

Proiect - Baze de date

Nume: Ciutacu Claudia

Grupa: 1050 C

Cuprins

- I. Tema Proiectului
- II. Descrierea succintă a proiectului
- III. Structura tabelelor
 - 1.descriere
 - 2.tipuri de date
 - 3.relații dintre tabele
- IV. Schema conceptuală a bazei de date
- V. Popularea tabelelor
- VI. Exerciții proiect

I. Tema proiectului

Am ales să proiectez și să implementez o bază de date pentru gestiunea activității unui supermarket.

II.Descrierea succintă a proiectului

Baza de date creată de mine reprezintă un mijloc prin care managerul unui supermarket poate gestiona detalii legate de angajați, furnizori, produsele puse la vânzare, cât și comenzile plasate de clienți.

În mare parte, angajații sunt caracterizați de funcția pe care o dețin și comenzile de care se ocupă fiecare. La rândul lor, comenzile sunt caracterizate de produsele ce intră în componența lor, fiecare produs având un anumit furnizor.

III.Structura tabelelor

- Angajati: tabelă ce conține datele personale ale angajaților, cât și id-ul, salariu și codurile asociate funcției și superiorului. Fiecare angajat ocupă o singură funcție, lucrează la un singur sediu și se poate ocupa de mai multe comenzi.

Nume coloană	Tip restricție	Tip de date
ID_ANGAJAT	Primary key	Number(3)
NUME	-	Varchar2(20)
PRENUME	-	Varchar2(20)
TELEFON	-	Varchar2(10)
COD_FUNCȚIE	Foreign key	Number(3)
NUME_ORAȘ	Foreign key	Varchar2(20)
DATA_ANGAJĂRII	-	Date
SALARIU	-	Number(8,2)

- Funcție: tabela conține detaliile funcțiilor la nivelul supermarketului. O funcție poate fi ocupată de mai mulți angajați.

Nume coloană	Tip restricție	Tip de date
COD_FUNCȚIE	Primary key	Number(3)
NUME_FUNCȚIE	-	Varchar2(30)
TRIBUȚII	-	Varchar2(50)
NR_ANGAJAȚI_PE_FUNCȚIE	-	Number(2)

- Sediul: într-un sediu pot exista mai mulți angajați

Nume coloană	Tip restricție	Tip de date
NUME_ORAȘ	Primary key	Varchar2(20)
ADRESĂ	-	Varchar2(30)

- Comenzi: un angajat se poate ocupa de mai multe comenzi, o comandă poate avea mai multe produse.

Nume coloană	Tip restricție	Tip de date
COD_COMANDĂ	Primary key	Number(5)
NUME_PRODUSE	-	Varchar2(70)
PREȚ_TOTAL	-	Number(5)
ID_ANGAJAT	Foreign key	Number(3)

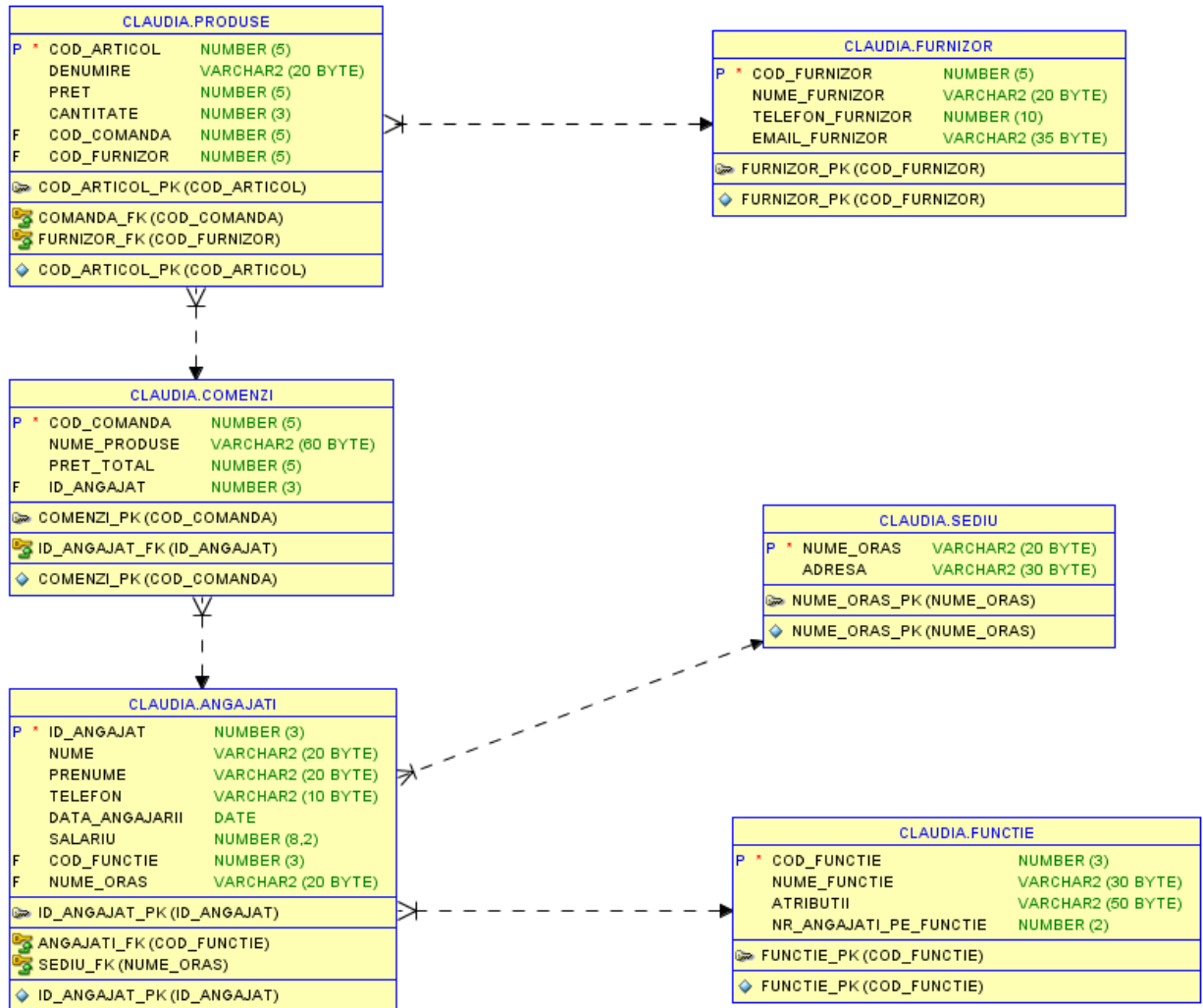
- Produse: un produs aparține unei singure comenzi și unui singur furnizor.

Nume coloană	Tip restricție	Tip de date
COD_ARTICOL	Primary key	Number(5)
DENUMIRE	-	Varchar2(20)
PREȚ	-	Number(5)
COD_COMANDĂ	Foreign key	Number(5)
COD_FURNIZOR	Foreign key	Number(5)
CANTITATE	-	Number(3)

- Furnizor_proiect: un furnizor poate aproviziona supermarketul cu mai multe produse.

Nume coloană	Tip restricție	Tip de date
COD_FURNIZOR	Primary key	Number(5)
NUME_FURNIZOR	-	Varchar2(20)
TELEFON_FURNIZOR	-	Number(10)
EMAIL_FURNIZOR	-	Varchar2(50)

Schema conceptuală a bazei de date



V.Popularea tabelelor

Crearea tabelelor

CREARE TABELA ANGAJATI CREATE TABLE ANGAJATI(ID_ANGAJAT NUMBER(3), NUME VARCHAR(20) , PRENUME VARCHAR(20), TELEFON VARCHAR(10), DATA_ANGAJARII DATE, SALARIU NUMBER(8,2), COD_FUNCTIE NUMBER(3), NUME_ORAS VARCHAR(20));	CREARE TABELA PRODUSE CREATE TABLE PRODUSE(COD_ARTICOL NUMBER(5), DENUMIRE VARCHAR(20), PRET NUMBER(5), CANTITATE NUMBER(3), COD_COMANDA NUMBER(5), COD_FURNIZOR NUMBER(5));	CREARE TABELA FUNCTIE CREATE TABLE FUNCTIE(COD_FUNCTIE NUMBER(3), NUME_FUNCTIE VARCHAR(30), ATRIBUTII VARCHAR(50), NR_ANGAJATI_PE_FUNCTIE NUMBER(2));
CREARE TABELA FURNIZOR CREATE TABLE FURNIZOR(COD_FURNIZOR NUMBER(5), NUME_FURNIZOR VARCHAR(20), TELEFON_FURNIZOR NUMBER(10), EMAIL_FURNIZOR VARCHAR(35));	CREARE TABELA COMENZI CREATE TABLE COMENZI(COD_COMANDA NUMBER(5), NUME_PRODUSE VARCHAR(20), PRET_TOTAL NUMBER(5), ID_ANGAJAT NUMBER(3));	CREARE TABELA SEDIU CREATE TABLE SEDIU(NUME_ORAS VARCHAR(20), ADRESA VARCHAR(30));

Restricțiile de integritate

RESTRICTII : CHEILE PRIMARE	RESTRICTII: CHEILE EXTERNE
<pre>ALTER TABLE ANGAJATI ADD CONSTRAINT ID_ANGAJAT_PK primary key (ID_ANGAJAT); ALTER TABLE PRODUSE ADD CONSTRAINT COD_ARTICOL_PK primary key (COD_ARTICOL); ALTER TABLE SEDIU ADD CONSTRAINT NUME_ORAS_PK primary key (NUME_ORAS); ALTER TABLE FUNCTIE ADD CONSTRAINT FUNCTIE_PK primary key (COD_FUNCTIE); ALTER TABLE COMENZI ADD CONSTRAINT COMENZI_PK primary key (COD_COMANDA); ALTER TABLE FURNIZOR ADD CONSTRAINT FURNIZOR_PK primary key (COD_FURNIZOR);</pre>	<pre>ALTER TABLE ANGAJATI ADD CONSTRAINT ANGAJATI_FK foreign key (COD_FUNCTIE) REFERENCES FUNCTIE(COD_FUNCTIE); ALTER TABLE ANGAJATI ADD CONSTRAINT SEDIU_FK foreign key (NUME_ORAS) REFERENCES SEDIU(NUME_ORAS); ALTER TABLE COMENZI ADD CONSTRAINT ID_ANGAJAT_FK foreign key (ID_ANGAJAT) REFERENCES ANGAJATI(ID_ANGAJAT); ALTER TABLE PRODUSE ADD CONSTRAINT COMANDA_FK foreign key (COD_COMANDA) REFERENCES COMENZI(COD_COMANDA); ALTER TABLE PRODUSE ADD CONSTRAINT FURNIZOR_FK foreign key (COD_FURNIZOR) REFERENCES FURNIZOR(COD_FURNIZOR);</pre>

Inserări

BEGIN

INSERT INTO SEDIU (nume_oras, adresa)

VALUES ('&nume_oras', '&adresa');

END;

/

BEGIN

INSERT INTO FURNIZOR (cod_furnizor, nume_furnizor, telefon_furnizor, email_furnizor)

VALUES (&cod_furnizor, '&nume_furnizor', &telefon_furnizor, '&email_furnizor');

END;

/

BEGIN

INSERT INTO FUNCTIE (cod_functie, nume_functie, atributii, nr_angajati_pe_functie)

VALUES (&cod_functie, '&nume_functie', '&atributii', &nr_angajati_pe_functie);

END;

/

BEGIN

INSERT INTO ANGAJATI (id_angajat, nume, prenume, telefon, data_angajarii, salariu, cod_functie, nume_oras)

VALUES (&id_angajat, '&nume', '&prenume', '&telefon', '&data_angajarii', &salariu, &cod_functie, '&nume_oras');

END;

/

BEGIN

INSERT INTO COMENZI (cod_comanda, nume_produce, pret_total, id_angajat)

VALUES (&cod_comanda, '&nume_produce', &pret_total, &id_angajat);

END;

/

BEGIN

INSERT INTO PRODUSE (cod_articol, denumire, pret, cantitate,cod_comanda,cod_furnizor)

VALUES (&cod_articol, '&denumire', &pret, &cantitate,&cod_comanda,&cod_furnizor);

END;

/

EXERCITII

Seminar 2

1.Se afiseaza numele angajatului cu codul 1.

SET SERVEROUTPUT ON

DECLARE

v_nume VARCHAR2(20);

BEGIN

SELECT nume

INTO v_nume

FROM angajati

WHERE id_angajat = 1;

DBMS_OUTPUT.PUT_LINE('NUMELE ANGAJATULUI ESTE:' || v_nume);

END;

/

```
NUMELE ANGAJATULUI ESTE:Ciutacu
```

```
PL/SQL procedure successfully completed.
```

2. Afiseaza numele si prenumele angajatului cu codul 1.

```
DECLARE

v_nume angajati.nume%TYPE;
v_prenume angajati.prenume%TYPE;

BEGIN

SELECT nume, prenume
INTO v_nume, v_prenume
FROM angajati
WHERE id_angajat = 1;

DBMS_OUTPUT.PUT_LINE('NUMELE ANGAJATULUI ESTE:' ||
v_nume || ' ' || v_prenume);

END;

/
```

3.Bind variables

```
SET SERVEROUTPUT ON

VARIABLE n number

BEGIN

select count(*) into :n
from comenzi
where id_angajat = 5;

END;

/

PRINT n

-- afiseaza 4

begin

:n:=:n+5;

dbms_output.put_line('n=' || :n);

-- afiseaza n=9
```

```

:n:=:n+5;

dbms_output.put_line('n='||:n);

-- afiseaza n=14

end;

/

PRINT n

-- afiseaza 14

```

```

PL/SQL procedure successfully completed.

      N
-----
      4

n=9
n=14

PL/SQL procedure successfully completed.

      N
-----
     14

```

4. Sa se selecteze comenzile si pretul acestora pentru acele comenzi care au pretul < pretul mediu al comenzii cu codul 234 fara a utiliza un select imbricate.

```

SET SERVEROUTPUT ON

SET AUTOPRINT ON

VARIABLE g_pret number

BEGIN

select avg(pret_total) into :g_pret

from comenzi

where cod_comanda = 234;

END;

/

select * from comenzi where pret_total< :g_pret;

```

PL/SQL procedure successfully completed.

```
G_PRET
--
34
>>Query Run In:Query Result 1
G_PRET
--
34
```

4. Se afiseaza numarul de comenzi ale angajatului al carui cod este introdus de utilizator prin intermediul variabilei de substitutie &id_angajat.

DECLARE

v_nr_comenzi number(2);

BEGIN

select count(cod_comanda) into v_nr_comenzi from comenzi

where id_angajat=&id_angajat;

dbms_output.put_line('Angajatul are: ' || v_nr_comenzi || ' comenzi');

END;

/

```
new:DECLARE
v_nr_comenzi number(2);
BEGIN
select count(cod_comanda) into v_nr_comenzi from comenzi
where id_angajat=9;
dbms_output.put_line('Angajatul are: ' || v_nr_comenzi || ' comenzi');
END;
Angajatul are: 2 comenzi
```

PL/SQL procedure successfully completed.

5. Se afiseaza salariul si prenumele angajatului cu numele Coman.

SET SERVEROUTPUT ON

VARIABLE g_salariul number

DEFINE s_nume=Coman

DECLARE

v_prenume angajati.nume%type;

BEGIN

select prenume,salariu into v_prenume, :g_salariul

from angajati where nume='&s_nume';

```
DBMS_OUTPUT.PUT_LINE ('Prenumele angajatului este: '||v_prenume);
```

```
END;
```

```
/
```

```
print g_salariul
```

```
-----
v_prenume angajati.nume%type;
BEGIN
select prenume,salariu into v_prenume, :g_salariul
from angajati where nume='Coman';
DBMS_OUTPUT.PUT_LINE ('Prenumele angajatului este: '||v_prenume);
END;
Prenumele angajatului este: Emilia

PL/SQL procedure successfully completed.

G_SALARIUL
-----
1800

G_SALARIUL
-----
      1800
```

Seminar 3

1. În funcție de prețul produsului având codul citit de la tastatură, se va afișa modificat pe ecran noua valoare.

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
v_lista produse.pret%type;
```

```
BEGIN
```

```
SELECT pret into v_lista from produse where cod_articol=&p;
```

```
dbms_output.put_line ('Pretul de lista initial este: '||v_lista);
```

```
IF v_lista < 23 THEN
```

```
    v_lista:=2* v_lista;
```

```
ELSIF v_lista between 23 and 70 THEN
```

```
    v_lista:=1.5 * v_lista;
```

```
ELSE
```

```
    v_lista:=1.25* v_lista;
```

```
END IF;
```

```
dbms_output.put_line('Pretul final este: '||v_lista);
```

```
end;
```

```
/
```

```
-----  
BEGIN  
SELECT pret into v_lista from produse where cod_articol=12;  
dbms_output.put_line ('Pretul de lista initial este: '||v_lista);  
IF v_lista < 23 THEN  
    v_lista:=2* v_lista;  
ELSIF v_lista between 23 and 70 THEN  
    v_lista:=1.5 * v_lista;  
ELSE  
    v_lista:=1.25* v_lista;  
END IF;  
dbms_output.put_line('Pretul final este: '||v_lista);  
end;  
Pretul de lista initial este: 3  
Pretul final este: 6
```

```
PL/SQL procedure successfully completed.
```

2. CASE WHEN :

SET SERVEROUTPUT ON

DECLARE

v_lista produse.pret%type;

BEGIN

SELECT pret into v_lista from produse where cod_articol=&p;

dbms_output.put_line ('Pretul de lista initial este: '||v_lista);

v_lista:= CASE WHEN v_lista < 23 THEN 2* v_lista

WHEN v_lista between 23 and 70 THEN 1.5 * v_lista

ELSE 1.25* v_lista END;

dbms_output.put_line('Pretul final este: '||v_lista);

end;

```
/
```

```
new:DECLARE  
v_lista produse.pret%type;  
BEGIN  
SELECT pret into v_lista from produse where cod_articol=12;  
dbms_output.put_line ('Pretul de lista initial este: '||v_lista);  
  
v_lista:= CASE WHEN v_lista < 23 THEN 2* v_lista  
    WHEN v_lista between 23 and 70 THEN 1.5 * v_lista  
    ELSE 1.25* v_lista END;  
  
dbms_output.put_line('Pretul final este: '||v_lista);  
end;  
Pretul de lista initial este: 3  
Pretul final este: 6
```

```
PL/SQL procedure successfully completed.
```


3.CASE WHEN:

SET SERVEROUTPUT ON

DECLARE

v_lista produse.pret%type;

BEGIN

SELECT pret into v_lista from produse where cod_articol=&p;

dbms_output.put_line ('Pretul de lista initial este: '||v_lista);

CASE

WHEN v_lista < 23 THEN

 v_lista:=2* v_lista;

WHEN v_lista between 23 and 70 THEN

 v_lista:=1.5 * v_lista;

ELSE

 v_lista:=1.25* v_lista;

END CASE;

dbms_output.put_line('Pretul final este: '||v_lista);

end;

/

```
PL/SQL procedure successfully completed.
```

```
BEGIN
SELECT pret into v_lista from produse where cod_articol=12;
dbms_output.put_line ('Pretul de lista initial este: '||v_lista);
CASE
WHEN v_lista < 23 THEN
    v_lista:=2* v_lista;
WHEN v_lista between 23 and 70 THEN
    v_lista:=1.5 * v_lista;
ELSE
    v_lista:=1.25* v_lista;
END CASE;
dbms_output.put_line('Pretul final este: '||v_lista);
end;
Pretul de lista initial este: 3
Pretul final este: 6
```

4. Se afișează în ordine angajații cu codurile în intervalul 1-10 atât timp cât salariul acestora este mai mic decât media(Structura LOOP.....END LOOP).

```
DECLARE
v_sal angajati.salariu%type;
v_salMediu v_sal%type;
i number(4):=1;
BEGIN
SELECT avg(salariu) into v_salmediu from angajati;
dbms_output.put_line('Salariul mediu este: ' || v_salmediu);
loop
select salariu into v_sal from angajati where id_angajat=i;
dbms_output.put_line('Salariatul cu codul ' || i || ' are salariul: ' || v_sal);
i:=i+1;
exit when v_sal<v_salmediu or i>10;
end loop;
end;
/
```

```
Salariul mediu este: 3215
Salariatul cu codul 1 are salariul: 2300

PL/SQL procedure successfully completed.
```

5. Se afișează în ordine angajații cu codurile în intervalul 1-10 atât timp cât salariul acestora este mai mic decât media(Structura WHILE.....LOOP....END LOOP).

```
DECLARE
v_sal angajati.salariu%type;
v_salMediu v_sal%type;
i number(4):=1;
BEGIN
```

```

SELECT avg(salariu) into v_salmediu from angajati;
dbms_output.put_line('Salariul mediu este: ' || v_salmediu);
while i<=10 loop
select salariu into v_sal from angajati where id_angajat=i;
dbms_output.put_line('Salariatul cu codul ' || i || ' are salariul: ' || v_sal);
i:=i+1;
exit when v_sal<v_salmediu;
end loop;
end;
/

Salariul mediu este: 3215
Salariatul cu codul 1 are salariul: 2300

PL/SQL procedure successfully completed.

```

6. Se afișează în ordine angajații cu codurile în intervalul 1-10 atâ timp cât salariul acestora este mai mic decât media (Structura FOR.....LOOP....END LOOP).

```

DECLARE
v_sal angajati.salariu%type;
v_salMediu v_sal%type;
-- i nu mai trebuie declarat
BEGIN
SELECT avg(salariu) into v_salmediu from angajati;
dbms_output.put_line('Salariul mediu este: ' || v_salmediu);
for i in 1..10 loop
select salariu into v_sal from angajati where id_angajat=i;
dbms_output.put_line('Salariatul cu codul ' || i || ' are salariul: ' || v_sal);
exit when v_sal<v_salmediu;
end loop;
end;
/

```

```
Salariul mediu este: 3215
Salariatul cu codul 1 are salariul: 2300

PL/SQL procedure successfully completed.
```

7. Utilizarea unui tablou indexat de tipul PRODUSE.DENPRODUS.

DECLARE

--declarare tip

type num_table is table of produse.denumire %type index by pls_integer;

-- declarare variabilă tablou

v_tab num_table;

i number(5):=12;

BEGIN

--incarcarea in tablou:

loop

SELECT denumire into v_tab(i) from produse where cod_articol=i;

i:=i+1;

exit when i>22;

end loop;

--extragerea din tablou

for i in v_tab.first..v_tab.last loop

IF v_tab.EXISTS(i) then

dbms_output.put_line('Nume produs: ' || v_tab(i));

end if;

end loop;

dbms_output.put_line('Total produse in tabloul indexat: ' || v_tab.count);

END;

/

SEMINAR 4

1. Sa se stearga produsele cu cod_furnizor=1000.

```
SET SERVEROUTPUT ON
```

```
BEGIN
```

```
DELETE FROM produse p
```

```
WHERE cod_furnizor=1000;
```

```
DBMS_OUTPUT.PUT_LINE (SQL%ROWCOUNT || ' randuri sterse');
```

```
ROLLBACK;
```

```
DBMS_OUTPUT.PUT_LINE (SQL%ROWCOUNT || ' randuri afectate');
```

```
END;
```

```
/
```

```
PL/SQL procedure successfully completed.
```

2. Se încearcă adăugarea unui sediu și apoi modificarea denumirii produsului având codul 3. În cazul în care acest produs nu există (comanda update nu realizează nici o modificare) va fi afișat un mesaj corespunzător.

```
BEGIN
```

```
INSERT INTO sediu VALUES('Sinaia','Str.Existentei,nr.56');
```

```
UPDATE produse
```

```
SET denumire='matura'
```

```
WHERE cod_articol=50;
```

```
IF SQL%NOTFOUND THEN
```

```
DBMS_OUTPUT.PUT_LINE('Nu exista produsul cu acest cod');
```

```
END IF;
```

```
ROLLBACK;
```

```
END; /
```

```

PL/SQL procedure successfully completed.

Nu exista produsul cu acest cod

PL/SQL procedure successfully completed.

```

3. Se șterge din tabela produse , produsul al cărui ID este introdus de utilizator prin intermediul variabilei de substituție *g_rid*. Mesajul este afișat folosind variabila de mediu *nr_sters*.

```

ACCEPT g_rid PROMPT 'Introduceti id-ul produsului'

VARIABLE nr_sters varchar2(100)

DECLARE

BEGIN

DELETE FROM produse WHERE cod_articol=&g_rid;

:nr_sters:=TO_CHAR(SQL%ROWCOUNT)|| ' INREGISTRARI STERSE';

END;

/

PRINT nr_sters

ROLLBACK;

```

```

PL/SQL procedure successfully completed.

NR_STERS
-----
1 INREGISTRARI STERSE

Rollback complete.

```

4. Să se afișeze lista cu numele și salariul angajaților care au salariul mai mare de 3000 folosind un cursor explicit și trei variabile scalare:

```

set serveroutput on

DECLARE

cursor ang_cursor is select id_angajat, nume, salariu from angajati where salariu>3000;

ang_id angajati.id_angajat%type;

ang_nume angajati.nume%type;

ang_sal angajati.salariu%type;

```



```

FETCH c1 INTO v_id, v_nume;
INSERT INTO mesaje VALUES(v_id, v_nume);
END LOOP;
CLOSE c1;
END;
/
SELECT * FROM mesaje;

```

	COD	NUME
1	1	Ciutacu
2	2	Coman
3	3	Daia
4	4	Craciun
5	5	Danila

6. Testul de ieșire din buclă în acest caz se poate face și cu ajutorul atributului %ROWCOUNT. Tabela mesaje nu are cheie primară deci pot fi adăugate aceleași rânduri de mai multe ori.

```

delete from mesaje;

DECLARE
v_id angajati.id_angajat%type;
v_nume angajati.nume%type;
CURSOR c1 IS SELECT id_angajat, nume FROM angajati;

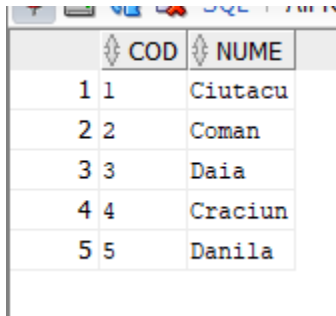
BEGIN
OPEN c1;
LOOP
FETCH c1 INTO v_id, v_nume;
EXIT WHEN c1%ROWCOUNT>5 OR c1%NOTFOUND;
INSERT INTO mesaje VALUES (v_id, v_nume);
END LOOP;
CLOSE c1;

```


END;

/

SELECT * FROM mesaje;



	COD	NUME
1	1	Ciutacu
2	2	Coman
3	3	Daia
4	4	Craciun
5	5	Danila

SEMINAR 5

1. Se afișează printr-un ciclu FOR numele și salariile angajaților care au salariul mai mare de 3000.

set serveroutput on

declare

cursor ang_cursor is select id_angajat, nume, salariu from angajati where salariu>3000;

begin

dbms_output.put_line('Lista cu salariile angajatilor cu salariul >3000');

for ang_rec in ang_cursor loop

dbms_output.put_line('Salariatul ' || ang_rec.nume || ' are salariul: ' || ang_rec.salariu);

end loop;

end;

/

```
Lista cu salariile angajatilor cu salariul >3000
Salariatul Daia are salariul: 6700
Salariatul Craciun are salariul: 4300
Salariatul Cozma are salariul: 3456
Salariatul Cristea are salariul: 3100
```

2. Să se afișeze produsele al căror cantitate totală comandată este mai mare decât o valoare primită drept parametru.

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
CURSOR c_prod (p_val NUMBER) IS
```

```
SELECT p.cod_articol, p.denumire, sum(p.cantitate) total
```

```
FROM produse p, comenzi c
```

```
WHERE p.cod_comanda =c.cod_comanda
```

```
GROUP BY p.cod_articol, p.denumire
```

```
HAVING sum(p.cantitate)>p_val
```

```
ORDER BY total desc;
```

```
v_val NUMBER(5);
```

```
rec_prod c_prod%rowtype;
```

```
BEGIN
```

```
v_val:=5;
```

```
DBMS_OUTPUT.PUT_LINE('Produsele al caror cantitate vânduta este mai mare decat '|| v_val);
```

```
IF NOT c_prod%ISOPEN THEN
```

```
OPEN c_prod (v_val);
```

```
END IF;
```

```
LOOP
```

```
FETCH c_prod into rec_prod;
```

```
EXIT WHEN c_prod%notfound;
```

```
DBMS_OUTPUT.PUT_LINE('Din produsul '||rec_prod.cod_articol||', '||rec_prod.denumire||', s-au  
vandut ' ||rec_prod.total|| ' unitati');
```

```
END LOOP;
```

```
CLOSE c_prod;
```

```
END;
```

```
/
```

```
Produsele al caror cantitate vânduta este mai mare decat 5
Din produsul 45, banane, s-au vandut 30 unitati
Din produsul 90, bere, s-au vandut 19 unitati
Din produsul 12, cartofi, s-au vandut 12 unitati
Din produsul 22, salata, s-au vandut 12 unitati
Din produsul 17, apa, s-au vandut 10 unitati
```

```
PL/SQL procedure successfully completed.
```

3.Sa se afiseze produsele si a carei comanda apartin.

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
--cursorul care va prelua comenzile incheiate
```

```
CURSOR c_com IS
```

```
SELECT cod_articol
```

```
FROM produse
```

```
Where pret>3
```

```
ORDER BY cod_articol;
```

```
--cursorul care, pentru fiecare comanda, va afisa produsele din cadrul acesteia, ordonate descrescator
```

```
CURSOR c_prod (p_cod_articol NUMBER) IS
```

```
SELECT p.cod_articol, p.denumire, c.cod_comanda
```

```
FROM produse p, comenzi c
```

```
WHERE p.cod_comanda=c.cod_comanda
```

```
ORDER BY c.cod_comanda desc;
```

```
BEGIN
```

```
FOR rec_com in c_com LOOP
```

```
DBMS_OUTPUT.PUT_LINE('Produsul cu codul ' || rec_com.cod_articol );
```

```
FOR rec_prod in c_prod(rec_com.cod_articol) LOOP --cursorul primeste drept parametru numarul comenzii care a fost afisata
```

```
DBMS_OUTPUT.PUT_LINE('apartine comenzii ' || rec_prod.cod_comanda);
```

```
END LOOP;
```

```

DBMS_OUTPUT.PUT_LINE('=====');

END LOOP;

END;

/

```

PL/SQL procedure successfully completed.

```

Produsul cu codul 13
apartine comenzii 954
apartine comenzii 888
apartine comenzii 565
apartine comenzii 444
apartine comenzii 235
apartine comenzii 235
apartine comenzii 235
apartine comenzii 235
apartine comenzii 234
apartine comenzii 234
apartine comenzii 224
apartine comenzii 116
apartine comenzii 116
apartine comenzii 112
apartine comenzii 112
=====
Produsul cu codul 18
apartine comenzii 954
apartine comenzii 888
apartine comenzii 565
apartine comenzii 444
apartine comenzii 235
apartine comenzii 235
apartine comenzii 235
apartine comenzii 235
apartine comenzii 234
apartine comenzii 234
apartine comenzii 224
apartine comenzii 116
apartine comenzii 116

```

4. Se creează tabela Situatie care pastreaza informatii despre comenzi: codul, valoarea comenzii. Se adaugă în aceasta coloana TVA, care va păstra valoarea TVA pentru fiecare comandă. Se creează un cursor căruia i se adaugă clauza FOR UPDATE pentru a bloca liniile afectate din tabelă, atunci când cursorul este deschis, iar pentru fiecare comandă din cursor se va calcula valoarea TVA.

```

DROP TABLE situatie;

CREATE TABLE situatie AS

SELECT c.cod_comanda cod, SUM(p.cantitate*p.pret) as valoare

FROM comenzi c, produse p

WHERE c. cod_comanda =p.cod_comanda

GROUP BY c. cod_comanda;

```

```
ALTER TABLE situatie
```

```
ADD(tva NUMBER(10));
```

```
DECLARE
```

```
CURSOR c_situatie IS
```

```
SELECT cod, valoare, tva
```

```
FROM situatie
```

```
FOR UPDATE OF tva NOWAIT;
```

```
BEGIN
```

```
FOR rec_situatie IN c_situatie LOOP
```

```
UPDATE situatie
```

```
SET tva=valoare*0.19
```

```
WHERE cod=rec_situatie.cod;
```

```
DBMS_OUTPUT.PUT_LINE('Comanda ' || rec_situatie.cod || ' are valoarea totala de  
' || rec_situatie.valoare || ' RON si TVA de: ' || rec_situatie.tva );
```

```
END LOOP;
```

```
END;
```

```
/
```

```
SELECT * FROM situatie;
```

	COD	VALOARE	TVA
1	888	8	2
2	565	24	5
3	954	20	4
4	116	12601	2394
5	112	38	7
6	235	490	93
7	444	16	3
8	234	152	29
9	224	120	23

5. Exemplu de mai sus poate fi rescris, actualizarea înregistrărilor din tabela SITUATIE realizându-se cu clauza WHERE CURRENT OF:

```
DROP TABLE situatie;
```

```
CREATE TABLE situatie AS
```

```
SELECT c.cod_comanda cod, SUM(p.cantitate*p.pret) as valoare
```

```
FROM comenzi c, produse p
```

```
WHERE c. cod_comanda =p.cod_comanda
```

```
GROUP BY c. cod_comanda;
```

```
ALTER TABLE situatie
```

```
ADD(tva NUMBER(10));
```

```
DECLARE
```

```
CURSOR c_situatie IS
```

```
SELECT cod, valoare, tva
```

```
FROM situatie
```

```
FOR UPDATE OF tva NOWAIT;
```

```
BEGIN
```

```
FOR rec_situatie IN c_situatie LOOP
```

```
UPDATE situatie
```

```
SET tva=valoare*0.19
```

```
WHERE CURRENT OF c_situatie;
```

```
DBMS_OUTPUT.PUT_LINE('Comanda '||rec_situatie.cod||' are valoarea totala de  
'||rec_situatie.valoare||' RON si tva de: '||rec_situatie.tva );
```

```
END LOOP;
```

```
END;
```

```
/
```

```
SELECT * FROM situatie;
```

	COD	VALOARE	TVA
1	888	8	2
2	565	24	5
3	954	20	4
4	116	12601	2394
5	112	38	7
6	235	490	93
7	444	16	3
8	234	152	29
9	224	120	23

SEMINAR 6

1. Să se afișeze angajatul cu codul 10. Să se trateze eroarea apărută în cazul în care nu există nici un angajat cu acest cod.

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
v_nume VARCHAR2(20);
```

```
BEGIN
```

```
SELECT nume INTO v_nume
```

```
FROM angajati
```

```
WHERE id_angajat=10;
```

```
dbms_output.put_line(v_nume);
```

```
EXCEPTION
```

```
WHEN NO_DATA_FOUND THEN
```

```
dbms_output.put_line('Nu exista angajatul cu acest ID!');
```

```
END;
```

```
/
```

```
Ion
```

```
PL/SQL procedure successfully completed.
```

2. Să se afișeze salariul angajatului cu prenumele Claudia. Să se trateze eroare apărută în cazul în care există mai mulți angajați cu același nume (interogarea SQL din bloc întoarce mai multe înregistrări).

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
sal angajati.salariu%type;
```

```
BEGIN
```

```
select salariu into sal from angajati where prenume='Claudia';
```

```
DBMS_OUTPUT.PUT_LINE('Claudia are salariul de: ' || sal);
```

```
EXCEPTION
```

```
WHEN TOO_MANY_ROWS THEN
```

```
DBMS_OUTPUT.PUT_LINE('Exista mai multi salariati cu numele John! Utilizati un cursor pentru selectie!');
```

```
END;
```

```
/
```

```
Claudia are salariul de: 2300
```

```
PL/SQL procedure successfully completed.
```

3. În exemplu următor se deschide un cursor folosind OPEN și mai apoi se încearcă parcurgerea sa folosind FOR. Instrucțiunea FOR încearcă să deschidă din nou cursorul rezultând excepția CURSOR_ALREADY_OPEN:

```
DECLARE
```

```
cursor c is select nume,prenume,salariul from angajati order by salariul desc;
```

```
BEGIN
```

```
open c;
```

```
for r in c loop
```

```
exit when c%rowcount>5;
```

```
dbms_output.put_line(r.num||' '||r.prenume||' '||r.salariul);
```



```

end loop;

EXCEPTION

WHEN CURSOR_ALREADY_OPEN then

    dbms_output.put_line('Cursorul este deja deschis');

end;

/

Cursorul este deja deschis

PL/SQL procedure successfully completed.

```

4. Să se insereze în tabela sediu un nou sediu cu ID-ul 200, fără a preciza denumirea acestuia. În acest caz va apare o eroare cu codul ORA-01400 prin care programatorul este avertizat de încălcarea unei restricții de integritate. Această excepție poate fi tratată astfel:

```

SET SERVEROUTPUT ON

DECLARE

-- se asociază un nume codului de eroare apărut

INSERT_EXCEPT EXCEPTION;

PRAGMA EXCEPTION_INIT(INSERT_EXCEPT, -01400);

BEGIN

insert into sediu (nume_oras, adresa) values ("Timisoara",NULL);

EXCEPTION

--se tratează eroarea prin numele său

WHEN insert_except THEN

DBMS_OUTPUT.PUT_LINE('Nu ati precizat informatii suficiente pentru sediu');

--se afișează mesajul erorii

DBMS_OUTPUT.PUT_LINE(SQLERRM);

END;

/

Nu ati precizat informatii suficiente pentru sediu
ORA-01400: cannot insert NULL into ("CLAUDIA"."SEDIU"."NUME_ORAS")

PL/SQL procedure successfully completed.

```

5. Să se șteargă toate înregistrările din tabela PRODUSE. Acest lucru va duce la apariția erorii cu codul –2292, reprezentând încălcarea restricției referențiale. Valorile SQLCODE și SQLERRM vor fi inserate în tabela ERORI. ATENTIE! Aceste variabile nu se pot utiliza direct într-o comandă SQL (cum ar fi SELECT, INSERT, UPDATE sau DELETE), drept pentru care vor fi încărcate mai întâi în variabilele PL/SQL COD și MESAJ și apoi utilizate în instrucțiuni SQL.

```
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 produse;

EXCEPTION
WHEN del_exception THEN
dbms_output.put_line('Nu puteti sterge produsul');
dbms_output.put_line('Exista comenzi asignate lui');
cod:=SQLCODE;
mesaj:=SQLERRM;
INSERT INTO erori VALUES(USER, SYSDATE, cod, mesaj);
```

```
END;  
  
/  
  
SELECT * FROM erori;
```

```
Table ERORI created.
```

```
PL/SQL procedure successfully completed.
```

```
>>Query Run In:Query Result
```

6. Să se invoce o excepție în cazul în care utilizatorul încearcă să execute blocul PL/SQL după ora 17.

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
e_exc1 EXCEPTION;
```

```
BEGIN
```

```
IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24'))>=17 THEN
```

```
RAISE e_exc1;
```

```
END IF;
```

```
EXCEPTION
```

```
WHEN e_exc1 THEN
```

```
dbms_output.put_line('Este ora '||TO_CHAR(SYSDATE, 'HH24'));
```

```
dbms_output.put_line('Operatiune permisa doar '||' in timpul programului');
```

```
END;
```

```
/
```

```
Este ora 22  
Operatiune permisa doar in timpul programului
```

```
PL/SQL procedure successfully completed.
```

7. Să se modifice denumirea produsului cu cod-ul 12. Dacă nu se produce nici o actualizare (valoarea atributului SQL%ROWCOUNT este 0) sau dacă apare o altă excepție (clauza OTHERS) atunci să se declanșeze o excepție prin care să fie avertizat utilizatorul:

```
DECLARE
```

```
invalid_prod EXCEPTION;
```

```
BEGIN
```

```
UPDATE produse
```

```
SET denumire='jeleuri'
```

```
WHERE cod_articol=12;
```

```
IF SQL%NOTFOUND THEN
```

```
RAISE invalid_prod;
```

```
END IF;
```

```
EXCEPTION
```

```
WHEN invalid_prod THEN
```

```
DBMS_OUTPUT.PUT_LINE('Nu exista produsul cu acest ID');
```

```
WHEN OTHERS THEN
```

```
DBMS_OUTPUT.PUT_LINE('A aparut o eroare! Nu se poate actualiza denumirea produsului!');
```

```
END;
```

```
/
```

```
PL/SQL procedure successfully completed.
```

COD_ARTICOL	DENUMIRE
12	jeleuri

8. Putem invoca în mod explicit și excepții pre-definite. In exemplul următor este invocată excepția NO_DATA_FOUND:

```
DECLARE
```

```
invalid_prod EXCEPTION;
```

```

BEGIN
UPDATE produse
SET denumire='Laptop ABC'
WHERE cod_articol=3;

IF SQL%NOTFOUND THEN
RAISE NO_DATA_FOUND;
END IF;

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
-----
ORA-01403: no data found

PL/SQL procedure successfully completed.

```

9. Să se atribuie excepției din exemplul anterior un cod și un mesaj de eroare și să se adauge aceste valori în tabela ERORI. Mai întâi construim tabela:

```

CREATE TABLE ERORI(
utilizator varchar2(32),
data_exc date,
cod_exc number(7),
mesaj_exc varchar2(128));

SET SERVEROUTPUT ON

DECLARE
cod NUMBER(7);

```

```

mesaj VARCHAR2(255);

invalid_prod EXCEPTION;

PRAGMA EXCEPTION_INIT(invalid_prod,-20999);


BEGIN

UPDATE produse

SET denumire='Laptop ABC'

WHERE cod_articol=3;


IF SQL%NOTFOUND THEN

RAISE_APPLICATION_ERROR (-20999,'Cod produs invalid!');

END IF;


EXCEPTION

WHEN invalid_prod THEN

DBMS_OUTPUT.PUT_LINE('Nu exista produsul cu acest ID');

cod:=SQLCODE;

mesaj:=SQLERRM;

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

END;

/

SELECT * FROM ERORI;

```

	UTILIZATOR	DATA	COD_EROARE	MESAJ_EROARE
1	CLAUDIA	18-APR-22	-20999	ORA-20999: Cod produs invalid!

SEMINAR 7

1. Procedura modifica_salariul primește doi parametrii: p_id_angajat și procent și majorează cu procentul specificat salariul angajatului cu id_angajat=p_id_angajat:

```
CREATE OR REPLACE
PROCEDURE modifica_salariul_procent
(p_id_angajat IN angajati.id_angajat%type, procent IN number)
IS
v_salariu angajati.salariu%type;
BEGIN
Select salariu into v_salariu from angajati where id_angajat=p_id_angajat;
dbms_output.put_line('Angajatul are salariul de ' || v_salariu);
Update angajati
Set salariu=salariu*(1+procent/100)
Where id_angajat=p_id_angajat;
Select salariu into v_salariu from angajati where id_angajat=p_id_angajat;
Dbms_output.put_line('Angajatul are acum salariul de ' || v_salariu);
END;
/
show errors;
```

```
SQL> CALL modifica_salariul_procent(176, 10)
SQL> EXECUTE modifica_salariul_procent(176, 10)
```

```
begin
  modifica_salariul_procent(10, 10);
end;
/
```

```
Angajatul are salariul de 2560
Angajatul are acum salariul de 2816
```

```
PL/SQL procedure successfully completed.
```

2. Procedura primește ca parametru de tip IN id-ul unui angajat și returnează prin parametrii de tip OUT numele și salariul acestuia:

```
CREATE OR REPLACE PROCEDURE cauta_angajat
```

```
(p_id_angajat IN angajati.id_angajat%type,
```

```
p_nume OUT angajati.nume%type,
```

```
p_salariul OUT angajati.salariu%type)
```

```
IS
```

```
BEGIN
```

```
Select nume, salariu into p_nume, p_salariul from angajati where id_angajat=p_id_angajat;
```

```
DBMS_OUTPUT.PUT_LINE(' Angajatul ' || p_nume || ' are salariul de: ' || p_salariul);
```

```
END;
```

```
/
```

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
v_nume angajati.nume%type;
```

```
v_salariul angajati.salariu%type;
```

```
BEGIN
```

```
Cauta_angajat(10, v_nume, v_salariul);
```

```
END;
```

```
/
```

```
Angajatul Ion are salariul de: 2816
```

```
PL/SQL procedure successfully completed.
```


3. Procedura calculează salariul mediu și îl returnează printr-un parametru de tip OUT:

```
CREATE or REPLACE PROCEDURE sal_mediu
(p_sal_mediu OUT number)
IS
BEGIN
Select AVG(salariu) into p_sal_mediu from angajati;
END;
/
show errors;
```

```
VARIABLE v_sal_mediu NUMBER
EXECUTE sal_mediu(:v_sal_mediu)
Print v_sal_mediu
```

```
Procedure SAL_MEDIU compiled
```

```
PL/SQL procedure successfully completed.
```

```
V_SAL_MEDIU
-----
      3240.6
```

4. Procedura modifică salariul unui angajat doar în cazul în care este mai mic decât media prin apelarea procedurii create mai sus, MODIFICA_SALARIUL_PROCENT. Procedura primește id-ul angajatului ca parametru de intrare și salariul mediu actual și prin returnează prin parametrul de tip IN OUT salariul mediu modificat prin apelul procedurii SAL_MEDIU.

```
CREATE or REPLACE PROCEDURE modifica_salariul_med
(p_id_angajat IN angajati.id_angajat%type, p_sal_mediu IN OUT number)
IS
nume angajati.nume%type;
sal angajati.salariu%type;
BEGIN
Select nume, salariu into nume, sal from angajati where id_angajat= p_id_angajat;
IF sal<p_sal_mediu then
      MODIFICA_SALARIUL_PROCENT (p_id_angajat, 15);
```

```
END IF;
```

```
SAL_MEDIU (p_sal_mediu);
```

```
End;
```

```
/
```

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
v_id_angajat angajati.id_angajat%type;
```

```
v_nume angajati.nume%type;
```

```
v_salariul angajati.salariu%type;
```

```
v_sal_mediu number;
```

```
BEGIN
```

```
--apelul cu id valid =10
```

```
v_id_angajat:=1;
```

```
--apelul procedurii pentru vizualizarea datelor angajatului
```

```
CAUTA_ANGAJAT(v_id_angajat, v_nume, v_salariul);
```

```
--apelul procedurii pentru aflarea salariului mediu. La afisare, se va rotunji la 2 zecimale
```

```
SAL_MEDIU (v_sal_mediu);
```

```
DBMS_OUTPUT.PUT_LINE('Salariul mediu este acum: ' || round(v_sal_mediu,2));
```

```
--apelul procedurii pentru modificarea salariului
```

```
modifica_salariul_med (v_id_angajat, v_sal_mediu);
```

```
CAUTA_ANGAJAT(v_id_angajat, v_nume, v_salariul);
```

```
--apelul cu id invalid
```

```
v_id_angajat:=1230;
```

```
CAUTA_ANGAJAT(v_id_angajat, v_nume, v_salariul);
```

```
modifica_salariul_med (v_id_angajat, v_sal_mediu);
```

Exception

```
When NO_DATA_FOUND then
```

```
DBMS_OUTPUT.PUT_LINE('Angajat inexistent! ID invalid');
```

```
END;
```

```
/
```

```
rollback;
```

5. Funcția `verifica_salariul` returnează TRUE/FALSE dacă salariatul are salariul mai mare/mai mic sau egal cu salariul mediu și NULL dacă salariatul nu există.

```
CREATE OR REPLACE FUNCTION verifica_salariul
```

```
(p_id_angajat IN angajati.id_angajat%type, p_sal_mediu IN number)
```

```
RETURN Boolean
```

```
IS
```

```
v_salariul angajati.salariu%type;
```

```
BEGIN
```

```
SELECT salariul into v_salariul from angajati where id_angajat=p_id_angajat;
```

```
IF v_salariul > p_sal_mediu then
```

```
return true;
```

```
ELSE
```

```
return false;
```

```
end if;
```

```
EXCEPTION
```

```
WHEN no_data_found THEN
```

```
return NULL;
```

```
end;
```

```
/
```

```
show errors
```

```
describe verifica_salariul;
```

Function VERIFICA_SALARIUL compiled

No errors.

FUNCTION verifica_salariul RETURNS PL/SQL BOOLEAN

Argument	Name	Type	In/Out	Default?
----------	------	------	--------	----------

-------	--	--	--	--

P_ID_ANGAJAT	NUMBER(3)	IN	unknown
--------------	-----------	----	---------

P_SAL_MEDIU	NUMBER	IN	unknown
-------------	--------	----	---------

-Apelul intr-un bloc anonim:

SET SERVEROUTPUT ON

DECLARE

v_sal_mediu number;

BEGIN

--apelul procedurii pentru calculul salariului mediu:

SAL_MEDIU (v_sal_mediu);

--primul apel al functiei

IF (verifica_salariul(10, v_sal_mediu) IS NULL) then

dbms_output.put_line('Angajat cu ID invalid!');

elsif (verifica_salariul(10, v_sal_mediu)) then

dbms_output.put_line('Salariatul are salariul mai mare decat media!');

else

dbms_output.put_line(' Salariatul are salariul mai mic decat media!');

end if;

--al doilea apel

IF (verifica_salariul(9, v_sal_mediu) IS NULL) then

dbms_output.put_line('Angajat cu ID invalid!');

elsif (verifica_salariul(9, v_sal_mediu)) then

```

dbms_output.put_line('Salariatul are salariul mai mare decat media!');
else
dbms_output.put_line(' Salariatul are salariul mai mic decat media!');
end if;

--al treilea apel
IF (verifica_salariul(104, v_sal_mediu) IS NULL) then
dbms_output.put_line('Angajat cu ID invalid!');
elsif (verifica_salariul(104, v_sal_mediu)) then
dbms_output.put_line('Salariatul are salariul mai mare decat media!');
else
dbms_output.put_line(' Salariatul are salariul mai mic decat media!');
end if;
END;
/

Angajat cu ID invalid!
Angajat cu ID invalid!
  Salariatul are salariul mai mic decat media!

PL/SQL procedure successfully completed.

```

SEMINAR 8

```

create or replace PACKAGE actualizare_produce IS
procedure adauga_produ
(p_codp produse.cod_articol%type,
p_denp PRODUSE.denumire%type,
p_pretp produse.pret%type,
p_cantitatep produse.cantitate%type,
p_cod_comandap produse.cod_comanda%type,

```

```
p_cod_furnizorp produse.cod_furnizor%type);
```

```
procedure modifica_produc
```

```
(p_codp produse.cod_articol%type,
```

```
p_denp produse.denumire%type,
```

```
p_pretp produse.pret%type,
```

```
p_cantitatep produse.cantitate%type,
```

```
p_cod_comandap produse.cod_comanda%type,
```

```
p_cod_furnizorp produse.cod_furnizor%type);
```

```
procedure modifica_produc
```

```
(p_codp produse.cod_articol%type,
```

```
p_denp produse.denumire%type);
```

```
procedure sterge_produc
```

```
(p_codp produse.cod_articol%type);
```

```
function exista_cod
```

```
(p_codp produse.cod_articol%type)
```

```
return boolean;
```

```
exceptie exception;
```

```
END;
```

```
/
```

```
create or replace PACKAGE BODY actualizare_produc IS
```

```
procedure adauga_produs
(p_codp produse.cod_articol%type,
p_denp produse.denumire%type,
p_pretp produse.pret%type,
p_cantitatep produse.cantitate%type,
p_cod_comandap produse.cod_comanda%type,
p_cod_furnizorp produse.cod_furnizor%type)
```

is

begin

if exista_cod(p_codp) then

raise exceptie;

else

insert into produse(cod_articol,denumire,pret,cantitate,cod_comanda,cod_furnizor) values
(p_codp, p_denp, p_pretp, p_cantitatep,p_cod_comandap, p_cod_furnizorp);

end if;

exception

when exceptie then

dbms_output.put_line('Produs existent!');

end;

```
procedure modifica_produs
```

```
(p_codp produse.cod_articol%type,
```

```
p_denp produse.denumire%type,
```

```
p_pretp produse.pret%type,
```

```
p_cantitatep produse.cantitate%type,
```

```
p_cod_comandap produse.cod_comanda%type,
```

```

p_cod_furnizorp produse.cod_furnizor%type)
is
begin
if exista_cod(p_codp) then
update produse
set denumire=p_denp, pret=p_pret,
cantitate=p_cantitatep,cod_comanda=p_cod_comandap,cod_furnizor=p_cod_furnizorp
where cod_articol=p_codp;
else
raise exceptie;
end if;
exception
when exceptie then
dbms_output.put_line('Produsul cu acest cod nu exista!');
end;

```

--supraîncarcare a procedurii modifica_produș

```

procedure modifica_produș
(p_codp produse.cod_articol%type,
p_denp PRODUSE.denumire%type)
is
begin
if exista_cod(p_codp) then
update produse
set denumire=p_denp
where cod_articol=p_codp;
else

```



```
raise exceptie;
end if;
exception
when exceptie then
dbms_output.put_line('Produsul cu acest cod nu exista!');
end;
```

```
procedure sterge_produus
(p_codp produse.cod_articol%type)
is
begin
if exista_cod(p_codp) then
delete from produse
where cod_articol=p_codp;
dbms_output.put_line('Produsul cu codul ' || p_codp || ' a fost sters!');
else
raise exceptie;
end if;
exception
when exceptie then
dbms_output.put_line('Produsul cu acest cod nu exista!');
end;
```

```
function exista_cod
(p_codp produse.cod_articol%type)
return boolean
is
```

```

v_unu number;

begin
select 1 into v_unu
from produse
where cod_articol=p_codp;
return true;
exception
when no_data_found then
return false;
end;

END;

/

execute actualizare_produse.adauga_produs(505,'ceai', 12, 14,298,1001);

select * from produse where cod_articol=505;

```

COD_ART...	DENUMIRE	PRET	CANTITATE	COD_COMANDA	COD_FURNIZOR
1	505 ceai	12	14	298	1001

--Apelarea procedurii supra-încărcate

```

execute actualizare_produse.modifica_produs(505,'ceai de tei');

select * from produse where cod_articol=505;

```

COD_ARTICOL	DENUMIRE	PRET	CANTI...	COD_COMANDA	COD_FURNIZOR
1	505 ceai de tei	12	14	298	1001

Seminar 9

1. Triggerul se declanșează la operațiile de INSERT, DELETE sau UPDATE pe tabela Produse. In tabela TEMP_LOG se introduce tipul operației, utilizatorul care a executat-o, data curentă.

```

CREATE TABLE temp_log
(tip CHAR(1),
utilizator VARCHAR2(50),

```

```

data DATE DEFAULT SYSDATE);

CREATE OR REPLACE TRIGGER produse_trig_log
BEFORE INSERT or UPDATE or DELETE on produse
DECLARE
v_tip temp_log.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_log(tip, utilizator, data) VALUES (v_tip, user, sysdate);

END;

/

--inserarea in tabela

insert into produse (cod_articol, denumire, pret, cantitate, cod_comanda, cod_furnizor) values (300,
'cafea', 23, 2, 444, 1007);

```

TIP	UTILIZATOR	DATA
1 I	CLAUDIA	22-MAY-22

2. Se creează un trigger care asigură unicitatea codului produsului folosind valorile generate de o secvență.

```

CREATE SEQUENCE produse_secv
START WITH 1
INCREMENT BY 1
MAXVALUE 100
NOCYCLE;

CREATE OR REPLACE TRIGGER generare_codprodus
BEFORE INSERT ON produse
FOR EACH ROW
BEGIN

```

```
SELECT produse_secv.nextval INTO :new.cod_articol FROM dual;
```

```
END;
```

```
/
```

```
show errors;
```

```
Sequence PRODUSE_SECV created.
```

```
Trigger GENERARE_CODPRODUS compiled
```

Name	Value
1 CREATED	22-MAY-22
2 LAST_DDL_TIME	22-MAY-22
3 SEQUENCE_OWNER	CLAUDIA
4 SEQUENCE_NAME	PRODUSE_SECV
5 MIN_VALUE	1
6 MAX_VALUE	100
7 INCREMENT_BY	1
8 CYCLE_FLAG	N
9 ORDER_FLAG	N
10 CACHE_SIZE	20
11 LAST_NUMBER	1
12 SCALE_FLAG	N
13 EXTEND_FLAG	N
14 SESSION_FLAG	N
15 KEEP_VALUE	N
16 DUPLICATED	N
17 SHARDED	N