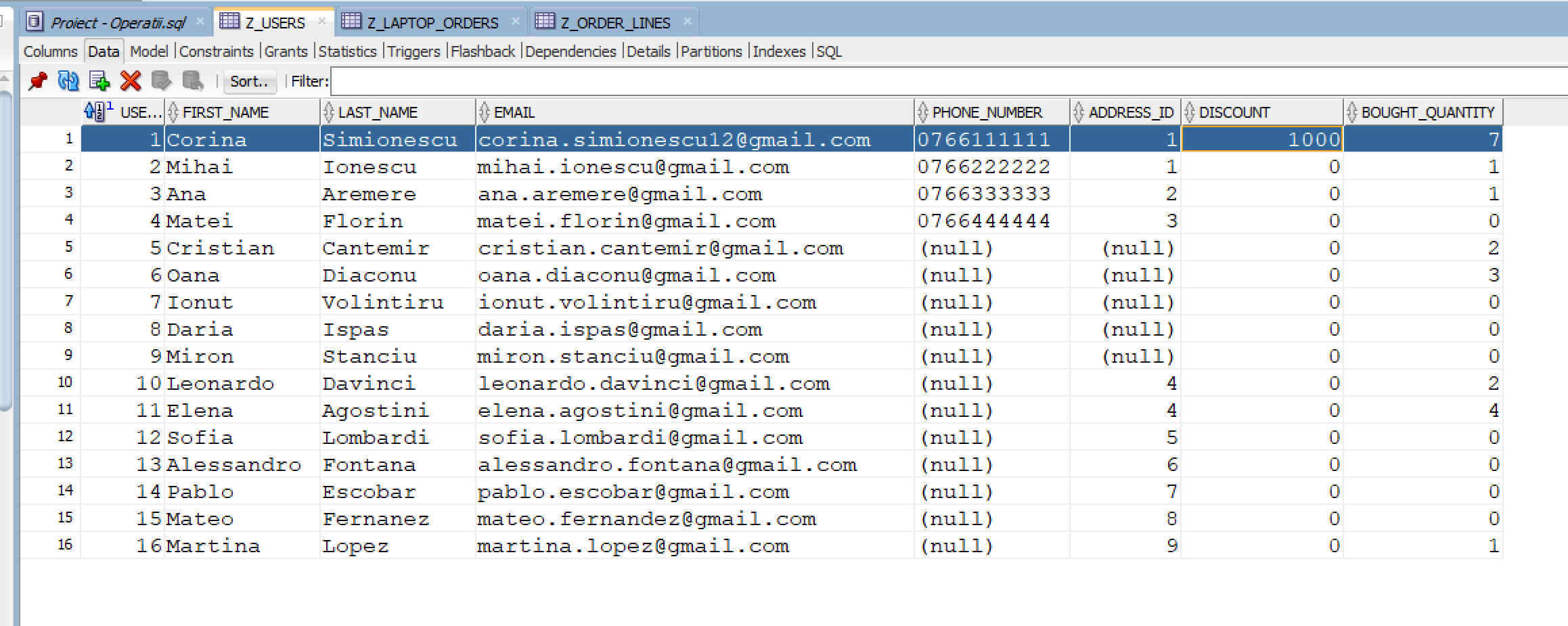
INSERT INTO z\_order\_lines (laptop\_order\_id, laptop\_id, quantity, price)

VALUES (12, 14, 10, 12000);

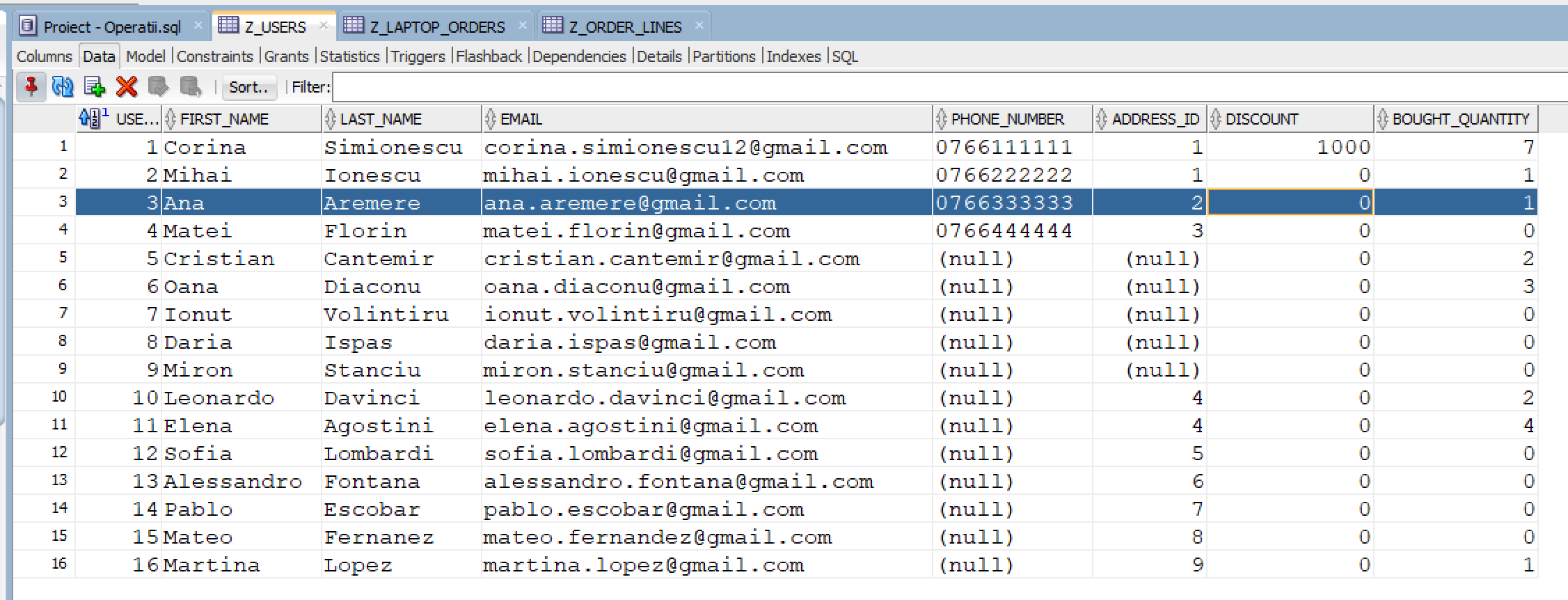
* Test pentru discount = 1000
  + Inaite de inserare



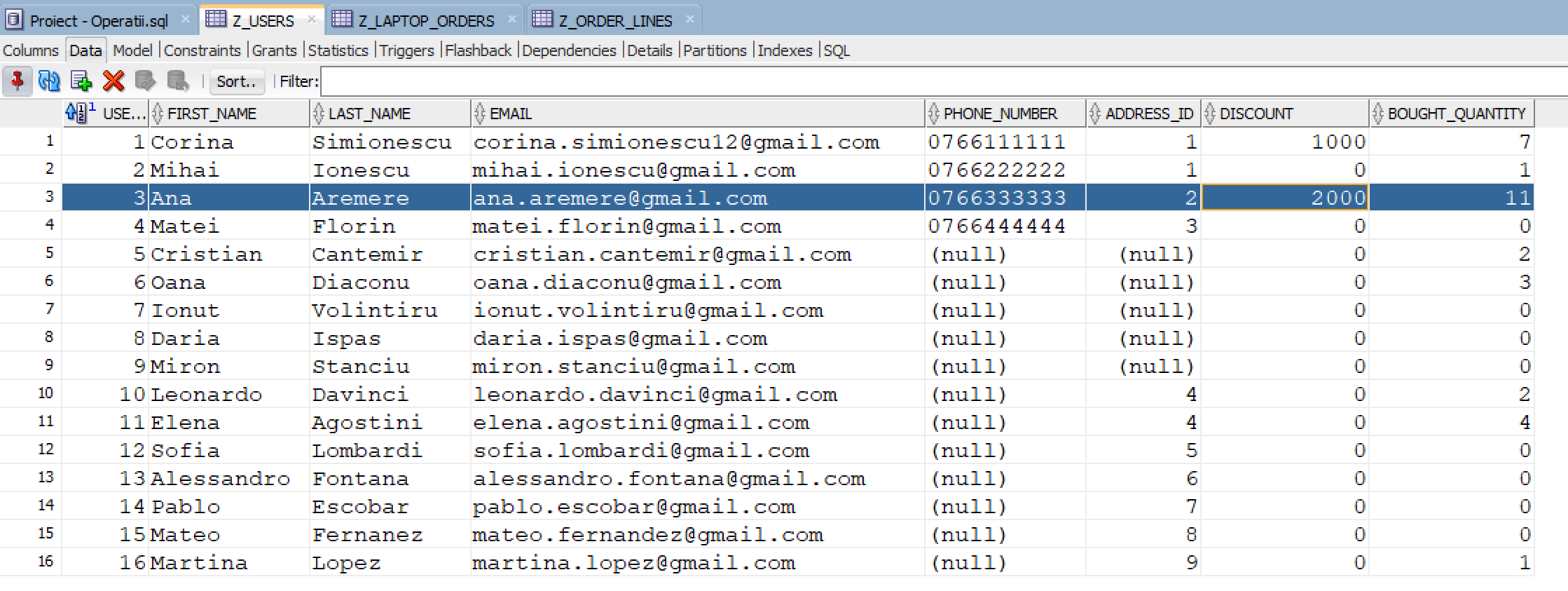
* + Dupa inserare



* Test pentru discount = 2000
  + Inainte de inserare



* Dupa inserare



**3)**

-- In tabela z\_laptops adaug coloana 'quantity\_sold'. Pentru fiecare laptop initializez coloana cu cantitatea

-- vanduta pana in acest moment.

-- Trigger: Cand se comanda un laptop, se schimba valoarea coloanei quantity\_sold, adaugand cantitatea comandata

ALTER TABLE z\_laptops

ADD quantity\_sold NUMBER;

/

DECLARE

    CURSOR c IS

        SELECT l.laptop\_id, SUM(ol.quantity) AS quantity\_sold\_per\_laptop

        FROM z\_order\_lines ol

        RIGHT JOIN z\_laptops l ON ol.laptop\_id = l.laptop\_id

        GROUP BY l.laptop\_id;

BEGIN

    FOR var IN c LOOP

        DBMS\_OUTPUT.PUT\_LINE(var.laptop\_id ||' - '||var.quantity\_sold\_per\_laptop);

        IF var.quantity\_sold\_per\_laptop IS NULL THEN

            UPDATE z\_laptops

            SET quantity\_sold = 0

            WHERE  laptop\_id = var.laptop\_id;

        ELSE

            UPDATE z\_laptops

            SET quantity\_sold = var.quantity\_sold\_per\_laptop

            WHERE laptop\_id = var.laptop\_id;

        END IF;

    END LOOP;

END;

/

/

CREATE OR REPLACE TRIGGER trg\_raise\_quantity\_sold

AFTER INSERT ON z\_order\_lines

FOR EACH ROW

DECLARE

BEGIN

    UPDATE z\_laptops

    SET quantity\_sold = quantity\_sold + :NEW.quantity

    WHERE laptop\_id = :NEW.laptop\_id;

END;

/

INSERT INTO z\_laptop\_orders (laptop\_order\_id, user\_id, order\_date, shipping\_method\_id, status)

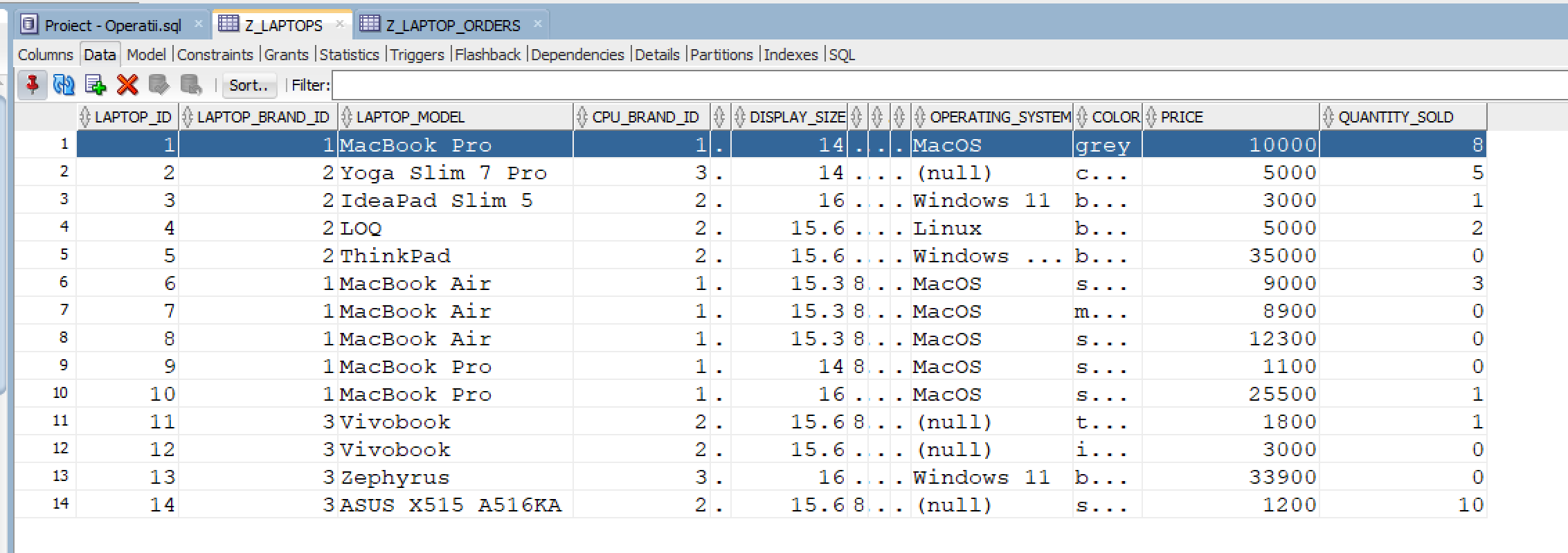
VALUES (13, 2, SYSDATE, 1, 'Coamnda este in procesare.');

-- Coamanda declansatoare

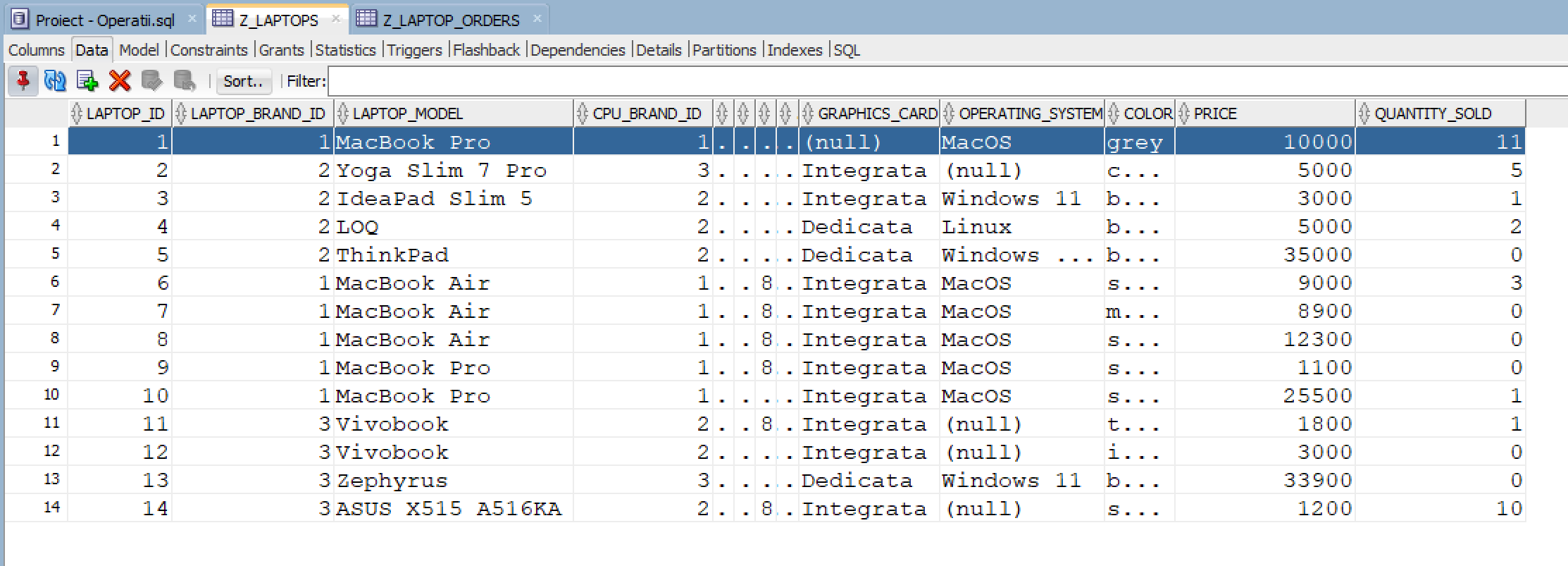
INSERT INTO z\_order\_lines (laptop\_order\_id, laptop\_id, quantity, price)

VALUES (13, 1, 3, 30000);

* Inainte de inserare



* Dupa inserare



**4)**

-- Trigger: Atunci cand inserez in z\_order\_lines, pretul sa fie calculat automat, fiind egal cu pretul laptopului

--          \* cantitatea.

/

CREATE OR REPLACE TRIGGER trg\_calculate\_price

BEFORE INSERT ON z\_order\_lines

FOR EACH ROW

DECLARE

    laptop\_price z\_laptops.price%TYPE;

BEGIN

    SELECT price

    INTO laptop\_price

    FROM z\_laptops

    WHERE laptop\_id = :NEW.laptop\_id;

    :NEW.price := :NEW.quantity \* laptop\_price;

END;

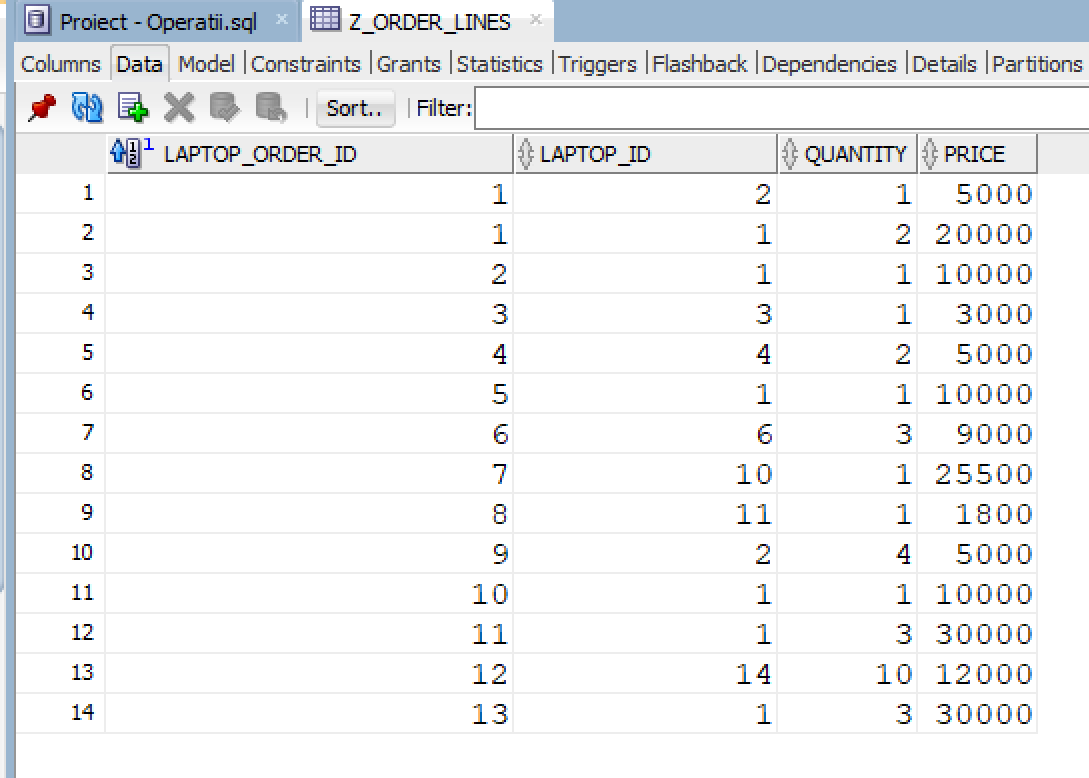
/

-- Comanda declansatoare

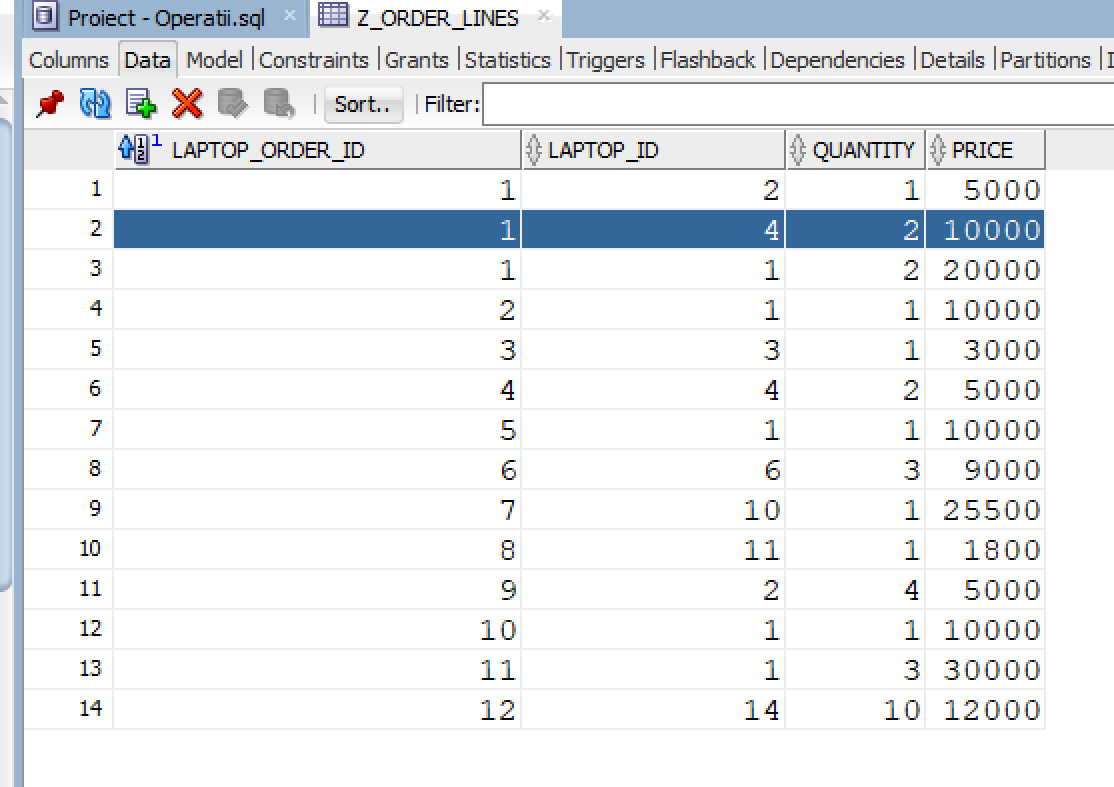
INSERT INTO z\_order\_lines (laptop\_order\_id, laptop\_id, quantity, price)

VALUES (1, 4, 2, 1); -- aici pretul ar trebui sa se calculeze automat si sa fie 2 \* 5000 = 10000 lei

* Inaite de inserare



* Dupa inserare



**5)**

-- Trigger: daca schimb idul unei metode de livrare din tabela z\_shipping\_methods, se schimba idul metodei de

--          livrare si in z\_laptop\_orders

/

CREATE OR REPLACE TRIGGER trg\_change\_shipping\_method\_name\_everywhere

AFTER UPDATE OF shipping\_method\_id ON z\_shipping\_methods

FOR EACH ROW

DECLARE

BEGIN

    UPDATE z\_laptop\_orders

    SET shipping\_method\_id = :NEW.shipping\_method\_id

    WHERE shipping\_method\_id = :OLD.shipping\_method\_id;

END;

/

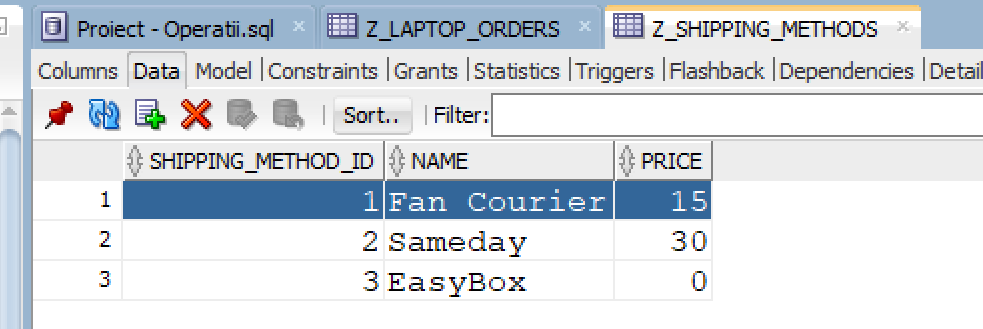
-- Comanda declansatoare

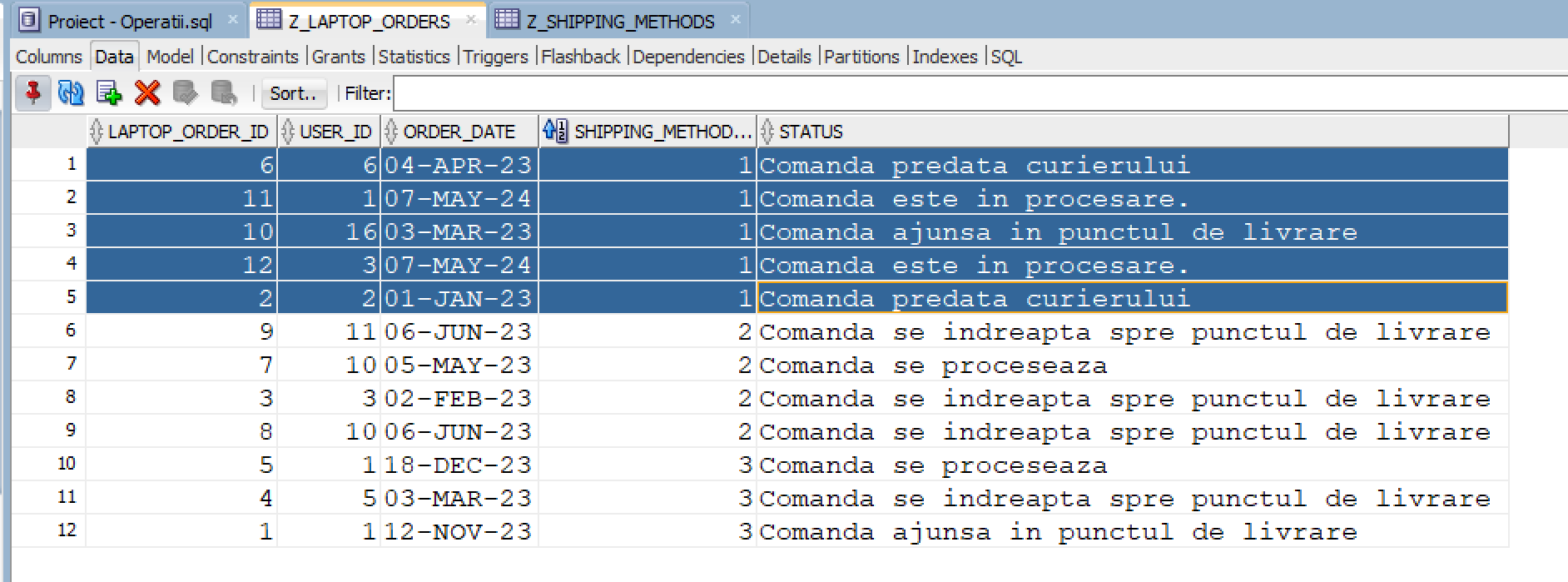
UPDATE z\_shipping\_methods

SET shipping\_method\_id = 9

WHERE shipping\_method\_id = 1;

* Inainte de executia comenzii declansatoare





* Dupa executia comenzii declansatoare

