PROIECT SGBD

Popa Diana-Maria

Seria D

Gr 1057

I.Cerințele proiectului

A. Descrierea problemei şi prezentarea schemei conceptuale a tabelelor.

B. Interacţiunea cu serverul Oracle prin intermediul comenzilor SQL (LDD şi LMD)

C. Structuri alternative şi repetitive.

D. Tratarea excepţiilor. (minim 2 implicite, 2 explicite)

E. Gestionarea cursorilor: impliciţi şi expliciţi (cu şi fără parametri)

F. Funcţii, proceduri, includerea acestora în pachete. (minim 3 funcţii, 2 proceduri şi

un pachet)

G. Declanşatori. (minim 2)

H. O aplicatie in Oracle Apex, care sa contina minim 4 pagini cu realizari diferite

(formulare, rapoarte, grafice, master-detail etc.).

II.Descrierea bazei de date

Baza de date pe care am creat-o gestionează un magazin alimentar, care are furnizori, informațille acestora, zona lor geografică, clienți, comenzi, înregistrarea plăților. Furnizorii dețin un nume, id-ul primary key și legătura cu zona geografică prin id-ul acesteia, foreign key. Astfel, cele mai importante informații despre un furnizor, pentru un magazin alimentar sunt stocate în tabela principală, numele fiind de la sine înțeles și realizând legătura cu tabela info\_furnizori, iar locația fiind importantă în vederea livrării. Celelalte informații, precum numărul de angajați și data înființării sunt stocate în info\_furnizori, în cazul în care este necesară o căutare mai amănunțită. În tabela comenzi, pe lângă primary key, id-ul comenzii, am adăugat id-ul produsului și id-ul furnizorului făcând legătura cu tabelele în cauză și cantitatea. Aceasta gestionează comenzile magazinului către furnizori. În tabela plăți am adșugat id-ul plății, al comenzii plătite și data plății. În produse, în afară de id, am adăugat informațiile necesare, precum stoc, data expirării,tipul, prețul și denumirea. Tabela clienți reprezintă practic vânzările către clienți, unde am înregistrat id-ul clientului, id-ul produsului și cantitatea din produs cumpărată.

III.Schema bazei de date

A diagram of a company

Description automatically generated with medium confidence

IV.Comenzi

**B. Interacţiunea cu serverul Oracle prin intermediul comenzilor SQL (LDD şi LMD)**

**1.Introduceți într-o nouă tabelă caracteristicile produsului cu id 9 și afișați un mesaj corespunzător.**

begin

execute immediate 'CREATE table prod AS SELECT \* FROM produse where 1=2';

end;

declare

v\_idp produse.id\_produs%type;

v\_denp produse.denumire%type;

v\_tip produse.tip%type;

v\_pret produse.pret%type;

v\_stoc produse.stoc%type;

v\_expirare produse.expirare%type;

begin

SELECT id\_produs, denumire, pret, stoc, expirare, tip INTO v\_idp, v\_denp, v\_pret, v\_stoc, v\_expirare, v\_tip

FROM produse where id\_produs=9;

INSERT INTO prod (id\_produs, denumire, pret, stoc, expirare, tip) VALUES (v\_idp, v\_denp, v\_pret, v\_stoc, v\_expirare, v\_tip);

DBMS\_OUTPUT.PUT\_LINE ('S-a adaugat in tabela prod produsul: '||v\_idp||' '||v\_denp||' '||v\_tip);

end;

A screenshot of a computer

Description automatically generated/

**2. La produsele unde prețul este mai mare decât un prag dat (10) scădeți prețul cu 10%.**

DECLARE

v\_procent number:=0.1;

v\_prag produse.pret%type:=10;

BEGIN

UPDATE produse

SET pret=pret\*(1-v\_procent)

WHERE pret>v\_prag;

END;

/

Select \* from produse;

A black screen with white text

Description automatically generatedA screenshot of a computer

Description automatically generated

**3. Ștergeți din tabela creată la primul exercițiu produsul care are în denumire „pufuleti”.**

DECLARE

BEGIN

DELETE FROM prod WHERE initcap(denumire) like '%pufuleti%';

ROLLBACK;

END;

/

A black screen with white text

Description automatically generated

**C. Structuri alternative şi repetitive.**

**1.Afișați prețul inițial, iar apoi pe cel modificat astfel: dacă este mai mic decât 10 i se aplică o creștere cu 50%, dacă este între 10 și 15 i se aplică o creștere de 25%, iar altfel, de 5%.**

DECLARE

v\_pret produse.pret%type;

BEGIN

SELECT pret into v\_pret from produse where id\_produs=9;

dbms\_output.put\_line ('Pretul initial este: '||v\_pret);

IF v\_pret < 10 THEN

v\_pret:=1.5\* v\_pret;

ELSIF v\_pret between 10 and 15 THEN

v\_pret:=1.25 \* v\_pret;

ELSE

v\_pret:=1.05\* v\_pret;

END IF;

dbms\_output.put\_line('Pretul final este: '||v\_pret);

end;

A screen shot of a computer

Description automatically generated

**2.Aceeași cerință ca la exercițiul anterior, dar folosind case.**

DECLARE

v\_pret produse.pret%type;

BEGIN

SELECT pret into v\_pret from produse where id\_produs=9;

dbms\_output.put\_line ('Pretul initial este: '||v\_pret);

v\_pret:= CASE WHEN v\_pret < 10 THEN 1.5\* v\_pret

WHEN v\_pret between 10 and 15 THEN 1.25 \* v\_pret

ELSE 1.05\* v\_pret END;

dbms\_output.put\_line('Pretul final este: '||v\_pret);

end;

/

A screenshot of a computer

Description automatically generated

**3. Afișați numărul mediu de angajați al furnizorilor și furnizorii până când se întâlnește unul cu număr mai mare de angajați decât media.**

DECLARE

v\_nr info\_furnizori.nrangajati%type;

v\_nrMediu v\_nr%type;

i number(4):=1;

BEGIN

SELECT avg(nrangajati) into v\_nrMediu from info\_furnizori;

dbms\_output.put\_line('Nr mediu de angajati este: '||v\_nrMediu);

loop

select i.nrangajati into v\_nr from info\_furnizori i, furnizori f where f.id\_furnizor=i and f.nume=i.nume;

exit when v\_nr>v\_nrMediu or i>10;

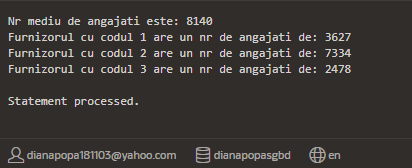
dbms\_output.put\_line('Furnizorul cu codul '||i||' are un nr de angajati de: '||v\_nr);

i:=i+1;

end loop;

end;

/



**4. Altă modalitate de a rezolva cerința anterioară.**

DECLARE

v\_nrMediu info\_furnizori.nrangajati%TYPE;

v\_nr info\_furnizori.nrangajati%TYPE;

BEGIN

SELECT AVG(nrangajati) INTO v\_nrMediu FROM info\_furnizori;

DBMS\_OUTPUT.PUT\_LINE('Nr mediu de angajati este: ' || v\_nrMediu);

FOR rec IN (

SELECT i.nrangajati, f.id\_furnizor

FROM info\_furnizori i

JOIN furnizori f ON f.nume = i.nume

)

LOOP

v\_nr := rec.nrangajati;

eXIT WHEN v\_nr > v\_nrMediu;

DBMS\_OUTPUT.PUT\_LINE('Furnizorul cu codul ' || rec.id\_furnizor || ' are un nr de angajati de: ' || v\_nr);

END LOOP;

END;

/

A screenshot of a computer

Description automatically generated

**E. Gestionarea cursorilor: impliciţi şi expliciţi (cu şi fără parametri)  
1.Ștergeți produsele unde tipul este „snacks” și afișați numărul de rânduri șterse.**

BEGIN

DELETE FROM prod p

WHERE tip='snacks';

DBMS\_OUTPUT.PUT\_LINE (SQL%ROWCOUNT || ' randuri sterse');

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE (SQL%ROWCOUNT || ' randuri afectate');

END;

/

A screenshot of a computer

Description automatically generated

**2. Afișați lista cu orașele din Muntenia.**

declare

cursor zg\_cursor is select id\_zonageo, oras from zona\_geografica where regiune like '%Muntenia%';

zg\_rec zg\_cursor%rowtype;

begin

dbms\_output.put\_line('Lista cu orasele din regiunea Muntenia');

open zg\_cursor;

loop

fetch zg\_cursor into zg\_rec;

exit when zg\_cursor%notfound;

dbms\_output.put\_line('Orasul '||zg\_rec.oras||' are id-ul: '||zg\_rec.id\_zonageo);

end loop;

close zg\_cursor;

end;

/

A screenshot of a computer

Description automatically generated

**3. Afișați produsele cu stocul mai mare decât 100.**

DECLARE

CURSOR c\_stoc (p\_val NUMBER) IS

SELECT p.id\_produs, p.denumire, p.stoc

FROM produse p

WHERE p.stoc>p\_val;

v\_val NUMBER(5);

rec\_prod c\_stoc%rowtype;

BEGIN

v\_val:=100;

DBMS\_OUTPUT.PUT\_LINE('Produsele cu stocul este mai mare decat '|| v\_val);

IF NOT c\_stoc%ISOPEN THEN

OPEN c\_stoc (v\_val);

END IF;

LOOP

FETCH c\_stoc into rec\_prod;

EXIT WHEN c\_stoc%notfound;

DBMS\_OUTPUT.PUT\_LINE('Pentru produsul '||rec\_prod.id\_produs||', '||rec\_prod.denumire||', este stoc ' ||rec\_prod.stoc||' unitati');

END LOOP;

CLOSE c\_stoc;

END;

A screenshot of a computer

Description automatically generated

**D. Tratarea excepţiilor. (minim 2 implicite, 2 explicite)**

**1.Căutați produsul cu id-ul 19. Dacă nu există, se aruncă o excepție predefinită.**

DECLARE

v\_denumire VARCHAR2(20);

BEGIN

SELECT denumire INTO v\_denumire

FROM produse

WHERE id\_produs=19;

dbms\_output.put\_line(v\_denumire);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('Nu exista produsul cu acest ID');

END;

/

A screenshot of a computer

Description automatically generated

**2. Creați o tabelă Erori unde adăugați mesajul, codul, utilizatorul și data erorii cauzate de ștergerea unui furnizor carea are informații asignate (excepție non-predefinită).**

CREATE TABLE erori

(utilizator VARCHAR2(40),

data DATE,

cod\_eroare NUMBER(10),

mesaj\_eroare VARCHAR2(255)

);

DECLARE

cod NUMBER;

mesaj VARCHAR2(255);

del\_exception EXCEPTION;

PRAGMA EXCEPTION\_INIT(del\_exception, -2292);

BEGIN

DELETE FROM furnizori;

EXCEPTION

WHEN del\_exception THEN

dbms\_output.put\_line('Nu puteti sterge furnizorul');

dbms\_output.put\_line('Exista informatii asignate lui');

cod:=SQLCODE;

mesaj:=SQLERRM;

INSERT INTO erori VALUES(USER, SYSDATE, cod, mesaj);

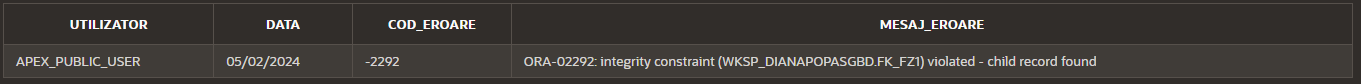
END;

/

SELECT \* FROM erori;

A screen shot of a computer

Description automatically generated



**3.Declarați o excepție care afișează un mesaj când se încearcă realizarea unei operații de UPDATE, dar nu există date.**

DECLARE

invalid\_fz EXCEPTION;

BEGIN

UPDATE furnizori

SET nume='Mega Image'

WHERE id\_furnizor=14;

IF SQL%NOTFOUND THEN

RAISE invalid\_fz;

END IF;

EXCEPTION

WHEN invalid\_fz THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista furnizorul cu acest ID');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare! Nu se poate actualiza denumirea furnizorului!');

END;

/

A black screen with white text

Description automatically generated

**F. Funcţii, proceduri, includerea acestora în pachete. (minim 3 funcţii, 2 proceduri şi un pachet)**

**1.Creați o procedură care caută un furnizor după nume, tratați excepțiile și apelați-o.**

CREATE OR REPLACE PROCEDURE cauta\_fz (

p\_id\_fz IN furnizori.id\_furnizor%TYPE,

p\_nume OUT furnizori.nume%TYPE

)

IS

BEGIN

SELECT nume INTO p\_nume FROM furnizori WHERE id\_furnizor = p\_id\_fz;

DBMS\_OUTPUT.PUT\_LINE('Furnizorul cu id ' || p\_id\_fz || ' are numele: ' || p\_nume);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu există furnizor cu id-ul ' || p\_id\_fz);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A apărut o eroare în căutarea furnizorului.');

END;

/

DECLARE

v\_denumire furnizori.nume%type;

BEGIN

Cauta\_fz(4, v\_denumire);

END;

/

A screenshot of a computer

Description automatically generated

**2.Creați o procedură care modifică prețul cu un procent și apelați-o.**

CREATE OR REPLACE

PROCEDURE modifica\_pret

(p\_id\_produs IN produse.id\_produs%type, procent IN number)

IS

v\_pret produse.pret%type;

BEGIN

Select pret into v\_pret from produse where id\_produs=p\_id\_produs;

dbms\_output.put\_line('Produsul are pretul de '||v\_pret);

Update produse

Set pret=pret\*(1+procent/100)

Where id\_produs=p\_id\_produs;

Select pret into v\_pret from produse where id\_produs=p\_id\_produs;

Dbms\_output.put\_line('Produsul are acum pretul de '||v\_pret);

END;

/

BEGIN

modifica\_pret(3, 10);

END;

A screenshot of a computer

Description automatically generated

**3. Creați o funcție care calculează TVA-ul și apelați-o într-un SELECT.**

CREATE OR REPLACE FUNCTION TVA (value IN NUMBER)

RETURN NUMBER IS

BEGIN

RETURN (value\*9/100);

END TVA;

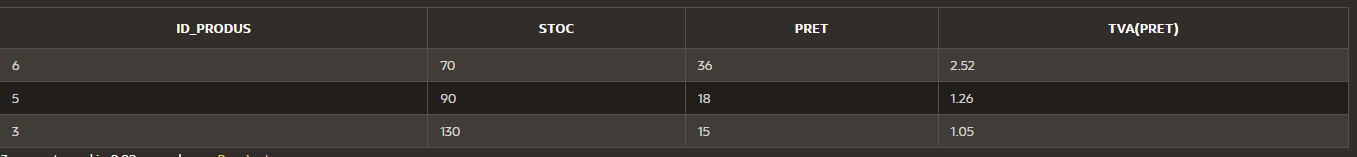
/

SELECT id\_produs, stoc, pret, tva (pret)

FROM produse

WHERE tva(pret)>(select avg(tva(pret)) from produse)

ORDER BY tva(pret) DESC;



**4. Creați o funcție care preia numele unui furnizor după id.**

CREATE OR REPLACE FUNCTION get\_nume\_furnizor(

p\_id\_furnizor IN furnizori.id\_furnizor%TYPE

) RETURN furnizori.nume%TYPE

IS

v\_nume furnizori.nume%TYPE;

BEGIN

SELECT nume INTO v\_nume FROM furnizori WHERE id\_furnizor = p\_id\_furnizor;

RETURN v\_nume;

EXCEPTION

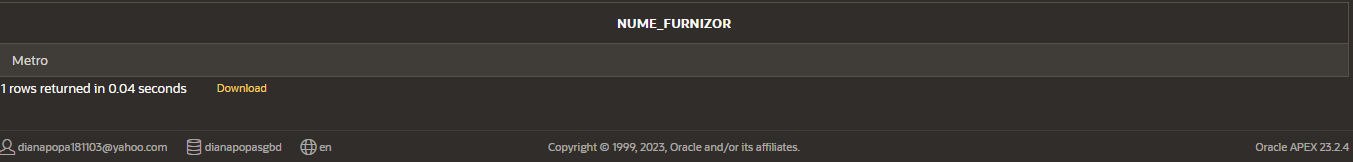
WHEN NO\_DATA\_FOUND THEN

RETURN NULL;

END;

/

SELECT get\_nume\_furnizor(1) AS nume\_furnizor FROM dual;



**5.Creați o funcție care realizează o listă de produse pe stoc dintr-un anumit tip dat ca parametru.**

CREATE OR REPLACE FUNCTION verifica\_stoc\_produse(

p\_tip\_produs IN PRODUSE.tip%TYPE

) RETURN VARCHAR2

IS

v\_lista\_produse VARCHAR2(1000) := '';

BEGIN

FOR rec IN (

SELECT denumire

FROM PRODUSE

WHERE tip = p\_tip\_produs AND stoc > 0

)

LOOP

v\_lista\_produse := v\_lista\_produse || rec.denumire || ', ';

END LOOP;

IF LENGTH(v\_lista\_produse) < 0 THEN

v\_lista\_produse := 'Nu există produse disponibile în stoc pentru tipul specificat.';

END IF;

RETURN v\_lista\_produse;

END;

/

SELECT verifica\_stoc\_produse('dulciuri') AS lista\_produse\_disponibile

FROM dual;

A screenshot of a computer

Description automatically generated

**6.Creați un pachet care realizează acțiunile de adăugare, modficare și ștergere asupra tabelei PRODUSE.**

CREATE OR REPLACE PACKAGE actualizare\_produse AS

PROCEDURE adauga\_produs(

p\_idp produse.id\_produs%TYPE,

p\_denp produse.denumire%TYPE,

p\_tip produse.tip%TYPE,

p\_pret produse.pret%TYPE,

p\_stoc produse.stoc%TYPE,

p\_expirare produse.expirare%TYPE

);

PROCEDURE modifica\_produs(

p\_idp produse.id\_produs%TYPE,

p\_denp produse.denumire%TYPE,

p\_tip produse.tip%TYPE,

p\_pret produse.pret%TYPE,

p\_stoc produse.stoc%TYPE,

p\_expirare produse.expirare%TYPE

);

PROCEDURE modifica\_denumire\_produs(

p\_idp produse.id\_produs%TYPE,

p\_denp produse.denumire%TYPE

);

PROCEDURE sterge\_produs(

p\_idp produse.id\_produs%TYPE

);

FUNCTION exista\_cod(

p\_idp produse.id\_produs%TYPE

) RETURN BOOLEAN;

END actualizare\_produse;

/

CREATE OR REPLACE PACKAGE BODY actualizare\_produse AS

PROCEDURE adauga\_produs(

p\_idp produse.id\_produs%TYPE,

p\_denp produse.denumire%TYPE,

p\_tip produse.tip%TYPE,

p\_pret produse.pret%TYPE,

p\_stoc produse.stoc%TYPE,

p\_expirare produse.expirare%TYPE

)

IS

BEGIN

IF exista\_cod(p\_idp) THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Produs existent!');

ELSE

INSERT INTO produse (id\_produs, denumire, tip, pret, stoc, expirare)

VALUES (p\_idp, p\_denp, p\_tip, p\_pret, p\_stoc, p\_expirare);

END IF;

END adauga\_produs;

PROCEDURE modifica\_produs(

p\_idp produse.id\_produs%TYPE,

p\_denp produse.denumire%TYPE,

p\_tip produse.tip%TYPE,

p\_pret produse.pret%TYPE,

p\_stoc produse.stoc%TYPE,

p\_expirare produse.expirare%TYPE

)

IS

BEGIN

IF exista\_cod(p\_idp) THEN

UPDATE produse

SET denumire = p\_denp,

tip = p\_tip,

pret = p\_pret,

stoc = p\_stoc,

expirare = p\_expirare

WHERE id\_produs = p\_idp;

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Produsul cu acest cod nu există!');

END IF;

END modifica\_produs;

PROCEDURE modifica\_denumire\_produs(

p\_idp produse.id\_produs%TYPE,

p\_denp produse.denumire%TYPE

)

IS

BEGIN

IF exista\_cod(p\_idp) THEN

UPDATE produse

SET denumire = p\_denp

WHERE id\_produs = p\_idp;

ELSE

RAISE\_APPLICATION\_ERROR(-20003, 'Produsul cu acest cod nu există!');

END IF;

END modifica\_denumire\_produs;

PROCEDURE sterge\_produs(

p\_idp produse.id\_produs%TYPE

)

IS

BEGIN

IF exista\_cod(p\_idp) THEN

DELETE FROM produse

WHERE id\_produs = p\_idp;

DBMS\_OUTPUT.PUT\_LINE('Produsul cu id-ul ' || p\_idp || ' a fost șters!');

ELSE

RAISE\_APPLICATION\_ERROR(-20004, 'Produsul cu acest cod nu există!');

END IF;

END sterge\_produs;

FUNCTION exista\_cod(

p\_idp produse.id\_produs%TYPE

) RETURN BOOLEAN

IS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM produse

WHERE id\_produs = p\_idp;

RETURN v\_count > 0;

END exista\_cod;

END actualizare\_produse;

/

BEGIN

actualizare\_produse.adauga\_produs(

p\_idp => 19,

p\_denp => 'Suc',

p\_tip => 'Bauturi',

p\_pret => 10,

p\_stoc => 100,

p\_expirare => TO\_DATE('2024-01-04', 'YYYY-MM-DD')

);

END;

/

select \* from produse where id\_produs=19;



BEGIN

actualizare\_produse.modifica\_produs(

19,

'Suc Fructe',

'sucuri',

15,

80,

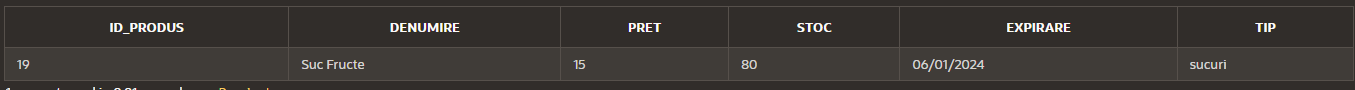
TO\_DATE('2024-06-01', 'YYYY-MM-DD')

);

END;

/

select \* from produse where id\_produs=19;



begin

actualizare\_produse.sterge\_produs(19);

end;

/

select \* from produse where id\_produs=449;

A screenshot of a computer

Description automatically generated

**G. Declanşatori. (minim 2)**

**1.Creați un trigger care adaugă într-o tabelă tipl acțiunii, utilizatorul și dată înainte de UPDATE, DELETE sau INSERT.**

CREATE TABLE temp

(tip CHAR(1),

utilizator VARCHAR2(50),

data DATE DEFAULT SYSDATE);

CREATE OR REPLACE TRIGGER produse\_trig

BEFORE INSERT or UPDATE or DELETE on produse

DECLARE

v\_tip temp.tip%TYPE;

BEGIN

case

when INSERTING then v\_tip :='I';

when UPDATING then v\_tip:='U';

ELSE v\_tip :='D';

END case;

INSERT INTO temp(tip, utilizator, data) VALUES (v\_tip, user, sysdate);

END;

/

insert into PRODUSE (id\_produs, denumire, pret, stoc, expirare, tip )

values (15,'biscuiti',7,160,TO\_DATE('2024-01-01', 'YYYY-MM-DD'), 'snacks');

delete from produse where id\_produs=15;

select\* from temp;

A screenshot of a computer

Description automatically generated

**2.Creați un trigger care se declanșează înainte de UPDATE-ul prețului, iar dacă prețul este mai mare de 20, modificați data expirării adăugându-i 30 de zile.**

CREATE OR REPLACE TRIGGER trg\_actualizare\_expirare

BEFORE UPDATE OF pret ON produse

FOR EACH ROW

DECLARE

v\_nou\_pret produse.pret%TYPE;

v\_data\_expirare produse.expirare%TYPE;

BEGIN

v\_nou\_pret := :new.pret;

v\_data\_expirare := :new.expirare;

IF v\_nou\_pret > 20 THEN

:new.expirare := v\_data\_expirare + INTERVAL '30' DAY;

ELSE

:new.expirare := v\_data\_expirare;

END IF;

END;

/

UPDATE PRODUSE

SET pret = 30

WHERE id\_produs = 3;

select\* from produse;

A screenshot of a computer

Description automatically generated

**H. O aplicatie in Oracle Apex, care sa contina minim 4 pagini cu realizari diferite (formulare, rapoarte, grafice, master-detail etc.).**

**1.Creați o aplicație pentru baza de date, care să cuprindă o pagină Master-Detail, un Interactive Report cu formular, un grafic și un calendar.**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated