ACADEMIA DE STUDII ECONOMICE BUCURESTI
FACULTATEA DE CIBERNETICA, STATISTICA SI INFORMATICA ECONOMICA

Proiect
-SGBD-
Gestiunea unei farmacii

Nume: Ioanovici Pojogeanu Andreea Gabriela

Grupa: 1050

CUPRINS:

1. Descrierea temei

Prin gestiune intelegem modul prin care se pot administra bunurile unei firme.

Farmaciile sunt locuri special amenajate unde se prepara si se comercializeaza medicamentele. Pentru a gestiona o farmacie trebuie sa ai in vedere cum se se prepară, se pastrează se distribuie si se primesc medicamente. De asemenea, tine evidenta aprovizionarea punctului farmaceutic, categoriile si produsele acestea, comenzile date si clientii punctul farmaceutic (fideli dar si unici).

S-a urmarit crearea unei baze de date pentru un punct farmaceutic ce se ocupa cu vanzarea medicamentelor.

Vom avea urmatoarele tabele:

1. Medicament
   ID_PRODUS
   DENUMIRE
   DESCRIERE
   CATEGORIE
   PRET
   STOC
   NATURA_SUBSTANTEI
   ID_MANAGER
   ID_LOCATIE
   ID_RETETA
   PRET_MINIM

2. Clienti_fideli
   ID_CLIENT
   NUME

PRENUME
TELEFON
EMAIL
DATA_NASTERE
CREDIT_DISPONIBIL

3.Orase

ID_ORAS

DENUMIRE_ORAS

TARA

4. Locatie

ID_LOCATIE
ADRESA
ID_ORAS
ZONA
STOC
ID_PRODUS

5.Farmacist

ID_ANGAJAT
NUME_ANGAJAT
PRENUME_ANGAJAT
TELEFON
EMAIL
STARE_CIVILA
SEX
VARSTA
SALARIU
ID_MANAGER
ID_FUNCTIE

6. Functii_angajati

ID_FUNCTIE

DENUMIRE_FUNCTIE

SALARIU

ID_ANGAJAT

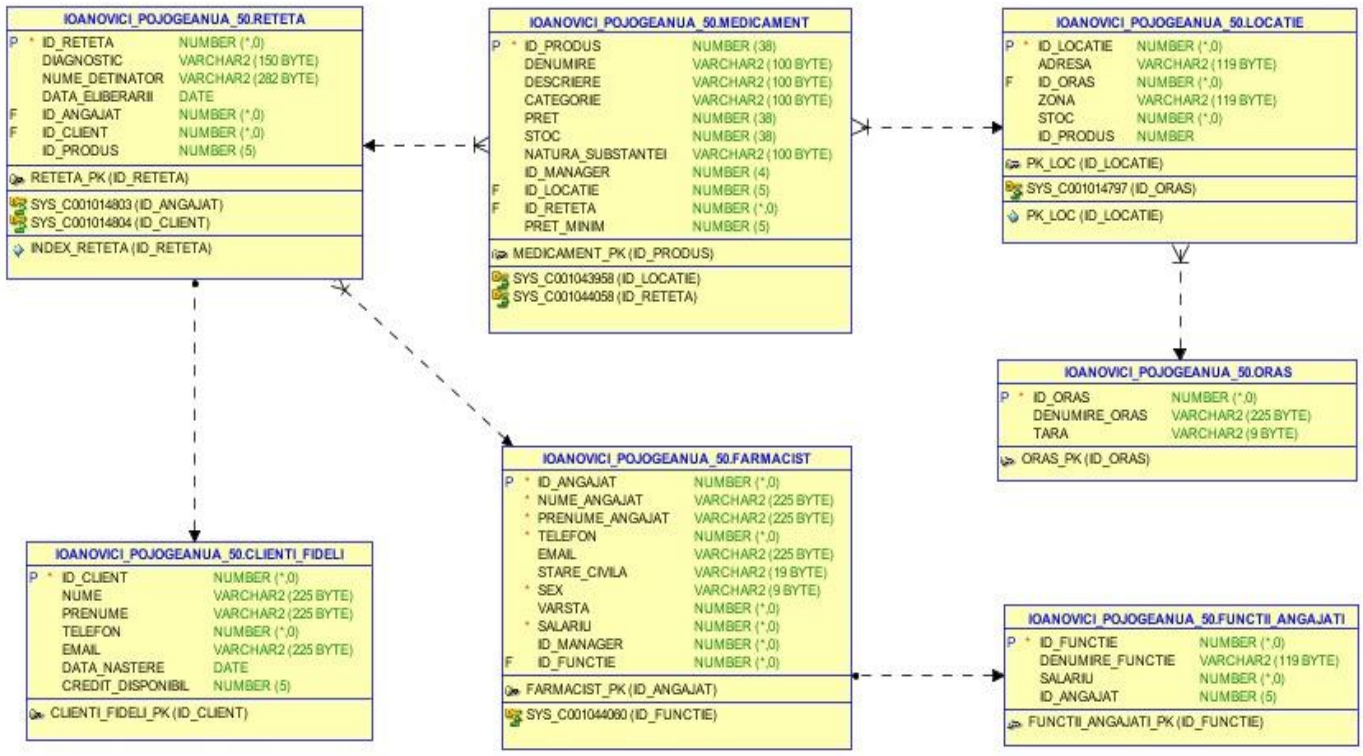
7. Reteta

ID_RETETA

DIAGNOSTIC
NUME_DETINATOR
DATA_ELIBERARII
ID_ANGAJAT
ID_CLIENT
ID_PRODUS

Schema conceptuala a proiectului

Rationamentul din spatele tabelelor este următorul. Baza de date este specifica unui lant de farmacii , fapt care face baza de date să nu aibă inregistrări din toate punctele sale farmaceutice.



CURSORI

1. Farmacistilor casatoriti le majorez salariul, apoi numar cati sunt.

```
BEGIN

UPDATE FARMACIST

SET SALARIU=0.2*SALARIU

WHERE STARE_CIVILA='CASATORIT';
```

```
IF SQL%NOTFOUND THEN

DBMS_OUTPUT.PUT_LINE('NU EXISTA FARMACISTI CARE SA SE INCADREZE');

ELSE

DBMS_OUTPUT.PUT_LINE('SUNT  ' || SQL%ROWCOUNT || '  ANGAJATI CASATORITI');

END IF;

END;
```

----------VERIFICARE----------

```
SUNT 5 ANGAJATI CASATORITI


PL/SQL procedure successfully completed.
```

---

2. Ma pregatesc sa livrez produsele de pe reteta primilor 4 cu data cea mai apropiata data

```
DECLARE

CURSOR C_RETETA IS SELECT *

        FROM (SELECT * FROM reteta ORDER BY DATA_ELIBERARII DESC)

        WHERE ROWNUM <= 4;

R C_RETETA%ROWTYPE;

BEGIN

FOR R IN C_RETETA

LOOP

dbms_output.put_line('LUI  '||R.NUME_DETINATOR|| '  TREBUIE SA II TRIMITETI PRODUSUL  '
||R.ID_produs);

END LOOP;

END;
```

----------VERIFICARE----------

```
LUI cara TREBUIE SA II TRIMITETI PRODUSUL 9
LUI Maria TREBUIE SA II TRIMITETI PRODUSUL 7
LUI Mira TREBUIE SA II TRIMITETI PRODUSUL 7
LUI Lana TREBUIE SA II TRIMITETI PRODUSUL 9


PL/SQL procedure successfully completed.
```

---

3. Ii fac fiecarui farmacist SUB 35 DE ANI SALARIUL DE MINIM salariu minim de 3000, IAR CELOR PESTE 35 SALARIUL DE 3500.

```
DECLARE

 l NUMBER := 3000;
 V NUMBER:=35;

 CURSOR c_f IS SELECT  *  FROM farmacist

 ORDER BY salariu;

  r_f c_f%ROWTYPE;

BEGIN

 OPEN c_f;

 LOOP

  FETCH  c_f  INTO r_f;

  EXIT WHEN c_f%NOTFOUND;

  UPDATE farmacist

  SET  salariu =

       CASE WHEN l > r_f.salariu

                THEN L

                WHEN L>R_F.SALARIU AND R_F.VARSTA>=V THEN L+500

                  ELSE r_f.salariu

                 end where varsta<55;

 DBMS_OUTPUT.PUT_LINE( 'Angajatul ' ||r_f.nume_angajat ||

' Are salariul ' || r_f.salariu || ' , iar salariul minim e:  ' || l );

 END LOOP;

 CLOSE c_f;

END;
```

---

4. Setam toti cumparatorii din 2019 ca le am trimis produsele

```
BEGIN
                        UPDATE reteta
   SET nume_detinator='Trimis'
   WHERE data_eliberarii LIKE '%19';
                        IF SQL%FOUND THEN
                            DBMS_OUTPUT.PUT_LINE('Produse trimise');
                        ELSE
                            DBMS_OUTPUT.PUT_LINE('Comenzile nu au fost trimise inca.');
                        END IF;
END;
```

----------VERIFICARE----------

```
Produse trimise


PL/SQL procedure successfully completed.
```

---

5. Scriu angajatii care au varsta intre 31 si 48 de ani si salariul in jur de 5000 lei.

```
DECLARE
CURSOR C_FARMACIST IS SELECT ID_angajaT, nume_angajat, SALARIU FROM FARMACIST
WHERE VARSTA BETWEEN 31 AND 48 AND SALARIU IN (5000,5005);
REC_C C_FARMACIST%ROWTYPE;
BEGIN
OPEN C_FARMACIST;
LOOP
FETCH C_FARMACIST INTO REC_C;
EXIT WHEN C_FARMACIST%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('FARMACISTUL' || rec_c.nume_angajat || 'CU ID-UL' ||
REC_C.ID_ANGAJAT|| 'ARE SALARIUL DE ' || REC_C.SALARIU);
END LOOP;
```

```
CLOSE C_FARMACIST;

END;
```

---

6. Selectam toti farmacistii care au adresa de email cu minim 15 caractere

```
select * from farmacist;

DECLARE

   CURSOR C_EM IS SELECT f.nume_angajat,f.prenume_angajat,f.email, LENGTH(f.email) AS
adresa

            FROM farmacist f

            ORDER BY adresa DESC;

   n varchar2(100);

   p varchar2(100);

   m varchar2(100);

   l NUMBER;

BEGIN

   OPEN c_em;

   LOOP

      FETCH c_em INTO n, p, m, l;

      EXIT WHEN l<=15;

      DBMS_OUTPUT.PUT_LINE('Farmacistul '|| n || ' ' || p || ' are mail-ul ' || m);

   END LOOP;

   CLOSE c_em;

END;
```

----------VERIFICARE----------

```
Farmacistul GURLUI ROBERTA are mail-ul robellybubbly@gmail.com
Farmacistul MIREA MARCEL are mail-ul MIRCEAMARC@YAHOO.COM
Farmacistul MITREA ANDI are mail-ul MITREEEA@YAHOO.COM
Farmacistul VICRES ALEX are mail-ul VICCALEX@YAHOO.COM
Farmacistul ILINCA DAN are mail-ul ILOIDAN@YAHOO.COM
Farmacistul GRECU DAVID are mail-ul GRECUT@YAHOO.COM


PL/SQL procedure successfully completed.
```

---

7. Vom mari stocul cu 5 pentru toate produsele care costa mai putin de 60 de lei.

```
declare
cursor c is select denumire, pret,stoc from medicament order by pret;
r_d varchar2(100);
r_p number;
r_s  number;
begin
open c;
loop
fetch c into r_d, r_p,r_s ;
exit when c%notfound;
WHILE r_p <= 60
LOOP
  r_s := r_s+5;
  dbms_output.put_line('sunt' || c%rowcount ||' medicamente carora le-am marit stocul');
END LOOP;
end loop;
close c;
end;
```

EXCEPTII

1. Returneaza adresa de email a fetei angajate.

```
DECLARE

   M farmacist.email%TYPE;

BEGIN

   SELECT email INTO m

   FROM farmacist

   WHERE prenume_angajat='Roberta';

                              DBMS_OUTPUT.PUT_LINE('Angajatul are adresa de mail: ' || m);


EXCEPTION

      WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.PUT_LINE('Exista mai multi angajati cu
acest prenume.');

      WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Nu exista acest angajat.');

      WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Eroare!');

END;
----------VERIFICARE----------
```

```
Angajatul are adresa de mail: CONRU@YAHOO.COM


PL/SQL procedure successfully completed.

Nu exista acest angajat.


PL/SQL procedure successfully completed.
```

---

2. Cautam sa dam card clientilor cu castiguri intre 1800 si 4000 si varsta intre 20 si 40.

Clientii care au sunt 1800 lei salariu au venitul prea mic pentru card, iar cei peste 4000 vor primi un card premium.

```
DECLARE

CURSOR c IS

SELECT nume, credit_disponibil

FROM clienti_fideli
```

```
WHERE EXTRACT(year FROM data_nastere) BETWEEN 1980 AND 2000;

exmic EXCEPTION;
exmar EXCEPTION;
BEGIN
FOR r IN c
LOOP
EXIT WHEN c%NOTFOUND;
  BEGIN
      IF r.credit_disponibil< 1800 THEN RAISE exmic;
      ELSE
        if r.credit_disponibil>4000 THEN RAISE exmar;
        ELSE DBMS_OUTPUT.PUT_LINE( r.nume || ' are venitul ' || r.credit_disponibil);
        END IF;
      END IF;

EXCEPTION
      WHEN exmic THEN DBMS_OUTPUT.PUT_LINE('Venit prea mic pentru card');
      WHEN exmar THEN DBMS_OUTPUT.PUT_LINE('Card premium.');
      WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Alta eroare.');
  END;
END LOOP;
END;
```

----------VERIFICARE----------

```
STINCEL  are venitul 1896
Venit prea mic pentru card
Card premium.
Venit prea mic pentru card


PL/SQL procedure successfully completed.
```

---

3. Pentru o anumita cantitate pe stoc, micsoram pretul cu 10.

```
declare
nu_exista_in_stoc exception;
   r_st medicament.stoc%type := &stoc;
begin
   update medicament
   set pret  = pret-10
   where stoc = r_st;
   if SQL%NOTFOUND then
      raise nu_exista_in_stoc;
   else
      dbms_output.put_line('S-a modificat pretul produsului cu stocul de '||r_st);
   end if;
exception
   when  nu_exista_in_stoc then dbms_output.put_line('Nu exista produsul cu stocul '||r_st);
end;
```

----------VERIFICARE----------

```
S-a modificat pretul produsului cu stocul de 555


PL/SQL procedure successfully completed.
```

----------VERIFICARE 2----------

```
Nu exista produsul cu stocul 8


PL/SQL procedure successfully completed.
```

---

4. Verific daca am o anumita cantitate dintr-un produs.

```
ACCEPT p_ID_produs   PROMPT 'Precizati produsul:'
ACCEPT p_cantit   PROMPT 'Cantitatea dorita:'
```

```
DECLARE
  stoc_insuficient EXCEPTION;
  PRAGMA EXCEPTION_INIT (stoc_insuficient, -20258);
  p_ID_produs NUMBER :=&p_ID_produs;
  p_cantit NUMBER :=&p_cantit;
  cantitate_in_stoc number;
  error_code number ;
  error_message varchar2(255);
BEGIN
  select stoc into cantitate_in_stoc from medicament where id_produs=p_id_produs;
  if p_cantit> cantitate_in_stoc then
    RAISE_APPLICATION_ERROR(-20211,'Nu avem destul produs in stoc!');
  else
       dbms_output.put_line('Avem destul stoc');
    end if;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Nu exista produsul!');
    error_code := SQLCODE ;
    error_message := SQLERRM ;
  WHEN stoc_insuficient THEN
    DBMS_OUTPUT.PUT_LINE('Nu exista cantitate suficienta in stoc pentru produsul selectat');
    error_code := SQLCODE ;
    error_message := SQLERRM ;
END;
```

----------VERIFICARE----------

```
Avem destul stoc


PL/SQL procedure successfully completed.
```

----------VERIFICARE 2----------

```
ORA-20211: Nu avem destul produs in stoc!
```

----------VERIFICARE 3----------

```
Nu exista produsul!


PL/SQL procedure successfully completed.
```

PROCEDURI

1. Facem o reducere unui anumit produs.

```
create or replace PROCEDURE reducere(p_id_produs IN medicament.id_produs%TYPE,
procent IN NUMBER)

as

   pret_prod medicament.pret%TYPE;

BEGIN

   SELECT pret INTO pret_prod FROM medicament WHERE id_produs = p_id_produs;

   DBMS_OUTPUT.PUT_LINE('Pretul medicamentului la inceput a fost ' || pret_prod);

   UPDATE medicament

   SET pret = pret*(1-procent/100)

   WHERE id_produs = p_id_produs;

   SELECT pret INTO pret_prod FROM medicament WHERE id_produs = p_id_produs;

   DBMS_OUTPUT.PUT_LINE('Pretul final este ' || pret_prod);

END;

execute reducere(3,5);
```

----------VERIFICARE----------

```
Pretul medicamentului la inceput a fost 190
Pretul final este 181


PL/SQL procedure successfully completed.
```

2.Afisam Primii 3 farmacisti cu salariul cel mai mare

```
create or replace PROCEDURE primii_3_farm

AS

CURSOR C IS SELECT ID_ANGAJAT, NUME_ANGAJAT, SALARIU FROM FARMACIST ORDER
BY SALARIU DESC;

BEGIN

   FOR R IN C LOOP

  EXIT WHEN C%ROWCOUNT>3;

   DBMS_OUTPUT.PUT_LINE('Angajatul cu ID-ul ' ||R.ID_ANGAJAT || ' si numele ' ||
R.NUME_ANGAJAT|| ' are salariul ' ||R.SALARIU);

   END LOOP;

END;

Execute primii_3_ang;
```

----------VERIFICARE----------

```
Angajatul cu ID-ul 1 si numele MITREA are salariul 10000
Angajatul cu ID-ul 2 si numele GRECU are salariul 9855
Angajatul cu ID-ul 6 si numele CON are salariul 8563


PL/SQL procedure successfully completed.
```

---

3. Creez o procedura care permite schimbarea unei adrese

```
create or replace PROCEDURE alta_adresa (p_id_oras IN locatie.id_oras%TYPE, adr_noua IN
locatie.adresa%TYPE)

AS

BEGIN

   UPDATE locatie
```

```
   SET adresa = adr_noua
   WHERE id_oras = p_id_oras;
   DBMS_OUTPUT.PUT_LINE('Am mutat adresa din orasul ' || p_id_oras || ' la ' || adr_noua);
END;
DECLARE
   orass locatie.id_oras%TYPE := '3';
   adresa locatie.adresa%TYPE := 'LILO MAP';
BEGIN
   alta_adresa(orass,adresa);
END;
```

----------VERIFICARE----------

```
Am mutat adresa din orasul 3 la LILO MAP


PL/SQL procedure successfully completed.
```

---

4. Maresc salariul farmacistilor cu salariu sub medie

```
create or replace procedure mareste_salariu (valoare in number)
as
 cursor c is select id_angajat, avg(salariu) medie_sal
   from farmacist
   group by id_angajat;


begin
 for i in c loop
   update farmacist
   set salariu = salariu + valoare
   where salariu < i.medie_sal
    and id_angajat = i.id_angajat;
```

```
  end loop;

end mareste_salariu;

execut mareste_salariu(150);
```

FUNCTII

1. Creeze o functie care sa calculeze in cat timp trebuie finalizata o reteta

```
create or replace FUNCTION timp_de_finalizare(p_id_reteta reteta.id_reteta%TYPE) RETURN
NUMBER

AS

   zile NUMBER;

BEGIN

   SELECT TRUNC((data_eliberarii-SYSDATE)) INTO zile

   FROM reteta

   WHERE id_reteta = p_id_reteta;



   RETURN zile;



EXCEPTION

   WHEN NO_DATA_FOUND THEN

   RETURN -1;

END;
DECLARE

z NUMBER;

BEGIN

   z := timp_de_finalizare(2);

   DBMS_OUTPUT.PUT_LINE('Preparatul urmator trebuie sa fie finalizat in ' || z
|| ' zile.');

   IF z = -1 THEN DBMS_OUTPUT.PUT_LINE('Nu exista reteta');

   END IF;
```

END;

----------VERIFICARE----------

```
Preparatul urmator trebuie sa fie finalizat in 124 zile.


PL/SQL procedure successfully completed.
```

---

2. Verificam printr-o functie daca un anumit angajat a depasit varsta de pensionare

```
create or replace FUNCTION verifica_varsta(p_nume_angajat farmacist.nume_angajat%type,
p_varsta number) RETURN Boolean

AS

varsta_ang farmacist.varsta%type;

BEGIN

SELECT varsta into varsta_ang from farmacist where nume_angajat=p_nume_angajat;

IF varsta_ang < p_varsta then

return true;
ELSE

return false;

end if;

EXCEPTION

WHEN no_data_found THEN

return NULL;

end;
show errors

describe verifica_varsta;

BEGIN

IF (verifica_varsta('con', 55) IS NULL) then

dbms_output.put_line('Acest angajat nu exista!');

elsif (verifica_varsta('con', 55)) then

dbms_output.put_line('Angajatul nu a trecut de varsta de pensionare!');
```

else

dbms_output.put_line(' Angajatul a depasit varsta de pensionare');

end if;

end;

----------VERIFICARE----------

```
No errors.

FUNCTION verifica_varsta RETURNS PL/SQL BOOLEAN
Argument Name  Type           In/Out Default?
-------------- -------------- ------ --------
P_NUME_ANGAJAT VARCHAR2(225) IN     unknown
P_VARSTA       NUMBER         IN     unknown
Acest angajat nu exista!


PL/SQL procedure successfully completed.
```

----------VERIFICARE 2 ----------

```
No errors.

FUNCTION verifica_varsta RETURNS PL/SQL BOOLEAN
Argument Name  Type           In/Out Default?
-------------- -------------- ------ --------
P_NUME_ANGAJAT VARCHAR2(225) IN     unknown
P_VARSTA       NUMBER         IN     unknown
Angajatul nu a trecut de varsta de pensionare!


PL/SQL procedure successfully completed.
```

---

3. Returnez printr-o functie numele angajatilor de sex feminin

create or replace FUNCTION farmaciste (p_gen IN farmacist.sex%TYPE) RETURN varchar2

IS

p_num varchar2(2555);

BEGIN

SELECT nume_angajat INTO p_num FROM farmacist WHERE sex = p_gen;

RETURN p_num;

exception

when too_many_rows

```
   then return 'mai multe femei';

END;
DECLARE

   s farmacist.sex%TYPE := 'f';

BEGIN

   DBMS_OUTPUT.PUT_LINE('Numele farmacistei e ' || farmaciste(s) );

END;
```

4.Returnam numarul angajatilor casatoriti

```
create or replace function f_cas(p_stare_civila in farmacist.stare_civila%type) RETURN
NUMBER

AS

num number(5);

begin

select count(stare_civila) into num  from farmacist

where stare_civila=p_stare_civila;

return num;

exception

when NO_DATA_FOUND

then return -1;
end;


DECLARE

   c farmacist.stare_civila%TYPE := 'casatorit';

BEGIN

   DBMS_OUTPUT.PUT_LINE('Numarul farmacistilot casatoriti e ' || f_cas(c) );

END;
```

PACHET

1. Creez un pachet care sa actualizeze farmacia astfel:
-vom adauga un oras nou
- vom modifica salariul farmacistilor cu un anume manager

-vom vedea ce retete au acelasi diagnostic
-vom verifica daca un anumit oras este in tara specificata
-vom reduce pretul medicamentelor dintr-o categorie aleasa.

```
create or replace PACKAGE actualizare_farmacie AS

PROCEDURE adauga_oras (p_id_oras varchar2, p_denum_oras oras.denumire_oras%type,
p_tara oras.tara%type);

PROCEDURE modifica_salariu (p_sal_nou IN farmacist.salariu%TYPE, p_id_mng IN
farMACist.id_manager%TYPE);

function acelasi_diagnostic (p_diagnostic IN reteta.diagnostic%TYPE) RETURN NUMBER;

FUNCTION verif_tara(p_oras oras.denumire_oras%type, p_tara oras.tara%type) RETURN
Boolean;

PROCEDURE reduceri_de_medicamente (p_cat medicament.categorie%TYPE);

END;

create or replace PACKAGE BODY actualizare_farmacie AS
PROCEDURE adauga_oras (p_id_oras varchar2, p_denum_oras oras.denumire_oras%type,
p_tara oras.tara%type)

AS

   ex EXCEPTION;

   ver number;

BEGIN

   SELECT count(*) INTO ver FROM oras where id_oras=p_id_oras;

   IF ver = 0 THEN

      INSERT INTO oras VALUES (p_id_oras, p_denum_oras, p_tara);

   ELSE

      RAISE ex;
   END IF;

EXCEPTION

   WHEN ex THEN DBMS_OUTPUT.PUT_LINE('Orasul a fost deja adaugat');

END;

PROCEDURE modifica_salariu (p_sal_nou IN farmacist.salariu%TYPE, p_id_mng IN
farMACist.id_manager%TYPE)

AS
```

```
BEGIN
 UPDATE farmacist
   SET salariu = p_sal_nou
   WHERE id_manager = p_id_mng;
   DBMS_OUTPUT.PUT_LINE('Angajatii cu managerul ' || p_id_mng || ' are acum salariul ' ||
P_SAL_NOU);
ENd;

function acelasi_diagnostic (p_diagnostic IN reteta.diagnostic%TYPE) RETURN NUMBER
AS
exc exception;
nr NUMBER;
suma_la_fel number;

BEGIN

   SELECT count(diagnostic), SUM(diagnostic) INTO nr, suma_la_fel FROM reteta WHERE
diagnostic = p_diagnostic;

   if nr>0 then

   RETURN suma_la_fel;

   else

   raise exc;

   end if;
   EXCEPTION

   when exc then return -1;
END;



FUNCTION verif_tara(p_oras oras.denumire_oras%type, p_tara oras.tara%type) RETURN
Boolean
As
tara_t oras.tara%type;
BEGIN
SELECT tara into tara_t from oras where p_oras=denumire_oras;

IF tara_t=p_tara then

return true;

ELSE
return false;

end if;
EXCEPTION

WHEN no_data_found THEN
return NULL;
end;
```

```
PROCEDURE reduceri_de_medicamente (p_cat medicament.categorie%TYPE)
AS
exe EXCEPTION;
p_stoc NUMBER;
CURSOR c IS SELECT id_produs, pret, stoc  FROM medicament WHERE categorie = p_cat
order by pret desc;
BEGIN

      FOR r IN c LOOP

       EXIT WHEN C%ROWCOUNT>10;

      DBMS_OUTPUT.PUT_LINE('Trebuie sa schimbam pretul medicamentului cu id-ul '||
r.id_produs|| ' si stocul '|| r.stoc);

      END LOOP;

EXCEPTION

   WHEN exe THEN DBMS_OUTPUT.PUT_LINE('Pentru categoria '|| p_cat|| ' am reduc cu 10 lei
pretul');

END;

end;

EXECUTE Actualizare_farmacie.reduceri_de_medicamente('campP');
```

----------VERIFICARE----------

```
Trebuie sa schimbam pretul medicamentului cu id-ul 2 si stocul 9


PL/SQL procedure successfully completed.
```

DECLANSATORI

1. Sa se creeze un trigger pe tabela clienti fideli prin care sa nu se permita stergerea lor

```
CREATE OR REPLACE TRIGGER trigger_cl
BEFORE DELETE ON clienti_fideli
BEGIN
raise_application_error (-20050, 'Nu putem sterge clientii');
END;

delete from clienti_fideli
where id_client=6;
```

----------VERIFICARE----------

```
Error report -
ORA-20050: Nu putem sterge clientii
ORA-06512: la "IOANOVICI_POJOGEANUA_50.TRIGGER_CL", linia 2
ORA-04088: eroare în timpul executiei triggerului 'IOANOVICI_POJOGEANUA_50.TRIGGER_CL'
```

---

2. Facem un trigger care nu ne permite sa punem un pret prea mare.

```
 create or replace TRIGGER lim_pr
BEFORE INSERT OR UPDATE ON medicament
FOR EACH ROW
DECLARE
pret_max number:=1000;
BEGIN
IF :new.pret>pret_max THEN
      RAISE_APPLICATION_ERROR (-20005, ' Pretul e prea mare!');
      else
 update medicament
set pret=:new.pret
where id_produs=:new.id_produs;
   END IF;
END;

update medicament
set pret=3000
where id_produs=5;
```

----------VERIFICARE----------

```
ORA-20005:  Pretul e prea mare!
ORA-06512: la "IOANOVICI_POJOGEANUA_50.LIM_PR", linia 5
ORA-04088: eroare în timpul executiei triggerului 'IOANOVICI_POJOGEANUA_50.LIM_PR'
```

---

3. Creez un trigger care sa nu permita introducerea valorilor negative.

```
create or replace TRIGGER VAL_NEG
BEFORE INSERT OR UPDATE OF stoc ON LOCATIE
FOR EACH ROW
BEGIN
IF :NEW.STOC<0 THEN
RAISE_APPLICATION_ERROR(-20000,'NU PUTEM INTRODUCE VALORI NEGATIVE!');
END IF;
END;

UPDATE LOCATIE
SET STOC=-33;
```

----------VERIFICARE----------

```
ORA-20000: NU PUTEM INTRODUCE VALORI NEGATIVE!
ORA-06512: la "IOANOVICI_POJOGEANUA_50.VAL_NEG", linia 3
ORA-04088: eroare în timpul executiei triggerului 'IOANOVICI_POJOGEANUA_50.VAL_NEG'
```
