Proiect la disciplina “Sisteme de gestiune a bazelor de date”

Hardware Haven

Nume și grupă:

- Sasu Alexandru-Cristian

- Grupa 242

1. Prezentarea bazei de date:

Modelul de date va gestiona informații despre un magazin care se ocupă cu vânzarea componentelor, accesoriilor, și serviciilor destinate calculatoarelor și laptop-urilor. Serviciile oferite constau în: diagnosticarea, montarea, asamblarea, și testarea produselor. În afara celor precizate anterior, magazinul mai are inclus în oferta sa atât sisteme deja asamblate, precum: calculatoare și laptop-uri.

Există mai multe categorii de produse, fiecare produs aparținându-i unei singure categorii.

Magazinul deține și un website, prin intermediul căruia clienții își pot crea unul sau mai multe conturi, cu scopul de a explora ofertele magazinului, de a plasa comenzi, de a comunica cu angajații în legătură cu anumite nelămuriri sau eventuale sfaturi, cât și de a adăuga recenzii produselor cumpărate de aceștia.

Magazinul are două tipuri de angajați: paznici și tehnicieni. De asemenea, angajații pot asista clienții în legătură cu diferite întrebări ale acestora, atât în incinta magazinului, cât și prin intermediul website-ului.

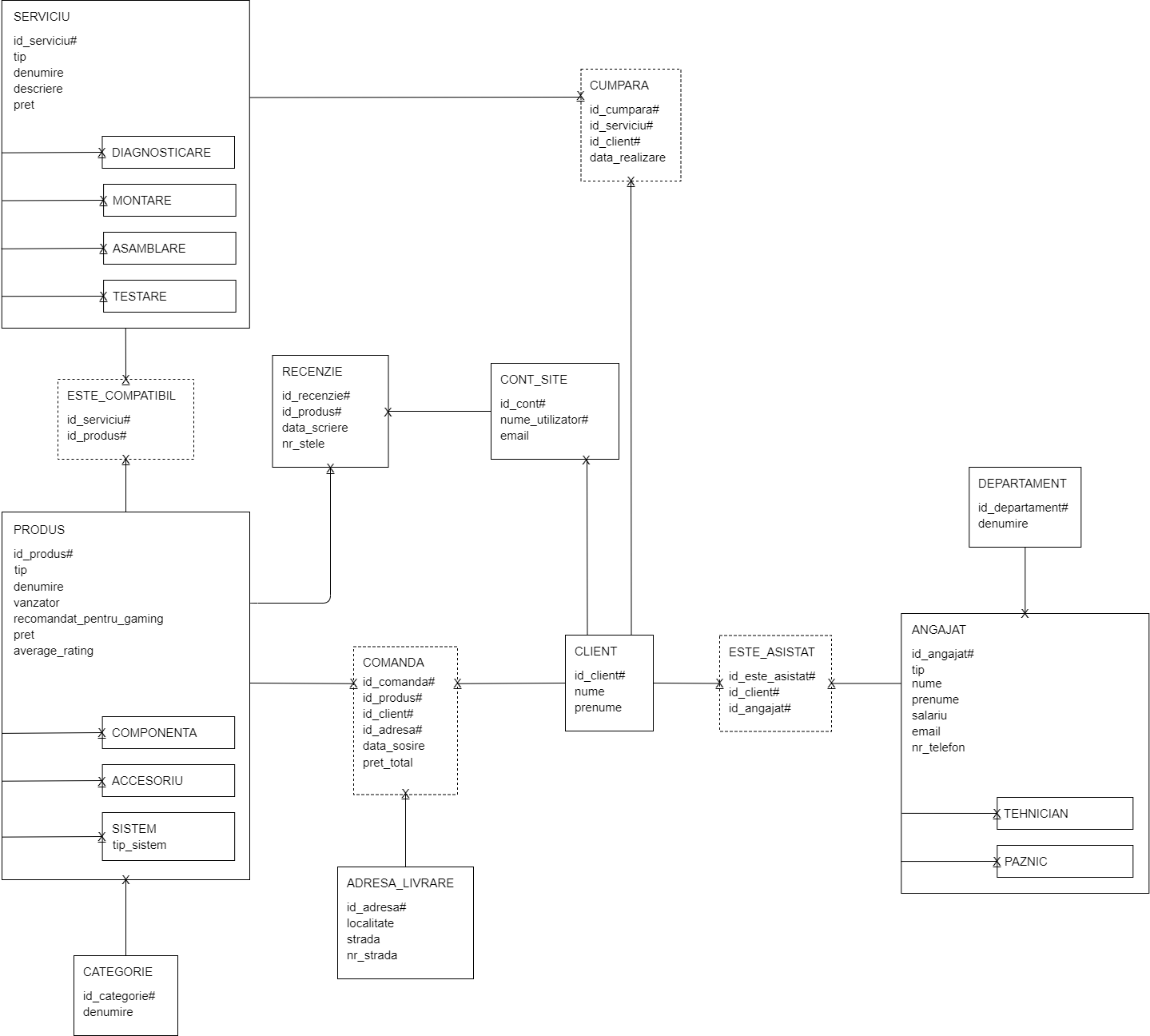
Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului:

* Un client poate șterge doar recenziile scrise de acesta;
* Un client nu poate să adauge mai mult de o recenzie pentru același produs;
* Atât tehnicienii cât și paznicii lucrează într-un singur department, iar paznicii lucrează Într-un singur tip de departament, anume cel de securitate;
* Clienții nu își pot șterge conturile;
* Nu există conturi cu același nume de utilizator;
* Un produs trebuie să facă parte dintr-o singură categorie;
* Orice cont trebuie să aibă specificată o adresă de e-mail;
* Orice angajat trebuie să aibă specificată o adresă de e-mail și un număr de telefon;
* Trebuie să se cunoască data sosirii și prețul total al tuturor comenzilor;
* Trebuie să se cunoască data realizării tuturor serviciilor.

1. Diagrama entitate-relație:



1. Diagrama conceptuală:



1. Crearea tabelelor:

CREATE TABLE CATEGORIE

(id\_categorie INTEGER,

denumire VARCHAR2(40) CONSTRAINT denumire\_categorie NOT NULL,

CONSTRAINT pk\_categorie PRIMARY KEY(id\_categorie),

CONSTRAINT u\_denumire\_categorie UNIQUE(denumire)

);

CREATE TABLE PRODUS

(id\_produs INTEGER,

id\_categorie INTEGER CONSTRAINT id\_categorie\_produs NOT NULL,

tip VARCHAR(10) CONSTRAINT tip\_produs NOT NULL,

denumire VARCHAR2(200) CONSTRAINT denumire\_produs NOT NULL,

tip\_sistem VARCHAR2(30),

vanzator VARCHAR2(40) CONSTRAINT vanzator\_produs NOT NULL,

recomandat\_pentru\_gaming VARCHAR2(2),

pret NUMBER CONSTRAINT pret\_produs NOT NULL,

average\_rating NUMBER,

CONSTRAINT pk\_produs PRIMARY KEY(id\_produs),

CONSTRAINT fk\_categorie\_in\_produs FOREIGN KEY(id\_categorie) REFERENCES CATEGORIE(id\_categorie) ON DELETE CASCADE

);

CREATE TABLE SERVICIU

(id\_serviciu INTEGER,

tip VARCHAR(13) CONSTRAINT tip\_serviciu NOT NULL,

denumire VARCHAR2(70) CONSTRAINT denumire\_serviciu NOT NULL,

descriere VARCHAR2(1000),

pret NUMBER CONSTRAINT pret\_serviciu NOT NULL,

CONSTRAINT pk\_serviciu PRIMARY KEY(id\_serviciu),

CONSTRAINT u\_denumire\_serviciu UNIQUE(denumire)

);

CREATE TABLE ADRESA\_LIVRARE

(id\_adresa INTEGER,

localitate VARCHAR2(30) CONSTRAINT localitate\_adresa\_livrare NOT NULL,

strada VARCHAR2(80) CONSTRAINT strada\_adresa\_livrare NOT NULL,

nr\_strada INTEGER CONSTRAINT nr\_strada\_adresa\_livrare NOT NULL,

CONSTRAINT pk\_adresa\_livrare PRIMARY KEY(id\_adresa)

);

CREATE TABLE CLIENT

(id\_client INTEGER,

nume VARCHAR2(40) CONSTRAINT nume\_client NOT NULL,

prenume VARCHAR2(40) CONSTRAINT prenume\_client NOT NULL,

CONSTRAINT pk\_client PRIMARY KEY(id\_client)

);

CREATE TABLE CONT\_SITE

(id\_cont INTEGER,

nume\_utilizator VARCHAR2(40),

id\_client INTEGER,

email VARCHAR2(40) CONSTRAINT email NOT NULL,

CONSTRAINT pk\_compus\_cont\_site PRIMARY KEY(id\_cont, nume\_utilizator),

CONSTRAINT u\_nume\_utilizator UNIQUE(nume\_utilizator),

CONSTRAINT fk\_client\_in\_cont\_site FOREIGN KEY(id\_client) REFERENCES CLIENT(id\_client) ON DELETE CASCADE

);

CREATE TABLE RECENZIE

(id\_recenzie INTEGER,

id\_produs INTEGER CONSTRAINT id\_produs\_recenzie NOT NULL,

nume\_utilizator VARCHAR2(40) CONSTRAINT nume\_utilizator\_recenzie NOT NULL,

data\_scriere DATE CONSTRAINT data\_scriere\_recenzie NOT NULL,

nr\_stele NUMBER,

CONSTRAINT pk\_compus\_recenzie PRIMARY KEY(id\_recenzie, id\_produs),

CONSTRAINT fk\_produs\_in\_recenzie FOREIGN KEY(id\_produs) REFERENCES PRODUS(id\_produs) ON DELETE CASCADE,

CONSTRAINT fk\_cont\_site\_in\_recenzie FOREIGN KEY(nume\_utilizator) REFERENCES CONT\_SITE(nume\_utilizator) ON DELETE CASCADE

);

CREATE TABLE DEPARTAMENT

(id\_departament INTEGER,

denumire VARCHAR2(30) CONSTRAINT denumire\_departament NOT NULL,

CONSTRAINT pk\_departament PRIMARY KEY(id\_departament),

CONSTRAINT u\_denumire\_departament UNIQUE(denumire)

);

CREATE TABLE ANGAJAT

(id\_angajat INTEGER,

id\_departament INTEGER,

tip VARCHAR2(9) CONSTRAINT tip\_angajat NOT NULL,

nume VARCHAR2(30) CONSTRAINT nume\_angajat NOT NULL,

prenume VARCHAR2(30) CONSTRAINT prenume\_angajat NOT NULL,

salariu NUMBER CONSTRAINT salariu\_angajat NOT NULL,

email VARCHAR2(40) CONSTRAINT email\_angajat NOT NULL,

nr\_telefon VARCHAR2(10) CONSTRAINT nr\_telefon\_angajat NOT NULL,

CONSTRAINT pk\_angajat PRIMARY KEY(id\_angajat),

CONSTRAINT fk\_departament\_in\_angajat FOREIGN KEY(id\_departament) REFERENCES DEPARTAMENT(id\_departament) ON DELETE CASCADE,

CONSTRAINT u\_email\_angajat UNIQUE(email)

);

CREATE TABLE ESTE\_COMPATIBIL

(id\_serviciu INTEGER,

id\_produs INTEGER,

CONSTRAINT pk\_compus\_este\_compatibil PRIMARY KEY(id\_serviciu, id\_produs),

CONSTRAINT fk\_serviciu\_in\_este\_compatibil FOREIGN KEY(id\_serviciu) REFERENCES SERVICIU(id\_serviciu) ON DELETE CASCADE,

CONSTRAINT fk\_produs\_in\_este\_compatibil FOREIGN KEY(id\_produs) REFERENCES PRODUS(id\_produs) ON DELETE CASCADE

);

CREATE TABLE COMANDA

(id\_comanda INTEGER,

id\_produs INTEGER,

id\_client INTEGER,

id\_adresa INTEGER,

data\_sosire DATE CONSTRAINT data\_sosire\_comanda NOT NULL,

pret\_total NUMBER CONSTRAINT pret\_total\_comanda NOT NULL,

CONSTRAINT pk\_compus\_comanda PRIMARY KEY(id\_comanda, id\_produs, id\_client, id\_adresa),

CONSTRAINT fk\_produs\_in\_comanda FOREIGN KEY(id\_produs) REFERENCES PRODUS(id\_produs) ON DELETE CASCADE,

CONSTRAINT fk\_client\_in\_comanda FOREIGN KEY(id\_client) REFERENCES CLIENT(id\_client) ON DELETE CASCADE,

CONSTRAINT fk\_adresa\_livrare\_in\_comanda FOREIGN KEY(id\_adresa) REFERENCES ADRESA\_LIVRARE(id\_adresa) ON DELETE CASCADE

);

CREATE TABLE CUMPARA

(id\_cumpara INTEGER,

id\_serviciu INTEGER,

id\_client INTEGER,

data\_realizare DATE CONSTRAINT data\_realizare\_cumpara NOT NULL,

CONSTRAINT pk\_compus\_cumpara PRIMARY KEY(id\_cumpara, id\_serviciu, id\_client),

CONSTRAINT fk\_serviciu\_in\_cumpara FOREIGN KEY(id\_serviciu) REFERENCES SERVICIU(id\_serviciu) ON DELETE CASCADE,

CONSTRAINT fk\_client\_in\_cumpara FOREIGN KEY(id\_client) REFERENCES CLIENT(id\_client) ON DELETE CASCADE

);

CREATE TABLE ESTE\_ASISTAT

(id\_este\_asistat INTEGER,

id\_client INTEGER,

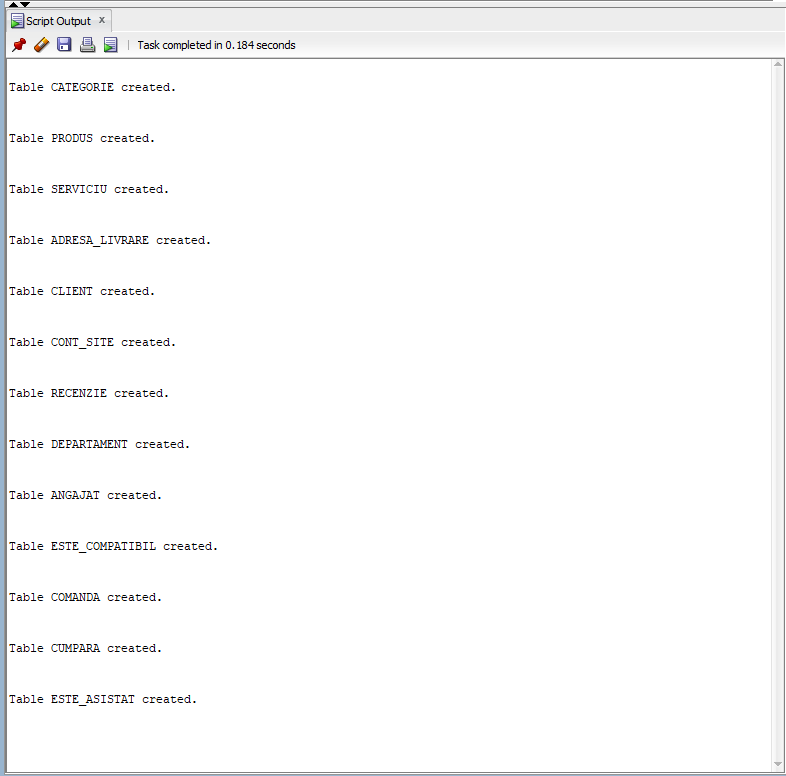
id\_angajat INTEGER,

CONSTRAINT pk\_compus\_este\_asistat PRIMARY KEY(id\_este\_asistat, id\_client, id\_angajat),

CONSTRAINT fk\_client\_in\_este\_asistat FOREIGN KEY(id\_client) REFERENCES CLIENT(id\_client) ON DELETE CASCADE,

CONSTRAINT fk\_angajat\_in\_este\_asistat FOREIGN KEY(id\_angajat) REFERENCES ANGAJAT(id\_angajat) ON DELETE CASCADE

);



1. Inserarea datelor în tabele:

-- Secventa pentru inserarea inregistrarilor in tabelul "CATEGORIE":

CREATE SEQUENCE SEQ\_CATEGORIE

INCREMENT by 1

START WITH 0

MINVALUE -1

MAXVALUE 399

NOCYCLE;

-- DROP SEQUENCE SEQ\_CATEGORIE;

INSERT INTO CATEGORIE

VALUES(SEQ\_CATEGORIE.NEXTVAL, 'Placi video');

INSERT INTO CATEGORIE

VALUES(SEQ\_CATEGORIE.NEXTVAL, 'Procesoare');

INSERT INTO CATEGORIE

VALUES(SEQ\_CATEGORIE.NEXTVAL, 'Memorii');

INSERT INTO CATEGORIE

VALUES(SEQ\_CATEGORIE.NEXTVAL, 'HDD');

INSERT INTO CATEGORIE

VALUES(SEQ\_CATEGORIE.NEXTVAL, 'Adaptoare');

INSERT INTO CATEGORIE

VALUES(SEQ\_CATEGORIE.NEXTVAL, 'PC-uri');

INSERT INTO CATEGORIE

VALUES(SEQ\_CATEGORIE.NEXTVAL, 'Laptop-uri');

-- ROLLBACK;

-- COMMIT;

-- Secventa pentru inserarea inregistrarilor in tabelul "PRODUS":

CREATE SEQUENCE SEQ\_PRODUS

INCREMENT by 5

START WITH 0

MINVALUE -1

NOCYCLE;

-- DROP SEQUENCE SEQ\_PRODUS;

INSERT INTO PRODUS

VALUES(SEQ\_PRODUS.NEXTVAL, 1, 'Componenta', 'Placa video Palit GeForce RTX 3090 GamingPro 24GB GDDR6X 384-bit', NULL, 'GIGABYTE', 'Da', 14999.99, 5);

INSERT INTO PRODUS

VALUES(SEQ\_PRODUS.NEXTVAL, 2, 'Componenta', 'Procesor AMD Ryzen 5 3600 3.6GHz box', NULL, 'AMD', 'Da', 999.99, 5);

INSERT INTO PRODUS

VALUES(SEQ\_PRODUS.NEXTVAL, 5, 'Accesoriu', 'Adaptor Gembird 1x HDMI 1.4 Male - 1x VGA Female', NULL, 'Gembird', 'Nu', 35.99, NULL);

INSERT INTO PRODUS

VALUES(SEQ\_PRODUS.NEXTVAL, 6, 'Sistem', 'PC Gaming Raptor5, Intel i5-9400F 2.9GHz Coffee Lake, 16GB DDR4, 960GB SSD, RX 5600 XT 6GB GDDR6, Iluminare RGB', 'PC', 'Raptor5', 'Da', 4799.99, 4);

INSERT INTO PRODUS

VALUES(SEQ\_PRODUS.NEXTVAL, 7, 'Sistem', 'Laptop Lenovo 15.6'' ThinkPad E15 Gen 2, FHD, Procesor Intel® Core™ i5-1135G7 (8M Cache, up to 4.20 GHz), 8GB DDR4, 256GB SSD, Intel Iris Xe, No OS, Black', 'Lenovo', 'Laptop', 'Nu', 3398.99, 5);

INSERT INTO PRODUS

VALUES(SEQ\_PRODUS.NEXTVAL, 1, 'Componenta', 'Placa video PowerColor Radeon RX 6900 XT Red Devil 16GB GDDR6 256-bit', NULL, 'Red Devil', 'Da', 11999.99, NULL);

INSERT INTO PRODUS

VALUES(SEQ\_PRODUS.NEXTVAL, 3, 'Componenta', 'Memorie Corsair Vengeance LPX Black 16GB DDR4 3200MHz CL16 Dual Channel Kit', NULL, 'Corsair', 'Da', 514.99, NULL);

INSERT INTO PRODUS

VALUES(SEQ\_PRODUS.NEXTVAL, 4, 'Componenta', 'Hard disk Seagate BarraCuda 2TB SATA-III 7200RPM 256MB', NULL, 'Seagate', 'Da', 262.99, NULL);

INSERT INTO PRODUS

VALUES(SEQ\_PRODUS.NEXTVAL, 5, 'Accesoriu', 'Adaptor Gembird 1x HDMI 1.4 Male - 1x VGA Female', NULL, 'Vanzator 1', 'Nu', 47.99, 2);

-- ROLLBACK;

-- COMMIT;

-- Secventa pentru inserarea inregistrarilor in tabelul "SERVICIU":

CREATE SEQUENCE SEQ\_SERVICIU

INCREMENT by 1

START WITH 0

MINVALUE -1

NOCYCLE;

-- DROP SEQUENCE SEQ\_SERVICIU;

INSERT INTO SERVICIU

VALUES(SEQ\_SERVICIU.NEXTVAL, 'Asamblare', 'Serviciu Asamblare Standard',

'Asamblare sistem de calcul desktop de catre un specialist calificat.

Instalare sistem de operare test. Verificare compatibilitate componente.

Preinstalare sisteme de operare si aplicatii (daca s-au achizitionat)',

159);

INSERT INTO SERVICIU

VALUES(SEQ\_SERVICIU.NEXTVAL, 'Testare', 'Serviciu Testare Produs',

'Verificare functionalitate produs.

Aplicabil oricarui produs comercializat.',

135.99);

INSERT INTO SERVICIU

VALUES(SEQ\_SERVICIU.NEXTVAL, 'Montare', 'Serviciu Montare Componenta',

'Montarea diverselor componente in sistemul de calcul: placa video,

procesor, RAM, HDD, cooler pentru procesor, etc.',

59.99);

INSERT INTO SERVICIU

VALUES(SEQ\_SERVICIU.NEXTVAL, 'Diagnosticare', 'Serviciu Diagnosticare Produs',

'Identificarea problemelor de functionare si furnizarea de recomandari.',

119.99);

INSERT INTO SERVICIU

VALUES(SEQ\_SERVICIU.NEXTVAL, 'Asamblare', 'Serviciu Asamblare Premium',

'Asamblare si testare sistem de calcul desktop de catre un specialist calificat.

Cooling management. Wire management. Instalare sistem de operare test.

Verificare compatibilitate componente. Testare sistem (rulare in conditii de stres 24 ore).

Generarea unui raport operatiune asamblare premium. Updateuri la ultimele versiuni stabile (BIOS placa de baza).

Preinstalare sisteme de operare si aplicatii (daca s-au achizitionat).',

249.99);

-- ROLLBACK;

-- COMMIT;

INSERT INTO ADRESA\_LIVRARE

VALUES(1, 'Bucuresti', 'Mihai Eminescu', 1);

INSERT INTO ADRESA\_LIVRARE

VALUES(2, 'Iasi', 'Morii', 12);

INSERT INTO ADRESA\_LIVRARE

VALUES(3, 'Bucuresti', 'Unirii', 34);

INSERT INTO ADRESA\_LIVRARE

VALUES(4, 'Constanta', 'Stejarul mic', 2);

INSERT INTO ADRESA\_LIVRARE

VALUES(5, 'Bucuresti', 'Eroilor', 10);

INSERT INTO ADRESA\_LIVRARE

VALUES(6, 'Bucuresti', 'Mihail Kogalniceanu', 8);

-- ROLLBACK;

-- COMMIT;

INSERT INTO CLIENT

VALUES(1, 'Aurel', 'Popescu-Mihai');

INSERT INTO CLIENT

VALUES(2, 'Carstea', 'Darian-Stefan');

INSERT INTO CLIENT

VALUES(3, 'Castan', 'Mirela');

INSERT INTO CLIENT

VALUES(4, 'Popa', 'Marcel-Radu');

INSERT INTO CLIENT

VALUES(5, 'Manole', 'Iuliana-Elena');

INSERT INTO CLIENT

VALUES(6, 'Florian', 'Andrei-Cosmin');

INSERT INTO CLIENT

VALUES(7, 'Aurel', 'Marcel-Stoian');

-- ROLLBACK;

-- COMMIT;

INSERT INTO CONT\_SITE

VALUES(1, 'Utilizator Mihai', NULL, 'uzmihai2005@gmail.com');

INSERT INTO CONT\_SITE

VALUES(2, 'Darian Stefan', 2, 'carsteastef@yahoo.com');

INSERT INTO CONT\_SITE

VALUES(3, 'Mirela Castan', 3, 'mirela\_c@gmail.com');

INSERT INTO CONT\_SITE

VALUES(4, 'ANTON PAVEL MIHAI', NULL, 'apav\_mihai@gmail.com');

INSERT INTO CONT\_SITE

VALUES(5, 'Elena 2008', 5, 'elenamanole@yahoo.com');

-- ROLLBACK;

-- COMMIT;

INSERT INTO RECENZIE

VALUES(1, 5, 'Utilizator Mihai', TO\_DATE('15-01-2021','dd-mm-yyyy'), 3);

INSERT INTO RECENZIE

VALUES(2, 10, 'Utilizator Mihai', TO\_DATE('15-01-2021','dd-mm-yyyy'), 5);

INSERT INTO RECENZIE

VALUES(3, 20, 'Darian Stefan', TO\_DATE('23-05-2020','dd-mm-yyyy'), 4);

INSERT INTO RECENZIE

VALUES(4, 25, 'Mirela Castan', TO\_DATE('02-08-2019','dd-mm-yyyy'), 5);

INSERT INTO RECENZIE

VALUES(5, 25, 'Elena 2008', TO\_DATE('11-09-2020','dd-mm-yyyy'), 5);

INSERT INTO RECENZIE

VALUES(6, 15, 'Darian Stefan', TO\_DATE('15-02-2020','dd-mm-yyyy'), 3);

INSERT INTO RECENZIE

VALUES(7, 15, 'Utilizator Mihai', TO\_DATE('14-07-2020','dd-mm-yyyy'), 3);

INSERT INTO RECENZIE

VALUES(8, 35, 'Utilizator Mihai', TO\_DATE('06-09-2021','dd-mm-yyyy'), 4);

-- ROLLBACK;

-- COMMIT;

INSERT INTO DEPARTAMENT

VALUES(1, 'Relationare clienti');

INSERT INTO DEPARTAMENT

VALUES(2, 'Prestare servicii');

INSERT INTO DEPARTAMENT

VALUES(3, 'Intretinere website');

INSERT INTO DEPARTAMENT

VALUES(4, 'Inventar produse');

INSERT INTO DEPARTAMENT

VALUES(5, 'Securitate');

-- ROLLBACK;

-- COMMIT;

INSERT INTO ANGAJAT

VALUES(1, 2, 'Tehnician', 'Aurel', 'Tudor-Mihai', 7000, 'aurel\_mihai@gmail.com', '0793333333');

INSERT INTO ANGAJAT

VALUES(2, 1, 'Tehnician', 'Mihail', 'Maria-Andreea', 4500, 'mariaandreea09@yahoo.com', '0791111111');

INSERT INTO ANGAJAT

VALUES(3, 1, 'Tehnician', 'Vladoi', 'Rares', 4000, 'raresVld@gmail.com', '0749999999');

INSERT INTO ANGAJAT

VALUES(4, 4, 'Tehnician', 'Grigorescu', 'Stefan', 4500, 'grgrsc\_stf@yahoo.com', '0755555555');

INSERT INTO ANGAJAT

VALUES(5, 5, 'Paznic', 'Marian', 'Alexandru', 3300, 'marianalex3@yahoo.com', '0788888885');

-- ROLLBACK;

-- COMMIT;

INSERT INTO ESTE\_COMPATIBIL

VALUES(2, 5);

INSERT INTO ESTE\_COMPATIBIL

VALUES(2, 10);

INSERT INTO ESTE\_COMPATIBIL

VALUES(2, 20);

INSERT INTO ESTE\_COMPATIBIL

VALUES(2, 25);

INSERT INTO ESTE\_COMPATIBIL

VALUES(2, 30);

INSERT INTO ESTE\_COMPATIBIL

VALUES(4, 5);

INSERT INTO ESTE\_COMPATIBIL

VALUES(4, 20);

INSERT INTO ESTE\_COMPATIBIL

VALUES(4, 25);

INSERT INTO ESTE\_COMPATIBIL

VALUES(3, 5);

INSERT INTO ESTE\_COMPATIBIL

VALUES(3, 10);

INSERT INTO ESTE\_COMPATIBIL

VALUES(3, 30);

INSERT INTO ESTE\_COMPATIBIL

VALUES(1, 10);

-- ROLLBACK;

-- COMMIT;

INSERT INTO COMANDA

VALUES(1, 5, 1, 1, TO\_DATE('20-09-2020','dd-mm-yyyy'), 14999.99);

INSERT INTO COMANDA

VALUES(11, 5, 1, 1, TO\_DATE('20-09-2020','dd-mm-yyyy'), 14999.99);

INSERT INTO COMANDA

VALUES(2, 10, 1, 1, TO\_DATE('20-09-2020','dd-mm-yyyy'), 999.99);

INSERT INTO COMANDA

VALUES(3, 20, 2, 3, TO\_DATE('07-12-2019','dd-mm-yyyy'), 4799.99);

INSERT INTO COMANDA

VALUES(4, 25, 3, 4, TO\_DATE('16-02-2019','dd-mm-yyyy'), 3398.99);

INSERT INTO COMANDA

VALUES(5, 25, 5, 2, TO\_DATE('11-09-2020','dd-mm-yyyy'), 3398.99);

INSERT INTO COMANDA

VALUES(6, 30, 4, 5, TO\_DATE('07-02-2021','dd-mm-yyyy'), 11999.99);

INSERT INTO COMANDA

VALUES(7, 15, 1, 5, TO\_DATE('07-02-2021','dd-mm-yyyy'), 71.98);

INSERT INTO COMANDA

VALUES(8, 10, 4, 3, TO\_DATE('07-02-2021','dd-mm-yyyy'), 999.99);

INSERT INTO COMANDA

VALUES(9, 15, 2, 3, TO\_DATE('10-12-2019','dd-mm-yyyy'), 35.99);

INSERT INTO COMANDA

VALUES(10, 20, 6, 6, TO\_DATE('20-05-2021','dd-mm-yyyy'), 4799.99);

-- ROLLBACK;

-- COMMIT;

INSERT INTO CUMPARA

VALUES(1, 2, 1, TO\_DATE('20-09-2020','dd-mm-yyyy'));

INSERT INTO CUMPARA

VALUES(2, 5, 1, TO\_DATE('20-09-2020','dd-mm-yyyy'));

INSERT INTO CUMPARA

VALUES(3, 2, 6, TO\_DATE('14-04-2020','dd-mm-yyyy'));

INSERT INTO CUMPARA

VALUES(4, 3, 6, TO\_DATE('14-04-2020','dd-mm-yyyy'));

INSERT INTO CUMPARA

VALUES(5, 2, 3, TO\_DATE('16-02-2019','dd-mm-yyyy'));

INSERT INTO CUMPARA

VALUES(6, 4, 3, TO\_DATE('16-02-2019','dd-mm-yyyy'));

INSERT INTO CUMPARA

VALUES(7, 2, 5, TO\_DATE('11-09-2020','dd-mm-yyyy'));

INSERT INTO CUMPARA

VALUES(8, 4, 5, TO\_DATE('11-09-2020','dd-mm-yyyy'));

INSERT INTO CUMPARA

VALUES(9, 2, 2, TO\_DATE('07-12-2019','dd-mm-yyyy'));

INSERT INTO CUMPARA

VALUES(10, 2, 4, TO\_DATE('07-02-2021','dd-mm-yyyy'));

INSERT INTO CUMPARA

VALUES(11, 3, 4, TO\_DATE('08-02-2021','dd-mm-yyyy'));

INSERT INTO CUMPARA

VALUES(12, 2, 1, TO\_DATE('20-10-2021','dd-mm-yyyy'));

-- ROLLBACK;

-- COMMIT;

INSERT INTO ESTE\_ASISTAT

VALUES(1, 1, 1);

INSERT INTO ESTE\_ASISTAT

VALUES(2, 1, 2);

INSERT INTO ESTE\_ASISTAT

VALUES(3, 2, 1);

INSERT INTO ESTE\_ASISTAT

VALUES(4, 2, 2);

INSERT INTO ESTE\_ASISTAT

VALUES(5, 3, 1);

INSERT INTO ESTE\_ASISTAT

VALUES(6, 3, 2);

INSERT INTO ESTE\_ASISTAT

VALUES(7, 5, 1);

INSERT INTO ESTE\_ASISTAT

VALUES(8, 5, 2);

INSERT INTO ESTE\_ASISTAT

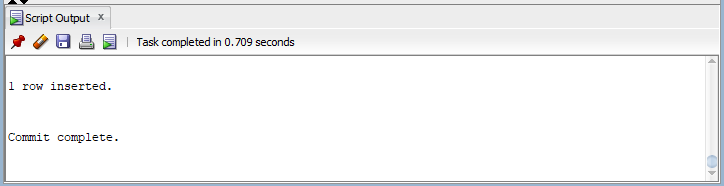
VALUES(9, 4, 3);

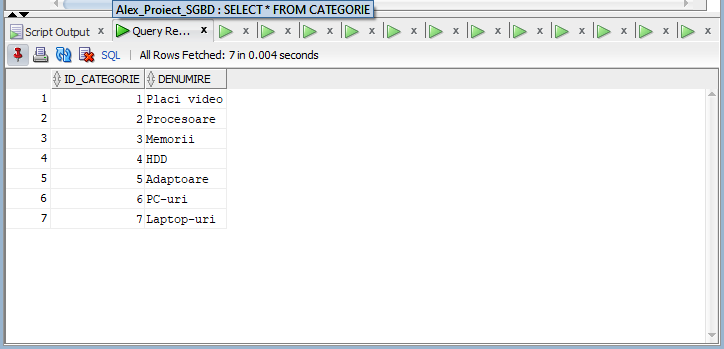
INSERT INTO ESTE\_ASISTAT

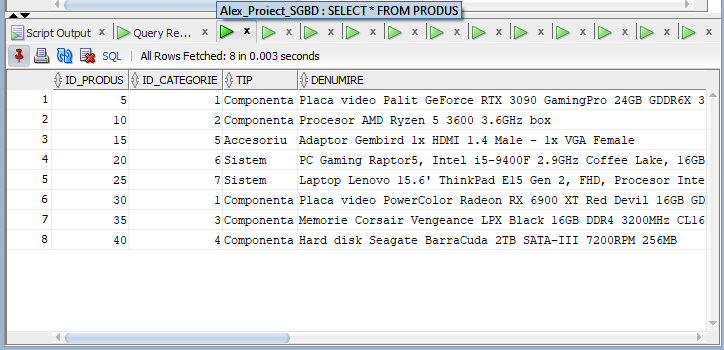
VALUES(10, 4, 2);

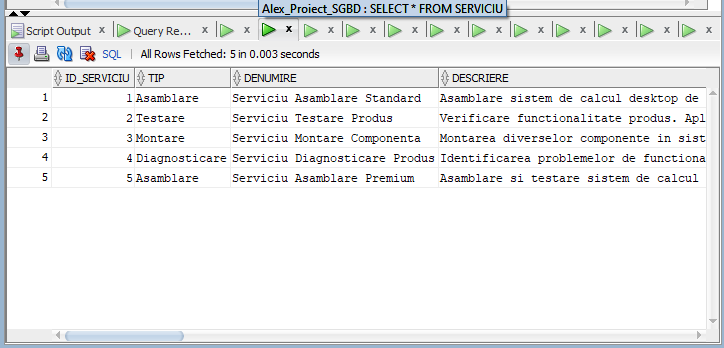
-- ROLLBACK;

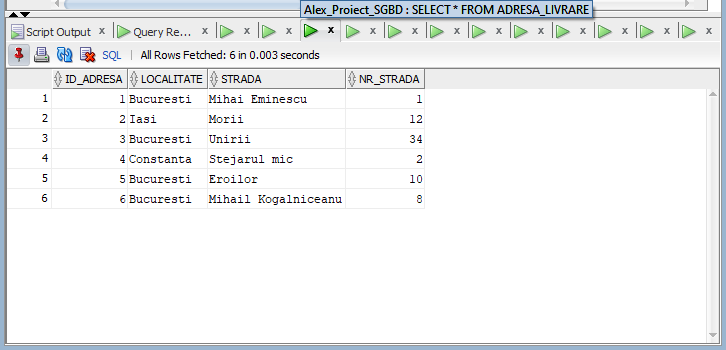
COMMIT;

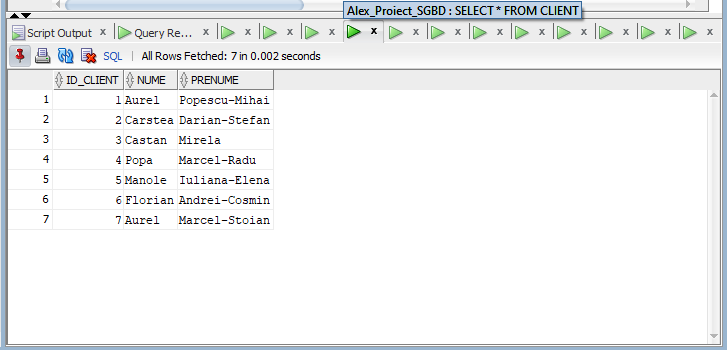


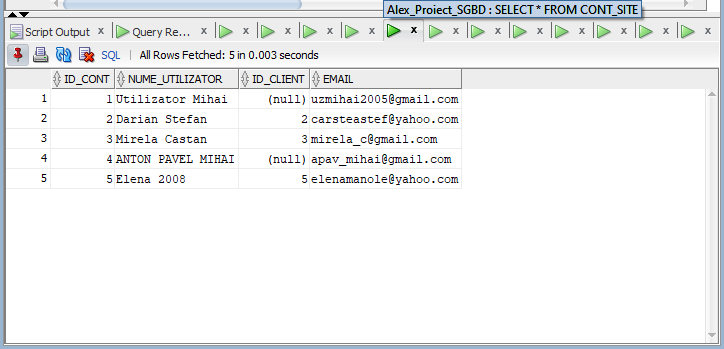


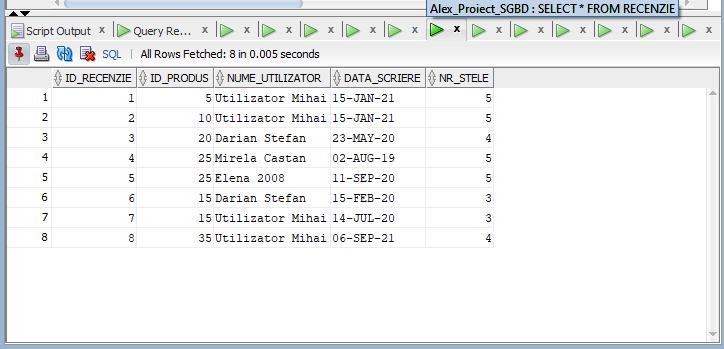


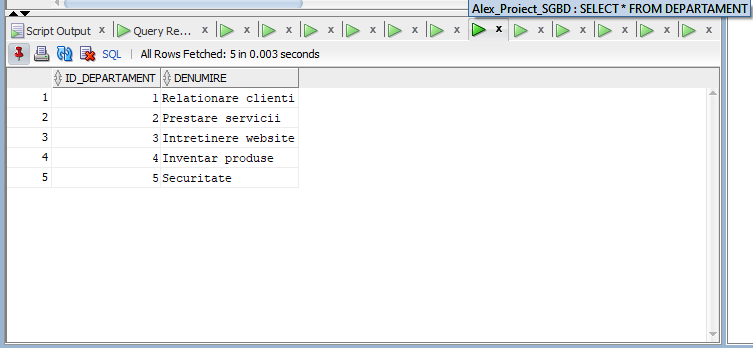


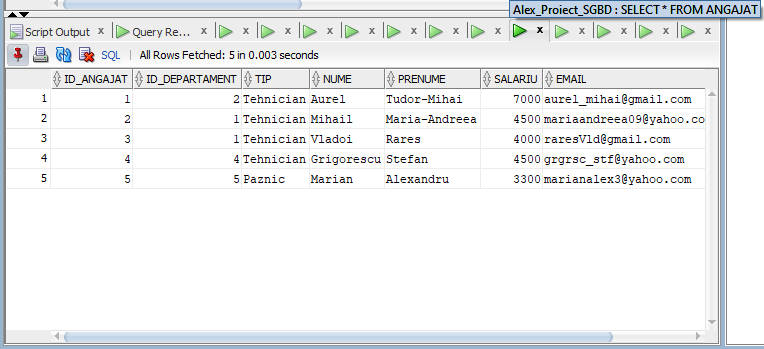


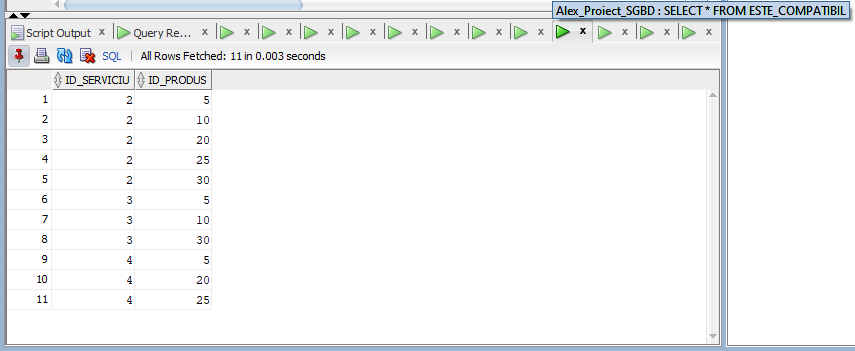


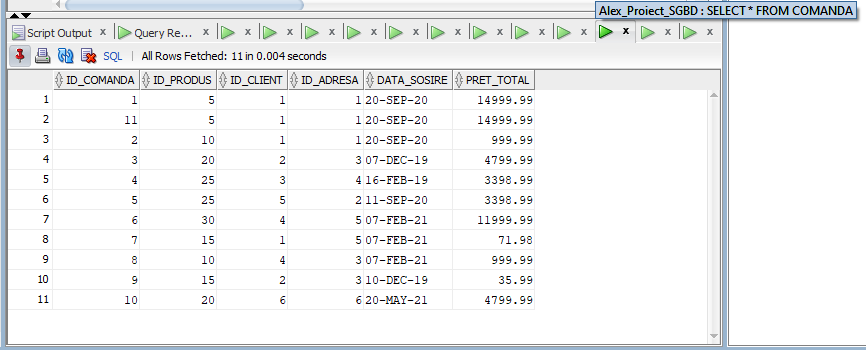


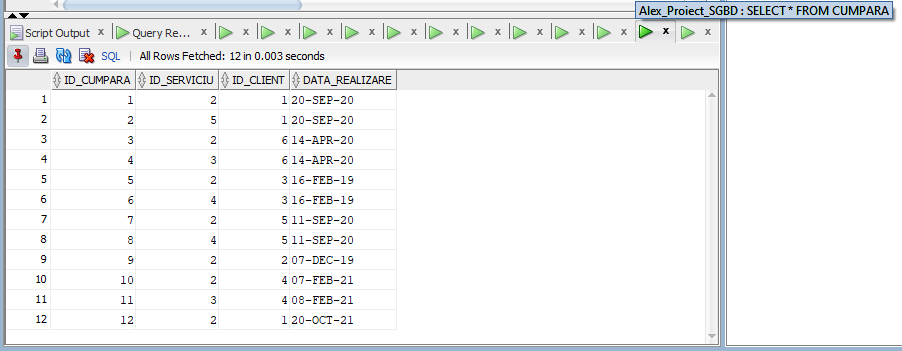


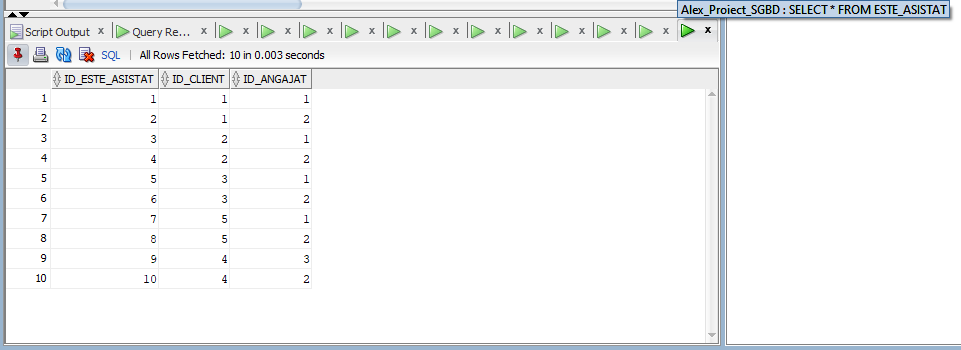












1. Subprogram stocat care utilizează două tipuri de colecții:

-- Cerinta 6.:

-- Enunt:

-- Pentru toti clientii care au acelasi nume ca un nume introdus de la tastatura,

-- sa se afiseze id-ul, numele complet, si pretul tuturor produselor care au fost comandate

-- de catre acestia, in ordine alfabetica a numelor produselor. Pentru fiecare produs

-- al fiecarui client se va afisa un numar de ordine. In cazul in care un anumit client

-- nu a comandat niciun produs, se va afisa un mesaj corespunzator.

-- Sa se trateze exceptii care pot aparea.

CREATE OR REPLACE PROCEDURE EX6\_PROC

(v\_nume IN CLIENT.nume%TYPE)

IS

v\_prenume CLIENT.prenume%TYPE;

v\_id CLIENT.id\_client%TYPE;

TYPE tab\_imb\_id\_cli IS TABLE OF CLIENT.id\_client%TYPE;

t\_id\_cli tab\_imb\_id\_cli;

v\_nr\_clienti NUMBER;

TYPE tab\_imb\_id\_prod IS TABLE OF PRODUS.id\_produs%TYPE;

t\_id\_prod tab\_imb\_id\_prod;

TYPE tab\_ind\_nume\_prod IS TABLE OF PRODUS.denumire%TYPE

INDEX BY PLS\_INTEGER;

t\_nume\_prod tab\_ind\_nume\_prod;

TYPE vector\_pret\_prod IS VARRAY(2000) OF PRODUS.pret%TYPE;

vect\_pret\_prod vector\_pret\_prod := vector\_pret\_prod();

v\_contor NUMBER;

BEGIN

SELECT COUNT(nume)

INTO v\_nr\_clienti

FROM CLIENT

WHERE UPPER(nume) = UPPER(v\_nume);

IF v\_nr\_clienti = 0 THEN

RAISE NO\_DATA\_FOUND;

ELSE

SELECT id\_client

BULK COLLECT INTO t\_id\_cli

FROM CLIENT

WHERE UPPER(nume) = UPPER(v\_nume);

FOR i IN t\_id\_cli.FIRST..t\_id\_cli.LAST LOOP

SELECT DISTINCT p.id\_produs, denumire, pret

BULK COLLECT INTO t\_id\_prod, t\_nume\_prod, vect\_pret\_prod

FROM PRODUS p JOIN COMANDA co ON p.id\_produs = co.id\_produs

JOIN CLIENT cl ON co.id\_client = cl.id\_client

WHERE cl.id\_client = t\_id\_cli(i)

ORDER BY 2;

SELECT prenume

INTO v\_prenume

FROM CLIENT

WHERE id\_client = t\_id\_cli(i);

IF t\_id\_prod.COUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Clientul cu id-ul ' || t\_id\_cli(i) || ', pe nume ' || v\_nume || ' ' || v\_prenume || ', nu a comandat niciun produs!');

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

ELSE

v\_contor := 1;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Clientul cu id-ul ' || t\_id\_cli(i) || ', pe nume ' || v\_nume || ' ' || v\_prenume || ', a achizitionat produsele: ');

DBMS\_OUTPUT.NEW\_LINE;

FOR i IN t\_id\_prod.FIRST..t\_id\_prod.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(v\_contor || '.');

DBMS\_OUTPUT.PUT\_LINE('Id produs: ' || t\_id\_prod(i));

DBMS\_OUTPUT.PUT\_LINE('Denumire produs: ' || t\_nume\_prod(i));

DBMS\_OUTPUT.PUT\_LINE('Pret produs: ' || vect\_pret\_prod(i));

IF i <> t\_id\_prod.LAST THEN

DBMS\_OUTPUT.NEW\_LINE;

END IF;

v\_contor := v\_contor + 1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

END IF;

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista niciun client cu numele ' || v\_nume || '!');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut alta eroare!');

END EX6\_PROC;

/

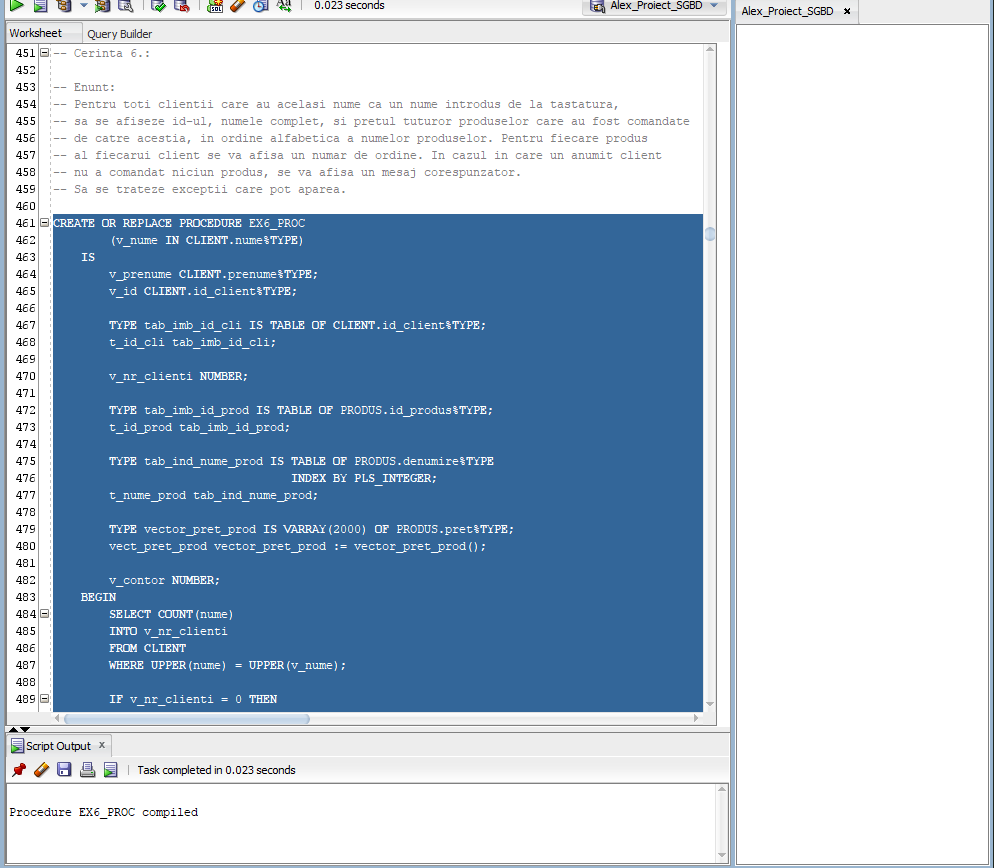
DECLARE

v\_nume CLIENT.nume%TYPE := '&p\_nume';

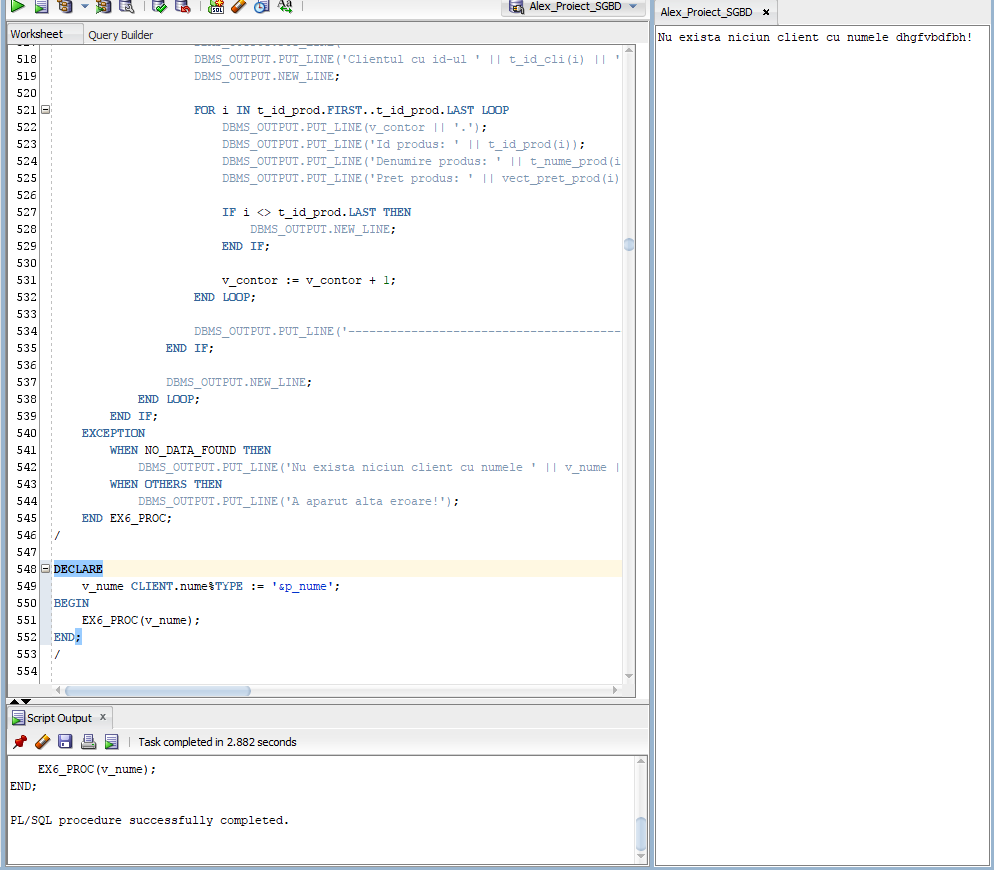
BEGIN

EX6\_PROC(v\_nume);

END;

/





1. Subprogram stocat care utilizează un tip de cursor:

-- Cerinta 7.:

-- Enunt:

-- Prin intermediul unei functii stocate, sa se afiseze, pentru fiecare angajat care a

-- asistat clienti, numele complet al acestuia (nume + prenume), alaturi de lista numelor

-- complete a clientilor asistati, ordonand rezultatele alfabetic dupa numele angajatilor.

-- Tot prin intermediul functiei stocate, se va returna numarul de angajati care au

-- asistat clienti, urmand sa se afiseze mesajul "Personal inactiv!" in cazul in care

-- numarul este mai mic decat jumatate din numarul total de angajati, respectiv mesajul

-- "Personal activ!" in caz contrar, si se va obtine (printr-un parametru de tip OUT) si

-- afisa numarul de clienti asistati de catre un angajat al carui cod este introdus de

-- la tastatura.

-- Sa se trateze exceptii care pot aparea.

-- Rezolvati problema folosind ciclu cursoare cu subcereri, stiind ca nu este permisa

-- folosirea colectiilor de date.

CREATE OR REPLACE FUNCTION EX7\_FUNC

(p\_id\_ang IN ANGAJAT.id\_angajat%TYPE,

p\_nr\_cli\_asist\_ang OUT NUMBER)

RETURN NUMBER

IS

v\_nr\_ang\_activi NUMBER;

v\_nr\_cli\_asist\_ang NUMBER;

BEGIN

p\_nr\_cli\_asist\_ang := 0;

FOR i IN (SELECT id\_angajat, nume || ' ' || prenume nume\_ang

FROM ANGAJAT a

ORDER BY 2) LOOP

SELECT COUNT(DISTINCT nume || ' ' || prenume)

INTO v\_nr\_cli\_asist\_ang

FROM CLIENT c JOIN ESTE\_ASISTAT ea ON c.id\_client = ea.id\_client

WHERE ea.id\_angajat = i.id\_angajat;

IF v\_nr\_cli\_asist\_ang > 0 THEN

v\_nr\_ang\_activi := v\_nr\_ang\_activi + 1;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Id angajat: ' || i.id\_angajat);

DBMS\_OUTPUT.PUT\_LINE('Nume angajat: ' || i.nume\_ang);

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Clienti asistati de catre angajat: ');

FOR j IN (SELECT DISTINCT nume || ' ' || prenume nume\_cli

FROM CLIENT c JOIN ESTE\_ASISTAT ea ON c.id\_client = ea.id\_client

WHERE ea.id\_angajat = i.id\_angajat) LOOP

DBMS\_OUTPUT.PUT\_LINE(j.nume\_cli);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.NEW\_LINE;

END IF;

IF i.id\_angajat = p\_id\_ang THEN

p\_nr\_cli\_asist\_ang := v\_nr\_cli\_asist\_ang;

END IF;

END LOOP;

RETURN v\_nr\_ang\_activi;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare necunoscuta!');

END EX7\_FUNC;

/

DECLARE

v\_id\_ang ANGAJAT.id\_angajat%TYPE := &p\_id;

v\_nr\_cli\_asist\_ang NUMBER;

v\_nr\_ang NUMBER;

BEGIN

SELECT COUNT(id\_angajat)

INTO v\_nr\_ang

FROM ANGAJAT;

SELECT id\_angajat

INTO v\_id\_ang

FROM ANGAJAT

WHERE id\_angajat = v\_id\_ang;

IF EX7\_FUNC(v\_id\_ang, v\_nr\_cli\_asist\_ang) < v\_nr\_ang/2 THEN

DBMS\_OUTPUT.PUT\_LINE('Status personal:');

DBMS\_OUTPUT.PUT\_LINE('Personal inactiv!');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Status personal:');

DBMS\_OUTPUT.PUT\_LINE('Personal activ!');

END IF;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Numarul de clienti asistati de catre angajatul cu id-ul ' || v\_id\_ang || ': ' || v\_nr\_cli\_asist\_ang || '.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

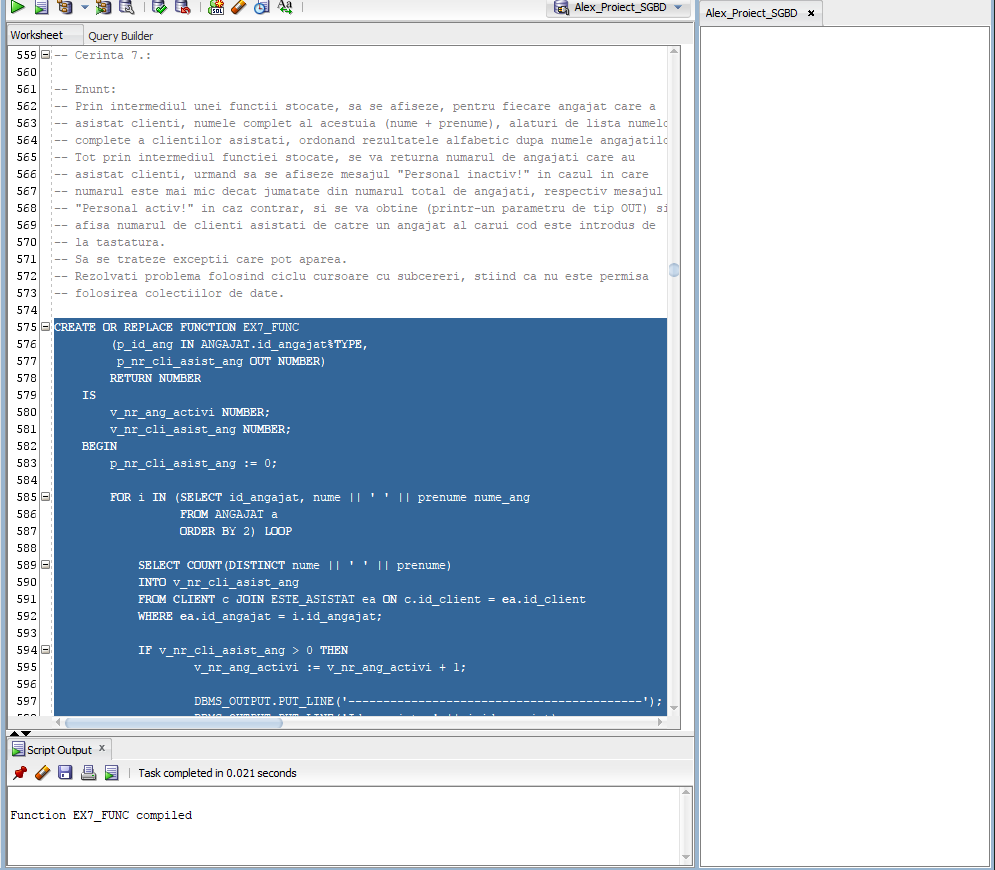
DBMS\_OUTPUT.PUT\_LINE('Nu exista niciun angajat cu id-ul ' || v\_id\_ang || '!');

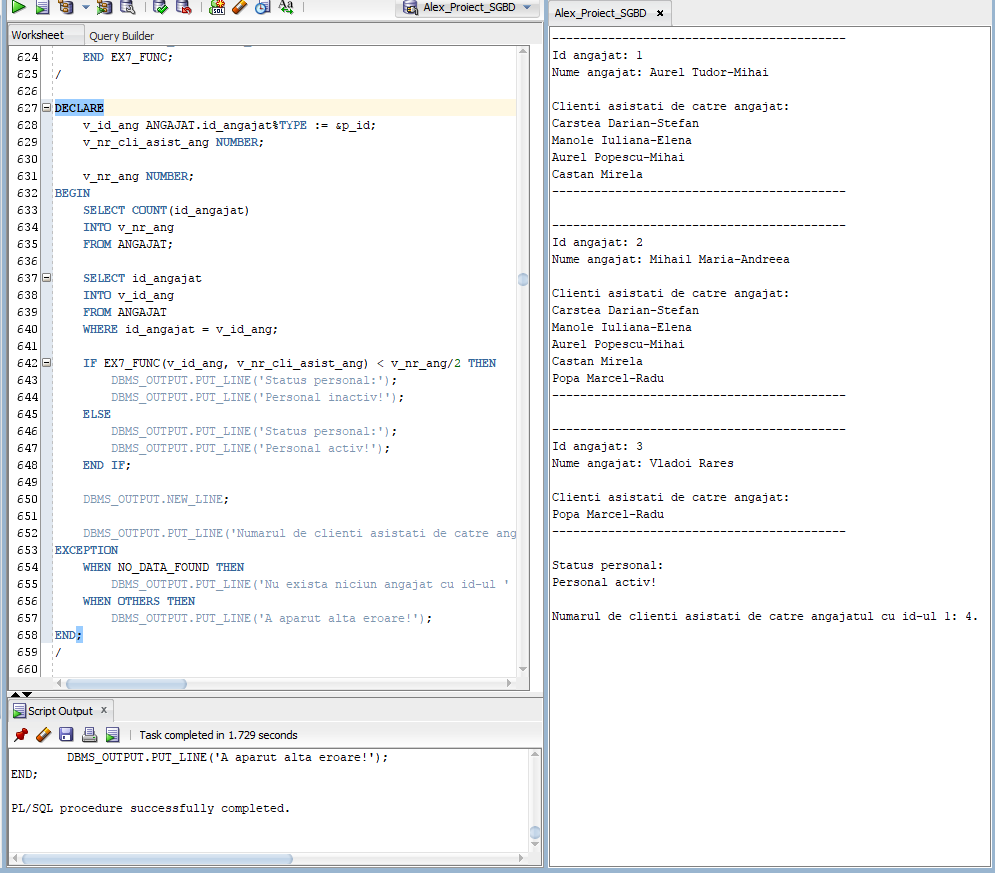
WHEN OTHERS THEN

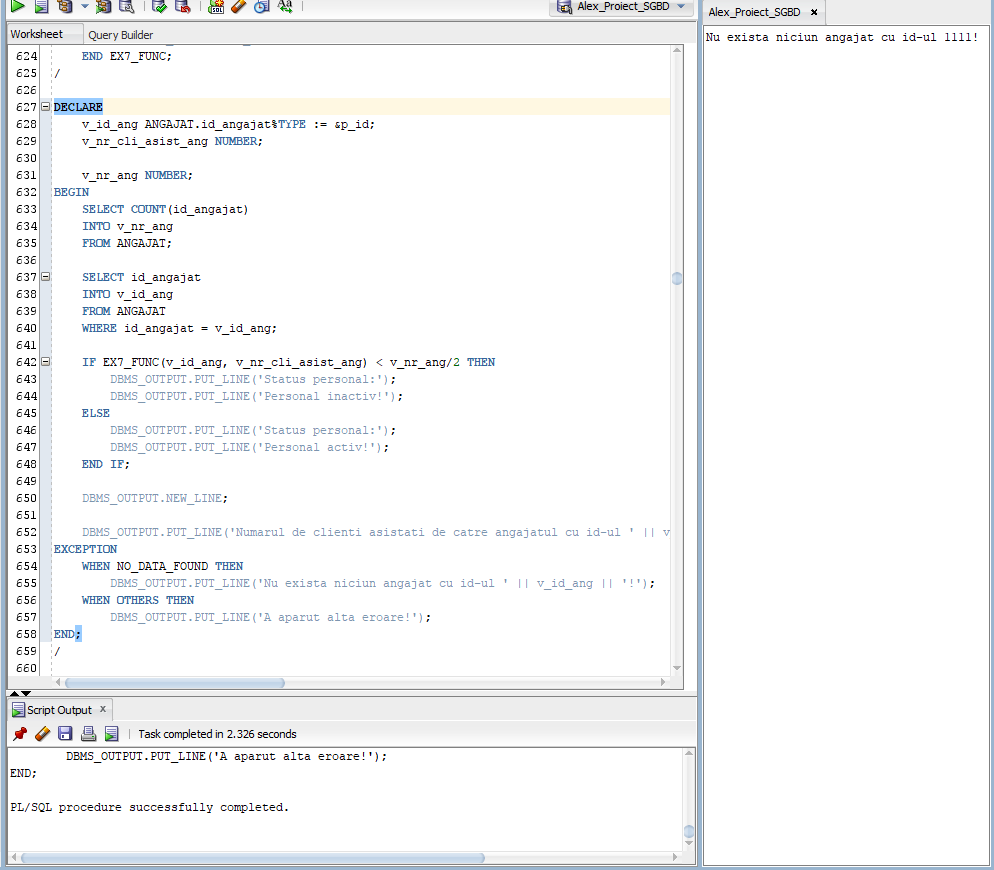
DBMS\_OUTPUT.PUT\_LINE('A aparut alta eroare!');

END;

/







1. Subprogram stocat de tip funcție care utilizează într-o singură comandă SQL 3 dintre tabelele definite:

-- Cerinta 8.:

-- Enunt:

-- Pentru un cont al carui nume de utilizator este dat de la tastatura, sa se

-- afiseze id-ul clientului care il detine. In cazul in care contul nu este detinut

-- de catre un client cunoscut, se va afisa un mesaj corespunzator. Totodata, se va

-- afisa, pentru top 3 recenzii (in functie de numarul de stele) scrise de catre cont,

-- id-ul recenziei, numarul de stele, si numele produsului pentru care a fost scrisa

-- recenzia. In caz ca exista mai multe recenzii care au acelasi numar de stele, si

-- sunt eligibile pentru a se afla in top, se vor afisa informatii despre toate aceste

-- recenzii. De asemenea, daca utilizatorul nu a scris nicio recenzie, se va afisa un

-- mesaj corespunzator.

-- Mai mult, pentru numele de utilizator specificat, sa se returneze si afiseze numarul

-- de recenzii scrise de catre contul asociat.

-- Sa se trateze toate exceptiile care pot aparea.

CREATE OR REPLACE FUNCTION EX8\_FUNC

(v\_nume\_utilizator IN CONT\_SITE.nume\_utilizator%TYPE)

RETURN NUMBER

IS

v\_id\_cont CONT\_SITE.id\_cont%TYPE;

v\_id\_client CLIENT.id\_client%TYPE;

v\_id\_recenzie RECENZIE.id\_recenzie%TYPE;

v\_nr\_stele RECENZIE.nr\_stele%TYPE;

v\_nr\_stele\_precedent RECENZIE.nr\_stele%TYPE := 0;

v\_nume\_produs PRODUS.denumire%TYPE;

v\_nr\_stele\_min RECENZIE.nr\_stele%TYPE;

TYPE ref\_info\_recenzii IS REF CURSOR;

c\_info\_recenzii ref\_info\_recenzii;

CURSOR c\_recenzii IS

SELECT id\_client,

CURSOR (SELECT id\_recenzie, nr\_stele, denumire

FROM RECENZIE r JOIN PRODUS p ON r.id\_produs = p.id\_produs

WHERE nume\_utilizator = v\_nume\_utilizator

ORDER BY 2 DESC)

FROM CONT\_SITE

WHERE nume\_utilizator = v\_nume\_utilizator;

v\_contor NUMBER := 0;

v\_top NUMBER := 0;

BEGIN

SELECT id\_cont

INTO v\_id\_cont

FROM CONT\_SITE

WHERE nume\_utilizator = v\_nume\_utilizator;

DBMS\_OUTPUT.PUT\_LINE('Id cont: ' || v\_id\_cont);

DBMS\_OUTPUT.PUT\_LINE('Nume utilizator cont: ' || v\_nume\_utilizator);

OPEN c\_recenzii;

FETCH c\_recenzii INTO v\_id\_client, c\_info\_recenzii;

IF v\_id\_client IS NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Contul nu este detinut de catre un client cunoscut.');

ElSE

DBMS\_OUTPUT.PUT\_LINE('Id client: ' || v\_id\_client);

END IF;

DBMS\_OUTPUT.NEW\_LINE;

SELECT MIN(distincte)

INTO v\_nr\_stele\_min

FROM (SELECT DISTINCT nr\_stele distincte

FROM RECENZIE

WHERE nume\_utilizator = v\_nume\_utilizator

ORDER BY distincte DESC)

WHERE ROWNUM <= 3;

FETCH c\_info\_recenzii INTO v\_id\_recenzie, v\_nr\_stele, v\_nume\_produs;

IF c\_info\_recenzii%ROWCOUNT = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('Top 3 recenzii ale contului: ');

LOOP

v\_contor := v\_contor + 1;

IF v\_nr\_stele <> v\_nr\_stele\_precedent THEN

v\_top := v\_top + 1;

v\_nr\_stele\_precedent := v\_nr\_stele;

END IF;

IF v\_nr\_stele >= v\_nr\_stele\_min THEN

DBMS\_OUTPUT.PUT\_LINE(v\_top || '. Id recenzie: ' || v\_id\_recenzie || ' | Nr. stele: ' || v\_nr\_stele || ' | Nume produs: ' || v\_nume\_produs);

END IF;

FETCH c\_info\_recenzii INTO v\_id\_recenzie, v\_nr\_stele, v\_nume\_produs;

EXIT WHEN c\_info\_recenzii%NOTFOUND;

END LOOP;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Utilizatorul nu a scris nicio recenzie.');

END IF;

DBMS\_OUTPUT.NEW\_LINE;

CLOSE c\_recenzii;

RETURN v\_contor;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista niciun cont cu numele de utilizator ' || v\_nume\_utilizator || '!');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'A aparut alta eroare!');

END EX8\_FUNC;

/

DECLARE

v\_nume\_utilizator CONT\_SITE.nume\_utilizator%TYPE := '&p\_nume\_utilizator';

v\_nr\_recenzii NUMBER;

BEGIN

v\_nr\_recenzii := EX8\_FUNC(v\_nume\_utilizator);

IF v\_nr\_recenzii > 0 THEN

IF v\_nr\_recenzii > 1 THEN

DBMS\_OUTPUT.PUT\_LINE('Contul cu numele de utilizator ' || v\_nume\_utilizator || ' a scris ' || v\_nr\_recenzii || ' recenzii.');

ELSE

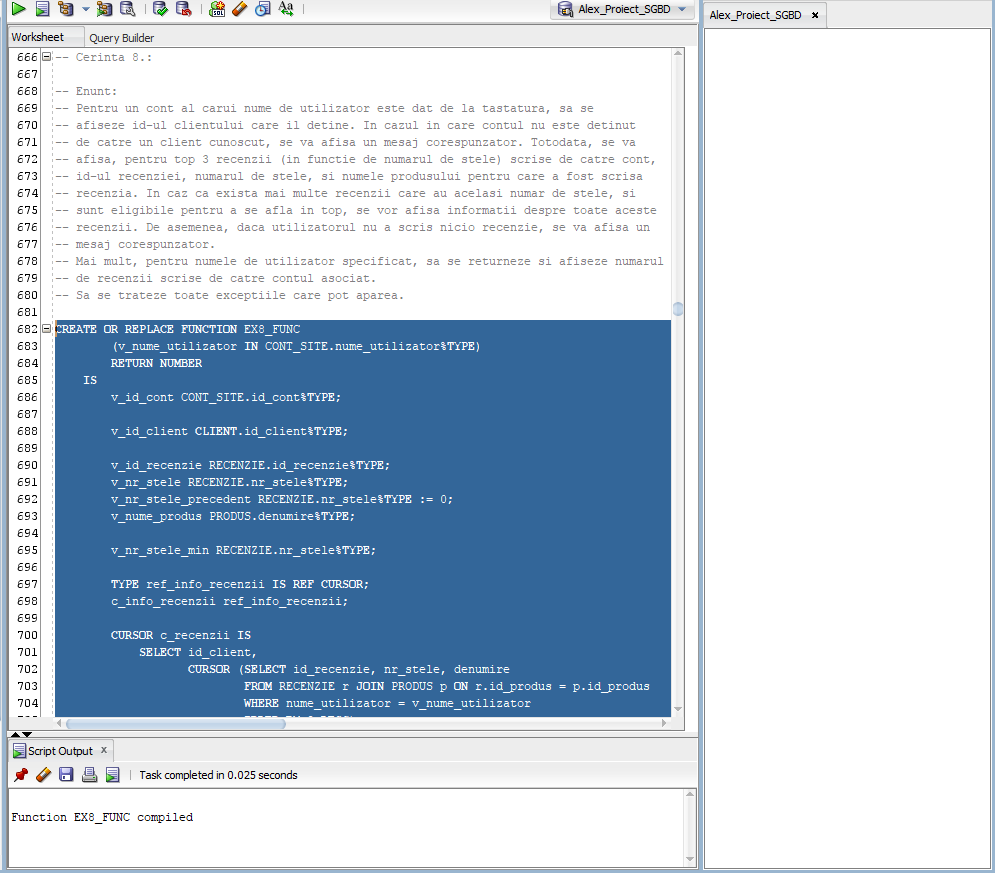
DBMS\_OUTPUT.PUT\_LINE('Contul cu numele de utilizator ' || v\_nume\_utilizator || ' a scris o recenzie.');

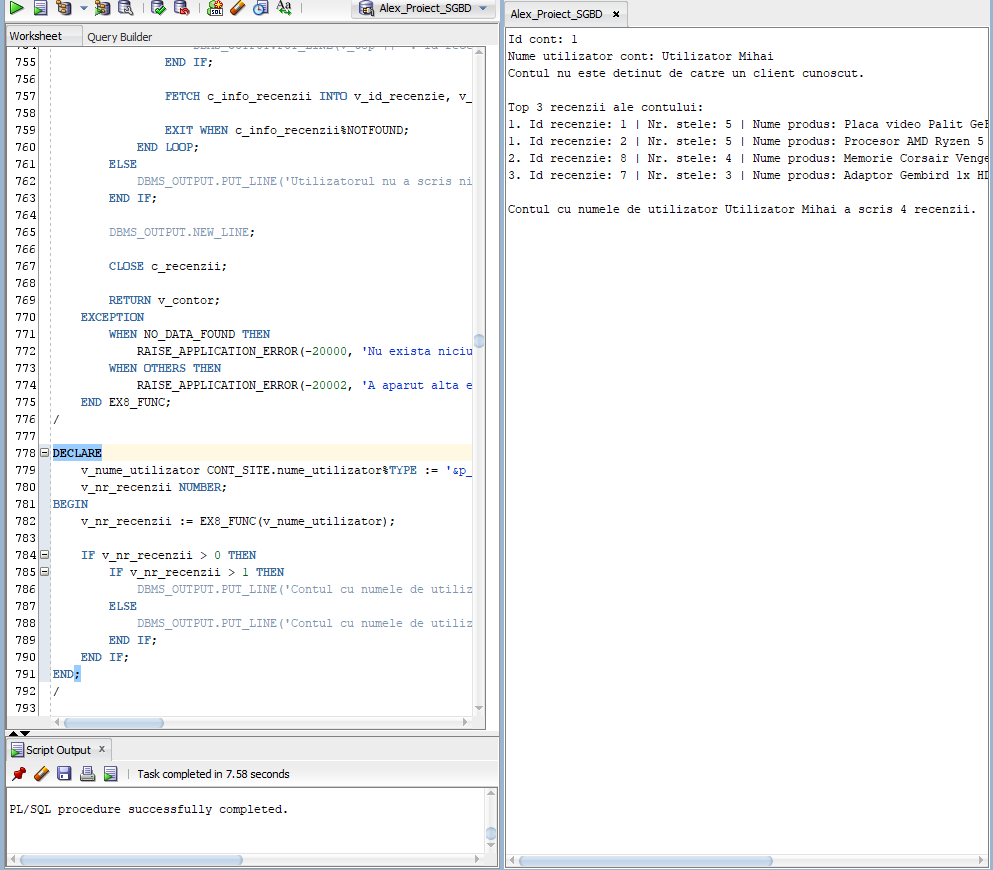
END IF;

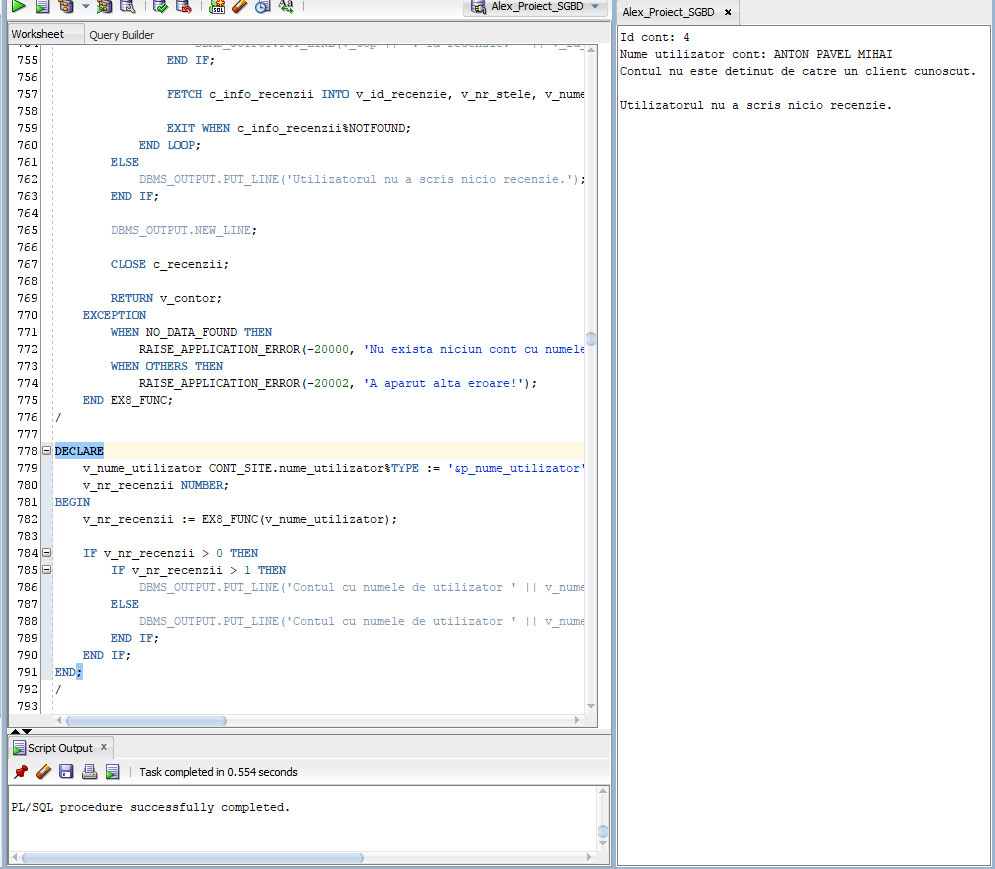
END IF;

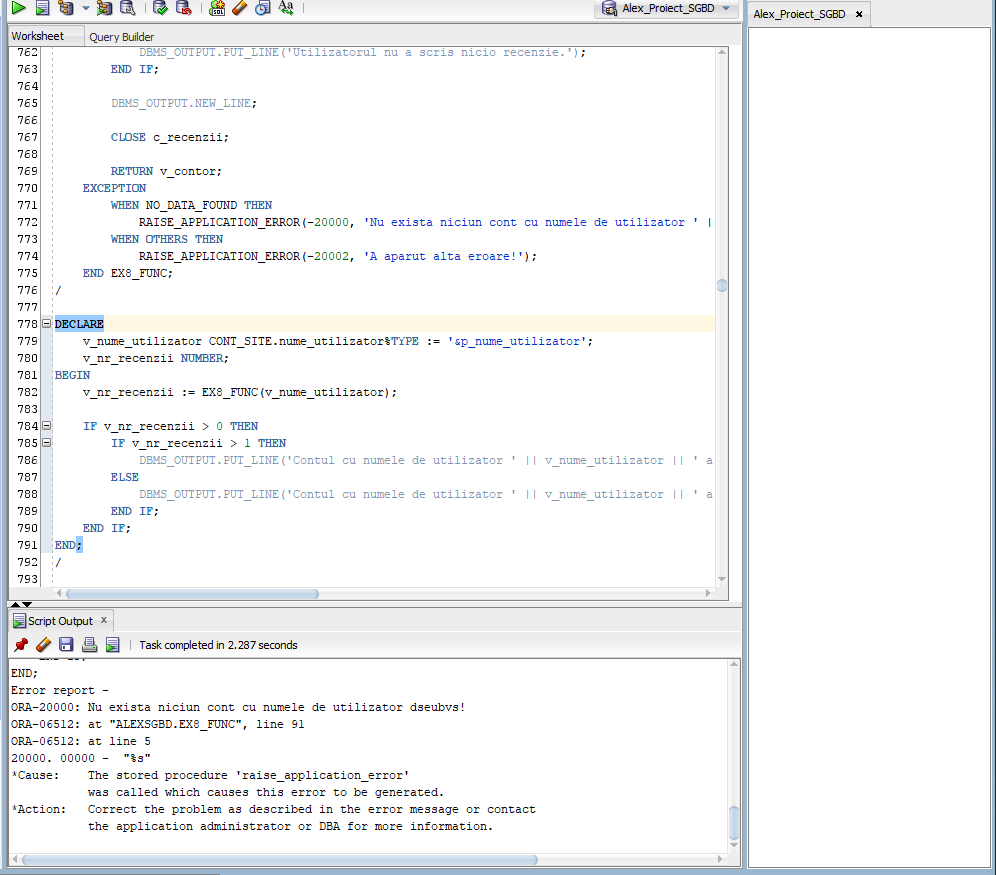
END;

/









1. Subprogram stocat de tip procedură care utilizează într-o singură comandă SQL 5 dintre tabelele definite:

-- Cerinta 9.:

-- Enunt:

-- Pentru un produs al carui nume este introdus de la tastatura, sa se afiseze id-ul

-- si numele tuturor serviciilor compatibile cu acesta. In plus, sa se afiseze si

-- id-ul, alaturi de numele complet (nume + prenume) al tuturor clientilor care au

-- cumparat un anumit serviciu compatibil cu produsul. In caz ca produsul nu este

-- compatibil cu niciun serviciu, sau in caz ca nu exista clienti care au cumparat

-- un anumit serviciu compatibil cu produsul, se vor afisa mesaje corespunzatoare.

-- Rezultatele se vor afisa in ordine alfabetica a numelor serviciilor, cat si a

-- numelor clientilor.

-- Sa se trateze toate exceptiile care pot aparea.

CREATE OR REPLACE PROCEDURE EX9\_PROC

(v\_nume\_produs IN PRODUS.denumire%TYPE)

IS

v\_id\_produs PRODUS.id\_produs%TYPE;

v\_id\_serviciu SERVICIU.id\_serviciu%TYPE;

v\_nume\_serviciu SERVICIU.denumire%TYPE;

TYPE tab\_imb\_id\_cli IS TABLE OF CLIENT.id\_client%TYPE;

t\_id\_cli tab\_imb\_id\_cli;

TYPE tab\_imb\_nume\_cli IS TABLE OF CLIENT.nume%TYPE;

t\_nume\_cli tab\_imb\_nume\_cli;

TYPE ref\_info\_cli IS REF CURSOR;

c\_info\_cli ref\_info\_cli;

CURSOR c\_date\_servicii IS

SELECT s.id\_serviciu, s.denumire,

CURSOR (SELECT DISTINCT cl.id\_client, nume || ' ' || prenume

FROM CLIENT cl JOIN CUMPARA cu ON cl.id\_client = cu.id\_client

JOIN SERVICIU si ON cu.id\_serviciu = si.id\_serviciu

WHERE si.id\_serviciu = s.id\_serviciu

ORDER BY 2)

FROM SERVICIU s JOIN ESTE\_COMPATIBIL ec ON s.id\_serviciu = ec.id\_serviciu

JOIN PRODUS p ON ec.id\_produs = p.id\_produs

WHERE UPPER(p.denumire) = UPPER(v\_nume\_produs)

ORDER BY 2;

BEGIN

SELECT id\_produs

INTO v\_id\_produs

FROM PRODUS

WHERE UPPER(denumire) = UPPER(v\_nume\_produs);

OPEN c\_date\_servicii;

DBMS\_OUTPUT.PUT\_LINE('Id produs: ' || v\_id\_produs);

DBMS\_OUTPUT.PUT\_LINE('Nume produs: ' || v\_nume\_produs);

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Lista serviciilor compatibile cu produsul:');

LOOP

FETCH c\_date\_servicii INTO v\_id\_serviciu, v\_nume\_serviciu, c\_info\_cli;

IF c\_date\_servicii%NOTFOUND THEN

IF c\_date\_servicii%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista servicii compatibile!');

END IF;

EXIT;

END IF;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Id serviciu: ' || v\_id\_serviciu);

DBMS\_OUTPUT.PUT\_LINE('Nume serviciu: ' || v\_nume\_serviciu);

DBMS\_OUTPUT.NEW\_LINE;

FETCH c\_info\_cli BULK COLLECT INTO t\_id\_cli, t\_nume\_cli;

DBMS\_OUTPUT.PUT\_LINE('Lista clientilor care au cumparat serviciul:');

IF t\_id\_cli.COUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista clienti care au cumparat serviciul.');

ELSE

FOR i IN t\_id\_cli.FIRST..t\_id\_cli.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE('Id client: ' || t\_id\_cli(i) || ' | Nume client: ' || t\_nume\_cli(i));

END LOOP;

END IF;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

CLOSE c\_date\_servicii;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista niciun produs cu numele ' || v\_nume\_produs || '!');

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Exista mai multe produse cu numele ' || v\_nume\_produs || '!');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'A aparut alta eroare!');

END EX9\_PROC;

/

DECLARE

v\_nume\_produs PRODUS.denumire%TYPE := '&p\_nume\_produs';

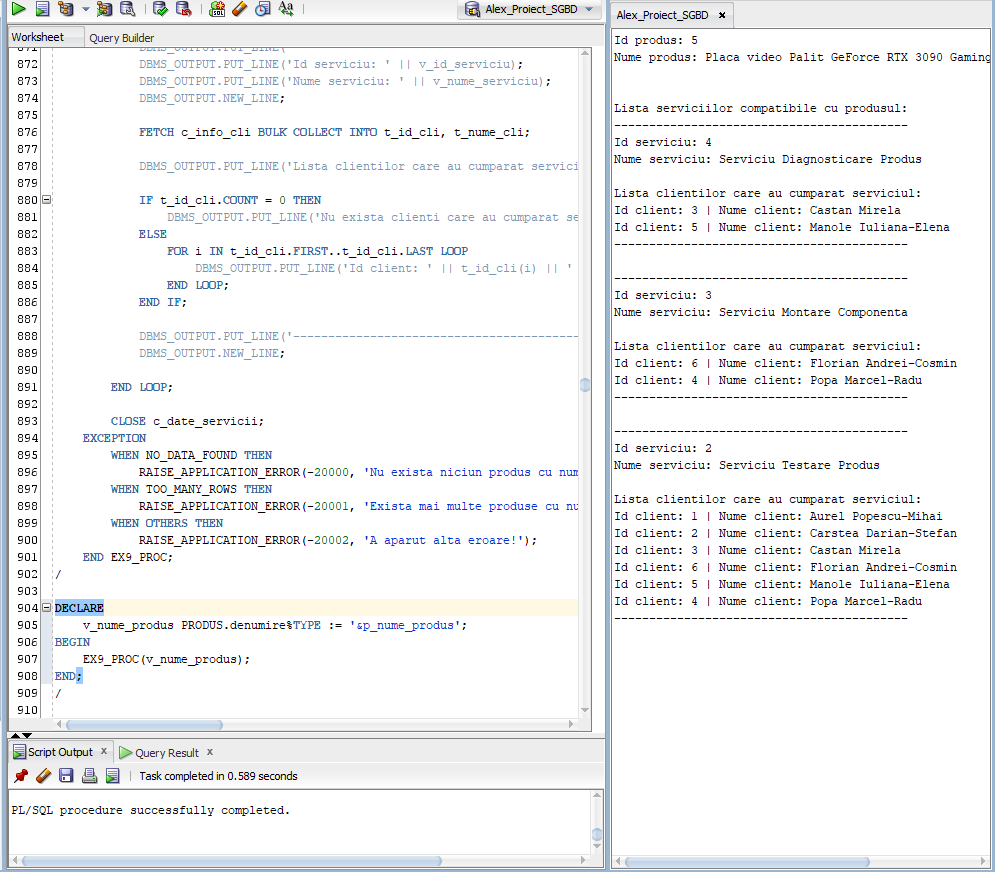
BEGIN

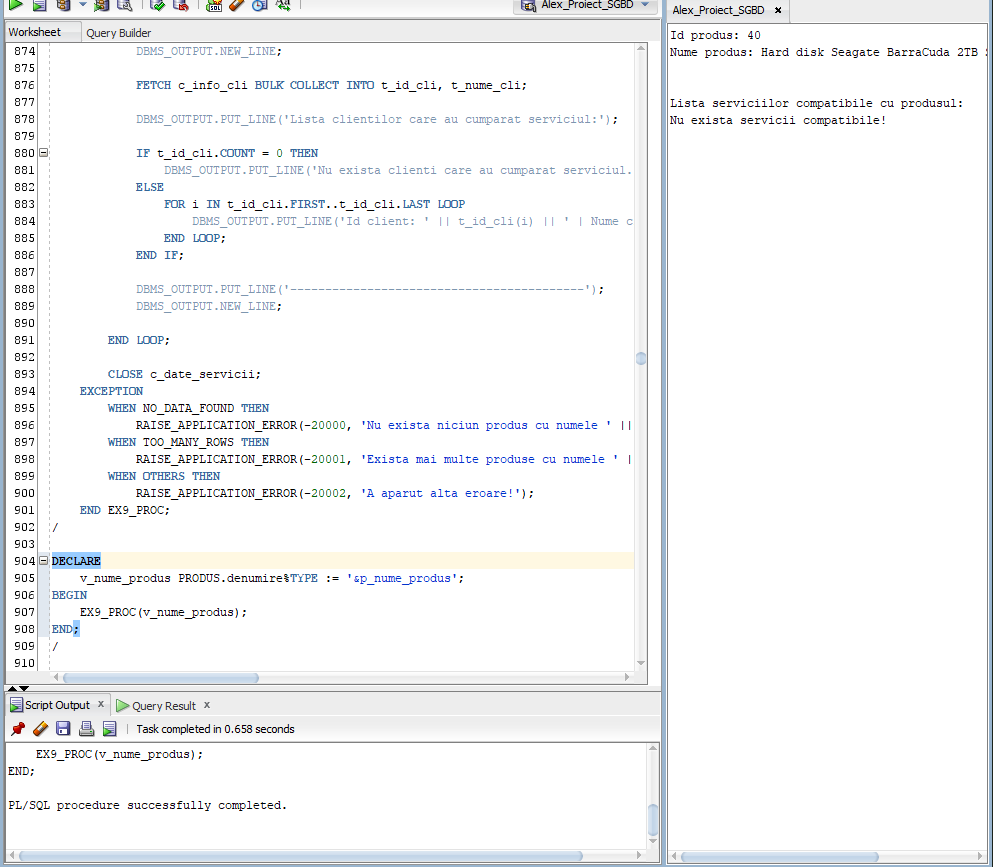
EX9\_PROC(v\_nume\_produs);

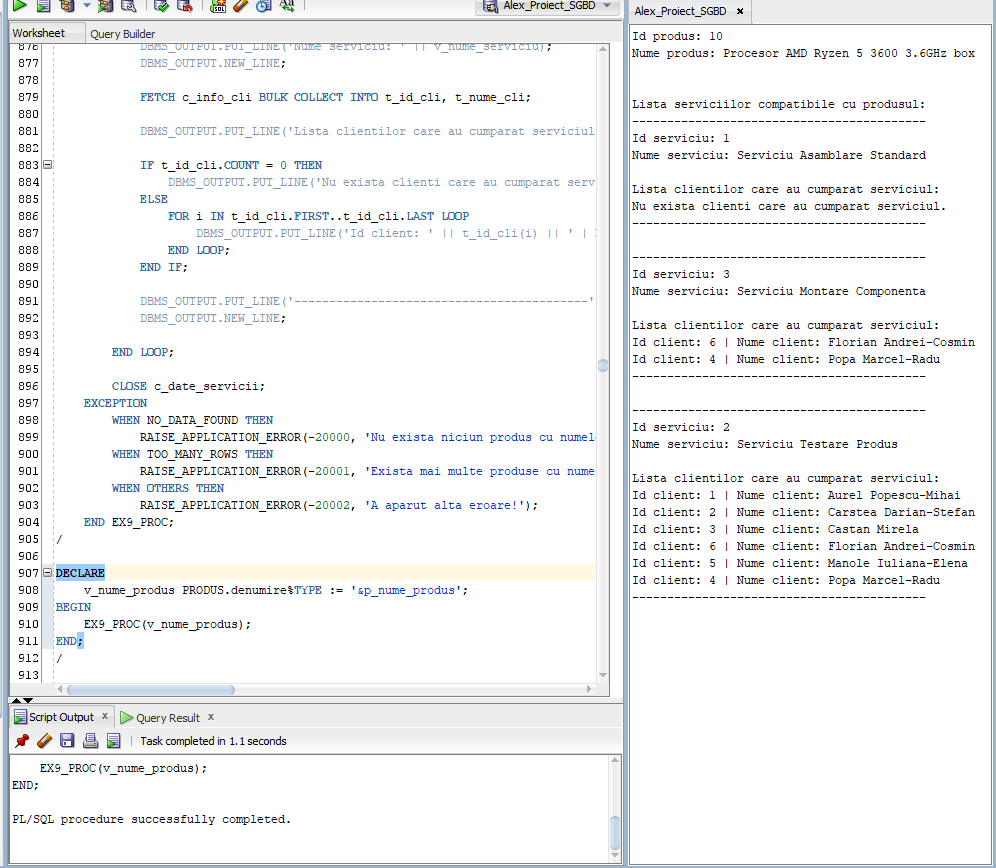
END;

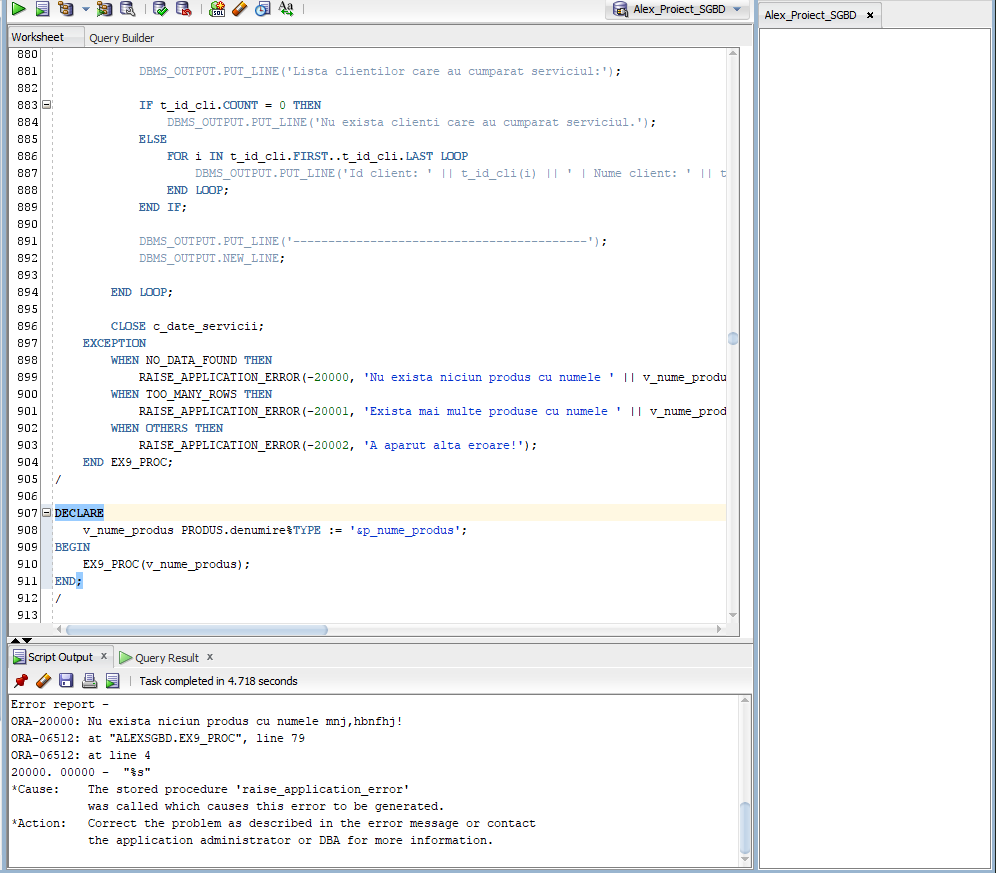
/

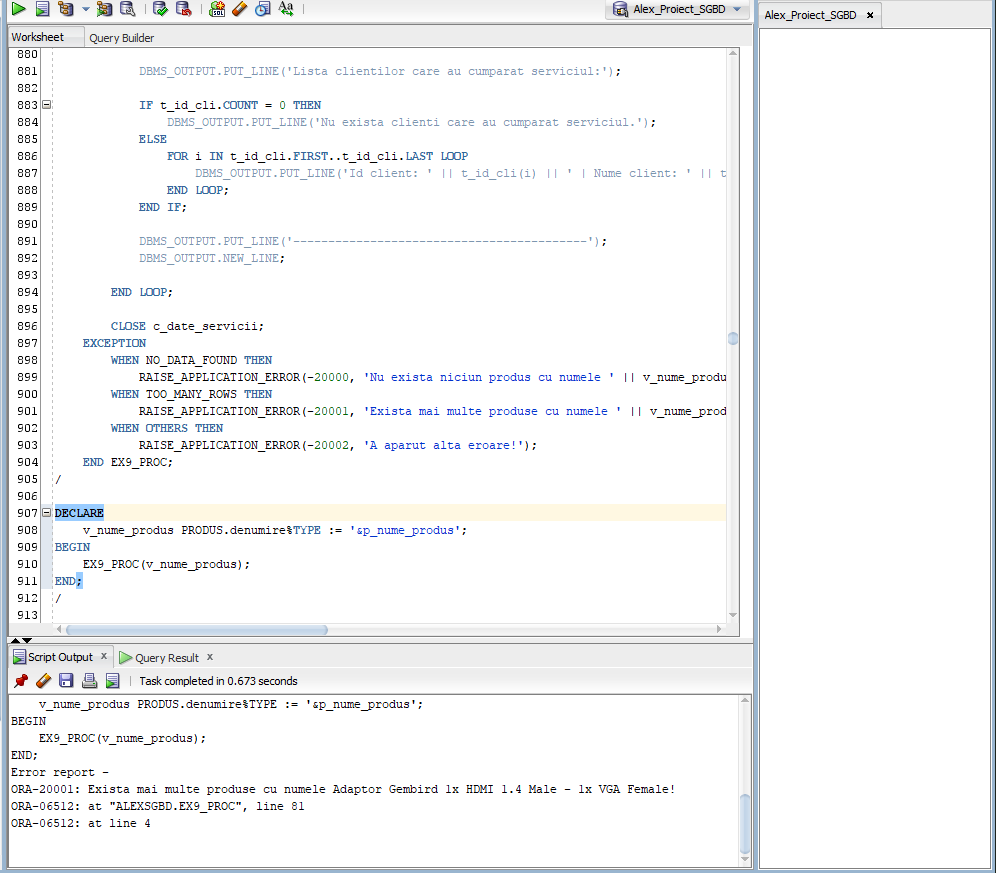












10. Trigger de tip LMD la nivel de comandă:

-- Cerinta 10.:

CREATE OR REPLACE PROCEDURE EX10\_PROC\_AUX

IS

TYPE tab\_imb\_tipuri\_servicii IS TABLE OF SERVICIU.tip%TYPE;

t\_tipuri\_servicii tab\_imb\_tipuri\_servicii;

v\_nr\_servicii NUMBER;

BEGIN

SELECT DISTINCT tip

BULK COLLECT INTO t\_tipuri\_servicii

FROM SERVICIU

ORDER BY 1;

IF t\_tipuri\_servicii.COUNT > 0 THEN

FOR i IN t\_tipuri\_servicii.FIRST..t\_tipuri\_servicii.LAST LOOP

SELECT COUNT(\*)

INTO v\_nr\_servicii

FROM SERVICIU

WHERE tip = t\_tipuri\_servicii(i);

IF v\_nr\_servicii = 1 THEN

DBMS\_OUTPUT.PUT\_LINE(' un serviciu de tipul ' || t\_tipuri\_servicii(i));

ELSE

DBMS\_OUTPUT.PUT\_LINE(' ' || v\_nr\_servicii || ' servicii de tipul ' || t\_tipuri\_servicii(i));

END IF;

END LOOP;

END IF;

END EX10\_PROC\_AUX;

/

CREATE OR REPLACE TRIGGER EX10\_TRIG\_LMD\_COM\_BEFORE

BEFORE INSERT OR DELETE ON SERVICIU

DECLARE

v\_nr\_inregistrari NUMBER;

BEGIN

SELECT COUNT (\*) INTO v\_nr\_inregistrari FROM SERVICIU;

IF INSERTING THEN

IF v\_nr\_inregistrari >= 1000 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Nu se mai pot insera inregistrari in tabel, deoarece acesta este prea plin!');

END IF;

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');

IF v\_nr\_inregistrari <> 1 THEN

DBMS\_OUTPUT.PUT('Inainte de inserare, in tabelul SERVICIU se afla ' || v\_nr\_inregistrari || ' servicii');

IF v\_nr\_inregistrari = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(':');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Inainte de inserare, in tabelul SERVICIU se afla un serviciu:');

END IF;

ELSE

IF v\_nr\_inregistrari <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Nu se mai pot sterge inregistrari din tabel, deoarece acesta este gol!');

END IF;

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');

IF v\_nr\_inregistrari <> 1 THEN

DBMS\_OUTPUT.PUT('Inainte de stergere, in tabelul SERVICIU se afla ' || v\_nr\_inregistrari || ' servicii');

IF v\_nr\_inregistrari = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(':');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Inainte de stergere, in tabelul SERVICIU se afla un serviciu:');

END IF;

END IF;

EX10\_PROC\_AUX;

END EX10\_TRIG\_LMD\_COM\_BEFORE;

/

CREATE OR REPLACE TRIGGER EX10\_TRIG\_LMD\_COM\_AFTER

AFTER INSERT OR DELETE ON SERVICIU

DECLARE

v\_nr\_inregistrari NUMBER;

BEGIN

SELECT COUNT (\*) INTO v\_nr\_inregistrari FROM SERVICIU;

DBMS\_OUTPUT.NEW\_LINE;

IF INSERTING THEN

IF v\_nr\_inregistrari <> 1 THEN

DBMS\_OUTPUT.PUT('Dupa inserare, in tabelul SERVICIU se afla ' || v\_nr\_inregistrari || ' servicii');

IF v\_nr\_inregistrari = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(':');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Dupa inserare, in tabelul SERVICIU se afla un serviciu:');

END IF;

ELSE

IF v\_nr\_inregistrari <> 1 THEN

DBMS\_OUTPUT.PUT('Dupa stergere, in tabelul SERVICIU au mai ramas ' || v\_nr\_inregistrari || ' servicii');

IF v\_nr\_inregistrari = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(':');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Dupa stergere, in tabelul SERVICIU a mai ramas un serviciu:');

END IF;

END IF;

EX10\_PROC\_AUX;

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');

DBMS\_OUTPUT.NEW\_LINE;

END EX10\_TRIG\_LMD\_COM\_AFTER;

/

-- ALTER TRIGGER EX10\_TRIG\_LMD\_COM\_BEFORE DISABLE;

-- ALTER TRIGGER EX10\_TRIG\_LMD\_COM\_BEFORE ENABLE;

-- DROP TRIGGER EX10\_TRIG\_LMD\_COM\_BEFORE;

-- ALTER TRIGGER EX10\_TRIG\_LMD\_COM\_AFTER DISABLE;

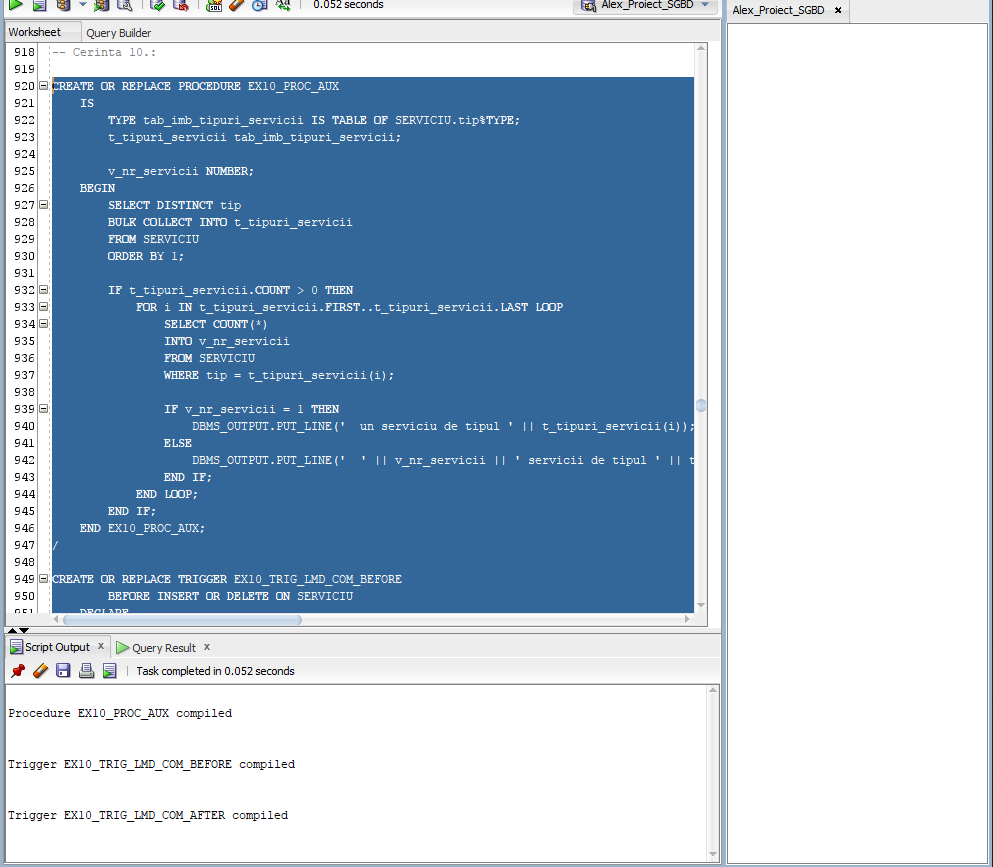
-- ALTER TRIGGER EX10\_TRIG\_LMD\_COM\_AFTER ENABLE;

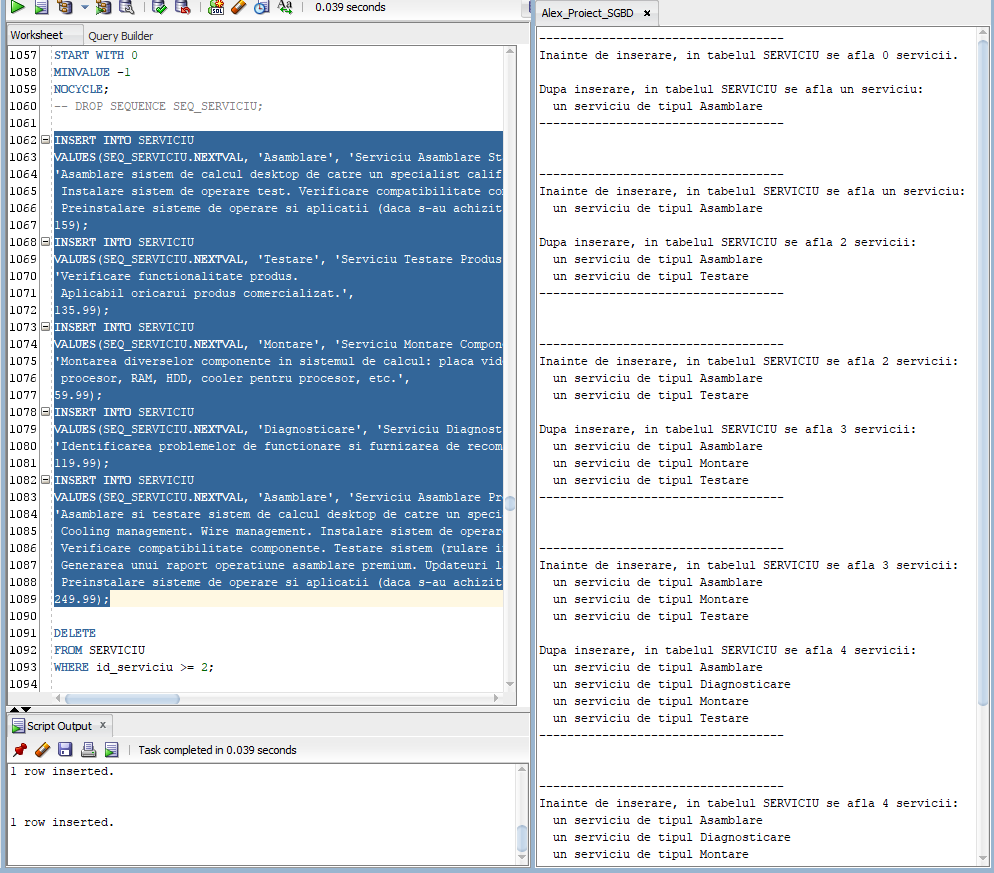
-- DROP TRIGGER EX10\_TRIG\_LMD\_COM\_AFTER;

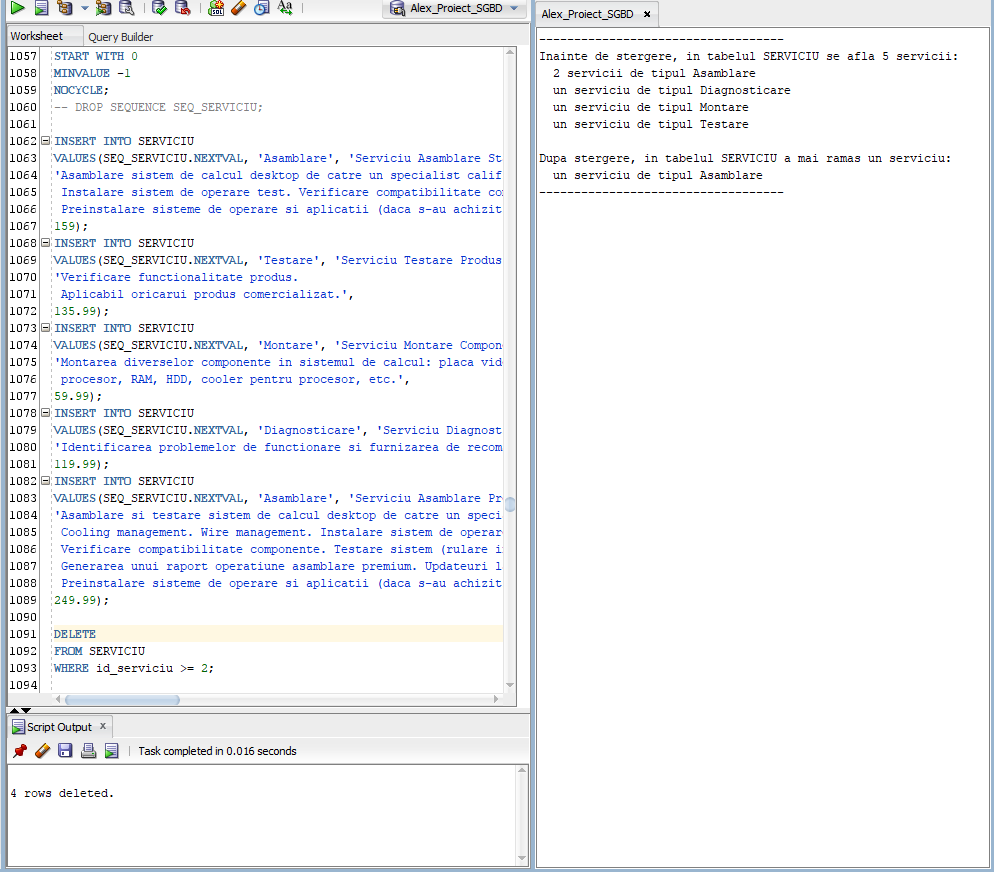
DELETE

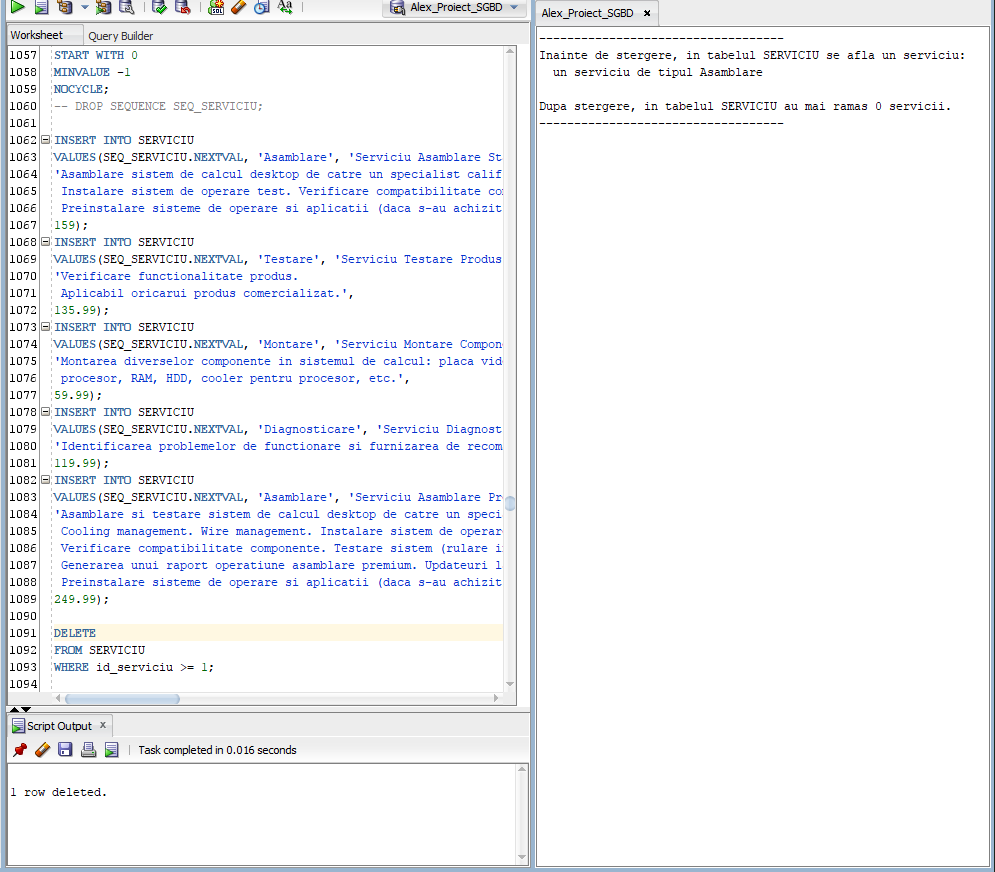
FROM SERVICIU

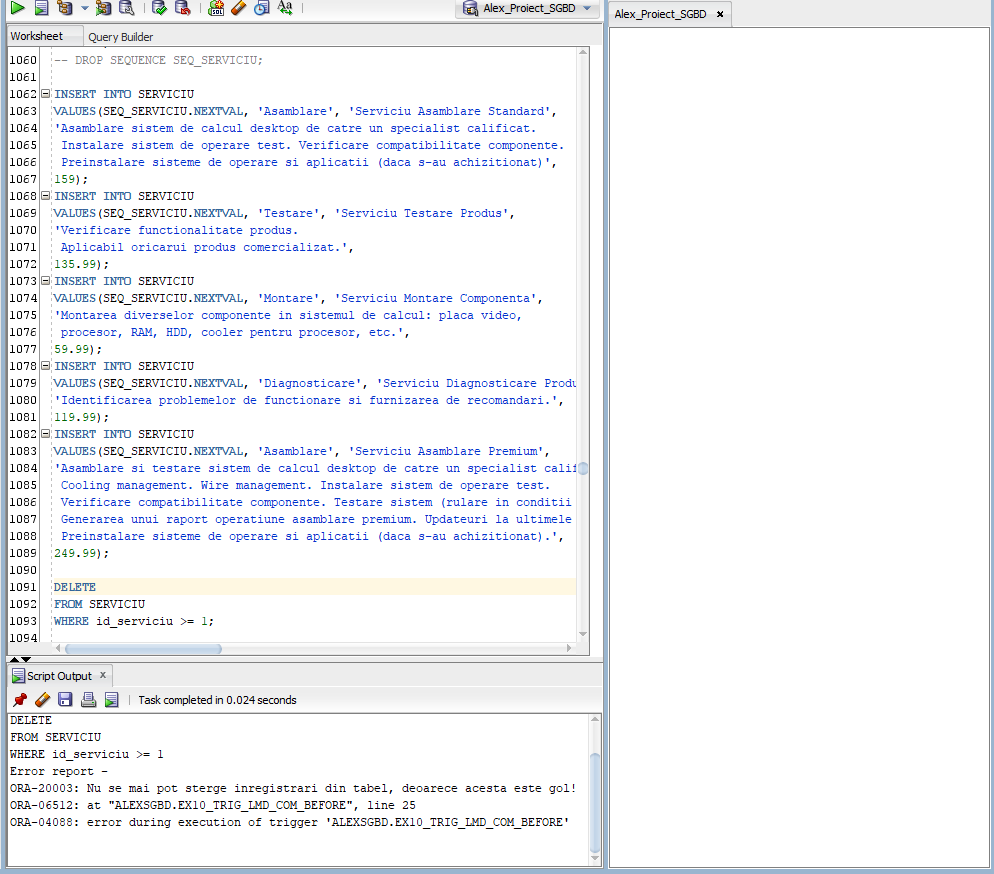
WHERE id\_serviciu >= 1;











11. Trigger de tip LMD la nivel de linie:

-- Cerinta 11.:

CREATE OR REPLACE PACKAGE EX11\_PACHET\_AUX

AS

TYPE tab\_imb\_id\_min IS TABLE OF SERVICIU.id\_serviciu%TYPE;

TYPE tab\_imb\_id\_max IS TABLE OF SERVICIU.id\_serviciu%TYPE;

TYPE PRET\_SERVICIU\_REC IS RECORD

(min SERVICIU.pret%TYPE, t\_id\_min tab\_imb\_id\_min,

max SERVICIU.pret%TYPE, t\_id\_max tab\_imb\_id\_max,

avg SERVICIU.pret%TYPE);

v\_pret PRET\_SERVICIU\_REC;

END EX11\_PACHET\_AUX;

/

CREATE OR REPLACE TRIGGER EX11\_TRIG\_LMD\_COM\_BEFORE

BEFORE UPDATE OF pret ON SERVICIU

BEGIN

SELECT MIN(pret), MAX(pret), ROUND(AVG(pret), 2)

INTO EX11\_PACHET\_AUX.v\_pret.min, EX11\_PACHET\_AUX.v\_pret.max, EX11\_PACHET\_AUX.v\_pret.avg

FROM SERVICIU;

SELECT id\_serviciu

BULK COLLECT INTO EX11\_PACHET\_AUX.v\_pret.t\_id\_min

FROM SERVICIU

WHERE pret = EX11\_PACHET\_AUX.v\_pret.min;

SELECT id\_serviciu

BULK COLLECT INTO EX11\_PACHET\_AUX.v\_pret.t\_id\_max

FROM SERVICIU

WHERE pret = EX11\_PACHET\_AUX.v\_pret.max;

END EX11\_TRIG\_LMD\_COM\_BEFORE;

/

CREATE OR REPLACE TRIGGER EX11\_TRIG\_LMD\_LIN\_BEFORE

BEFORE UPDATE OF pret ON SERVICIU

FOR EACH ROW

BEGIN

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Serviciul cu id-ul ' || :NEW.id\_serviciu || ', si pretul egal cu ' || :OLD.pret || ', va avea pretul egal cu ' || :NEW.pret || '.');

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Date actuale despre preturile din tabelul SERVICIU:');

DBMS\_OUTPUT.PUT(' Id-ul serviciilor care au pretul minim, adica pretul de ' || EX11\_PACHET\_AUX.v\_pret.min || ': ');

FOR i IN EX11\_PACHET\_AUX.v\_pret.t\_id\_min.FIRST..EX11\_PACHET\_AUX.v\_pret.t\_id\_min.LAST LOOP

DBMS\_OUTPUT.PUT(EX11\_PACHET\_AUX.v\_pret.t\_id\_min(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT(' Id-ul serviciilor care au pretul maxim, adica pretul de ' || EX11\_PACHET\_AUX.v\_pret.max || ': ');

FOR i IN EX11\_PACHET\_AUX.v\_pret.t\_id\_max.FIRST..EX11\_PACHET\_AUX.v\_pret.t\_id\_max.LAST LOOP

DBMS\_OUTPUT.PUT(EX11\_PACHET\_AUX.v\_pret.t\_id\_max(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT(' Pretul mediu al serviciilor: ' || EX11\_PACHET\_AUX.v\_pret.avg);

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.NEW\_LINE;

END EX11\_TRIG\_LMD\_LIN\_BEFORE;

/

-- ALTER TRIGGER EX11\_TRIG\_LMD\_COM\_BEFORE DISABLE;

-- ALTER TRIGGER EX11\_TRIG\_LMD\_COM\_BEFORE ENABLE;

-- DROP TRIGGER EX11\_TRIG\_LMD\_COM\_BEFORE;

-- ALTER TRIGGER EX11\_TRIG\_LMD\_LIN\_BEFORE DISABLE;

-- ALTER TRIGGER EX11\_TRIG\_LMD\_LIN\_BEFORE ENABLE;

-- DROP TRIGGER EX11\_TRIG\_LMD\_LIN\_BEFORE;

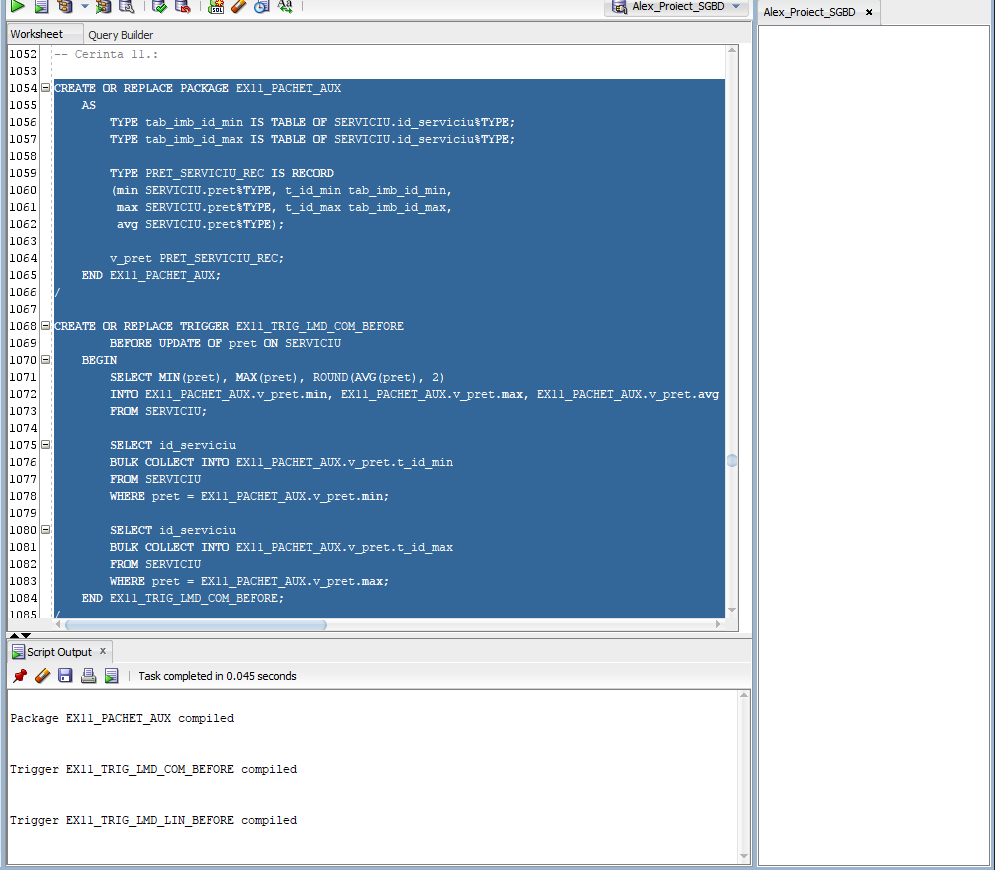
UPDATE SERVICIU

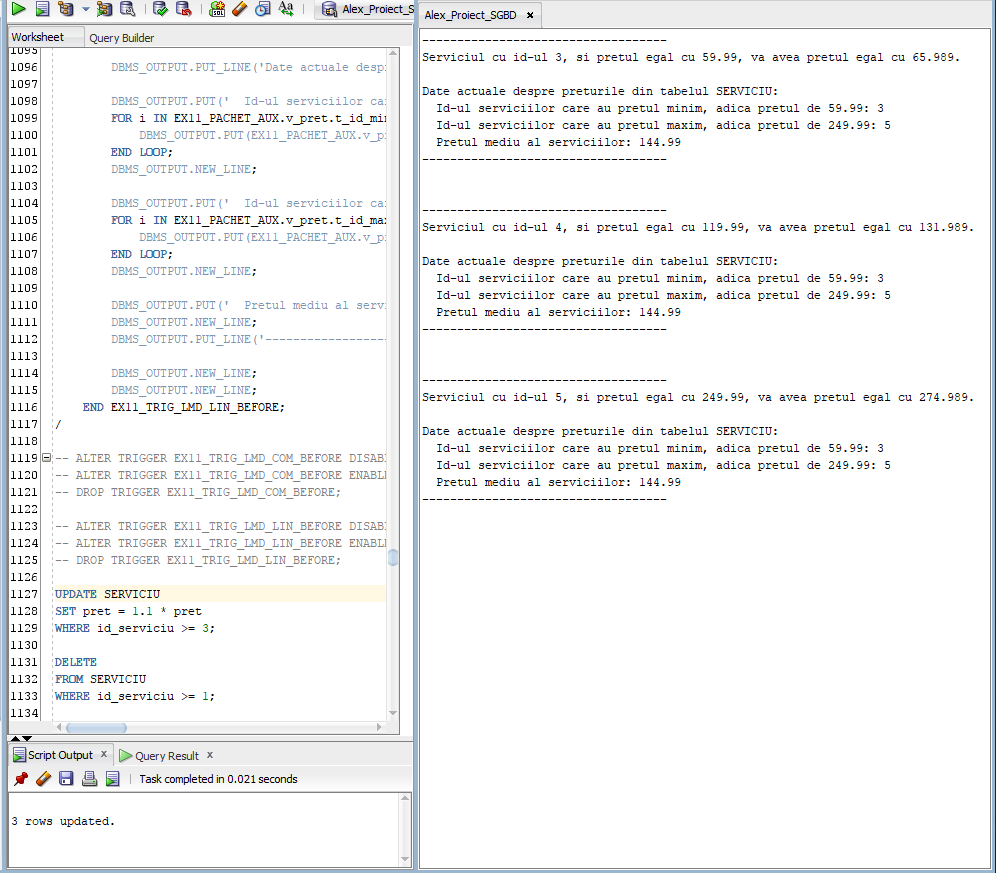
SET pret = 1.1 \* pret

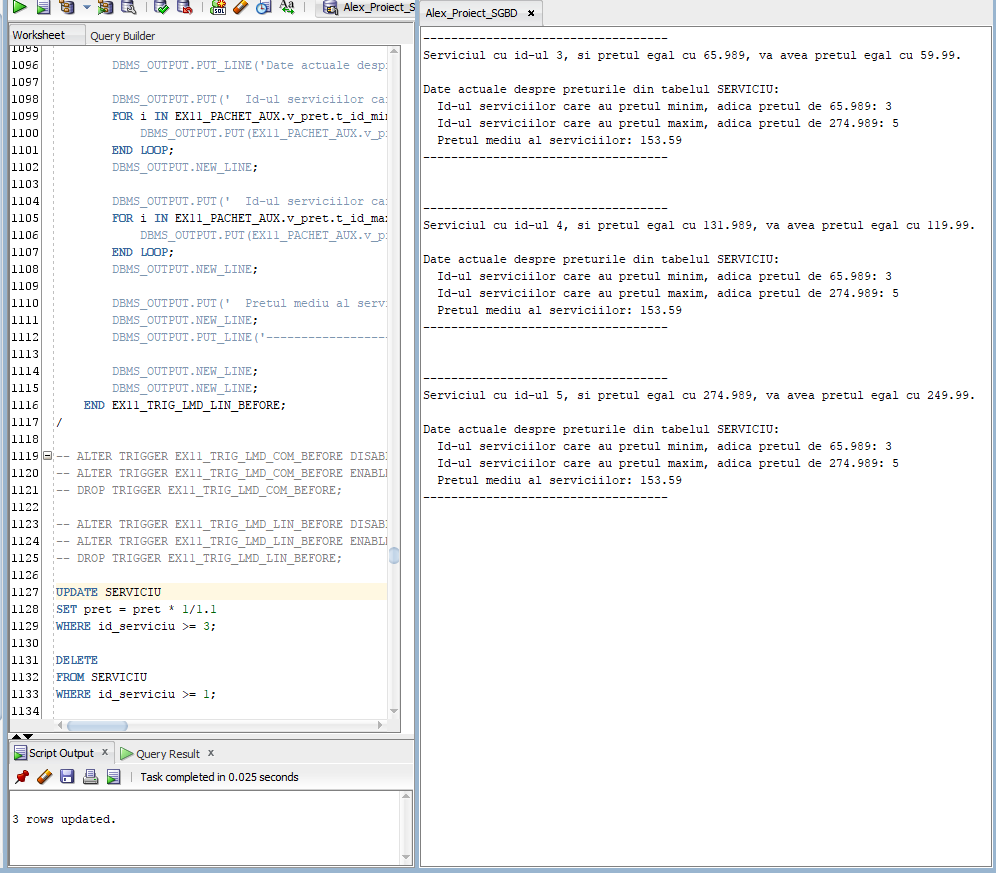
WHERE id\_serviciu >= 3;

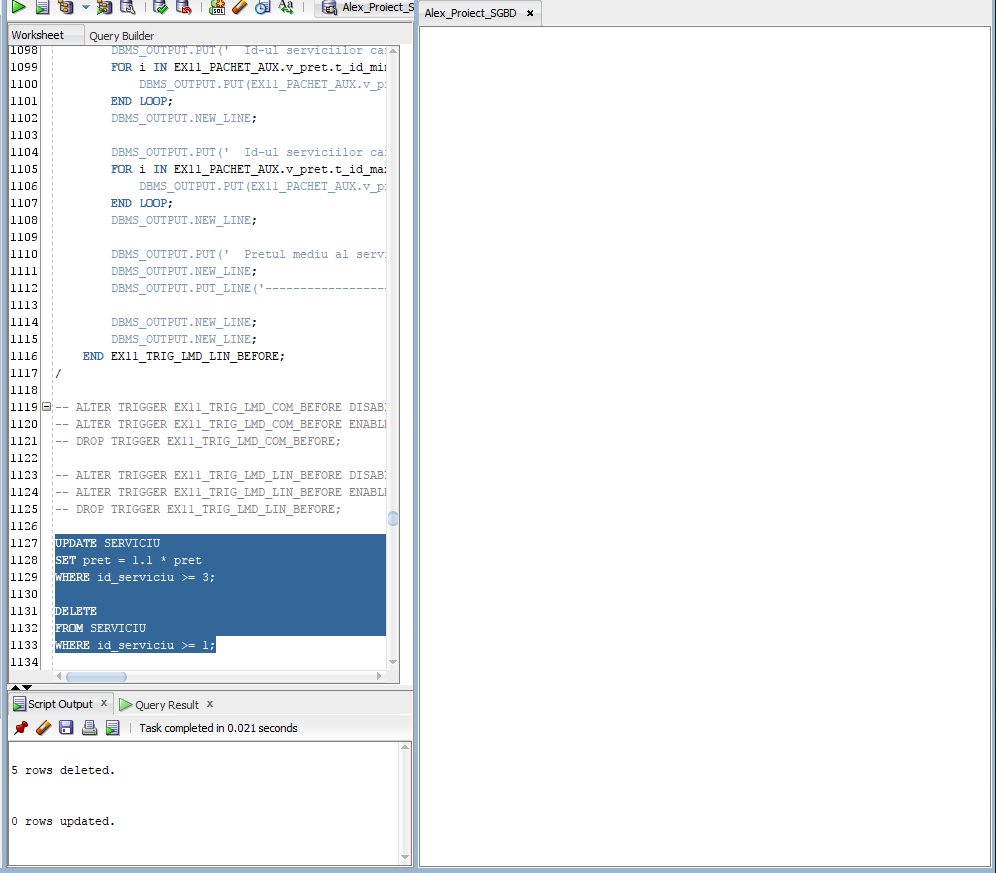
DELETE

FROM SERVICIU

WHERE id\_serviciu >= 1;







12. Trigger de tip LDD:

CREATE TABLE INFORMATII\_COMANDA\_LDD

(nume\_bd VARCHAR2(50),

user\_logat VARCHAR2(30),

eveniment VARCHAR2(20),

data\_comanda TIMESTAMP(3),

tip\_obiect\_referit VARCHAR2(30),

nume\_obiect\_referit VARCHAR2(30)

);

CREATE OR REPLACE TRIGGER EX12\_TRIG\_LDD

AFTER CREATE OR DROP ON SCHEMA

BEGIN

IF SYS.SYSEVENT = 'DROP' THEN

DBMS\_OUTPUT.PUT\_LINE('A avut loc un eveniment de tipul DROP asupra schemei personale, pe un obiect de tipul ' || SYS.DICTIONARY\_OBJ\_TYPE || ', numit ' || SYS.DICTIONARY\_OBJ\_NAME || '.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('A avut loc un eveniment de tipul CREATE asupra schemei personale, pe un obiect de tipul ' || SYS.DICTIONARY\_OBJ\_TYPE || ', numit ' || SYS.DICTIONARY\_OBJ\_NAME || '.');

END IF;

INSERT INTO INFORMATII\_COMANDA\_LDD

VALUES (SYS.DATABASE\_NAME,

SYS.LOGIN\_USER,

SYS.SYSEVENT,

SYSTIMESTAMP(3),

SYS.DICTIONARY\_OBJ\_TYPE,

SYS.DICTIONARY\_OBJ\_NAME

);

END EX12\_TRIG\_LDD;

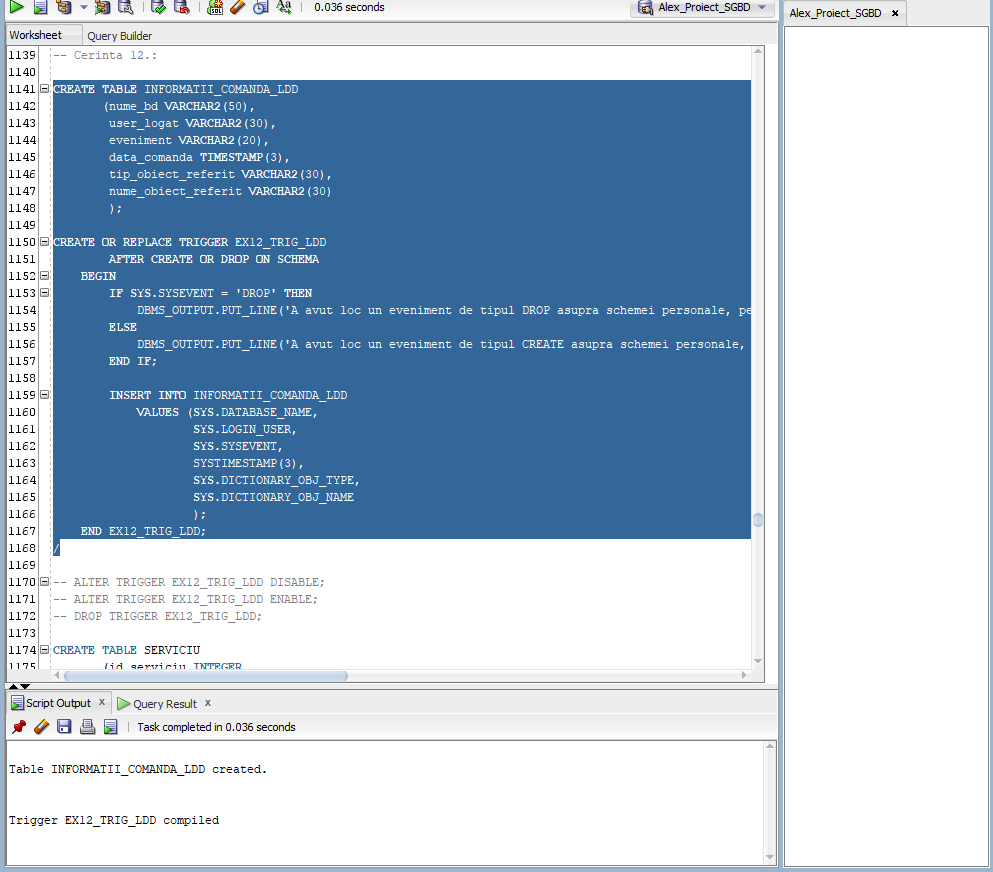
/

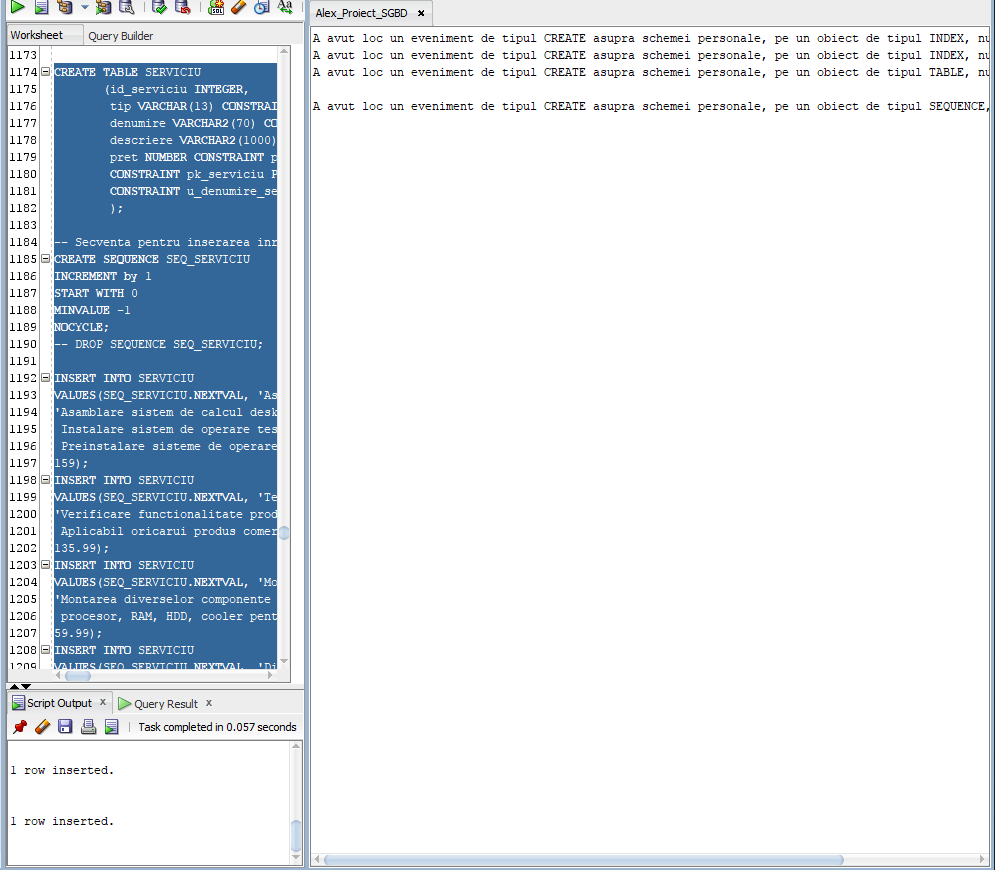
-- ALTER TRIGGER EX12\_TRIG\_LDD DISABLE;

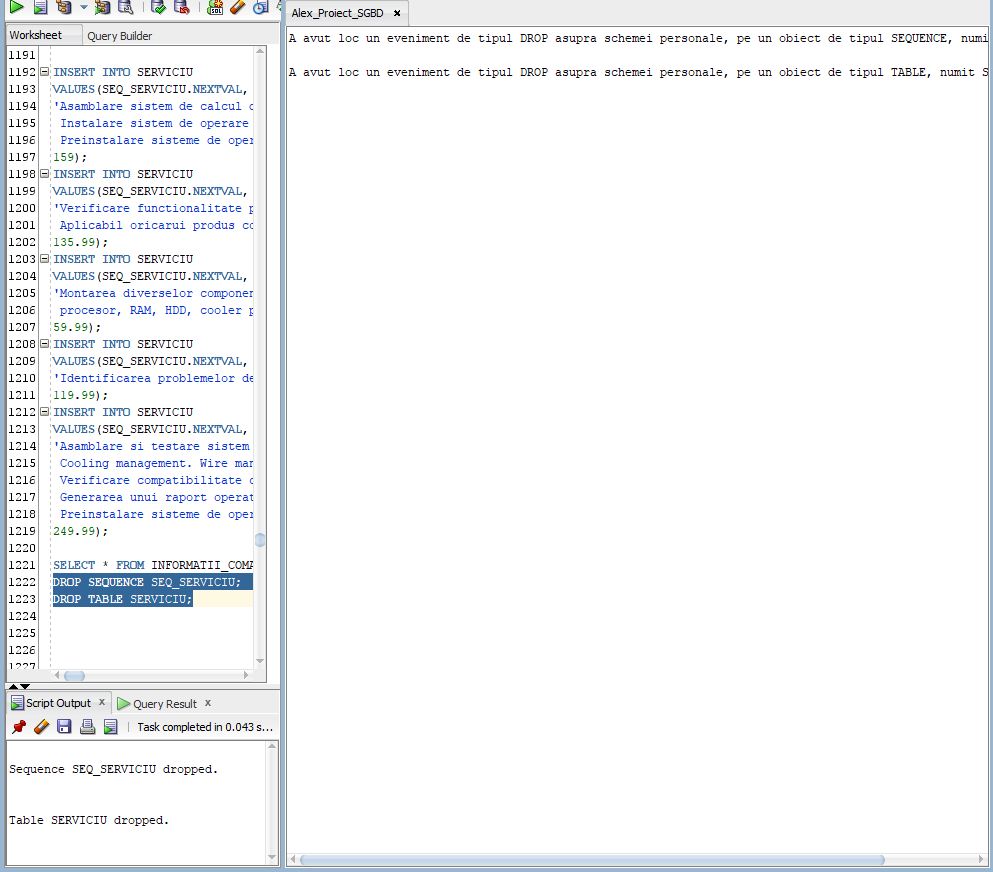
-- ALTER TRIGGER EX12\_TRIG\_LDD ENABLE;

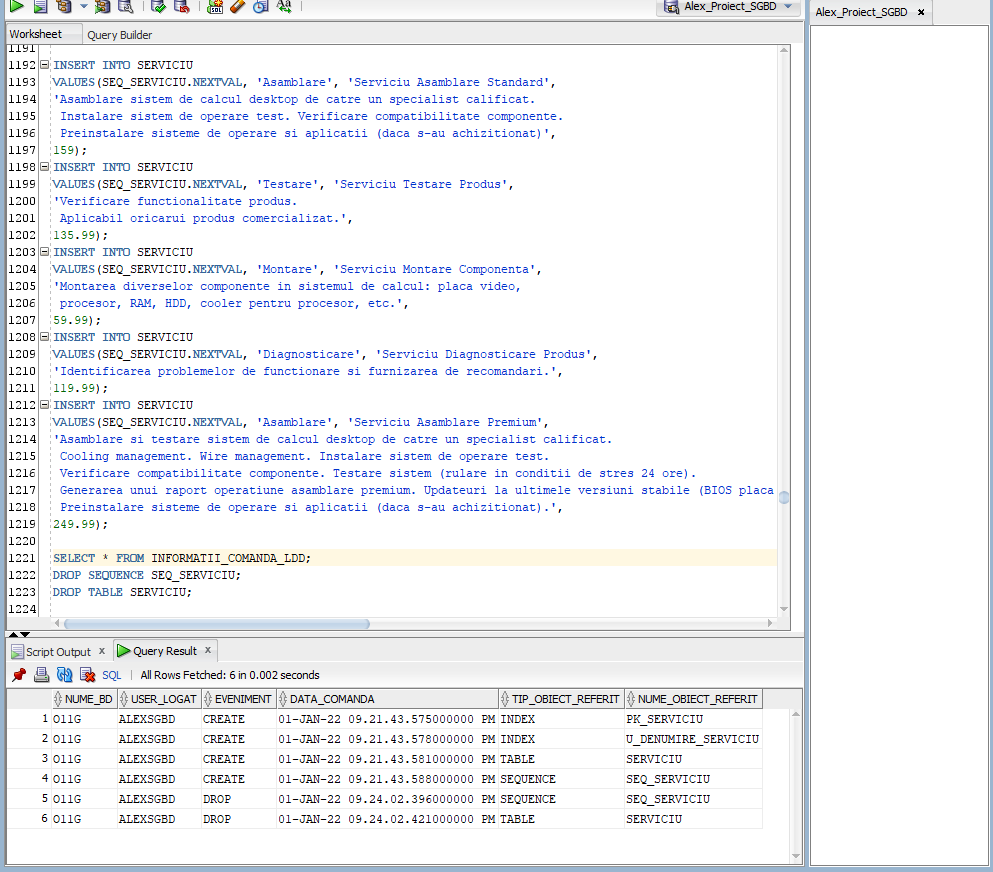
-- DROP TRIGGER EX12\_TRIG\_LDD;

DROP TABLE SERVICIU;

SELECT \* FROM INFORMATII\_COMANDA\_LDD;







13. Pachet care conține toate obiectele definite în cadrul proiectului:

-- Cerinta 13.:

CREATE OR REPLACE PACKAGE EX13\_PACK

IS

PROCEDURE EX6\_PROC(v\_nume IN CLIENT.nume%TYPE);

FUNCTION EX7\_FUNC(p\_id\_ang IN ANGAJAT.id\_angajat%TYPE,

p\_nr\_cli\_asist\_ang OUT NUMBER)

RETURN NUMBER;

FUNCTION EX8\_FUNC(v\_nume\_utilizator IN CONT\_SITE.nume\_utilizator%TYPE)

RETURN NUMBER;

PROCEDURE EX9\_PROC(v\_nume\_produs IN PRODUS.denumire%TYPE);

PROCEDURE EX10\_PROC\_AUX;

END EX13\_PACK;

/

CREATE OR REPLACE PACKAGE BODY EX13\_PACK

IS

PROCEDURE EX6\_PROC

(v\_nume IN CLIENT.nume%TYPE)

IS

v\_prenume CLIENT.prenume%TYPE;

v\_id CLIENT.id\_client%TYPE;

TYPE tab\_imb\_id\_cli IS TABLE OF CLIENT.id\_client%TYPE;

t\_id\_cli tab\_imb\_id\_cli;

v\_nr\_clienti NUMBER;

TYPE tab\_imb\_id\_prod IS TABLE OF PRODUS.id\_produs%TYPE;

t\_id\_prod tab\_imb\_id\_prod;

TYPE tab\_ind\_nume\_prod IS TABLE OF PRODUS.denumire%TYPE

INDEX BY PLS\_INTEGER;

t\_nume\_prod tab\_ind\_nume\_prod;

TYPE vector\_pret\_prod IS VARRAY(2000) OF PRODUS.pret%TYPE;

vect\_pret\_prod vector\_pret\_prod := vector\_pret\_prod();

v\_contor NUMBER;

BEGIN

SELECT COUNT(nume)

INTO v\_nr\_clienti

FROM CLIENT

WHERE UPPER(nume) = UPPER(v\_nume);

IF v\_nr\_clienti = 0 THEN

RAISE NO\_DATA\_FOUND;

ELSE

SELECT id\_client

BULK COLLECT INTO t\_id\_cli

FROM CLIENT

WHERE UPPER(nume) = UPPER(v\_nume);

FOR i IN t\_id\_cli.FIRST..t\_id\_cli.LAST LOOP

SELECT DISTINCT p.id\_produs, denumire, pret

BULK COLLECT INTO t\_id\_prod, t\_nume\_prod, vect\_pret\_prod

FROM PRODUS p JOIN COMANDA co ON p.id\_produs = co.id\_produs

JOIN CLIENT cl ON co.id\_client = cl.id\_client

WHERE cl.id\_client = t\_id\_cli(i)

ORDER BY 2;

SELECT prenume

INTO v\_prenume

FROM CLIENT

WHERE id\_client = t\_id\_cli(i);

IF t\_id\_prod.COUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Clientul cu id-ul ' || t\_id\_cli(i) || ', pe nume ' || v\_nume || ' ' || v\_prenume || ', nu a comandat niciun produs!');

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

ELSE

v\_contor := 1;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Clientul cu id-ul ' || t\_id\_cli(i) || ', pe nume ' || v\_nume || ' ' || v\_prenume || ', a achizitionat produsele: ');

DBMS\_OUTPUT.NEW\_LINE;

FOR i IN t\_id\_prod.FIRST..t\_id\_prod.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(v\_contor || '.');

DBMS\_OUTPUT.PUT\_LINE('Id produs: ' || t\_id\_prod(i));

DBMS\_OUTPUT.PUT\_LINE('Denumire produs: ' || t\_nume\_prod(i));

DBMS\_OUTPUT.PUT\_LINE('Pret produs: ' || vect\_pret\_prod(i));

IF i <> t\_id\_prod.LAST THEN

DBMS\_OUTPUT.NEW\_LINE;

END IF;

v\_contor := v\_contor + 1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

END IF;

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista niciun client cu numele ' || v\_nume || '!');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut alta eroare!');

END EX6\_PROC;

FUNCTION EX7\_FUNC

(p\_id\_ang IN ANGAJAT.id\_angajat%TYPE,

p\_nr\_cli\_asist\_ang OUT NUMBER)

RETURN NUMBER

IS

v\_nr\_ang\_activi NUMBER;

v\_nr\_cli\_asist\_ang NUMBER;

BEGIN

p\_nr\_cli\_asist\_ang := 0;

FOR i IN (SELECT id\_angajat, nume || ' ' || prenume nume\_ang

FROM ANGAJAT a

ORDER BY 2) LOOP

SELECT COUNT(DISTINCT nume || ' ' || prenume)

INTO v\_nr\_cli\_asist\_ang

FROM CLIENT c JOIN ESTE\_ASISTAT ea ON c.id\_client = ea.id\_client

WHERE ea.id\_angajat = i.id\_angajat;

IF v\_nr\_cli\_asist\_ang > 0 THEN

v\_nr\_ang\_activi := v\_nr\_ang\_activi + 1;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Id angajat: ' || i.id\_angajat);

DBMS\_OUTPUT.PUT\_LINE('Nume angajat: ' || i.nume\_ang);

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Clienti asistati de catre angajat: ');

FOR j IN (SELECT DISTINCT nume || ' ' || prenume nume\_cli

FROM CLIENT c JOIN ESTE\_ASISTAT ea ON c.id\_client = ea.id\_client

WHERE ea.id\_angajat = i.id\_angajat) LOOP

DBMS\_OUTPUT.PUT\_LINE(j.nume\_cli);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.NEW\_LINE;

END IF;

IF i.id\_angajat = p\_id\_ang THEN

p\_nr\_cli\_asist\_ang := v\_nr\_cli\_asist\_ang;

END IF;

END LOOP;

RETURN v\_nr\_ang\_activi;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare necunoscuta!');

END EX7\_FUNC;

FUNCTION EX8\_FUNC

(v\_nume\_utilizator IN CONT\_SITE.nume\_utilizator%TYPE)

RETURN NUMBER

IS

v\_id\_cont CONT\_SITE.id\_cont%TYPE;

v\_id\_client CLIENT.id\_client%TYPE;

v\_id\_recenzie RECENZIE.id\_recenzie%TYPE;

v\_nr\_stele RECENZIE.nr\_stele%TYPE;

v\_nr\_stele\_precedent RECENZIE.nr\_stele%TYPE := 0;

v\_nume\_produs PRODUS.denumire%TYPE;

v\_nr\_stele\_min RECENZIE.nr\_stele%TYPE;

TYPE ref\_info\_recenzii IS REF CURSOR;

c\_info\_recenzii ref\_info\_recenzii;

CURSOR c\_recenzii IS

SELECT id\_client,

CURSOR (SELECT id\_recenzie, nr\_stele, denumire

FROM RECENZIE r JOIN PRODUS p ON r.id\_produs = p.id\_produs

WHERE nume\_utilizator = v\_nume\_utilizator

ORDER BY 2 DESC)

FROM CONT\_SITE

WHERE nume\_utilizator = v\_nume\_utilizator;

v\_contor NUMBER := 0;

v\_top NUMBER := 0;

BEGIN

SELECT id\_cont

INTO v\_id\_cont

FROM CONT\_SITE

WHERE nume\_utilizator = v\_nume\_utilizator;

DBMS\_OUTPUT.PUT\_LINE('Id cont: ' || v\_id\_cont);

DBMS\_OUTPUT.PUT\_LINE('Nume utilizator cont: ' || v\_nume\_utilizator);

OPEN c\_recenzii;

FETCH c\_recenzii INTO v\_id\_client, c\_info\_recenzii;

IF v\_id\_client IS NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Contul nu este detinut de catre un client cunoscut.');

ElSE

DBMS\_OUTPUT.PUT\_LINE('Id client: ' || v\_id\_client);

END IF;

DBMS\_OUTPUT.NEW\_LINE;

SELECT MIN(distincte)

INTO v\_nr\_stele\_min

FROM (SELECT DISTINCT nr\_stele distincte

FROM RECENZIE

WHERE nume\_utilizator = v\_nume\_utilizator

ORDER BY distincte DESC)

WHERE ROWNUM <= 3;

FETCH c\_info\_recenzii INTO v\_id\_recenzie, v\_nr\_stele, v\_nume\_produs;

IF c\_info\_recenzii%ROWCOUNT = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('Top 3 recenzii ale contului: ');

LOOP

v\_contor := v\_contor + 1;

IF v\_nr\_stele <> v\_nr\_stele\_precedent THEN

v\_top := v\_top + 1;

v\_nr\_stele\_precedent := v\_nr\_stele;

END IF;

IF v\_nr\_stele >= v\_nr\_stele\_min THEN

DBMS\_OUTPUT.PUT\_LINE(v\_top || '. Id recenzie: ' || v\_id\_recenzie || ' | Nr. stele: ' || v\_nr\_stele || ' | Nume produs: ' || v\_nume\_produs);

END IF;

FETCH c\_info\_recenzii INTO v\_id\_recenzie, v\_nr\_stele, v\_nume\_produs;

EXIT WHEN c\_info\_recenzii%NOTFOUND;

END LOOP;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Utilizatorul nu a scris nicio recenzie.');

END IF;

DBMS\_OUTPUT.NEW\_LINE;

CLOSE c\_recenzii;

RETURN v\_contor;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista niciun cont cu numele de utilizator ' || v\_nume\_utilizator || '!');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'A aparut alta eroare!');

END EX8\_FUNC;

PROCEDURE EX9\_PROC

(v\_nume\_produs IN PRODUS.denumire%TYPE)

IS

v\_id\_produs PRODUS.id\_produs%TYPE;

v\_id\_serviciu SERVICIU.id\_serviciu%TYPE;

v\_nume\_serviciu SERVICIU.denumire%TYPE;

TYPE tab\_imb\_id\_cli IS TABLE OF CLIENT.id\_client%TYPE;

t\_id\_cli tab\_imb\_id\_cli;

TYPE tab\_imb\_nume\_cli IS TABLE OF CLIENT.nume%TYPE;

t\_nume\_cli tab\_imb\_nume\_cli;

TYPE ref\_info\_cli IS REF CURSOR;

c\_info\_cli ref\_info\_cli;

CURSOR c\_date\_servicii IS

SELECT s.id\_serviciu, s.denumire,

CURSOR (SELECT DISTINCT cl.id\_client, nume || ' ' || prenume

FROM CLIENT cl JOIN CUMPARA cu ON cl.id\_client = cu.id\_client

JOIN SERVICIU si ON cu.id\_serviciu = si.id\_serviciu

WHERE si.id\_serviciu = s.id\_serviciu

ORDER BY 2)

FROM SERVICIU s JOIN ESTE\_COMPATIBIL ec ON s.id\_serviciu = ec.id\_serviciu

JOIN PRODUS p ON ec.id\_produs = p.id\_produs

WHERE UPPER(p.denumire) = UPPER(v\_nume\_produs)

ORDER BY 2;

BEGIN

SELECT id\_produs

INTO v\_id\_produs

FROM PRODUS

WHERE UPPER(denumire) = UPPER(v\_nume\_produs);

OPEN c\_date\_servicii;

DBMS\_OUTPUT.PUT\_LINE('Id produs: ' || v\_id\_produs);

DBMS\_OUTPUT.PUT\_LINE('Nume produs: ' || v\_nume\_produs);

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Lista serviciilor compatibile cu produsul:');

LOOP

FETCH c\_date\_servicii INTO v\_id\_serviciu, v\_nume\_serviciu, c\_info\_cli;

IF c\_date\_servicii%NOTFOUND THEN

IF c\_date\_servicii%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista servicii compatibile!');

END IF;

EXIT;

END IF;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Id serviciu: ' || v\_id\_serviciu);

DBMS\_OUTPUT.PUT\_LINE('Nume serviciu: ' || v\_nume\_serviciu);

DBMS\_OUTPUT.NEW\_LINE;

FETCH c\_info\_cli BULK COLLECT INTO t\_id\_cli, t\_nume\_cli;

DBMS\_OUTPUT.PUT\_LINE('Lista clientilor care au cumparat serviciul:');

IF t\_id\_cli.COUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista clienti care au cumparat serviciul.');

ELSE

FOR i IN t\_id\_cli.FIRST..t\_id\_cli.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE('Id client: ' || t\_id\_cli(i) || ' | Nume client: ' || t\_nume\_cli(i));

END LOOP;

END IF;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

CLOSE c\_date\_servicii;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista niciun produs cu numele ' || v\_nume\_produs || '!');

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Exista mai multe produse cu numele ' || v\_nume\_produs || '!');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'A aparut alta eroare!');

END EX9\_PROC;

PROCEDURE EX10\_PROC\_AUX

IS

TYPE tab\_imb\_tipuri\_servicii IS TABLE OF SERVICIU.tip%TYPE;

t\_tipuri\_servicii tab\_imb\_tipuri\_servicii;

v\_nr\_servicii NUMBER;

BEGIN

SELECT DISTINCT tip

BULK COLLECT INTO t\_tipuri\_servicii

FROM SERVICIU

ORDER BY 1;

IF t\_tipuri\_servicii.COUNT > 0 THEN

FOR i IN t\_tipuri\_servicii.FIRST..t\_tipuri\_servicii.LAST LOOP

SELECT COUNT(\*)

INTO v\_nr\_servicii

FROM SERVICIU

WHERE tip = t\_tipuri\_servicii(i);

IF v\_nr\_servicii = 1 THEN

DBMS\_OUTPUT.PUT\_LINE(' un serviciu de tipul ' || t\_tipuri\_servicii(i));

ELSE

DBMS\_OUTPUT.PUT\_LINE(' ' || v\_nr\_servicii || ' servicii de tipul ' || t\_tipuri\_servicii(i));

END IF;

END LOOP;

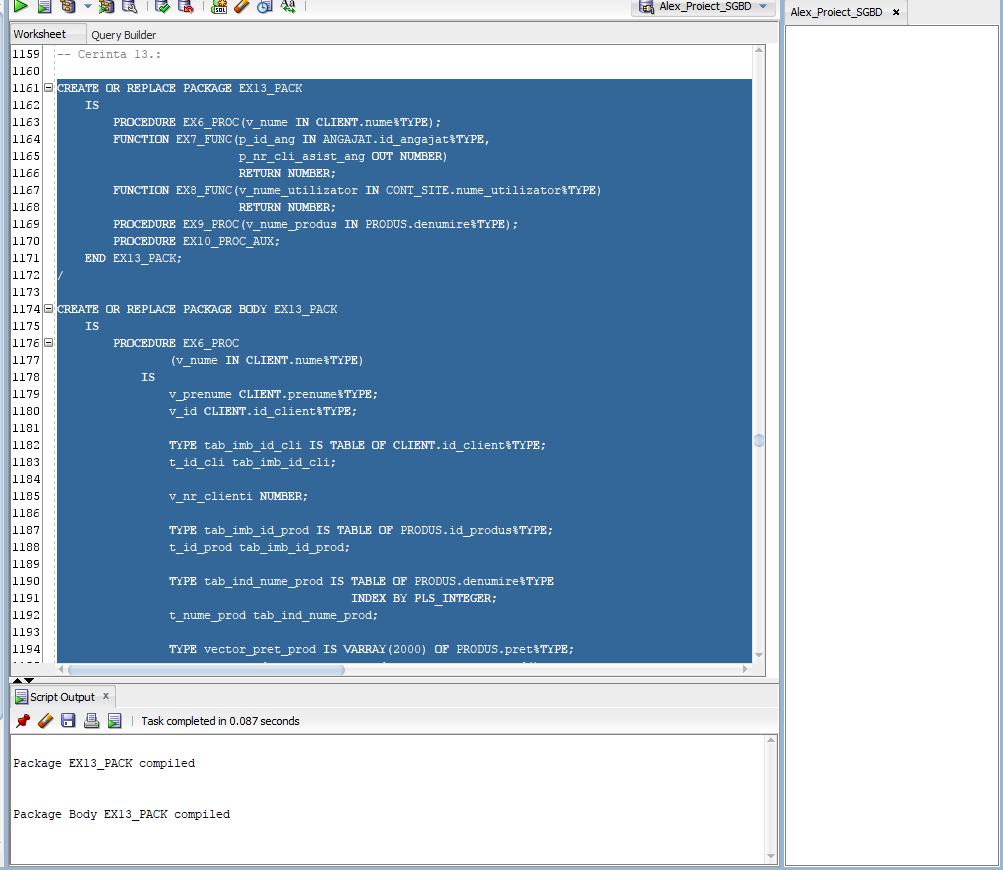
END IF;

END EX10\_PROC\_AUX;

END EX13\_PACK;

/

-- DROP PACKAGE EX13\_PACK;



-- PENTRU EX13\_PACK.EX6\_PROC

DECLARE

v\_nume CLIENT.nume%TYPE := '&p\_nume';

BEGIN

EX13\_PACK.EX6\_PROC(v\_nume);

END;

/

-- PENTRU EX13\_PACK.EX7\_FUNC

DECLARE

v\_id\_ang ANGAJAT.id\_angajat%TYPE := &p\_id;

v\_nr\_cli\_asist\_ang NUMBER;

v\_nr\_ang NUMBER;

BEGIN

SELECT COUNT(id\_angajat)

INTO v\_nr\_ang

FROM ANGAJAT;

SELECT id\_angajat

INTO v\_id\_ang

FROM ANGAJAT

WHERE id\_angajat = v\_id\_ang;

IF EX13\_PACK.EX7\_FUNC(v\_id\_ang, v\_nr\_cli\_asist\_ang) < v\_nr\_ang/2 THEN

DBMS\_OUTPUT.PUT\_LINE('Status personal:');

DBMS\_OUTPUT.PUT\_LINE('Personal inactiv!');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Status personal:');

DBMS\_OUTPUT.PUT\_LINE('Personal activ!');

END IF;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Numarul de clienti asistati de catre angajatul cu id-ul ' || v\_id\_ang || ': ' || v\_nr\_cli\_asist\_ang || '.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista niciun angajat cu id-ul ' || v\_id\_ang || '!');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut alta eroare!');

END;

/

-- PENTRU EX13\_PACK.EX8\_FUNC

DECLARE

v\_nume\_utilizator CONT\_SITE.nume\_utilizator%TYPE := '&p\_nume\_utilizator';

v\_nr\_recenzii NUMBER;

BEGIN

v\_nr\_recenzii := EX13\_PACK.EX8\_FUNC(v\_nume\_utilizator);

IF v\_nr\_recenzii > 0 THEN

IF v\_nr\_recenzii > 1 THEN

DBMS\_OUTPUT.PUT\_LINE('Contul cu numele de utilizator ' || v\_nume\_utilizator || ' a scris ' || v\_nr\_recenzii || ' recenzii.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Contul cu numele de utilizator ' || v\_nume\_utilizator || ' a scris o recenzie.');

END IF;

END IF;

END;

/

-- PENTRU EX13\_PACK.EX9\_PROC

DECLARE

v\_nume\_produs PRODUS.denumire%TYPE := '&p\_nume\_produs';

BEGIN

EX13\_PACK.EX9\_PROC(v\_nume\_produs);

END;

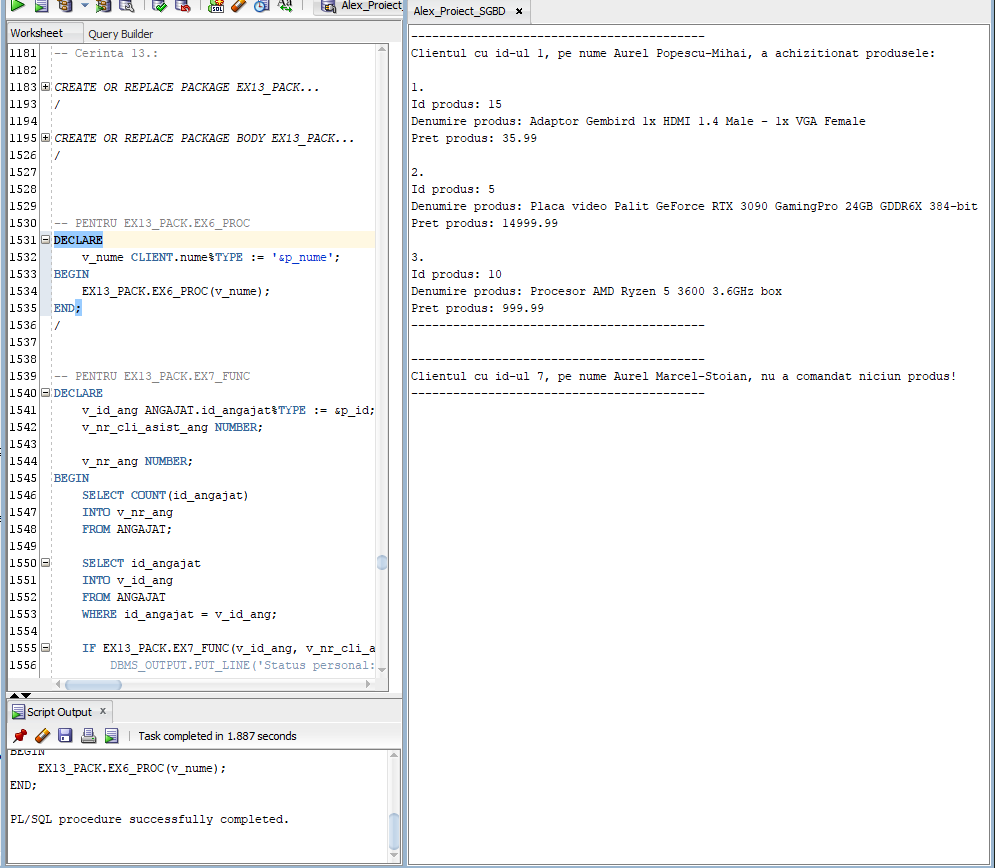
/

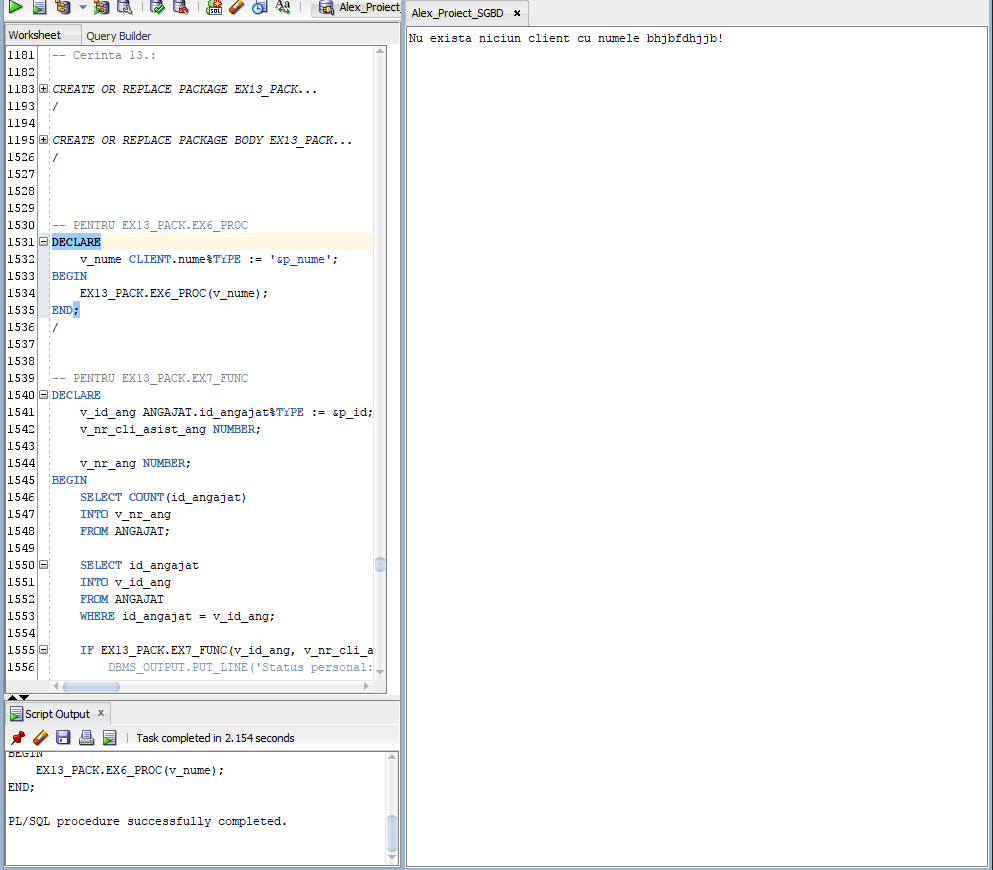
-- PENTRU EX13\_PACK.EX10\_PROC\_AUX

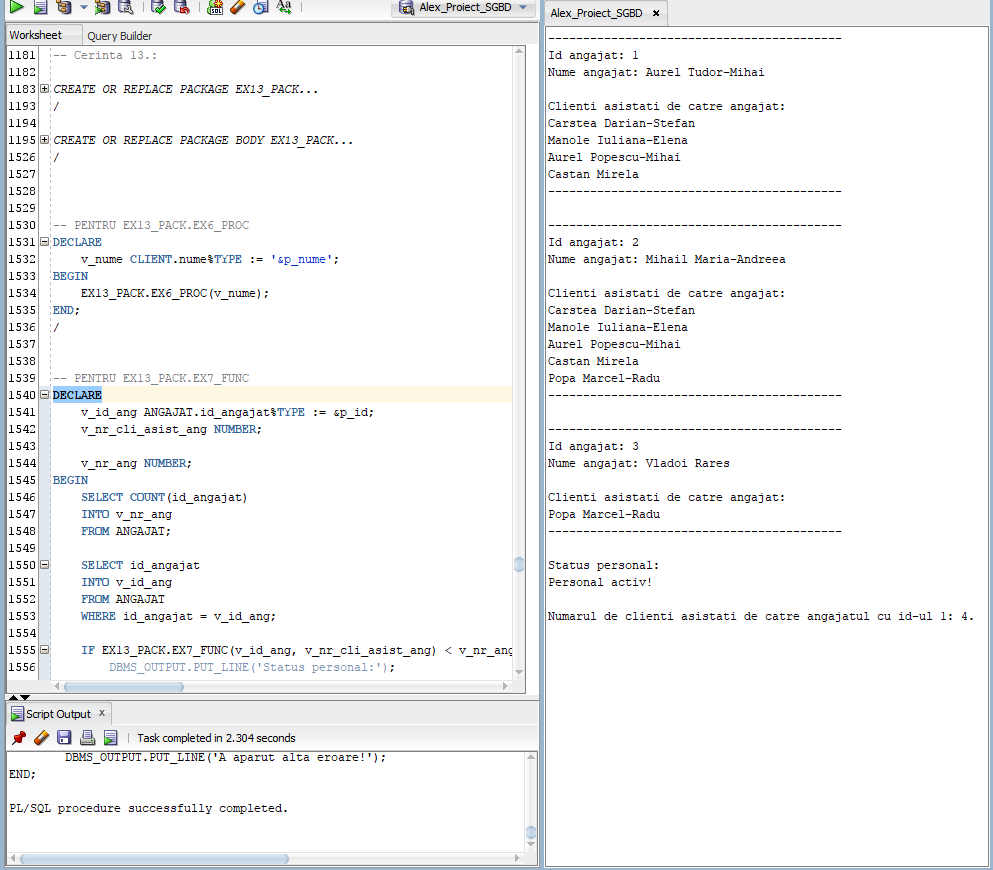
BEGIN

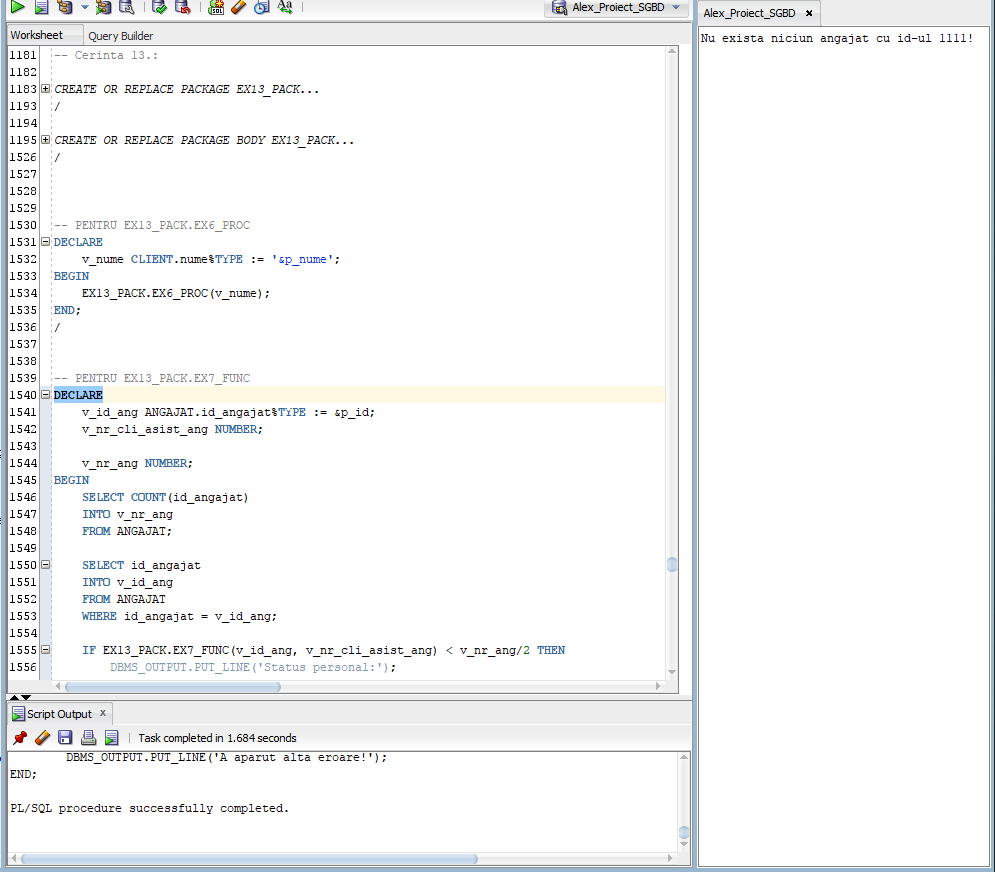
EX13\_PACK.EX10\_PROC\_AUX;

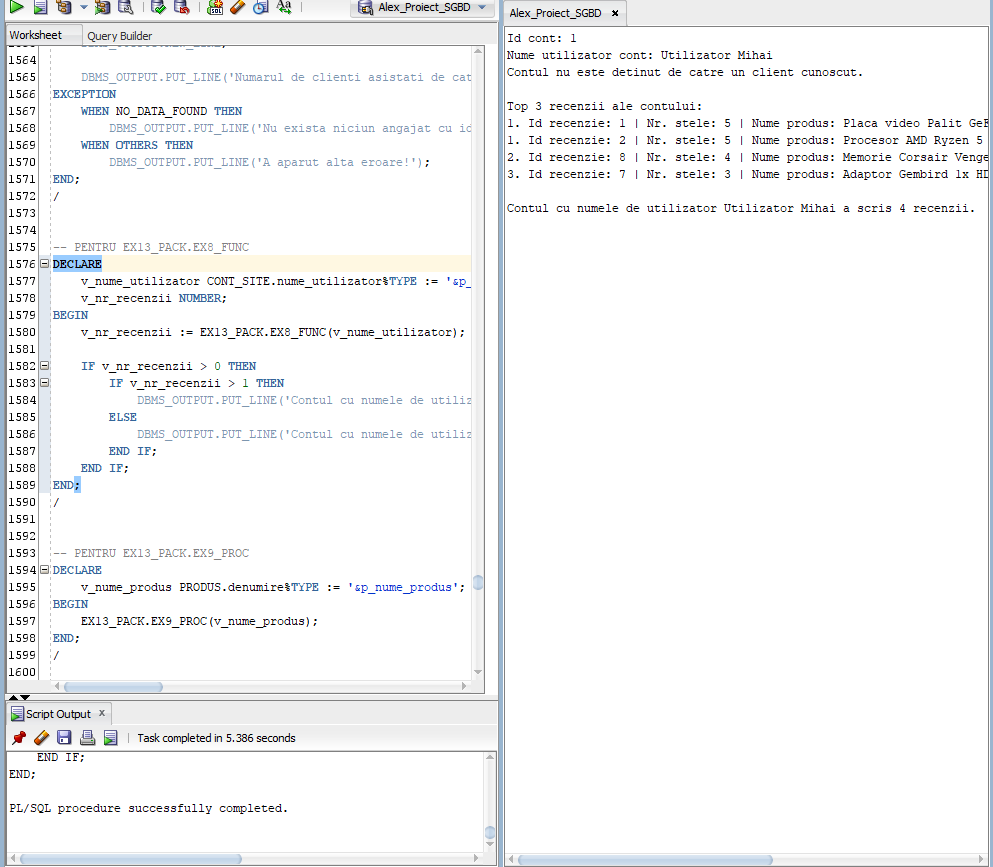
END;

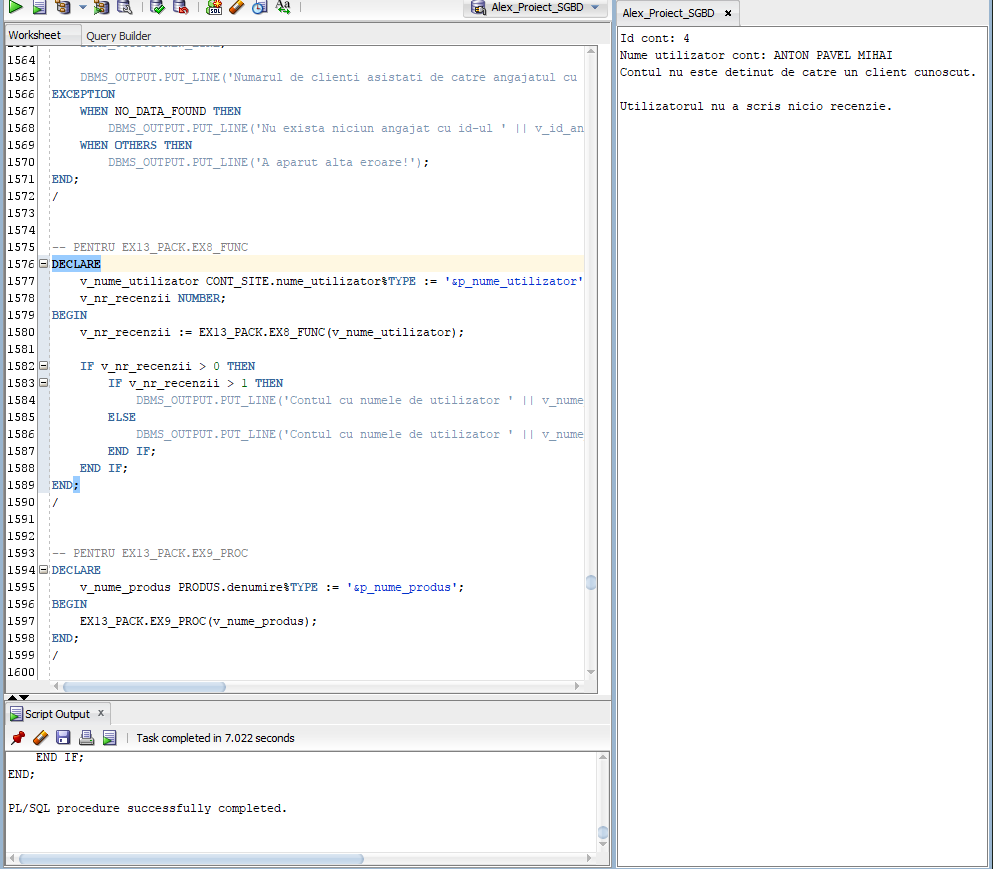
/

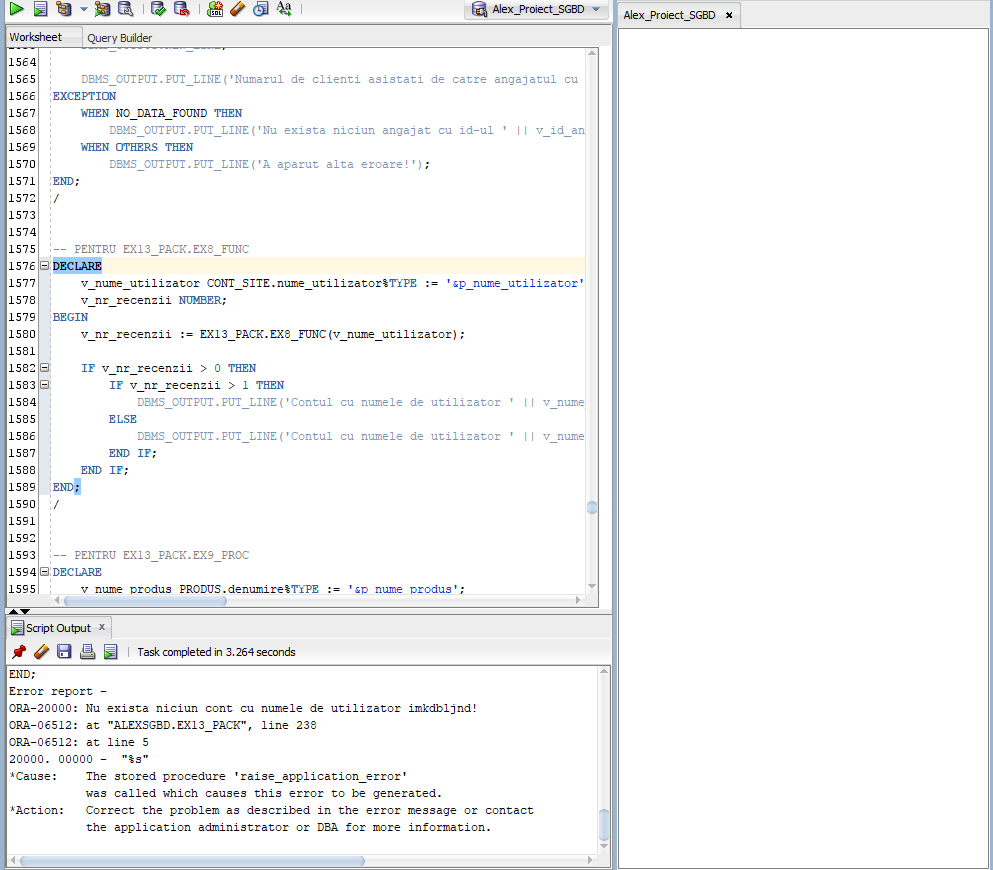


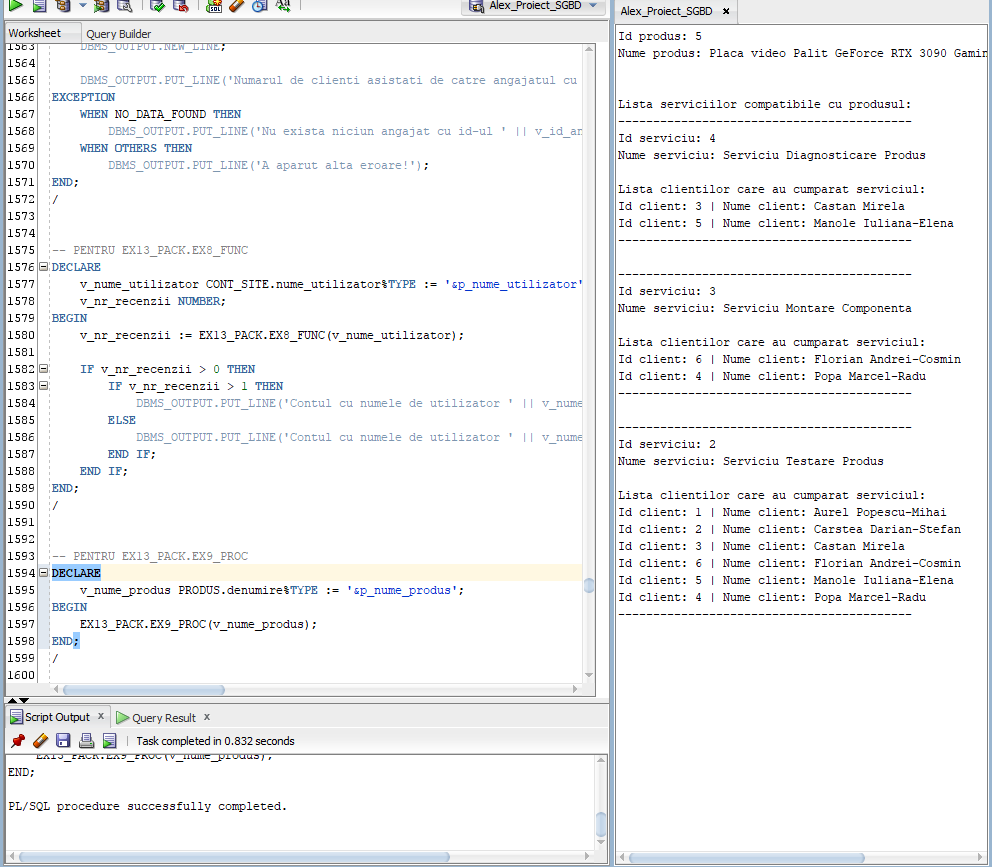


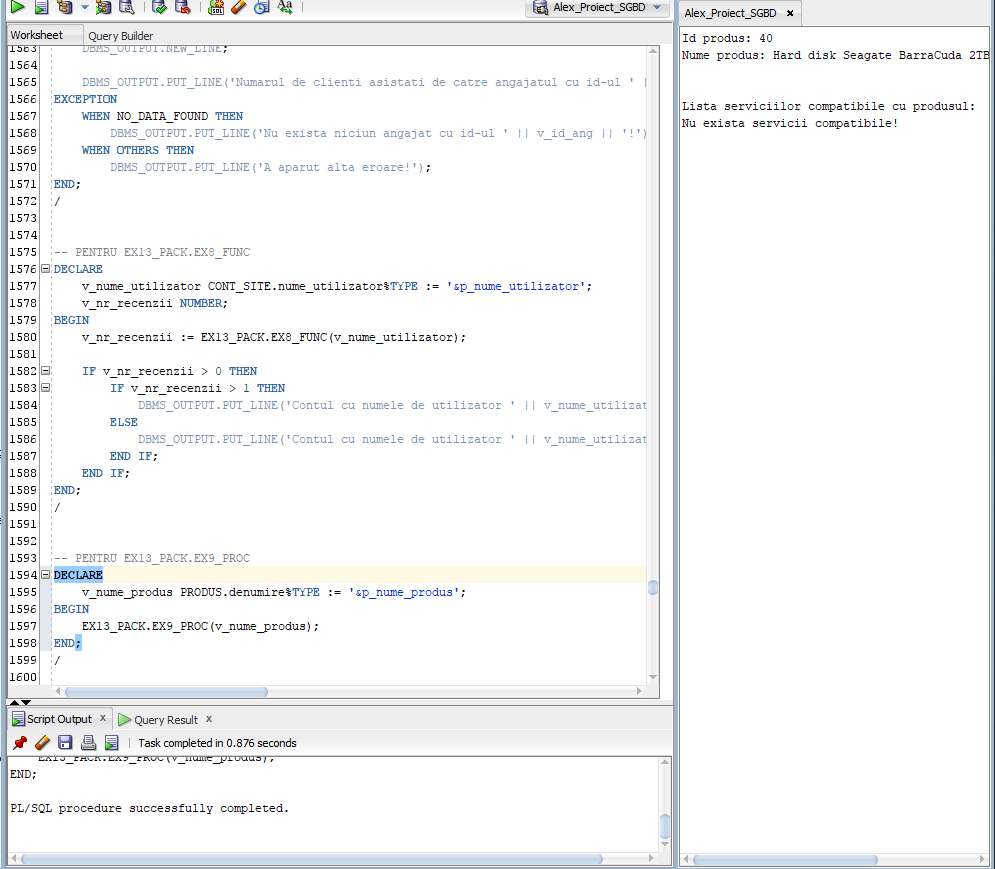


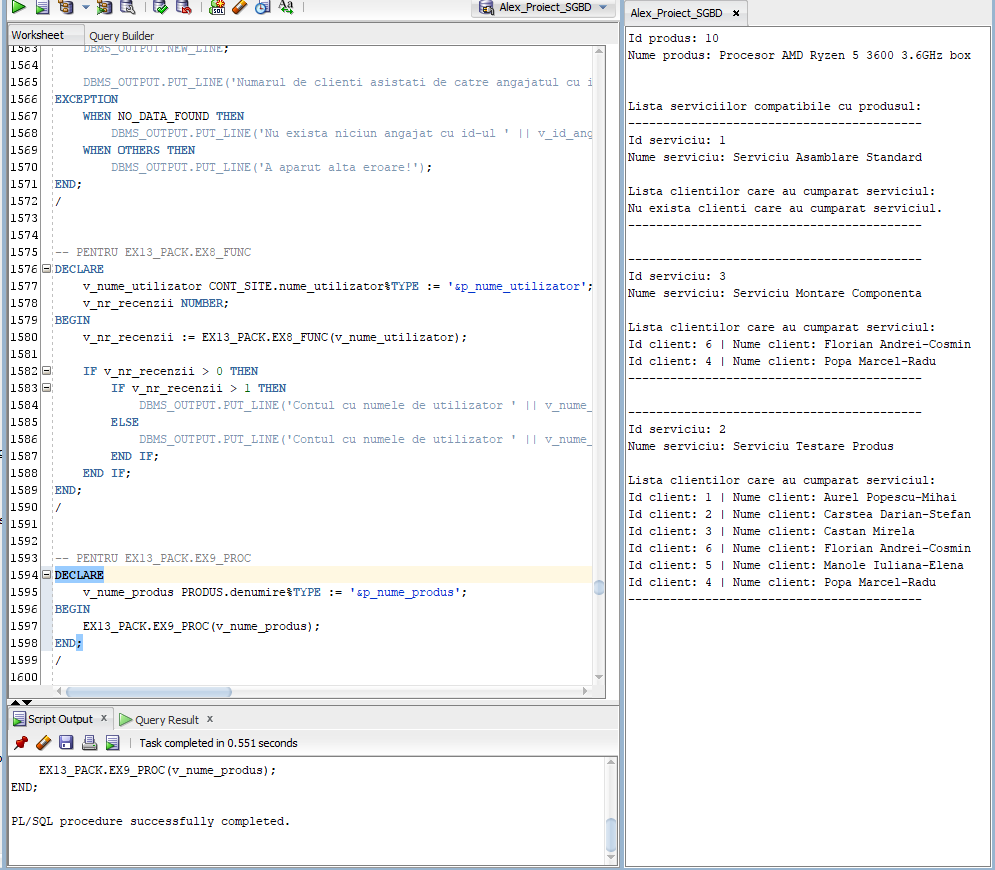


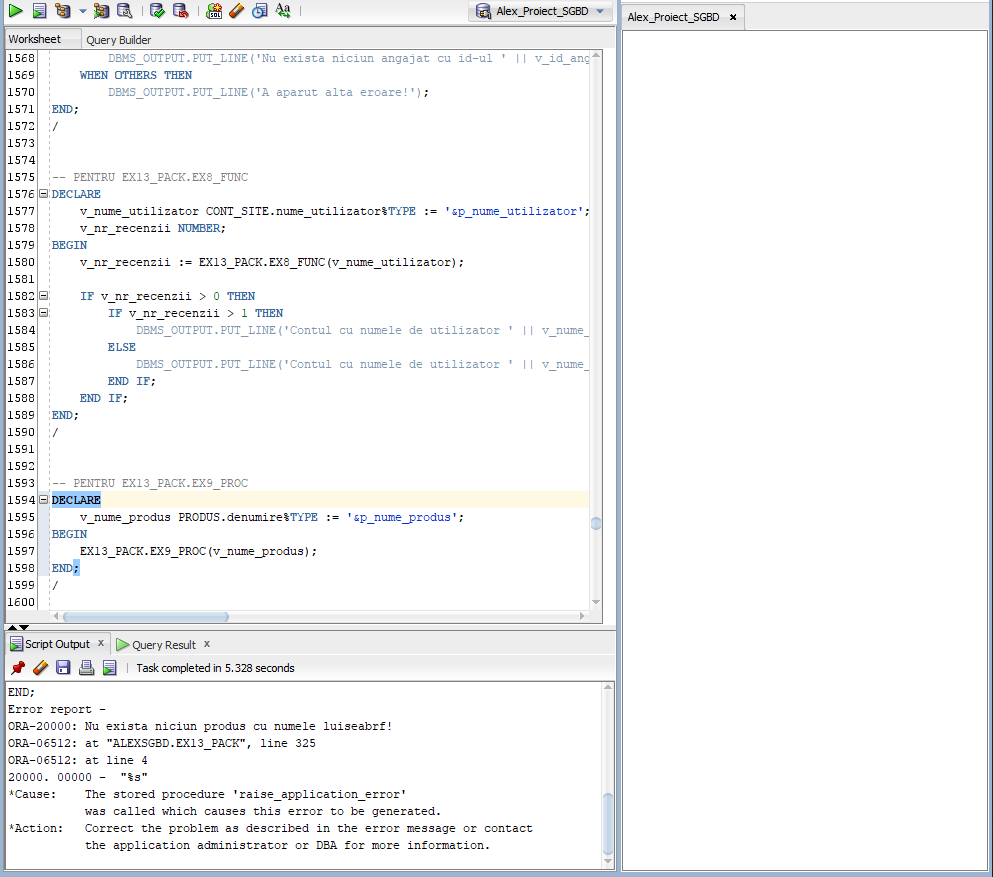


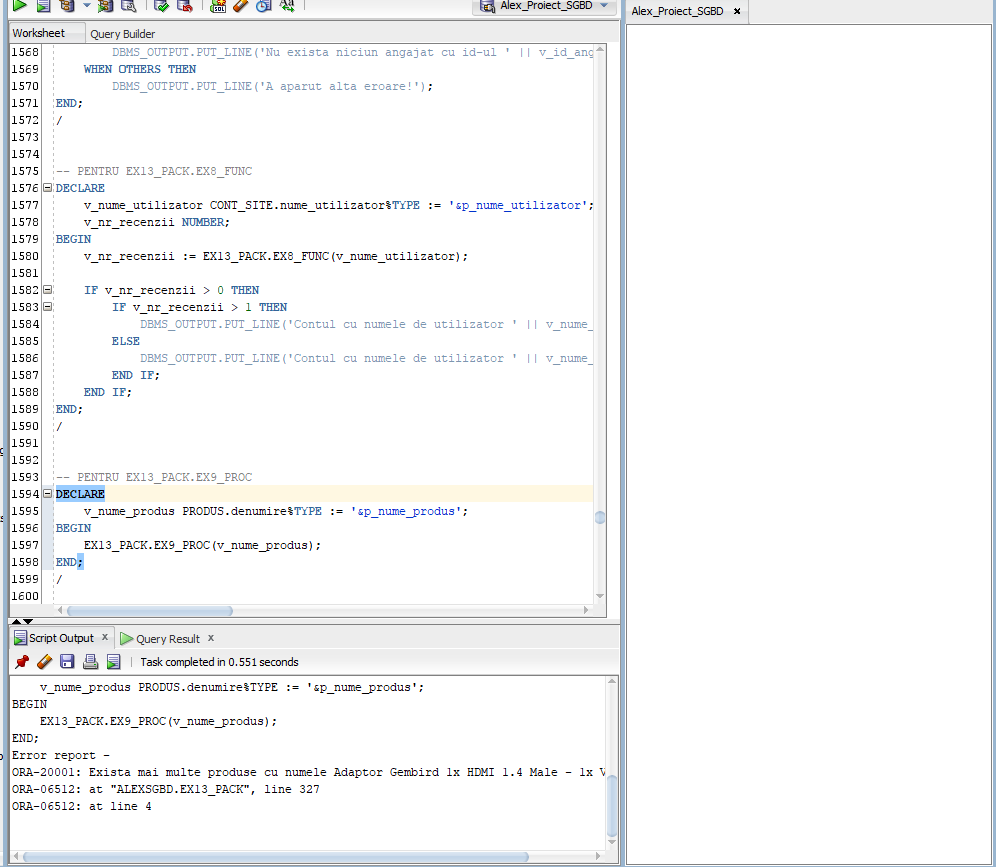


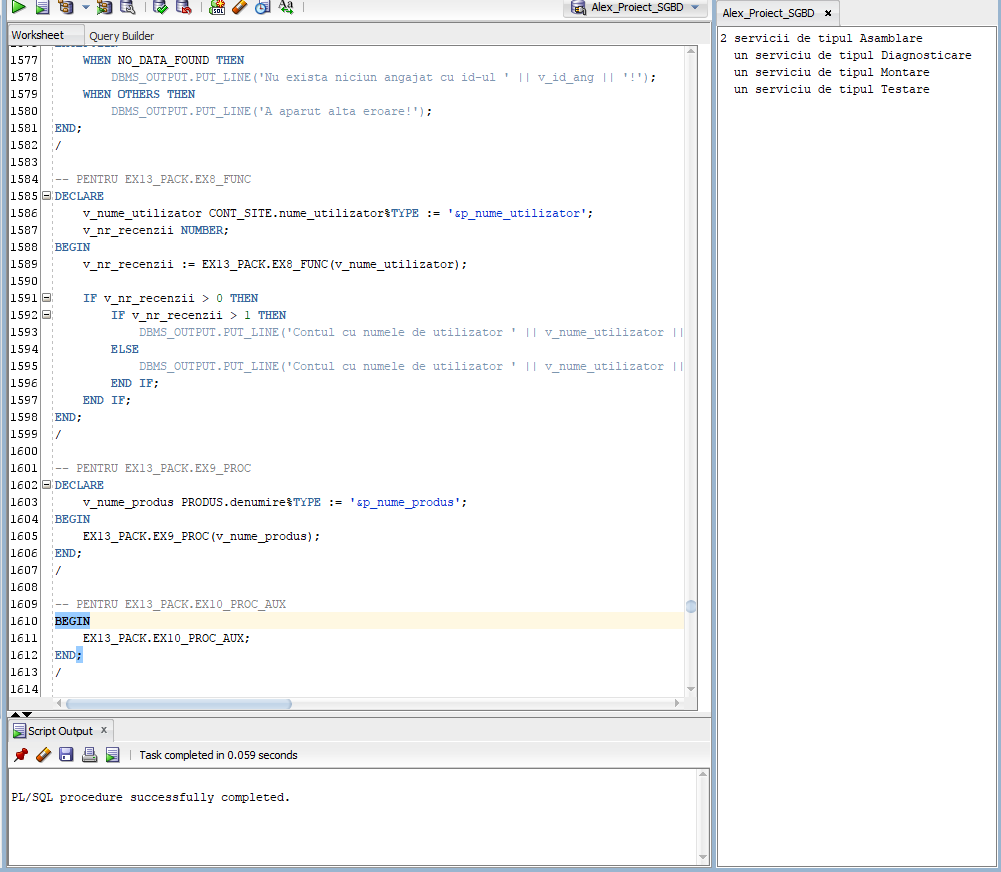












14. Pachet care include tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite:

-- Cerinta 14.:

-- Enunt:

-- Definiti un pachet, care contine minimum 2 tipuri de date, minimum 2 functii, si minimum 2 proceduri.

-- Prin intermediul pachetului, determinati, pentru un produs al carui id este dat de la tastatura, daca

-- i se poate aplica vreuna din reducerile:

-- - In caz ca produsul are cel mult 10 recenzii, iar numarul mediu de stele al sau este cel mult 3,

-- i se va aplica o reducere de 10%. In caz contrar, produsul nu este eligibil pentru reducere;

-- - In caz ca produsul a fost cumparat de cel mult 10 clienti, i se va aplica o reducere de 10%.

-- In caz contrar, produsul nu este eligibil pentru reducere.

-- Sa se afiseze date despre produs, cat si despre recenziile si comenzile sale.

CREATE OR REPLACE PACKAGE EX14\_PACK

IS

TYPE DATE\_RECENZIE\_PROD IS RECORD

(id\_recenzie RECENZIE.id\_recenzie%TYPE,

stele\_recenzie RECENZIE.nr\_stele%TYPE);

TYPE tab\_ind\_date\_recenzii\_prod IS TABLE OF DATE\_RECENZIE\_PROD

INDEX BY PLS\_INTEGER;

t\_date\_recenzii\_prod tab\_ind\_date\_recenzii\_prod;

v\_total\_nr\_stele\_prod NUMBER;

TYPE DATE\_COMANDA\_PROD IS RECORD

(id\_comanda COMANDA.id\_comanda%TYPE,

pret\_comanda COMANDA.pret\_total%TYPE);

TYPE tab\_ind\_date\_comenzi\_prod IS TABLE OF DATE\_COMANDA\_PROD

INDEX BY PLS\_INTEGER;

t\_date\_comenzi\_prod tab\_ind\_date\_comenzi\_prod;

TYPE DATE\_PRODUS IS RECORD

(id\_produs PRODUS.id\_produs%TYPE,

nume\_produs PRODUS.denumire%TYPE,

nume\_categorie\_produs CATEGORIE.denumire%TYPE,

pret\_produs PRODUS.pret%TYPE);

v\_date\_produs DATE\_PRODUS;

FUNCTION EX14\_FUNC1

(id\_prod PRODUS.id\_produs%TYPE)

RETURN NUMBER;

FUNCTION EX14\_FUNC2

(id\_prod PRODUS.id\_produs%TYPE)

RETURN NUMBER;

PROCEDURE EX14\_PROC1

(id\_prod IN OUT PRODUS.id\_produs%TYPE);

PROCEDURE EX14\_PROC2

(id\_prod PRODUS.id\_produs%TYPE,

id\_apel NUMBER);

END EX14\_PACK;

/

CREATE OR REPLACE PACKAGE BODY EX14\_PACK

IS

PROCEDURE EX14\_PROC2

(id\_prod PRODUS.id\_produs%TYPE,

id\_apel NUMBER)

IS

exceptie EXCEPTION;

BEGIN

IF id\_apel = 1 THEN

IF (t\_date\_recenzii\_prod.COUNT <= 10) AND (v\_total\_nr\_stele\_prod/t\_date\_recenzii\_prod.COUNT <= 3) THEN

UPDATE PRODUS

SET pret = pret \* 0.9

WHERE id\_produs = id\_prod;

DBMS\_OUTPUT.PUT\_LINE('Produsul cu id-ul ' || id\_prod || ' a primit o reducere a pretului de 10%, deoarece are un numar de recenzii mai mic sau egal cu 10, iar media numarului de stele este mai mica sau egala cu 3.');

DBMS\_OUTPUT.PUT\_LINE('Medie nr. stele: ' || ROUND(v\_total\_nr\_stele\_prod/t\_date\_recenzii\_prod.COUNT, 2));

ELSE

DBMS\_OUTPUT.PUT\_LINE('Produsul nu este eligibil pentru reducere, deoarece fie are un numar de recenzii mai mare decat 10, fie media numarului de stele este mai mare decat 3, fie ambele.');

DBMS\_OUTPUT.PUT\_LINE('Medie nr. stele: ' || ROUND(v\_total\_nr\_stele\_prod/t\_date\_recenzii\_prod.COUNT, 2));

END IF;

ELSIF id\_apel = 2 THEN

IF t\_date\_comenzi\_prod.COUNT <= 10 THEN

UPDATE PRODUS

SET pret = pret \* 0.9

WHERE id\_produs = id\_prod;

DBMS\_OUTPUT.PUT\_LINE('Produsul cu id-ul ' || id\_prod || ' a primit o reducere a pretului de 10%, deoarece a fost cumparat de catre un numar mai mic sau egal cu 10 clienti.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Produsul nu este eligibil pentru reducere, deoarece a fost cumparat de catre un numar mai mare de 10 clienti.');

END IF;

ELSIF id\_apel <> 1 OR id\_apel <> 2 THEN

RAISE exceptie;

END IF;

EXCEPTION

WHEN exceptie THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Id-ul apelului este incorect!');

END EX14\_PROC2;

FUNCTION EX14\_FUNC1

(id\_prod PRODUS.id\_produs%TYPE)

RETURN NUMBER

IS

v\_recenzie\_rec DATE\_RECENZIE\_PROD;

v\_counter NUMBER := 0;

BEGIN

v\_total\_nr\_stele\_prod := 0;

FOR i IN (SELECT id\_recenzie id

FROM RECENZIE

WHERE id\_produs = id\_prod) LOOP

v\_counter := v\_counter + 1;

SELECT id\_recenzie, nr\_stele

INTO v\_recenzie\_rec

FROM RECENZIE

WHERE id\_recenzie = i.id;

t\_date\_recenzii\_prod(v\_counter) := v\_recenzie\_rec;

v\_total\_nr\_stele\_prod := v\_total\_nr\_stele\_prod + v\_recenzie\_rec.stele\_recenzie;

END LOOP;

EX14\_PROC2(id\_prod, 1);

RETURN t\_date\_recenzii\_prod.COUNT;

END EX14\_FUNC1;

FUNCTION EX14\_FUNC2

(id\_prod PRODUS.id\_produs%TYPE)

RETURN NUMBER

IS

v\_comanda\_rec DATE\_COMANDA\_PROD;

v\_counter NUMBER := 0;

BEGIN

FOR i IN (SELECT id\_comanda id

FROM PRODUS p JOIN COMANDA c ON p.id\_produs = c.id\_produs

WHERE p.id\_produs = id\_prod) LOOP

v\_counter := v\_counter + 1;

SELECT id\_comanda, pret\_total

INTO v\_comanda\_rec

FROM COMANDA

WHERE id\_comanda = i.id;

t\_date\_comenzi\_prod(v\_counter) := v\_comanda\_rec;

END LOOP;

EX14\_PROC2(id\_prod, 2);

RETURN t\_date\_comenzi\_prod.COUNT;

END EX14\_FUNC2;

PROCEDURE EX14\_PROC1

(id\_prod IN OUT PRODUS.id\_produs%TYPE)

IS

BEGIN

SELECT id\_produs, p.denumire, c.denumire, pret

INTO v\_date\_produs.id\_produs, v\_date\_produs.nume\_produs, v\_date\_produs.nume\_categorie\_produs, v\_date\_produs.pret\_produs

FROM PRODUS p JOIN CATEGORIE c USING (id\_categorie)

WHERE id\_produs = id\_prod;

DBMS\_OUTPUT.PUT\_LINE('Id produs: ' || v\_date\_produs.id\_produs);

DBMS\_OUTPUT.PUT\_LINE('Nume produs: ' || v\_date\_produs.nume\_produs);

DBMS\_OUTPUT.PUT\_LINE('Nume categorie: ' || v\_date\_produs.nume\_categorie\_produs);

DBMS\_OUTPUT.PUT\_LINE('Pret produs: ' || ROUND(v\_date\_produs.pret\_produs, 2));

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Nr. recenzii produs = ' || EX14\_FUNC1(id\_prod) || ':');

FOR i IN t\_date\_recenzii\_prod.FIRST..t\_date\_recenzii\_prod.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(' Id recenzie: ' || t\_date\_recenzii\_prod(i).id\_recenzie || ' | Nr. stele recenzie: ' || t\_date\_recenzii\_prod(i).stele\_recenzie);

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Nr. comenzi produs = ' || EX14\_FUNC2(id\_prod) || ':');

FOR i IN t\_date\_comenzi\_prod.FIRST..t\_date\_comenzi\_prod.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(' Id comanda: ' || t\_date\_comenzi\_prod(i).id\_comanda || ' | Pret comanda: ' || t\_date\_comenzi\_prod(i).pret\_comanda);

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

SELECT pret

INTO v\_date\_produs.pret\_produs

FROM PRODUS

WHERE id\_produs = id\_prod;

DBMS\_OUTPUT.PUT\_LINE('Pretul final al produsului: ' || ROUND(v\_date\_produs.pret\_produs, 2));

t\_date\_recenzii\_prod.DELETE;

t\_date\_comenzi\_prod.DELETE;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista niciun produs cu id-ul ' || id\_prod || '!');

END EX14\_PROC1;

END EX14\_PACK;

/

-- DROP PACKAGE EX14\_PACK;

DECLARE

v\_id\_produs PRODUS.id\_produs%TYPE := 223;

BEGIN

EX14\_PACK.EX14\_PROC1(v\_id\_produs);

END;

