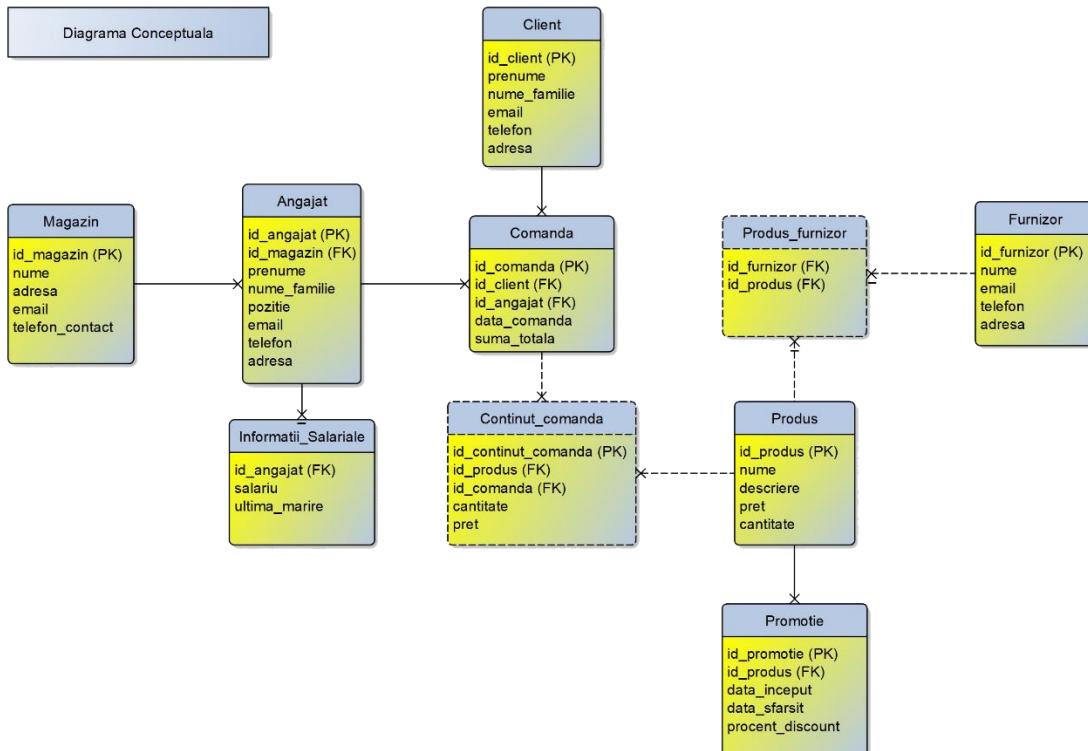
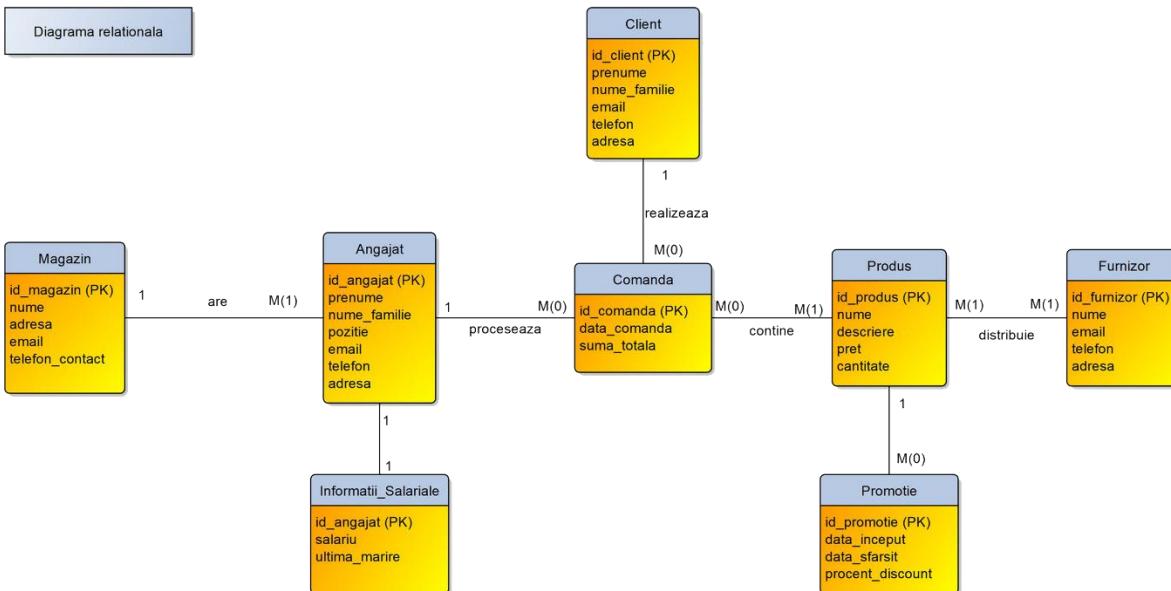


## Supermarket management

Această bază de date va ține evidența stocului dintr-un supermarket, va avea evidența angajaților, a furnizorilor de produse și a comenzielor plasate.

Administrarea acestui supermarket constă în urmărirea comenzielor, a profitului, a furnizorilor și a stocului laolaltă cu date despre angajați și clienți pentru a asigura o experiență cât mai plăcută pentru toate persoanele implicate.



### Cerinta 4 – Crearea tabelelor

```

CREATE TABLE Magazin (
    id_magazin number(4) not null,
    nume varchar(25) not null,
    adresa varchar(30) not null,
    email varchar(30) unique not null,
    telefon_contact varchar(11) unique not null,
    constraint magazin_pk primary key (id_magazin)
);

CREATE TABLE Angajat (
    id_angajat number(4) not null,
    id_magazin number(4) not null,
    prenume varchar(25) not null,
    nume_familie varchar(25) not null,
    pozitie varchar(30) not null,
    email varchar(30) unique not null,
    telefon varchar(11) unique not null,
    adresa varchar(30) not null,
    constraint angajat_pk primary key (id_angajat),
    constraint magazin_angajat_fk foreign key (id_magazin) references
magazin(id_magazin)
);

CREATE TABLE Client (
    id_client number(4) not null,
    prenume varchar(25) not null,
    nume_familie varchar(25) not null,
    email varchar(30) unique not null,
    telefon varchar(11) unique not null,
    adresa varchar(30) not null,
    constraint client_pk primary key (id_client)
);

CREATE TABLE Comanda (
    id_comanda number(4) not null,
    id_client number(4) not null,
    id_angajat number(4) not null,
    data_comanda date default SYSDATE not null,
    suma_totala number(8,2),
    constraint comanda_pk primary key (id_comanda),
    constraint comanda_client_fk foreign key (id_client) references
client(id_client),
    constraint comanda_angajat_fk foreign key (id_angajat) references
angajat(id_angajat) ON DELETE CASCADE
);

CREATE TABLE Produs (
    id_produs number(4) not null,
    nume varchar(20) not null,
    descriere varchar(100) not null,
    pret number(8,2) not null,
    cantitate number(5) not null,
    constraint produs_pk primary key (id_produs)
);

```

```

CREATE TABLE Continut_comanda (
    id_continut_comanda number(4) not null,
    id_produs number(4) not null,
    id_comanda number(4) not null,
    cantitate number(5) not null,
    pret number(8,2) not null,
    constraint continut_comanda_pk primary key (id_continut_comanda),
    constraint continut_comanda_produs_fk foreign key (id_produs)
    references produs(id_produs),
    constraint continut_comanda_comanda_fk foreign key (id_comanda)
    references comanda(id_comanda) ON DELETE CASCADE
);

CREATE TABLE Promotie (
    id_promotie number(4) not null,
    id_produs number(4) not null,
    data_inceput date default SYSDATE not null,
    data_sfarsit date default SYSDATE not null,
    procent_discount number(4,2) not null,
    constraint promotie_pk primary key (id_promotie),
    constraint promotie_produs_fk foreign key (id_produs) references
    produs(id_produs)
);

CREATE TABLE Furnizor (
    id_furnizor number(4) not null,
    nume varchar(20) not null,
    email varchar(30) unique not null,
    telefon varchar(11) unique not null,
    adresa varchar(30) not null,
    constraint furnizor_pk primary key (id_furnizor)
);

CREATE TABLE Produs_furnizor (
    id_furnizor number(4) not null,
    id_produs number(4) not null,
    constraint furnizor_fk foreign key (id_furnizor) references
    furnizor(id_furnizor),
    constraint produs_fk foreign key (id_produs) references
    produs(id_produs)
);

CREATE TABLE informatii_salariale (
    id_angajat number(4) not null,
    salariu number(10) not null,
    ultima_marire date default SYSDATE not null,
    constraint angajat_fk foreign key (id_angajat) references
    angajat(id_angajat) ON DELETE CASCADE,
    constraint info_pk primary key (id_angajat)
);

```

The screenshot shows a SQL worksheet interface with the following details:

- Toolbar:** Includes "Live SQL", "Feedback", "Help", "narcis.necula@my.fmi.unibuc.ro", and a theme switch.
- Header:** "SQL Worksheet" with buttons for "Clear", "Find", "Actions", "Save", and "Run".
- Code Area:** Displays 75 numbered SQL statements (1 to 75) for creating tables Magazin, Angajat, Client, Comanda, Produs, Continut\_comanda, and Promotie, along with their constraints and foreign keys.
- Output Area:** Shows the results of each table creation command as "Table created." repeated 75 times.

```

1 v CREATE TABLE Magazin (
2     id_magazin number(4) not null,
3     nume varchar(25) not null,
4     adresa varchar(30) not null,
5     email varchar(30) unique not null,
6     telefon_contact varchar(11) unique not null,
7
8     constraint magazin_pk primary key (id_magazin)
9 );
10
11 v CREATE TABLE Angajat (
12     id_angajat number(4) not null,
13     id_magazin number(4) not null,
14     prenume varchar(25) not null,
15     nume_familie varchar(25) not null,
16     pozitie varchar(30) not null,
17     email varchar(30) unique not null,
18     telefon varchar(11) unique not null,
19     adresa varchar(30) not null,
20
21     constraint angajat_pk primary key (id_angajat),
22     constraint magazin_angajat_fk foreign key (id_magazin) references magazin(id_magazin)
23 );
24
25 v CREATE TABLE Client (
26     id_client number(4) not null,
27     prenume varchar(25) not null,
28     nume_familie varchar(25) not null,
29     email varchar(30) unique not null,
30     telefon varchar(11) unique not null,
31     adresa varchar(30) not null,
32
33     constraint client_pk primary key (id_client)
34 );
35
36 v CREATE TABLE Comanda (
37     id_comanda number(4) not null,
38     id_client number(4) not null,
39     id_angajat number(4) not null,
40     data_comanda date default SYSDATE not null,
41     suma_totala number(8,2),
42
43     constraint comanda_pk primary key (id_comanda),
44     constraint comanda_client_fk foreign key (id_client) references client(id_client),
45     constraint comanda_angajat_fk foreign key (id_angajat) references angajat(id_angajat) ON DELETE CASCADE
46 );
47
48 v CREATE TABLE Produs (
49     id_produs number(4) not null,
50     nume varchar(20) not null,
51     descriere varchar(100) not null,
52     pret number(8,2) not null,
53     cantitate number(5) not null,
54
55     constraint produs_pk primary key (id_produs)
56 );
57
58 v CREATE TABLE Continut_comanda (
59     id_continut_comanda number(4) not null,
60     id_produs number(4) not null,
61     id_comanda number(4) not null,
62     cantitate number(5) not null,
63     pret number(8,2) not null,
64
65     constraint continut_comanda_pk primary key (id_continut_comanda),
66     constraint continut_comanda_produs_fk foreign key (id_produs) references produs(id_produs),
67     constraint continut_comanda_comanda_fk foreign key (id_comanda) references comanda(id_comanda) ON DELETE CASCADE
68 );
69
70 v CREATE TABLE Promotie (
71     id_promotie number(4) not null,
72     id_produs number(4) not null,
73     data_inceput date default SYSDATE not null,
74     data_sfarsit date default SYSDATE not null,
75     procent_discount number(4,2) not null,

```

Table created.  
Table created.

### Cerință 5 – Adăugare de date:

```

CREATE SEQUENCE SEQ_MAGAZIN
INCREMENT by 1
START WITH 1
MAXVALUE 10000
NOCYCLE;

insert into Magazin
values (SEQ_MAGAZIN.NEXTVAL, 'Lidl', 'Str. V. olt, 139', 'lidl@first.com',
'0700000001');
insert into Magazin
values (SEQ_MAGAZIN.NEXTVAL, 'Kaufland', 'Str. V. olt, 100',
'Kaufland@DrTaberei.com', '0700000002');
insert into Magazin
values (SEQ_MAGAZIN.NEXTVAL, 'Lidl', 'Sos. Berceni 8', 'lidl@second.com',
'0700000003');
insert into Magazin
values (SEQ_MAGAZIN.NEXTVAL, 'Profi', 'Sos. Berceni 9', 'profi@contact.com',
'0700000004');
insert into Magazin
values (SEQ_MAGAZIN.NEXTVAL, 'Mega Image', 'Str. Nitu Vasile, 1', 'mega@image.com',
'0700000005');
insert into Magazin
values (SEQ_MAGAZIN.NEXTVAL, 'Cora', 'Sos. VACARESTI, 12', 'Cora@sunplaza.com',
'0700000006');
commit;
select* from Magazin;

CREATE SEQUENCE SEQ_ANGAJAT
INCREMENT by 1
START WITH 1
MAXVALUE 10000
NOCYCLE;

insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 1, 'Andrei', 'Pascu', 'Casier',
'andrei.pascu@gmail.com', '0700000007', 'Str Albu 1, b1');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 1, 'Sergiu', 'Dan', 'Logistica', 'sergiu.dan@gmail.com',
'0700000008', 'Str Barbu 1, b2');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 1, 'Alex', 'Moraru', 'Manager', 'alex.moraru@gmail.com',
'0700000009', 'Str Coltea 1, 123');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 2, 'Alexandru', 'Marinescu', 'Casier',
'alexandru.marinescu@gmail.com', '0700000010', 'Sos Vacaresti 4, 14');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 2, 'Florin', 'Constantinescu', 'Logistica',
'florin.constantin@gmail.com', '0700000011', 'Str Albu 1, 52');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 2, 'Aniela', 'Macovei', 'Manager',
'aniela.macovei@gmail.com', '0700000012', 'Bld Berceni 4, 31');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 3, 'Corina', 'Caldararu', 'Casier',
'corina.caldararu@gmail.com', '0700000013', 'Sos Albu 1, 45');

```

```

insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 3, 'Ana', 'Matei', 'Logistica', 'ana.matei@gmail.com',
'0700000014', 'Bld Berceni 15, 44');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 3, 'Mihaela', 'Anton', 'Manager',
'mihaela.anton@gmail.com', '0700000015', 'Sos Berceni 1, 554');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 4, 'Narcis', 'Andrei', 'Casier',
'narcis.andrei@gmail.com', '0700000016', 'Str Albu 6, 3');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 4, 'Denis', 'Constantin', 'Logistica',
'denis.constantin@gmail.com', '0700000017', 'Str Oltenitei 1, C34');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 4, 'Andrada', 'Olteanu', 'Manager',
'andrada.olteanu@gmail.com', '0700000018', 'Str Vacaresti 1, b1');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 5, 'Marian', 'Nisoi', 'Casier',
'marian.nisoi@gmail.com', '0700000019', 'Bld Albu 1, b1');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 5, 'Mihai', 'Paraschiva', 'Logistica',
'mihai.paraschiva@gmail.com', '0700000020', 'Str Oltenitei 1, b1');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 5, 'Florin', 'Sandoiu', 'Manager',
'florin.sandoiu@gmail.com', '0700000021', 'Str Oltenitei 1, b1');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 6, 'Simona', 'Constantin', 'Casier',
'simona.constantin@gmail.com', '0700000022', 'Sos Vacaresti 9, b1');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 6, 'Andrei', 'Zidaru', 'Logistica',
'andrei.zidaru@gmail.com', '0700000023', 'Bld Oltenitei 3, b1');
insert into Angajat
values (SEQ_ANGAJAT.NEXTVAL, 6, 'Iulian', 'Tutu', 'Manager', 'iulian.tutu@gmail.com',
'0700000024', 'Sos Vacaresti 1, b1');
commit;
select* from Angajat;

CREATE SEQUENCE SEQ_CLIENT
INCREMENT by 1
START WITH 1
MAXVALUE 10000
NOCYCLE;

INSERT INTO Client
VALUES ( SEQ_CLIENT.NEXTVAL, 'Felicia', 'Bunescu', 'felicia.bunescu@gmail.com',
'0700000025', 'Bld. Marasesti 12');
INSERT INTO Client
VALUES ( SEQ_CLIENT.NEXTVAL, 'Denisa', 'Predescu', 'denisa.predescu@gmail.com',
'0700000026', 'Str Astronomului 12');
INSERT INTO Client
VALUES ( SEQ_CLIENT.NEXTVAL, 'Marius', 'Anca', 'marius.anca@gmail.com', '0700000027',
'Bld. Marasesti 33');
INSERT INTO Client
VALUES ( SEQ_CLIENT.NEXTVAL, 'Constantin', 'Corban', 'constantin.corban@gmail.com',
'0700000028', 'Bld. Dacia 24');
INSERT INTO Client
VALUES ( SEQ_CLIENT.NEXTVAL, 'Andrei', 'Avram', 'andrei.avram@gmail.com',
'0700000029', 'Str. nitu Vasile 11');

```

```

INSERT INTO Client
VALUES ( SEQ_CLIENT.NEXTVAL, 'Bogdan', 'Maftei', 'bogdan.maftei@gmail.com',
'0700000030', 'Sos. Oltenitei 7');
commit;
SELECT * FROM Client;

CREATE SEQUENCE SEQ_COMANDA
INCREMENT by 1
START WITH 1
MAXVALUE 10000
NOCYCLE;

INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 1, 1, to_date('12-01-2022','dd-mm-yyyy'), 123.2);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 3, 4, to_date('22-01-2022','dd-mm-yyyy'), 100.21);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 3, 7, to_date('01-01-2022','dd-mm-yyyy'), 23);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 5, 10, to_date('05-02-2022','dd-mm-yyyy'), 24.54);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 5, 13, to_date('14-02-2022','dd-mm-yyyy'), 129.99);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 2, 16, to_date('14-02-2022','dd-mm-yyyy'), 56.05);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 2, 4, to_date('16-02-2022','dd-mm-yyyy'), 59.99);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 2, 7, to_date('16-03-2022','dd-mm-yyyy'), 114.11);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 6, 4, to_date('09-04-2022','dd-mm-yyyy'), 185.22);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 4, 7, to_date('10-04-2022','dd-mm-yyyy'), 265.99);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 5, 4, to_date('06-05-2022','dd-mm-yyyy'), 157.22);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 6, 7, to_date('05-06-2022','dd-mm-yyyy'), 321);
INSERT INTO Comanda
VALUES ( SEQ_COMANDA.NEXTVAL, 6, 1, to_date('28-06-2022','dd-mm-yyyy'), 12.6);
commit;
SELECT * FROM Comanda;

CREATE SEQUENCE SEQ_PRODUS
INCREMENT by 1
START WITH 1
MAXVALUE 10000
NOCYCLE;

INSERT INTO Produs
VALUES ( SEQ_PRODUS.NEXTVAL, 'Banane', 'Banane dole pret per KG', 6.99, 100);
INSERT INTO Produs
VALUES ( SEQ_PRODUS.NEXTVAL, 'Rosii', 'Rosii pret per KG', 4.49, 100);
INSERT INTO Produs
VALUES ( SEQ_PRODUS.NEXTVAL, 'Rosii cherry', 'Pret la bucată', 12.00, 100);
INSERT INTO Produs
VALUES ( SEQ_PRODUS.NEXTVAL, 'Paine graham', 'Paine cu faina integrala', 7.35, 100);
INSERT INTO Produs

```

```

VALUES ( SEQ_PRODUS.NEXTVAL, 'Painea Campionilor', 'Paine cu multe seminte', 9.25,
100);
INSERT INTO Produs
VALUES ( SEQ_PRODUS.NEXTVAL, 'Mango', 'Mango pret per bucata', 8.10, 100);
INSERT INTO Produs
VALUES ( SEQ_PRODUS.NEXTVAL, 'Lapte Pilos 1.5', 'Lapte 1.5% grasime', 4.99, 100);
INSERT INTO Produs
VALUES ( SEQ_PRODUS.NEXTVAL, 'Lapte Pilos 3.5', 'Lapte 3.5%', 5.79, 100);
INSERT INTO Produs
VALUES ( SEQ_PRODUS.NEXTVAL, 'Telemea Pilos', 'Telemea de vaca, per kg', 23.99, 100);
INSERT INTO Produs
VALUES ( SEQ_PRODUS.NEXTVAL, 'Sushi', 'sushi autentic', 26.99, 100);
INSERT INTO Produs
VALUES ( SEQ_PRODUS.NEXTVAL, 'Sushi XL', 'Sushi XL autentic', 46.99, 100);
INSERT INTO Produs
VALUES ( SEQ_PRODUS.NEXTVAL, 'Gogosi 6 Buc', 'Gogosi gem', 9, 100);
commit;
SELECT * FROM Produs;

CREATE SEQUENCE SEQ_CONTINUT_COMANDA
INCREMENT by 1
START WITH 1
MAXVALUE 10000
NOCYCLE;

INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 11, 1, 2, 46.99);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 10, 1, 1, 29.22);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 6, 2, 10, 10.02);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 3, 3, 2, 11.5);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 2, 4, 6, 4.09);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 9, 5, 4, 32.5);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 8, 6, 10, 5.6);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 8, 7, 10, 6);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 2, 8, 22, 5.18);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 4, 9, 10, 7.35);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 5, 9, 11, 10.15);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 5, 10, 25, 10.64);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 12, 11, 15, 10.48);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 10, 12, 2, 26.99);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 11, 12, 1, 46.99);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 12, 12, 4, 9);

```

```

INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 9, 12, 8, 23);
INSERT INTO Continut_comanda
VALUES ( SEQ_CONTINUT_COMANDA.NEXTVAL, 12, 13, 1, 12.6);
commit;
SELECT * FROM Continut_comanda;

CREATE SEQUENCE SEQ_PROMOTIE
INCREMENT by 1
START WITH 1
MAXVALUE 10000
NOCYCLE;

INSERT INTO Promotie
VALUES ( SEQ_PROMOTIE.NEXTVAL, 3, TO_DATE('16-02-2022','dd-mm-yyyy'), TO_DATE('26-02-2022','dd-mm-yyyy'), 10.00);
INSERT INTO Promotie
VALUES ( SEQ_PROMOTIE.NEXTVAL, 5, TO_DATE('01-04-2021','dd-mm-yyyy'), TO_DATE('16-04-2021','dd-mm-yyyy'), 33.00);
INSERT INTO Promotie
VALUES ( SEQ_PROMOTIE.NEXTVAL, 6, TO_DATE('11-02-2022','dd-mm-yyyy'), TO_DATE('11-03-2022','dd-mm-yyyy'), 25.00);
INSERT INTO Promotie
VALUES ( SEQ_PROMOTIE.NEXTVAL, 7, TO_DATE('09-05-2022','dd-mm-yyyy'), TO_DATE('29-05-2022','dd-mm-yyyy'), 5.00);
INSERT INTO Promotie
VALUES ( SEQ_PROMOTIE.NEXTVAL, 2, TO_DATE('26-02-2021','dd-mm-yyyy'), TO_DATE('20-03-2021','dd-mm-yyyy'), 33.33);
INSERT INTO Promotie
VALUES ( SEQ_PROMOTIE.NEXTVAL, 12, TO_DATE('02-03-2021','dd-mm-yyyy'), TO_DATE('12-03-2021','dd-mm-yyyy'), 15.00);
INSERT INTO Promotie
VALUES ( SEQ_PROMOTIE.NEXTVAL, 9, TO_DATE('11-04-2021','dd-mm-yyyy'), TO_DATE('21-04-2021','dd-mm-yyyy'), 60.00);
INSERT INTO Promotie
VALUES ( SEQ_PROMOTIE.NEXTVAL, 10, TO_DATE('08-07-2022','dd-mm-yyyy'), TO_DATE('18-09-2022','dd-mm-yyyy'), 75.00);
INSERT INTO Promotie
VALUES ( SEQ_PROMOTIE.NEXTVAL, 10, TO_DATE('01-10-2022','dd-mm-yyyy'), TO_DATE('29-11-2022','dd-mm-yyyy'), 25.00);
commit;
SELECT * FROM Promotie;

CREATE SEQUENCE SEQ_FURNIZOR
INCREMENT by 1
START WITH 1
MAXVALUE 10000
NOCYCLE;

INSERT INTO Furnizor
VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Pilos', 'pilos@contact.com', '0700000031', 'Str. Business 43');
INSERT INTO Furnizor
VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Star', 'star@contact.com', '0700000032', 'Str. Business 44');
INSERT INTO Furnizor

```

```
VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Milbona', 'milbona@contact.com', '0700000033', 'Str.  
Business 45');  
INSERT INTO Furnizor  
VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Dole', 'dole@contact.com', '0700000034', 'Str.  
Business 46');  
INSERT INTO Furnizor  
VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Coca-Cola', 'coca.cola@contact.com', '0700000035',  
'Str. Business 47');  
INSERT INTO Furnizor  
VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Uniliver', 'uniliver@contact.com', '0700000036',  
'Str. Business 48');  
commit;  
SELECT * FROM Furnizor;  
  
INSERT INTO Produs_furnizor  
VALUES ( 1, 1);  
INSERT INTO Produs_furnizor  
VALUES ( 1, 2);  
INSERT INTO Produs_furnizor  
VALUES ( 1, 3);  
INSERT INTO Produs_furnizor  
VALUES ( 1, 4);  
INSERT INTO Produs_furnizor  
VALUES ( 1, 5);  
INSERT INTO Produs_furnizor  
VALUES ( 2, 2);  
INSERT INTO Produs_furnizor  
VALUES ( 2, 4);  
INSERT INTO Produs_furnizor  
VALUES ( 2, 5);  
INSERT INTO Produs_furnizor  
VALUES ( 2, 6);  
INSERT INTO Produs_furnizor  
VALUES ( 2, 3);  
INSERT INTO Produs_furnizor  
VALUES ( 3, 6);  
INSERT INTO Produs_furnizor  
VALUES ( 3, 7);  
INSERT INTO Produs_furnizor  
VALUES ( 3, 4);  
INSERT INTO Produs_furnizor  
VALUES ( 3, 3);  
INSERT INTO Produs_furnizor  
VALUES ( 3, 5);  
INSERT INTO Produs_furnizor  
VALUES ( 4, 2);  
INSERT INTO Produs_furnizor  
VALUES ( 4, 9);  
INSERT INTO Produs_furnizor  
VALUES ( 4, 10);  
INSERT INTO Produs_furnizor  
VALUES ( 4, 11);  
INSERT INTO Produs_furnizor  
VALUES ( 4, 12);  
INSERT INTO Produs_furnizor  
VALUES ( 4, 8);  
INSERT INTO Produs_furnizor
```

```
VALUES ( 4, 9);
commit;
SELECT * FROM Produs_furnizor;

insert into informatii_salariale
values (1, 2000, to_date('05-06-2021','dd-mm-yyyy')));
insert into informatii_salariale
values (2, 2200, to_date('02-06-2022','dd-mm-yyyy')));
insert into informatii_salariale
values (3, 3000, to_date('03-05-2023','dd-mm-yyyy')));
insert into informatii_salariale
values (4, 2100, to_date('04-06-2022','dd-mm-yyyy')));
insert into informatii_salariale
values (5, 2500, to_date('05-06-2021','dd-mm-yyyy')));
insert into informatii_salariale
values (6, 4000, to_date('06-06-2022','dd-mm-yyyy')));
insert into informatii_salariale
values (7, 1900, to_date('01-06-2022','dd-mm-yyyy')));
insert into informatii_salariale
values (8, 2200, to_date('01-06-2021','dd-mm-yyyy')));
insert into informatii_salariale
values (9, 2900, to_date('08-06-2021','dd-mm-yyyy')));
insert into informatii_salariale
values (10, 2300, to_date('09-06-2022','dd-mm-yyyy')));
insert into informatii_salariale
values (11, 2700, to_date('05-06-2022','dd-mm-yyyy')));
insert into informatii_salariale
values (12, 3500, to_date('11-06-2022','dd-mm-yyyy')));
insert into informatii_salariale
values (13, 2100, to_date('01-02-2023','dd-mm-yyyy')));
insert into informatii_salariale
values (14, 2900, to_date('03-03-2023','dd-mm-yyyy')));
insert into informatii_salariale
values (15, 4100, to_date('04-04-2023','dd-mm-yyyy')));
insert into informatii_salariale
values (16, 2900, to_date('01-02-2023','dd-mm-yyyy')));
insert into informatii_salariale
values (17, 3200, to_date('01-03-2023','dd-mm-yyyy')));
insert into informatii_salariale
values (18, 4900, to_date('03-02-2023','dd-mm-yyyy')));

commit;
select* from informatii_salariale;
```

Live SQL

SQL Worksheet

Feedback Help narcis.necula@my.fmi.unibuc.ro Actions Clear Find Save Run

```

281 v INSERT INTO Produs_furnizor
282   VALUES ( 3, 6);
283 v INSERT INTO Produs_furnizor
284   VALUES ( 3, 7);
285 v INSERT INTO Produs_furnizor
286   VALUES ( 3, 4);
287 v INSERT INTO Produs_furnizor
288   VALUES ( 3, 3);
289 v INSERT INTO Produs_furnizor
290   VALUES ( 3, 5);
291 v INSERT INTO Produs_furnizor
292   VALUES ( 4, 2);
293 v INSERT INTO Produs_furnizor
294   VALUES ( 4, 9);
295 v INSERT INTO Produs_furnizor
296   VALUES ( 4, 18);
297 v INSERT INTO Produs_furnizor
298   VALUES ( 4, 11);
299 v INSERT INTO Produs_furnizor
300   VALUES ( 4, 12);
301 v INSERT INTO Produs_furnizor
302   VALUES ( 4, 8);
303 v INSERT INTO Produs_furnizor
304   VALUES ( 4, 9);
305 commit;
306 SELECT * FROM Produs_furnizor;
307
308 v insert into informatii_salariale
309   values (1, 2000, to_date('05-06-2021','dd-mm-yyyy'));
310 v insert into informatii_salariale
311   values (2, 2200, to_date('02-06-2022','dd-mm-yyyy'));
312 v insert into informatii_salariale
313   values (3, 3000, to_date('03-05-2023','dd-mm-yyyy'));
314 v insert into informatii_salariale
315   values (4, 2100, to_date('04-06-2022','dd-mm-yyyy'));
316 v insert into informatii_salariale
317   values (5, 2500, to_date('05-06-2021','dd-mm-yyyy'));
318 v insert into informatii_salariale
319   values (6, 4000, to_date('06-06-2022','dd-mm-yyyy'));
320 v insert into informatii_salariale
321   values (7, 1900, to_date('01-06-2022','dd-mm-yyyy'));
322 v insert into informatii_salariale
323   values (8, 2200, to_date('01-06-2021','dd-mm-yyyy'));
324 v insert into informatii_salariale
325   values (9, 2900, to_date('08-06-2021','dd-mm-yyyy'));
326 v insert into informatii_salariale
327   values (10, 2300, to_date('09-06-2022','dd-mm-yyyy'));
328 v insert into informatii_salariale
329   values (11, 2700, to_date('05-06-2022','dd-mm-yyyy'));
330 v insert into informatii_salariale
331   values (12, 3500, to_date('11-06-2022','dd-mm-yyyy'));
332 v insert into informatii_salariale
333   values (13, 2100, to_date('01-02-2023','dd-mm-yyyy'));
334 v insert into informatii_salariale
335   values (14, 2900, to_date('03-03-2023','dd-mm-yyyy'));
336 v insert into informatii_salariale
337   values (15, 4100, to_date('04-04-2023','dd-mm-yyyy'));
338 v insert into informatii_salariale
339   values (16, 2900, to_date('01-02-2023','dd-mm-yyyy'));
340 v insert into informatii_salariale
341   values (17, 3200, to_date('01-03-2023','dd-mm-yyyy'));
342 v insert into informatii_salariale
343   values (18, 4900, to_date('03-02-2023','dd-mm-yyyy'));
344
345 commit;
346 select* from informatii_salariale;
347
    7      1900  01-JUN-22
    8      2200  01-JUN-21
    9      2900  08-JUN-21
   10      2300  09-JUN-22
   11      2700  05-JUN-22
   12      3500  11-JUN-22
   13      2100  01-FEB-23
   14      2900  03-MAR-23
   15      4100  04-APR-23
   16      2900  01-FEB-23
   17      3200  01-MAR-23
   18      4900  03-FEB-23

```

Download CSV

18 rows selected.

## Cerință 6:

*Enunț:* Determinați produsul care se vinde cel mai bine. Dacă există mai multe cu același număr de vânzări se afișează toate.

```

CREATE OR REPLACE PROCEDURE cel_mai_bine_vandut_produc
IS
    TYPE tabel_indexat IS TABLE OF continut_comanda.cantitate%type INDEX BY
PLS_INTEGER;
    TYPE tabel_imbricat IS TABLE OF continut_comanda.id_produs%type;

    nr_unitati_vandute tabel_indexat; -- numarul de unitati vandute din fiecare
produs vandut
    produse_vandute tabel_imbricat := tabel_imbricat(); --Idurile produselor
vandute

    cantitate_maxima number := 0;
    i number := 0; -- indice de parcurgere
    nume_produs produs.nume%type;
BEGIN

    FOR vanzare IN ( SELECT id_produs, cantitate
                      FROM continut_comanda) LOOP
        i := 1;

        WHILE i <= produse_vandute.COUNT AND produse_vandute(i) != vanzare.id_produs LOOP -- Cautăm dacă există deja înregistrare de vânzare a acestui
produs
            i := i + 1;
        END LOOP;

        IF i = produse_vandute.COUNT + 1 THEN -- În caz că nu am găsit produsul
deja adăugat îl adăugăm în lista de produse vândute
            produse_vandute.EXTEND;
            produse_vandute(produse_vandute.COUNT) := vanzare.id_produs;
            nr_unitati_vandute(produse_vandute.COUNT) := vanzare.cantitate;

            IF nr_unitati_vandute(produse_vandute.COUNT) > cantitate_maxima
THEN -- Calculăm cantitatea maximă în timp real pt eficientă
                cantitate_maxima :=
nr_unitati_vandute(produse_vandute.COUNT);
            END IF;
            ELSE -- în cazul unui produs deja înregistrat că vândut doar adăugăm
cantitatea nouă la cea existentă
                nr_unitati_vandute(i) := nr_unitati_vandute(i) + vanzare.cantitate;

                IF nr_unitati_vandute(i) > cantitate_maxima THEN-- Calculăm
cantitatea maximă în timp real pt eficientă
                    cantitate_maxima := nr_unitati_vandute(i);
                END IF;
            END IF;
        END LOOP;

        IF cantitate_maxima = 0 THEN
            dbms_output.put_line('Nu avem vânzări înregistrate');
        END IF;
    END IF;
END;

```

```

    ELSE
        FOR i IN 1..produse_vandute.COUNT LOOP -- Cautam numele produsului cu
        cantitate maxima vanduta si ii aflisam un mesaj intuitiv
            IF nr_unitati_vandute(i) = cantitate_maxima THEN
                SELECT nume INTO nume_produs
                FROM produs
                WHERE id_produs = produse_vandute(i);

                dbms_output.put_line('Produs cel mai bine vandut: ' || nume_produs || ' '
in cantitate de: ' || cantitate_maxima || ' unitati');
            END IF;
        END LOOP;
    END IF;
END;
/
EXECUTE cel_mai_bine_vandut_produs;

```

```

1 CREATE OR REPLACE PROCEDURE cel_mai_bine_vandut_produs
2 IS
3     TYPE tabel_indexat IS TABLE OF continut_comanda.cantitate%type INDEX BY PLS_INTEGER;
4     TYPE tabel_imbricat IS TABLE OF continut_comanda.id_produs%type;
5
6     nr_unitati_vandute tabel_indexat; -- numarul de unitati vandute din fiecare produs vandut
7     produse_vandute tabel_imbricat(); --Idurile produselor vandute
8
9     cantitate_maxima number := 0;
10    i number := 0; -- indice de parcursare
11    nume_produs produs.nume%type;
12
13 BEGIN
14
15     FOR vanzare IN ( SELECT id_produs, cantitate
16                         FROM continut_comanda) LOOP
17         i := 1;
18
19         WHILE i <= produse_vandute.COUNT AND produse_vandute(i) != vanzare.id_produs LOOP -- Cautam daca exista deja inregistrare de vanzare a acestui produs
20             i := i + 1;
21         END LOOP;
22
23         IF i = produse_vandute.COUNT + 1 THEN -- In caz ca nu am gasit produsul deja adaugat il adaugam in lista de produse vandute
24             produse_vandute.EXTEND;
25             produse_vandute(produse_vandute.COUNT) := vanzare.id_produs;
26             nr_unitati_vandute(produse_vandute.COUNT) := vanzare.cantitate;
27
28             IF nr_unitati_vandute(produse_vandute.COUNT) > cantitate_maxima THEN -- Calculam cantitatea maxima in timp real pt eficienta
29                 cantitate_maxima := nr_unitati_vandute(produse_vandute.COUNT);
30             END IF;
31         ELSE -- in cazul unui produs deja inregistrat ca vandut doar adaugam cantitatea noua la cea existenta
32             nr_unitati_vandute(i) := nr_unitati_vandute(i) + vanzare.cantitate;
33
34             IF nr_unitati_vandute(i) > cantitate_maxima THEN-- Calculam cantitatea maxima in timp real pt eficienta
35                 cantitate_maxima := nr_unitati_vandute(i);
36             END IF;
37         END IF;
38     END LOOP;
39
40     IF cantitate_maxima = 0 THEN
41         dbms_output.put_line('Nu avem vanzari inregistrate');
42     ELSE
43         FOR i IN 1..produse_vandute.COUNT LOOP -- Cautam numele produsului cu cantitate maxima vanduta si ii aflisam un mesaj intuitiv
44             IF nr_unitati_vandute(i) = cantitate_maxima THEN
45                 SELECT nume INTO nume_produs
46                 FROM produs
47                 WHERE id_produs = produse_vandute(i);

48                 dbms_output.put_line('Produs cel mai bine vandut: ' || nume_produs || ' in cantitate de: ' || cantitate_maxima || ' unitati');
49             END IF;
50         END LOOP;
51     END IF;
52 END;
53 /
54
55 EXECUTE cel_mai_bine_vandut_produs;

```

Procedure created.

Statement processed.  
Produs cel mai bine vandut: Painea Campionilor in cantitate de: 36 unitati

### Cerinta 7:

*Enunț: Determinați magazinul/magazinele cu cele mai bune salarii în medie, pentru angajații săi și afișați doar numele acestuia/acestora.*

```

CREATE OR REPLACE PROCEDURE calcularea_magazinului_cu_plata_cea_mai_buna
IS
    TYPE lista_id IS TABLE OF angajat.id_magazin%type;
    CURSOR salariile_medii IS -- creez cursor pentru aflarea salariilor medii oferite
de fiecare magazin
        SELECT AVG(salariu), id_magazin
        FROM informatii_salariale JOIN (SELECT id_angajat, id_magazin
                                         FROM angajat) USING
(id_angajat)
        GROUP BY id_magazin;

    media_curenta number;
    magazin_curent angajat.id_magazin%type;
    maxim_medie_salarii number := 0;
    lista_magazine lista_id := lista_id();
    nume_magazin magazin.nume%type;
BEGIN
    OPEN salariile_medii;
    LOOP
        FETCH salariile_medii into media_curenta, magazin_curent; -- parcurg
datele din cursor una cate una
        EXIT WHEN salariile_medii%NOTFOUND;

        IF media_curenta > maxim_medie_salarii THEN -- aflu media maxima si
salvez toate magazinele care o ofera
            maxim_medie_salarii := media_curenta;
            lista_magazine.DELETE;
            lista_magazine.extend;
            lista_magazine(lista_magazine.COUNT) := magazin_curent;
        ELSIF media_curenta = maxim_medie_salarii THEN
            lista_magazine.extend;
            lista_magazine(lista_magazine.COUNT) := magazin_curent;
        END IF;
    END LOOP;
    CLOSE salariile_medii;

    IF lista_magazine.COUNT = 0 THEN -- in cazul in care nu avem date sause
intampla ceva la preluarea datelor.
        dbms_output.put_line('NU avem date despre salariile oferite');
    ELSE

        FOR i IN 1..lista_magazine.COUNT LOOP -- preiau numele magazinului bazat pe
id-ul salvat mai devreme
            SELECT nume INTO nume_magazin
            FROM magazin
            WHERE id_magazin = lista_magazine(i);
    END IF;
END;

```

```

        dbms_output.put_line('Cele mai bune salarii sunt oferite de ' || 
nume_magazin || ' ');
        END LOOP;
    END IF;
END calcularea_magazinului_cu_plata_cea_mai_buna;
/
EXECUTE calcularea_magazinului_cu_plata_cea_mai_buna;

```

The screenshot shows the Oracle SQL Developer interface with the 'Live SQL' tab selected. In the SQL Worksheet, the following PL/SQL code is displayed and executed:

```

1 v CREATE OR REPLACE PROCEDURE calcularea_magazinului_cu_plata_cea_mai_buna
2 IS
3     TYPE lista_id IS TABLE OF angajat.id_magazin%type;
4     CURSOR salariile_medi IS -- creez cursor pentru afisarea salariilor medii oferite de fiecare magazin
5         SELECT AVG(salariu), id_magazin
6             FROM informatii_salariale JOIN (SELECT id_angajat, id_magazin
7                                         FROM angajat) USING (id_angajat)
8             GROUP BY id_magazin;
9
10    media_curenta number;
11    magazin_curent angajat.id_magazin%type;
12    maxim_medic_salarii number := 0;
13    lista_magazine lista_id := lista_id();
14    nume_magazin magazin.nume%type;
15 v BEGIN
16     OPEN salariile_medi;
17 v     LOOP
18         FETCH salariile_medi into media_curenta, magazin_curent; -- parcurg datele din cursor una cate una
19         EXIT WHEN salariile_medi%NOTFOUND;
20
21 v         IF media_curenta > maxim_medic_salarii THEN -- aflu media maxima si salvez toate magazinele care o ofera
22             maxim_medic_salarii := media_curenta;
23             lista_magazine.DELETE;
24             lista_magazine.extend;
25             lista_magazine(lista_magazine.COUNT) := magazin_curent;
26 v         ELSIF media_curenta = maxim_medic_salarii THEN
27             lista_magazine.extend;
28             lista_magazine(lista_magazine.COUNT) := magazin_curent;
29         END IF;
30     END LOOP;
31     CLOSE salariile_medi;
32
33 v     IF lista_magazine.COUNT = 0 THEN -- in cazul in care nu avem date sause intampla ceva la preluarea datelor.
34         dbms_output.put_line('NU avem date despre salariile oferite');
35 v     ELSE
36
37         FOR i IN 1..lista_magazine.COUNT LOOP -- preiau numele magazinului bazat pe id-ul salvat mai devreme
38             SELECT nume INTO nume_magazin
39                 FROM magazin
40                 WHERE id_magazin = lista_magazine(i);
41
42             dbms_output.put_line('Cele mai bune salarii sunt oferite de ' || nume_magazin || ' ');
43         END LOOP;
44     END IF;
45
46 END calcularea_magazinului_cu_plata_cea_mai_buna;
47 v /
48
49 EXECUTE calcularea_magazinului_cu_plata_cea_mai_buna;
50
51

```

After execution, the output is shown at the bottom of the worksheet:

```

Procedure created.

Statement processed.
Cele mai bune salarii sunt oferite de Cora

```

### Cerință 8:

*Enunț: Efectuați măririle de salariu pentru angajații unui magazin dat care sunt la rând să le primească (are mai mult de un an de la ultima mărire). Afipați numele, salariul vechi și cel nou, și poziția angajatului care primește mărirea.*

```
-- Creez noi tipuri de date globale pentru a putea returna toate datele de care am nevoie într-o lista de obiecte
CREATE OR REPLACE TYPE tabel_de_returnat_mariri IS OBJECT ( nume varchar(100),
                                                               salariu_vechi number(10),
                                                               salariu_nou number(10),
                                                               pozitie varchar(100));
/
CREATE OR REPLACE TYPE tabel_mariri IS TABLE OF tabel_de_returnat_mariri;
/

CREATE OR REPLACE FUNCTION angajati_acordare_marire(nume_magazin magazin.nume%type,
                                                       procent_de_marire number)
RETURN tabel_mariri IS
    nr_magazine_numite_la_fel number := 0;
    lista_mariri tabel_mariri := tabel_mariri();
    TOO_MANY_STORES EXCEPTION;
    NO_STORE_FOUND EXCEPTION;
BEGIN
    SELECT COUNT(id_magazin) INTO nr_magazine_numite_la_fel
    FROM magazin
    WHERE ((UPPER(nume) LIKE UPPER(nume_magazin)) OR (UPPER(adresa) LIKE
    UPPER(nume_magazin)));
    -- Verific dacă magazinul introdus are același nume cu alt magazin din lista
    -- sau dacă adresa sau numele introdus este valid
    IF nr_magazine_numite_la_fel > 1 THEN -- în cazul în care avem mai multe
    magazine cu același nume trebuie introdusa adresa
        raise TOO_MANY_STORES;
    ELSIF nr_magazine_numite_la_fel = 0 THEN -- în cazul în care numele sau adresa
    introdusa nu au fost gasite
        raise NO_STORE_FOUND;
    END IF;
    FOR linie IN (  SELECT prenume || ' ' || nume_familie as nume,
                           salariu as salariu_vechi,
                           (salariu + (salariu * procent_de_marire / 100))
    as salariu_nou,
                           pozitie,
                           a.id_angajat -- preluam datele ce trebuie scris
    returnate
                           FROM angajat a JOIN informatii_salariale i_s ON (a.id_angajat =
    i_s.id_angajat)
                           JOIN magazin m ON (m.id_magazin =
    a.id_magazin) -- folosim 3 tabele - magazin.informatii_salariale și angajat
                           WHERE ( (UPPER(m.nume) LIKE UPPER(nume_magazin)) OR
    --verific dacă numele sau adresa se potrivesc și condiția de acordare a maririi
```

```

        (UPPER(m.adresa) LIKE UPPER(nume_magazin)))
AND
        i_s.ultima_marire NOT BETWEEN (SYSDATE-365)
AND SYSDATE) LOOP
    lista_mariri.extend;
    lista_mariri(lista_mariri.COUNT) := tabel_de_returnat_mariri(linie.nume,
linie.salariu_vechi, linie.salariu_nou, linie.pozitie); -- adaugam in lista de
returnat

    UPDATE informatii_salariale -- facem schimarea salariului cu cel marit
    SET salariu = linie.salariu_nou,
        ultima_marire = SYSDATE
    WHERE id_angajat = linie.id_angajat;
END LOOP;

    IF lista_mariri.COUNT = 0 THEN -- in cazul in care pt magazinul ales nu exista
angajati care primesc marire
        raise NO_DATA_FOUND;
    END IF;

    RETURN lista_mariri;

EXCEPTION -- definim exceptiile
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Nu avem angajati care se califica pentru
mariri');
    WHEN TOO_MANY_STORES THEN
        RAISE_APPLICATION_ERROR(-20003, 'Prea multe magazine cu acelasi nume,
introduceti adresa si incercati din nou');
    WHEN NO_STORE_FOUND THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nu am gasit magazin cu acest nume sau
adresa');
    WHEN others THEN
        RAISE_APPLICATION_ERROR(-20001, SQLERRM);

END angajati_acordare_marire;
/

DECLARE
    lista_afisare tabel_mariri := tabel_mariri();
BEGIN
    -- Decommentati cate una si rulati individual
    -- lista_afisare := angajati_acordare_marire('CORA', 10); -- nu avem mariri
deci NO_DATA_FOUND
    -- lista_afisare := angajati_acordare_marire('lidl', 10); -- TOO_MANY_STORES
    -- lista_afisare := angajati_acordare_marire('Str. V. olt, 139', 10); -- OK ,
fix pt cea anterioara
    -- lista_afisare := angajati_acordare_marire('Metro', 10); -- NO_STORE_FOUND

    FOR i IN 1..lista_afisare.COUNT LOOP
        dbms_output.put_line(i || '.' || lista_afisare(i).nume || ' -- Salariul
vechi: ' || lista_afisare(i).salariu_vechi || ' -- Salariul nou: ' ||
lista_afisare(i).salariu_nou || ' -- Pozitia: ' || lista_afisare(i).pozitie);
    END LOOP;
END;
/

```

## Crearea tipurilor de date și a funcției

```

1 v CREATE OR REPLACE TYPE tabel_de_returnat_mariri IS OBJECT ( nume varchar(100),
2                                         salariu_vechi number(10),
3                                         salariu_nou number(10),
4                                         pozitie varchar(100));
5 v /
6 CREATE OR REPLACE TYPE tabel_mariri IS TABLE OF tabel_de_returnat_mariri;
7 v /
8
9 CREATE OR REPLACE FUNCTION angajati_acordare_marire(nume_magazin magazin.numextype, procent_de_marire number)
10 RETURN tabel_mariri IS
11
12     nr_magazine_numite_la_fel number := 0;
13     lista_mariri tabel_mariri := tabel_mariri();
14
15     TOO_MANY_STORES EXCEPTION;
16     NO_STORE_FOUND EXCEPTION;
17 v BEGIN
18     SELECT COUNT(id_magazin) INTO nr_magazine_numite_la_fel
19     FROM magazin
20     WHERE ((UPPER(nume) LIKE UPPER(nume_magazin)) OR (UPPER(adresa) LIKE UPPER(nume_magazin))); -- Verific daca magazinul introdus are acelasi nume cu magazinul din tabela
21                                         -- sau daca adresa sau numele introdus este valid
22
23 v     IF nr_magazine_numite_la_fel > 1 THEN -- in cazul in care avem mai multe magazine cu acelasi nume trebuie introdusa adresa
24         raise TOO_MANY_STORES;
25 v     ELSIF nr_magazine_numite_la_fel = 0 THEN -- in cazul in care numele sau adresa introdusa nu au fost gasite
26         raise NO_STORE_FOUND;
27     END IF;
28
29 v     FOR linie IN ( SELECT prenume || ' ' || nume_familie as nume,
30                     salariu as salariu_vechi,
31                     (salariu + (salariu * procent_de_marire / 100)) as salariu_nou,
32                     pozitie,
33                     a.id_angajat -- preluam datele ce trebuie returnate
34                 FROM angajat a JOIN informatii_salariale_i_s ON (a.id_angajat = i_s.id_angajat)
35                 JOIN magazin m ON (m.id_magazin = a.id_magazin) -- folosim 3 table - magazin, informatii_salariale si angajat
36                 WHERE ( (UPPER(m.nume) LIKE UPPER(nume_magazin)) OR
37                         (UPPER(m.adresa) LIKE UPPER(nume_magazin))) AND
38                         i_s.ulima_marire NOT BETWEEN (SYSDATE-365) AND SYSDATE) LOOP
39         lista_mariri.extend;
40         lista_mariri(lista_mariri.COUNT) := tabel_de_returnat_mariri(linie.nume, linie.salariu_vechi, linie.salariu_nou, linie.pozitie); -- adaugam
41
42 v     UPDATE informatii_salariale -- facem schimarea salariului cu cel marit
43         SET salariu = linie.salariu_nou,
44             ultima_marire = SYSDATE
45         WHERE id_angajat = linie.id_angajat;
46     END LOOP;
47
48 v     IF lista_mariri.COUNT = 0 THEN -- in cazul in care pt magazinul ales nu exista angajati care primesc marire
49         raise NO_DATA_FOUND;
50     END IF;
51
52     RETURN lista_mariri;
53
54 v     EXCEPTION -- definim exceptiile
55     WHEN NO_DATA_FOUND THEN
56         RAISE_APPLICATION_ERROR(-20004, 'Nu avem angajati care se califica pentru mariri');
57 v     WHEN TOO_MANY_STORES THEN
58         RAISE_APPLICATION_ERROR(-20003, 'Prea multe magazine cu acelasi nume, introduceti adresa si incercati din nou');
59 v     WHEN NO_STORE_FOUND THEN
60         RAISE_APPLICATION_ERROR(-20002, 'Nu am gasit magazin cu acest nume sau adresa');
61 v     WHEN others THEN
62         RAISE_APPLICATION_ERROR(-20001, SQLERRM);
63
64 END angajati_acordare_marire;
65 v /
66
67 DECLARE
68     lista_afisare tabel_mariri := tabel_mariri();
69 v BEGIN
70     -- Deconsemnat cate una si rulati individual
71     -- lista_afisare := angajati_acordare_marire('CORA', 10); -- nu avem mariri deci NO_DATA_FOUND
72     -- lista_afisare := angajati_acordare_marire('lidl', 10); -- TOO_MANY_STORES
73     -- lista_afisare := angajati_acordare_marire('Str. V. olt, 139', 10); -- OK , fix pt cea anterioara
74     -- lista_afisare := angajati_acordare_marire('Metro', 10); -- NO_STORE_FOUND
75
76     FOR i IN 1..lista_afisare.COUNT LOOP
77         dbms_output.put_line(i || ' ' || lista_afisare(i).nume || ' -- Salariul vechi: ' || lista_afisare(i).salariu_vechi || ' -- Salariul nou: ' ||
78     END LOOP;
79 END;
80 v /
81
82 select * from informatii_salariale;

```

Type created.

Type created.

Function created.

## Cazul NO\_DATA\_FOUND

```

1 v CREATE OR REPLACE TYPE tabel_de_returnat_mariri IS OBJECT ( nume varchar(100),
2                                         salariu_vechi number(10),
3                                         salariu_nou number(10),
4                                         pozitie varchar(100));
5 v /
6 CREATE OR REPLACE TYPE tabel_mariri IS TABLE OF tabel_de_returnat_mariri;
7 v /
8
9 CREATE OR REPLACE FUNCTION angajati_acordare_marire(nume_magazin magazin.nume%type, procent_de_marire number)
10 RETURN tabel_mariri IS
11
12     nr_magazine_numite_la_fel number := 0;
13     lista_mariri tabel_mariri := tabel_mariri();
14
15     TOO_MANY_STORES EXCEPTION;
16     NO_STORE_FOUND EXCEPTION;
17 v BEGIN
18     SELECT COUNT(id_magazin) INTO nr_magazine_numite_la_fel
19     FROM magazin
20     WHERE ((UPPER(nume) LIKE UPPER(nume_magazin)) OR (UPPER(adresa) LIKE UPPER(nume_magazin))); -- Verific daca magazinul introdus are acelasi nume cu magazinul din tabela
21                                         -- sau daca adresa sau numele introdus este valid
22
23 v     IF nr_magazine_numite_la_fel > 1 THEN -- in cazul in care avem mai multe magazine cu acelasi nume trebuie sa introducem adresa
24         raise TOO_MANY_STORES;
25 v     ELSIF nr_magazine_numite_la_fel = 0 THEN -- in cazul in care numele sau adresa introdusa nu au fost gasite
26         raise NO_STORE_FOUND;
27     END IF;
28
29 v     FOR linie IN (  SELECT prenume || ' ' || nume_familie as nume,
30                         salariu as salariu_vechi,
31                         (salariu + (salariu * procent_de_marire / 100)) as salariu_nou,
32                         pozitie,
33                         a.id_angajat -- preluam datele ce trebuie returnate
34                     FROM angajat_a JOIN informatii_salariale_i_s ON (a.id_angajat = i_s.id_angajat)
35                     JOIN magazin m ON (m.id_magazin = a.id_magazin) -- folosim 3 table - magazin, informatii_salariale si angajat
36                     WHERE ( (UPPER(m.nume) LIKE UPPER(nume_magazin)) OR (UPPER(m.adresa) LIKE UPPER(nume_magazin))) AND
37                         i_s.ultima_marire NOT BETWEEN (SYSDATE-365) AND SYSDATE) LOOP
38
39         lista_mariri.extend;
40         lista_mariri(lista_mariri.COUNT) := tabel_de_returnat_mariri(linie.nume, linie.salariu_vechi, linie.salariu_nou, linie.pozitie); -- adaugam
41
42 v         UPDATE informatii_salariale -- facem schimarea salariului cu cel marit
43             SET salariu = linie.salariu_nou,
44                 ultima_marire = SYSDATE
45             WHERE id_angajat = linie.id_angajat;
46     END LOOP;
47
48 v     IF lista_mariri.COUNT = 0 THEN -- in cazul in care pt magazinul ales nu exista angajati care primesc marire
49         raise NO_DATA_FOUND;
50     END IF;
51
52     RETURN lista_mariri;
53
54 v     EXCEPTION -- definim exceptiile
55     WHEN NO_DATA_FOUND THEN
56         RAISE_APPLICATION_ERROR(-20004, 'Nu avem angajati care se califica pentru mariri');
57 v     WHEN TOO_MANY_STORES THEN
58         RAISE_APPLICATION_ERROR(-20003, 'Prea multe magazine cu acelasi nume, introduceti adresa si incercati din nou');
59 v     WHEN NO_STORE_FOUND THEN
60         RAISE_APPLICATION_ERROR(-20002, 'Nu am gasit magazin cu acest nume sau adresa');
61 v     WHEN others THEN
62         RAISE_APPLICATION_ERROR(-20001, SQLERRM);
63
64 END angajati_acordare_marire;
65 v /
66
67 DECLARE
68     lista_afisare tabel_mariri := tabel_mariri();
69 v BEGIN
70     -- Deconectati cate una si rulati individual
71     lista_afisare := angajati_acordare_marire('CORA', 10); -- nu avem mariri deci NO_DATA_FOUND
72     -- lista_afisare := angajati_acordare_marire('lidi', 10); -- TOO_MANY_STORES
73     -- lista_afisare := angajati_acordare_marire('Str. V. olt, 139', 10); -- OK , fix pt cea anterioara
74     -- lista_afisare := angajati_acordare_marire('Metro', 10); -- NO_STORE_FOUND
75
76 v     FOR i IN 1..lista_afisare.COUNT LOOP
77         dbms_output.put_line(i || ' ' || lista_afisare(i).nume || ' -- Salariul vechi: ' || lista_afisare(i).salariu_vechi || ' -- Salariul nou: ' ||
78     END LOOP;
79 END;
80 v /
81
82 select * from informatii_salariale;

```

ORA-20004: Nu avem angajati care se califica pentru mariri ORA-06512: at "SQL\_YNSWEFXACCLFGFKDBXFDNPB.ANGAJATI\_ACORDARE\_MARIRE", line 48  
ORA-06512: at line 5  
ORA-06512: at "SYS.DBMS\_SQL", line 1721

## Cazul NO\_STORE\_FOUND

```

1 v CREATE OR REPLACE TYPE tabel_de_returnat_mariri IS OBJECT ( nume varchar(100),
2                                         salariu_vechi number(10),
3                                         salariu_nou number(10),
4                                         pozitie varchar(100));
5 v /
6 CREATE OR REPLACE TYPE tabel_mariri IS TABLE OF tabel_de_returnat_mariri;
7 v /
8
9 CREATE OR REPLACE FUNCTION angajati_acordare_marire(nume_magazin magazin.nume%type, procent_de_marire number)
10 RETURN tabel_mariri IS
11
12     nr_magazine_numite_la_fel number := 0;
13     lista_mariri tabel_mariri := tabel_mariri();
14
15     TOO_MANY_STORES EXCEPTION;
16     NO_STORE_FOUND EXCEPTION;
17 v BEGIN
18     SELECT COUNT(id_magazin) INTO nr_magazine_numite_la_fel
19     FROM magazin
20     WHERE ((UPPER(nume) LIKE UPPER(nume_magazin)) OR (UPPER(adresa) LIKE UPPER(nume_magazin))); -- Verific daca magazinul introdus are acelasi nume cu magazinul existent
21                                         -- sau daca adresa sau numele introdus este valid
22
23 v     IF nr_magazine_numite_la_fel > 1 THEN -- in cazul in care avem mai multe magazine cu acelasi nume trebuie inrtodusa adresa
24         raise TOO_MANY_STORES;
25 v     ELSIF nr_magazine_numite_la_fel = 0 THEN -- in cazul in care numele sau adresa introdusa nu au fost gasite
26         raise NO_STORE_FOUND;
27     END IF;
28
29 v     FOR linie IN (  SELECT prenume || ' ' || nume_familie as nume,
30                         salariu as salariu_vechi,
31                         (salariu + (salariu * procent_de_marire / 100)) as salariu_nou,
32                         pozitie,
33                         a.id_angajat -- preluam datele ce trebuie returnate
34                         FROM angajat a JOIN informatii_salariale i_s ON (a.id_angajat = i_s.id_angajat)
35                         JOIN magazin m ON (m.id_magazin = a.id_magazin) -- folosim 3 table - magazin, informatii_salariale si angajat
36                         WHERE ( (UPPER(m.nume) LIKE UPPER(nume_magazin)) OR
37                                 (UPPER(m.adresa) LIKE UPPER(nume_magazin))) AND
38                                 i_s.ultima_marire NOT BETWEEN (SYSDATE-365) AND SYSDATE) LOOP
39         lista_mariri.extend;
40         lista_mariri(lista_mariri.COUNT) := tabel_de_returnat_mariri(linie.nume, linie.salariu_vechi, linie.salariu_nou, linie.pozitie); -- adaugam
41
42 v         UPDATE informatii_salariale -- facem schimarea salariului cu cel marit
43             SET salariu = linie.salariu_nou,
44                 ultima_marire = SYSDATE
45             WHERE id_angajat = linie.id_angajat;
46     END LOOP;
47
48 v     IF lista_mariri.COUNT = 0 THEN -- in cazul in care pt magazinul ales nu exista angajati care primesc marire
49         raise NO_DATA_FOUND;
50     END IF;
51
52     RETURN lista_mariri;
53
54 v     EXCEPTION -- definim exceptiile
55         WHEN NO_DATA_FOUND THEN
56             RAISE_APPLICATION_ERROR(-20004, 'Nu avem angajati care se califica pentru marire');
57 v         WHEN TOO_MANY_STORES THEN
58             RAISE_APPLICATION_ERROR(-20003, 'Prea multe magazine cu acelasi nume, introduceti adresa si incercati din nou');
59 v         WHEN NO_STORE_FOUND THEN
60             RAISE_APPLICATION_ERROR(-20002, 'Nu am gasit magazin cu acest nume sau adresa');
61 v         WHEN others THEN
62             RAISE_APPLICATION_ERROR(-20001, SQLERRM);
63 |
64     END angajati_acordare_marire;
65 v /
66
67 DECLARE
68     lista_afisare tabel_mariri := tabel_mariri();
69 v BEGIN
70     -- Deconectati cate una si rulati individual
71     -- lista_afisare := angajati_acordare_marire('CORA', 10); -- nu avem mariri deci NO_DATA_FOUND
72     -- lista_afisare := angajati_acordare_marire('lidl', 10); -- TOO_MANY_STORES
73     -- lista_afisare := angajati_acordare_marire('Str. V. olt, 139', 10); -- OK , fix pt cea anterioara
74     lista_afisare := angajati_acordare_marire('Metro', 10); -- NO_STORE_FOUND
75
76 v     FOR i IN 1..lista_afisare.COUNT LOOP
77         dbms_output.put_line(i || ' ' || lista_afisare(i).nume || ' -- Salariul vechi: ' || lista_afisare(i).salariu_vechi || ' -- Salariul nou: ' ||
78     END LOOP;
79 END;
80 v /
81
82 select * from informatii_salariale;

ORA-20002: Nu am gasit magazin cu acest nume sau adresa ORA-06512: at "SQL_VNSWEFXACCLFGFKDBXFDNPARB.ANGAJATI_ACORDARE_MARIRE", line 52
ORA-06512: at line 8
ORA-06512: at "SYS.DBMS_SQL", line 1721

```

## Cazul TOO\_MANY\_STORES

```

1 v CREATE OR REPLACE TYPE tabel_de_returnat_mariri IS OBJECT ( nume varchar(100),
2                                         salariu_vechi number(10),
3                                         salariu_nou number(10),
4                                         pozitie varchar(100));
5 v /
6 CREATE OR REPLACE TYPE tabel_mariri IS TABLE OF tabel_de_returnat_mariri;
7 v /
8
9 CREATE OR REPLACE FUNCTION angajati_acordare_marire(nume_magazin magazin.nume%type, procent_de_marire number)
10 RETURN tabel_mariri IS
11
12     nr_magazine_numite_la_fel number := 0;
13     lista_mariri tabel_mariri := tabel_mariri();
14
15     TOO_MANY_STORES EXCEPTION;
16     NO_STORE_FOUND EXCEPTION;
17 v BEGIN
18     SELECT COUNT(id_magazin) INTO nr_magazine_numite_la_fel
19     FROM magazin
20     WHERE ((UPPER(nume) LIKE UPPER(nume_magazin)) OR (UPPER(adresa) LIKE UPPER(nume_magazin))); -- Verific daca magazinul introdus are acelasi nume cu magazinul existent
21                                         -- sau daca adresa sau numele introdus este valid
22
23 v     IF nr_magazine_numite_la_fel > 1 THEN -- in cazul in care avem mai multe magazine cu acelasi nume trebuie inrtodusa adresa
24         raise TOO_MANY_STORES;
25 v     ELSIF nr_magazine_numite_la_fel = 0 THEN -- in cazul in care numele sau adresa introdusa nu au fost gasite
26         raise NO_STORE_FOUND;
27     END IF;
28
29 v     FOR linie IN (  SELECT prenume || ' ' || nume_familie as nume,
30                         salariu as salariu_vechi,
31                         (salariu + (salariu * procent_de_marire / 100)) as salariu_nou,
32                         pozitie,
33                         a.id_angajat -- prelauam datele ce trebuieesc returnate
34                     FROM angajat a JOIN informatii_salariale i_s ON (a.id_angajat = i_s.id_angajat)
35                         JOIN magazin m ON (m.id_magazin = a.id_magazin) -- folosim 3 tabele - magazin, informatii_salariale si angajat
36                         WHERE ( (UPPER(m.nume) LIKE UPPER(nume_magazin)) OR
37                                 (UPPER(m.adresa) LIKE UPPER(nume_magazin))) AND
38                                 i_s.ulima_marire NOT BETWEEN (SYSDATE-365) AND SYSDATE ) LOOP
39         lista_mariri.extend;
40         lista_mariri(lista_mariri.COUNT) := tabel_de_returnat_mariri(linie.nume, linie.salariu_vechi, linie.salariu_nou, linie.pozitie); -- adaugam
41
42 v         UPDATE informatii_salariale -- facem schimarea salariului cu cel marit
43             SET salariu = linie.salariu_nou,
44                 ultima_marire = SYSDATE
45             WHERE id_angajat = linie.id_angajat;
46     END LOOP;
47
48 v     IF lista_mariri.COUNT = 0 THEN -- in cazul in care pt magazinul ales nu exista angajati care primesc marire
49         raise NO_DATA_FOUND;
50     END IF;
51
52     RETURN lista_mariri;
53
54 v     EXCEPTION -- definim exceptiile
55     WHEN NO_DATA_FOUND THEN
56         RAISE_APPLICATION_ERROR(-20004, 'Nu avem angajati care se califica pentru mariri');
57 v     WHEN TOO_MANY_STORES THEN
58         RAISE_APPLICATION_ERROR(-20003, 'Prea multe magazine cu acelasi nume, introduceti adresa si incercati din nou');
59 v     WHEN NO_STORE_FOUND THEN
60         RAISE_APPLICATION_ERROR(-20002, 'Nu am gasit magazin cu acest nume sau adresa');
61 v     WHEN others THEN
62         RAISE_APPLICATION_ERROR(-20001, SQLERRM);
63
64     END angajati_acordare_marire;
65 v /
66
67     DECLARE
68         lista_afisare tabel_mariri := tabel_mariri();
69 v BEGIN
70     -- Deconsemnat cate una si rulati individual
71     -- lista_afisare := angajati_acordare_marire('CORA', 10); -- nu avem mariri deci NO_DATA_FOUND
72     lista_afisare := angajati_acordare_marire('lid1', 10); -- TOO_MANY_STORES
73     -- lista_afisare := angajati_acordare_marire('Str. V. olt, 139', 10); -- OK , fix pt cea anterioara
74     -- lista_afisare := angajati_acordare_marire('Metro', 10); -- NO_STORE_FOUND
75
76 v     FOR i IN 1..lista_afisare.COUNT LOOP
77         dbms_output.put_line(i || ' ' || lista_afisare(i).nume || ' -- Salariul vechi: ' || lista_afisare(i).salariu_vechi || ' -- Salariul nou: ' ||
78     END LOOP;
79 END;
80 v /
81
82 select * from informatii_salariale;

```

DRA-20003: Prea multe magazine cu acelasi nume, introduceti adresa si incercati din nou ORA-06512: at "SQL\_YNSWEFXACLFGFKDBXFONPARB.ANGAJATI\_ACORDARE\_MARIRE", line 50  
ORA-06512: at line 6  
ORA-06512: at "SYS.DBMS\_SQL", line 1721

*Cazul care rulează fără erori*

```

1 v CREATE OR REPLACE TYPE tabel_de_returnat_mariri IS OBJECT ( nume varchar(100),
2                                     salariu_vechi number(10),
3                                     salariu_nou number(10),
4                                     pozitie varchar(100));
5 v /
6 CREATE OR REPLACE TYPE tabel_mariri IS TABLE OF tabel_de_returnat_mariri;
7 v /
8
9 CREATE OR REPLACE FUNCTION angajati_acordare_marire(nume_magazin magazin.nume%type, procent_de_marire number)
10 RETURN tabel_mariri IS
11
12     nr_magazine_numite_la_fel number := 0;
13     lista_mariri tabel_mariri := tabel_mariri();
14
15     TOO_MANY_STORES EXCEPTION;
16     NO_STORE_FOUND EXCEPTION;
17 v BEGIN
18     SELECT COUNT(id_magazin) INTO nr_magazine_numite_la_fel
19     FROM magazin
20     WHERE ((UPPER(nume) LIKE UPPER(nume_magazin)) OR (UPPER(adresa) LIKE UPPER(nume_magazin))); -- Verific daca magazinul introdus are acelasi nume cu magazinul din baza de date
21     -- sau daca adresa sau numele introdus este valid
22
23 v     IF nr_magazine_numite_la_fel > 1 THEN -- in cazul in care avem mai multe magazine cu acelasi nume trebuie introdusa adresa
24         raise TOO_MANY_STORES;
25 v     ELSIF nr_magazine_numite_la_fel = 0 THEN -- in cazul in care numele sau adresa introdusa nu au fost gasite
26         raise NO_STORE_FOUND;
27     END IF;
28
29 v     FOR linie IN (  SELECT prenume || ' ' || nume_familie as nume,
30                         salariu as salariu_vechi,
31                         (salariu + (salariu * procent_de_marire / 100)) as salariu_nou,
32                         pozitie,
33                         a.id_angajat -- preluam datele ce trebuie returnate
34                     FROM angajat A JOIN informatii_salariale i_s ON (a.id_angajat = i_s.id_angajat)
35                         JOIN magazin m ON (m.id_magazin = a.id_magazin) -- folosim 3 table - magazin, informatii_salariale si angajat
36                         WHERE ( (UPPER(m.nume) LIKE UPPER(nume_magazin)) OR (UPPER(m.adresa) LIKE UPPER(nume_magazin))) AND
37                               i_s.ultima_marire NOT BETWEEN (SYSDATE-365) AND SYSDATE ) LOOP
38
39         lista_mariri.extend;
40         lista_mariri(lista_mariri.COUNT) := tabel_de_returnat_mariri(linie.nume, linie.salariu_vechi, linie.salariu_nou, linie.pozitie); -- adaugam
41
42 v     UPDATE informatii_salariale -- facem schimarea salariului cu cel marit
43         SET salariu = linie.salariu_nou,
44             ultima_marire = SYSDATE
45         WHERE id_angajat = linie.id_angajat;
46     END LOOP;
47
48 v     IF lista_mariri.COUNT = 0 THEN -- in cazul in care pt magazinul ales nu exista angajati care primesc marire
49         raise NO_DATA_FOUND;
50     END IF;
51
52     RETURN lista_mariri;
53
54 v     EXCEPTION -- definim exceptiile
55     WHEN NO_DATA_FOUND THEN
56         RAISE_APPLICATION_ERROR(-20004, 'Nu avem angajati care se califica pentru mariri');
57 v     WHEN TOO_MANY_STORES THEN
58         RAISE_APPLICATION_ERROR(-20003, 'Prea multe magazine cu acelasi nume, introduceti adresa si incercati din nou');
59 v     WHEN NO_STORE_FOUND THEN
60         RAISE_APPLICATION_ERROR(-20002, 'Nu am gasit magazin cu acest nume sau adresa');
61 v     WHEN others THEN
62         RAISE_APPLICATION_ERROR(-20001, SQLERRM);
63
64     END angajati_acordare_marire;
65 v /
66
67 DECLARE
68     lista_afisare tabel_mariri := tabel_mariri();
69 v BEGIN
70     -- Deconectati cate una si rulati individual
71     -- lista_afisare := angajati_acordare_marire('CORA', 10); -- nu avem mariri deci NO_DATA_FOUND
72     -- lista_afisare := angajati_acordare_marire('lidl', 10); -- TOO_MANY_STORES
73     lista_afisare := angajati_acordare_marire('Str. V. olt, 139', 10); -- OK , fix pt cea anterioara
74     -- lista_afisare := angajati_acordare_marire('Metro', 10); -- NO_STORE_FOUND
75
76 v     FOR i IN 1..lista_afisare.COUNT LOOP
77         dbms_output.put_line(i || ' ' || lista_afisare(i).nume || ' -- Salariul vechi: ' || lista_afisare(i).salariu_vechi || ' -- Salariul nou: ' ||
78     END LOOP;
79 END;
80 v /
81
82 select * from informatii_salariale;

Statement processed.
1. Andrei Pascu -- Salariul vechi: 2000 -- Salariul nou: 2200 -- Pozitia: Casier

```

### Cerinta 9:

*Enunț: Un furnizor vine și anunță că toate produsele lor care conțin sau au fost în contact cu alune au fost contaminate și trebuie contactați toți clienții care au cumpărat produse ce conțin un produs dat, de la acest furnizor.*

*Trebuie să afișăm o listă cu emailurile și numerele de telefon ale tuturor clienților afectați.*

```

CREATE OR REPLACE PROCEDURE clienti_afectati_de_rechemare(furnizor_afectat
furnizor.nume%type,
element_rechemat produs.descriere%type) IS
    TYPE client_afectat IS TABLE OF VARCHAR(1000);

        lista_clienti_afectati client_afectat := client_afectat(); -- Definim lista ce
va fi afisata
        id_furnizor_afectat furnizor.id_furnizor%type;

        NO_AFFECTED_CLIENTS EXCEPTION; -- declar exceptia care nu e predefinita
BEGIN
    SELECT id_furnizor INTO id_furnizor_afectat -- Verificam daca furnizorul
inclus exista / este unic
        FROM furnizor
        WHERE UPPER(nume) LIKE UPPER(furnizor_afectat);

        SELECT 'Date de contact client - email: ' || email || ', numarul de telefon: '
|| telefon || ' va trebui contactat.' BULK COLLECT INTO lista_clienti_afectati -- Cream textul de afisat cu datele clientului
        FROM client
        WHERE id_client IN (SELECT id_client -- preluam toate comenziile care contin
produse afectate
                            FROM comanda
                            WHERE id_comanda IN (SELECT id_comanda
                                FROM
continut_comanda
                                WHERE id_produs IN
(SELECT id_produs -- preluam toate produsele care contin elementul rechemat pt ca
asta le afecteaza
                            FROM produs
                            WHERE (UPPER(descriere) LIKE '%' || UPPER(element_rechemat) || '%')
                                AND id_produs IN (SELECT id_produs
                                    FROM produs_furnizor
                                    WHERE id_furnizor =
id_furnizor_afectat))));

        IF lista_clienti_afectati.COUNT = 0 THEN
            RAISE NO_AFFECTED_CLIENTS;
        END IF;
    
```

```

FOR i IN 1..lista_clienti_afectati.COUNT LOOP
    dbms_output.put_line(lista_clienti_afectati(i));
END LOOP;

EXCEPTION -- definim exceptiile
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Nu exista furnizori afectati pentru
produsele vandute.');" -- suprascriu exceptiile cu mesaje customize
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20003, 'Furnizorul nu poate fi identificat
pentru ca exista mai multi cu acelasi nume.');" -- suprascriu exceptiile cu mesaje
customizate
    WHEN NO_AFFECTED_CLIENTS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nici un client nu a fost afectat de
aceste produse!');" -- definesc exceptia nou declarata
    WHEN others THEN
        RAISE_APPLICATION_ERROR(-20001, SQLERRM);
END clienti_afectati_de_rechemare;
/

-- EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- OK
-- EXECUTE clienti_afectati_de_rechemare('Pepsi', 'cola'); -- NO_DATA_FOUND

-- Urmatoarele comenzi (urmatoarele 7 linii) vor fi rulate impreuna
-- -- Adaug date dublura
-- INSERT INTO Furnizor
-- VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Pilos', 'pilosX@contact.com', '0700000099', 'Str.
Business xx');

-- EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- TOO_MANY_ROWS

-- -- Elimin datele adaugate
-- DELETE FROM furnizor
-- WHERE telefon like '0700000099'

-- EXECUTE clienti_afectati_de_rechemare('PILOS', 'Caseina'); -- NO_AFFECTED_CLIENTS

```

### *Crearea procedurii:*

```

1 CREATE OR REPLACE PROCEDURE clienti_afectati_de_rechemare(furnizor_afectat furnizor.nume%type,
2                                         element_rechemat produs.descriere%type) IS
3
4     TYPE client_afectat IS TABLE OF VARCHAR(1000);
5
6     lista_clienti_afectati client_afectat := client_afectat(); -- Definim lista ce va fi afisata
7     id_furnizor_afectat furnizor.id_furnizor%type;
8
9     NO_AFFECTED_CLIENTS EXCEPTION; -- declar exceptia care nu e predefinita
10    BEGIN
11        SELECT id_furnizor INTO id_furnizor_afectat -- Verificam daca furnizorul inclus exista / este unic
12        FROM furnizor
13        WHERE UPPER(nume) LIKE UPPER(furnizor_afectat);
14
15        SELECT 'Date de contact client - email: ' || email || ', numarul de telefon: ' || telefon || ' va trebui contactat.' BULK COLLECT INTO lista_clienti_afectati
16        FROM client
17        WHERE id_client IN (SELECT id_client -- preluam toate comenzi care contin produse afectate
18                            FROM comanda
19                            WHERE id_comanda IN (SELECT id_comanda
20                                      FROM continut_comanda
21                                      WHERE id_produs IN (SELECT id_produs -- preluam toate produsele care contin elementul rechemat pt ca asta sa fie
22                                              FROM produs
23                                              WHERE (UPPER(descriere) LIKE '%' || UPPER(element_rechemat) || '%')
24                                              AND id_produs IN (SELECT id_produs
25                                              FROM produs_furnizor
26                                              WHERE id_furnizor = id_furnizor_afectat)));
27
28        IF lista_clienti_afectati.COUNT = 0 THEN
29            RAISE NO_AFFECTED_CLIENTS;
30        END IF;
31
32        FOR i IN 1..lista_clienti_afectati.COUNT LOOP
33            dbms_output.put_line(lista_clienti_afectati(i));
34        END LOOP;
35
36        EXCEPTION -- definim exceptiile
37            WHEN NO_DATA_FOUND THEN
38                RAISE_APPLICATION_ERROR(-20004, 'Nu exista furnizori afectati pentru produsele vandute.); -- suprascriu exceptiile cu mesaje customizate
39            WHEN TOO_MANY_ROWS THEN
40                RAISE_APPLICATION_ERROR(-20003, 'Furnizorul nu poate fi identificat pentru ca exista mai multi cu acelasi nume.);-- suprascriu exceptiile
41            WHEN NO_AFFECTED_CLIENTS THEN
42                RAISE_APPLICATION_ERROR(-20002, 'Nici un client nu a fost afectat de aceste produse!); -- definesc exceptia nou declarata
43            WHEN others THEN
44                RAISE_APPLICATION_ERROR(-20001, SQLERRM);
45    END clienti_afectati_de_rechemare;
46 /
47
48 -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- OK
49 -- EXECUTE clienti_afectati_de_rechemare('Pepsi', 'cola'); -- NO_DATA_FOUND
50
51 -- Urmatoarele comenzi (urmatoarele 7 linii) vor fi rulate impreuna
52 -- -- Adaug date dublura
53 -- INSERT INTO Furnizor
54 -- VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Pilos', 'pilosX@contact.com', '0700000099', 'Str. Business xx');
55
56 -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- TOO_MANY_ROWS
57
58 -- -- Elimin datele adaugate
59 -- DELETE FROM furnizor
60 -- WHERE telefon like '0700000099'
61
62 -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'Caseina'); -- NO_AFFECTED_CLIENTS
63

```

Procedure created.

*Cazul în care nici un client nu e afectat:*

```

1
2 v CREATE OR REPLACE PROCEDURE clienti_afectati_de_rechemare(furnizor_afectat furnizor.nume%type,
3                                     element_rechemat produs.descriere%type) IS
4     TYPE client_afectat IS TABLE OF VARCHAR(1000);
5
6     lista_clienti_afectati client_afectat := client_afectat(); -- Definim lista ce va fi afisata
7     id_furnizor_afectat furnizor.id_furnizor%type;
8
9     NO_AFFECTED_CLIENTS EXCEPTION; -- declar exceptia care nu e predefinita
10 v BEGIN
11     SELECT id_furnizor INTO id_furnizor_afectat -- Verificam daca furnizorul inclus exista / este unic
12     FROM furnizor
13     WHERE UPPER(nume) LIKE UPPER(furnizor_afectat);
14
15 v     SELECT 'Date de contact client - email: ' || email || ', numarul de telefon: ' || telefon || ' va trebui contactat.' BULK COLLECT INTO lista_clie
16     FROM client
17     WHERE id_client IN (SELECT id_client -- preluam toate comenziile care contin produse afectate
18                           FROM comanda
19                           WHERE id_comanda IN (SELECT id_comanda
20                                     FROM continut_comanda
21                                     WHERE id_produs IN (SELECT id_produs -- preluam toate produsele care contin elementul rechemat pt ca asta sa fie
22                                           FROM produs
23                                           WHERE (UPPER(descriere) LIKE '%' || UPPER(element_rechemat) || '%')
24                                           AND id_produs IN (SELECT id_produs
25                                             FROM produs_furnizor
26                                             WHERE id_furnizor = id_furnizor_afectat)));
27
28 v     IF lista_clienti_afectati.COUNT = 0 THEN
29         RAISE NO_AFFECTED_CLIENTS;
30     END IF;
31
32 v     FOR i IN 1..lista_clienti_afectati.COUNT LOOP
33         dbms_output.put_line(lista_clienti_afectati(i));
34     END LOOP;
35
36 v     EXCEPTION -- definim exceptiile
37         WHEN NO_DATA_FOUND THEN
38             RAISE_APPLICATION_ERROR(-20004, 'Nu exista furnizori afectati pentru produsele vandute.');?>
39 v         WHEN TOO_MANY_ROWS THEN
40             RAISE_APPLICATION_ERROR(-20003, 'Furnizorul nu poate fi identificat pentru ca exista mai multi cu acelasi nume.');?>
41 v         WHEN NO_AFFECTED_CLIENTS THEN
42             RAISE_APPLICATION_ERROR(-20002, 'Nici un client nu a fost afectat de aceste produse!');?>
43 v         WHEN others THEN
44             RAISE_APPLICATION_ERROR(-20001, SQLERRM);
45 END clienti_afectati_de_rechemare;
46 v /
47
48 -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- OK
49 -- EXECUTE clienti_afectati_de_rechemare('Pepsi', 'cola'); -- NO_DATA_FOUND
50
51 -- Urmatoarele comenzi (urmatoarele 7 linii) vor fi rulate impreuna
52 -- - Adaug date dublura
53 -- INSERT INTO Furnizor
54 -- VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Pilos', 'pilosX@contact.com', '0700000099', 'Str. Business xx');
55
56 -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- TOO_MANY_ROWS
57
58 -- - Elimin datele adaugate
59 -- DELETE FROM furnizor
60 -- WHERE telefon like '0700000099'
61
62 EXECUTE clienti_afectati_de_rechemare('PILOS', 'Caserina'); -- NO_AFFECTED_CLIENTS
63

```

```

ORA-20002: Nici un client nu a fost afectat de aceste produse! ORA-06512: at "SQL_YNSWEFXACCLFGFKDBXFONPARB.CLIENTI_AFECTATI_DE_RECHEMARE", line 41
ORA-06512: at line 1
ORA-06512: at "SYS.DBMS_SQL", line 1721

```

Nu găsim date care să corespundă furnizorului dat:

```

1
2 v CREATE OR REPLACE PROCEDURE clienti_afectati_de_rechemare(furnizor_afectat furnizor.nume%type,
3                                     element_rechemat produs.descriere%type) IS
4     TYPE client_afectat IS TABLE OF VARCHAR(1000);
5
6     lista_clienti_afectati client_afectat := client_afectat(); -- Definim lista ce va fi afisata
7     id_furnizor_afectat furnizor.id_furnizor%type;
8
9     NO_AFFECTED_CLIENTS EXCEPTION; -- declar exceptia care nu e predefinita
10 v BEGIN
11     SELECT id_furnizor INTO id_furnizor_afectat -- Verificam daca furnizorul inclus exista / este unic
12     FROM furnizor
13     WHERE UPPER(nume) LIKE UPPER(furnizor_afectat);
14
15 v     SELECT 'Date de contact client - email: ' || email || ', numarul de telefon: ' || telefon || ' va trebui contactat.' BULK COLLECT INTO lista_clie
16     FROM client
17     WHERE id_client IN (SELECT id_comanda -- preluam toate comenzi care contin produse afectate
18                           FROM comanda
19                           WHERE id_comanda IN (SELECT id_comanda
20                                     FROM continut_comanda
21                                     WHERE id_produs IN (SELECT id_produs -- preluam toate produsele care contin elementul rechemat pt ca ast
22                                           FROM produs
23                                           WHERE (UPPER(descriere) LIKE '%' || UPPER(element_rechemat) || '%')
24                                           AND id_produs IN (SELECT id_produs
25                                             FROM produs_furnizor
26                                             WHERE id_furnizor = id_furnizor_afectat)));
27
28 v     IF lista_clienti_afectati.COUNT = 0 THEN
29         RAISE NO_AFFECTED_CLIENTS;
30     END IF;
31
32 v     FOR i IN 1..lista_clienti_afectati.COUNT LOOP
33         dbms_output.put_line(lista_clienti_afectati(i));
34     END LOOP;
35
36 v     EXCEPTION -- definim exceptiile
37         WHEN NO_DATA_FOUND THEN
38             RAISE_APPLICATION_ERROR(-20004, 'Nu exista furnizori afectati pentru produsele vandute.');?>
39 v         WHEN TOO_MANY_ROWS THEN
40             RAISE_APPLICATION_ERROR(-20003, 'Furnizorul nu poate fi identificat pentru ca exista mai multi cu acelasi nume.');?>
41 v         WHEN NO_AFFECTED_CLIENTS THEN
42             RAISE_APPLICATION_ERROR(-20002, 'Nici un client nu a fost afectat de aceste produse!');?>
43 v         WHEN others THEN
44             RAISE_APPLICATION_ERROR(-20001, SQLERRM);
45     END clienti_afectati_de_rechemare;
46 v /
47
48 -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- OK
49 EXECUTE clienti_afectati_de_rechemare('Pepsi', 'cola'); -- NO_DATA_FOUND
50
51 -- Urmatoarele comenzi (urmatoarele 7 linii) vor fi rulate impreuna
52 -- -- Adaug date dublura
53 -- INSERT INTO Furnizor
54 -- VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Pilos', 'pilosX@contact.com', '0700000099', 'Str. Business xx');
55
56 -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- TOO_MANY_ROWS
57
58 -- -- Elimin datele adaugate
59 -- DELETE FROM furnizor
60 -- WHERE telefon like '0700000099'
61
62 -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'Cafeina'); -- NO_AFFECTED_CLIENTS
63

```

ORA-20004: Nu exista furnizori afectati pentru produsele vandute. ORA-06512: at "SQL\_YNSWEFXACCLFGFKDBXFDPNPARB.CLIENTI\_AFECTATI\_DE\_RECHEMARE", line 37  
ORA-06512: at line 1  
ORA-06512: at "SYS.DBMS\_SQL", line 1721

*Cazul în care factorul identificator al furnizorului nu e unic:*

```

1 CREATE OR REPLACE PROCEDURE clienti_afectati_de_rechemare(furnizor_afectat furnizor.nume%type,
2                                         element_rechemat produs.descriere%type) IS
3     TYPE client_afectat IS TABLE OF VARCHAR(1000);
4
5     lista_clienti_afectati client_afectat := client_afectat(); -- Definim lista ce va fi afisata
6     id_furnizor_afectat furnizor.id_furnizor%type;
7
8     NO_AFFECTED_CLIENTS EXCEPTION; -- declar exceptia care nu e predefinita
9
10    BEGIN
11        SELECT id_furnizor INTO id_furnizor_afectat -- Verificam daca furnizorul inclus exista / este unic
12        FROM furnizor
13        WHERE UPPER(nume) LIKE UPPER(furnizor_afectat);
14
15        SELECT 'Date de contact client - email: ' || email || ', numarul de telefon: ' || telefon || ' va trebui contactat.' BULK COLLECT INTO lista_clie
16        FROM client
17        WHERE id_client IN (SELECT id_client -- preluam toate comenzi care contin produse afectate
18                           FROM comanda
19                           WHERE id_comanda IN (SELECT id_comanda
20                                     FROM continut_comanda
21                                     WHERE id_produs IN (SELECT id_produs -- preluam toate produsele care contin elementul rechemat pt ca ast
22                                           FROM produs
23                                           WHERE (UPPER(descriere) LIKE '%' || UPPER(element_rechemat) || '%')
24                                           AND id_produs IN (SELECT id_produs
25                                             FROM produs_furnizor
26                                             WHERE id_furnizor = id_furnizor_afectat)));
27
28        IF lista_clienti_afectati.COUNT = 0 THEN
29            RAISE NO_AFFECTED_CLIENTS;
30        END IF;
31
32        FOR i IN 1..lista_clienti_afectati.COUNT LOOP
33            dbms_output.put_line(lista_clienti_afectati(i));
34        END LOOP;
35
36        EXCEPTION -- definim exceptiile
37            WHEN NO_DATA_FOUND THEN
38                RAISE_APPLICATION_ERROR(-20004, 'Nu exista furnizori afectati pentru produsele vandute.');?>
39            WHEN TOO_MANY_ROWS THEN
40                RAISE_APPLICATION_ERROR(-20003, 'Furnizorul nu poate fi identificat pentru ca exista mai multi cu acelasi nume.');?>
41            WHEN NO_AFFECTED_CLIENTS THEN
42                RAISE_APPLICATION_ERROR(-20002, 'Nici un client nu a fost afectat de aceste produse!');?>
43            WHEN others THEN
44                RAISE_APPLICATION_ERROR(-20001, SQLERRM);
45    END clienti_afectati_de_rechemare;
46    /
47
48    -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- OK
49    -- EXECUTE clienti_afectati_de_rechemare('Pepsi', 'cola'); -- NO_DATA_FOUND
50
51    -- Urmatoarele comenzi (urmatoarele 7 linii) vor fi rulate impreuna
52    -- -- Adaug date dublura
53    INSERT INTO Furnizor
54    VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Pilos', 'pilosX@contact.com', '0700000099', 'Str. Business xx');
55
56    EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- TOO_MANY_ROWS
57
58    -- -- Elimin datele adaugate
59    -- DELETE FROM furnizor
60    -- WHERE telefon like '0700000099';
61
62    -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'Caseina'); -- NO_AFFECTED_CLIENTS
63

```

ORA-20003: Furnizorul nu poate fi identificat pentru ca exista mai multi cu acelasi nume. ORA-06512: at "SQL\_NNSWEFXACCLFGFKDBXFDNPARB.CLIENTI\_AFECTATI\_DE\_RECHEMARE", line 39  
ORA-06512: at line 1  
ORA-06512: at "SYS.DBMS\_SQL", line 1721

## Cazul în care găsim clienți afectați:

```

1 CREATE OR REPLACE PROCEDURE clienti_afectati_de_rechemare(furnizor_afectat furnizor.nume%type,
2                                                 element_rechemat produs.descriere%type) IS
3
4     TYPE client_afectat IS TABLE OF VARCHAR(1000);
5
6     lista_clienti_afectati client_afectat := client_afectat(); -- Definim lista ce va fi afisata
7     id_furnizor_afectat furnizor.id_furnizor%type;
8
9     NO_AFFECTED_CLIENTS EXCEPTION; -- declar exceptia care nu e predefinita
10    BEGIN
11        SELECT id_furnizor INTO id_furnizor_afectat -- Verificam daca furnizorul inclus exista / este unic
12        FROM furnizor
13        WHERE UPPER(nume) LIKE UPPER(furnizor_afectat);
14
15        SELECT 'Date de contact client - email: ' || email || ', numarul de telefon: ' || telefon || ' va trebui contactat.' BULK COLLECT INTO lista_clienti_afectati
16        FROM client
17        WHERE id_client IN (SELECT id_client -- preluam toate comenzi care contin produse afectate
18                            FROM comanda
19                            WHERE id_comanda IN (SELECT id_comanda
20                                      FROM continut_comanda
21                                      WHERE id_produs IN (SELECT id_produs -- preluam toate produsele care contin elementul rechemat pt ca asta sa fie
22                                              FROM produs
23                                              WHERE (UPPER(descriere) LIKE '%' || UPPER(element_rechemat) || '%')
24                                              AND id_produs IN (SELECT id_produs
25                                              FROM produs_furnizor
26                                              WHERE id_furnizor = id_furnizor_afectat)));
27
28        IF lista_clienti_afectati.COUNT = 0 THEN
29            RAISE NO_AFFECTED_CLIENTS;
30        END IF;
31
32        FOR i IN 1..lista_clienti_afectati.COUNT LOOP
33            dbms_output.put_line(lista_clienti_afectati(i));
34        END LOOP;
35
36        EXCEPTION -- definim exceptiile
37            WHEN NO_DATA_FOUND THEN
38                RAISE_APPLICATION_ERROR(-20004, 'Nu exista furnizori afectati pentru produsele vandute.');?>
39            WHEN TOO_MANY_ROWS THEN
40                RAISE_APPLICATION_ERROR(-20003, 'Furnizorul nu poate fi identificat pentru ca exista mai multi cu acelasi nume.');?>
41            WHEN NO_AFFECTED_CLIENTS THEN
42                RAISE_APPLICATION_ERROR(-20002, 'Nici un client nu a fost afectat de aceste produse!');-- definesc exceptia nou declarata
43            WHEN others THEN
44                RAISE_APPLICATION_ERROR(-20001, SQLERRM);
45    END clienti_afectati_de_rechemare;
46
47
48 EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- OK
49 -- EXECUTE clienti_afectati_de_rechemare('Pepsi', 'cola'); -- NO_DATA_FOUND
50
51 -- Urmatoarele comenzi (urmatoarele 7 linii) vor fi rulate impreuna
52 -- -- Adaug date dublura
53 -- INSERT INTO Furnizor
54 -- VALUES ( SEQ_FURNIZOR.NEXTVAL, 'Pilos', 'pilosX@contact.com', '0700000099', 'Str. Business xx');
55
56 -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'paine'); -- TOO_MANY_ROWS
57
58 -- -- Elimin datele adaugate
59 -- DELETE FROM furnizor
60 -- WHERE telefon like '0700000099'
61
62 -- EXECUTE clienti_afectati_de_rechemare('PILOS', 'Cafeina'); -- NO_AFFECTED_CLIENTS
63

```

Statement processed.  
 Date de contact client - email: constantin.corban@gmail.com, numarul de telefon: 0700000028 va trebui contactat.  
 Date de contact client - email: bogdan.maftei@gmail.com, numarul de telefon: 0700000030 va trebui contactat.

## Cerinta 10:

*Enunț: După ce se sterg angajați verificați dacă magazinele au cel puțin un angajat pe fiecare post (manager, logistics, casier).*

```

CREATE OR REPLACE TRIGGER verifica_angajati_departamente
    AFTER DELETE ON angajat
DECLARE
    CURSOR lista_magazine IS -- preiau lista de id-uri ale magazinelor
        SELECT id_magazin
        FROM magazin;

    numar_manageri number := 0;
    numar_logistica number := 0;
    numar_casieri number := 0;
BEGIN
    FOR elem IN lista_magazine LOOP
        SELECT COUNT(*) INTO numar_manageri -- numar cati angajati mai sunt in
        pozitia de manager la magazinul curent
        FROM angajat
        WHERE UPPER(pozitie) LIKE 'MANAGER' AND id_magazin = elem.id_magazin;

        SELECT COUNT(*) INTO numar_logistica -- numar cati angajati mai sunt in
        pozitia de logistica la magazinul curent
        FROM angajat
        WHERE UPPER(pozitie) LIKE 'LOGISTICA' AND id_magazin = elem.id_magazin;

        SELECT COUNT(*) INTO numar_casieri -- numar cati angajati mai sunt in
        pozitia de casier la magazinul curent
        FROM angajat
        WHERE UPPER(pozitie) LIKE 'CASIER' AND id_magazin = elem.id_magazin;
    END LOOP;

    IF numar_manageri = 0 OR numar_logistica = 0 OR numar_casieri = 0 THEN
        RAISE_APPLICATION_ERROR('-20008', 'NU putem avea deparamente fara nici
        un angajat. Fiecare magazin trebuie sa ramana cu cel putin un Manager, un angajat in
        logistica si un casier.');
    END IF;
END;
/
-- Rulam comanda si nu ne permite sa stergem pentru ca nu exista alt manager la
magazinul cu id-ul 6
-- DELETE FROM angajat
-- WHERE id_angajat = 18;

-- adaugam noi angajati pentru a avea cel putin 2 angajati pe pozitiile din care
vrem sa stergem
-- insert into Angajat
-- values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 1', 'Casier', 'test1@gmail.com',
'0700001001', 'Str Albu 1, b1');
-- insert into Angajat
-- values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 2', 'Manager', 'test2@gmail.com',
'0700001002', 'Str Albu 2, b1');
-- insert into Angajat

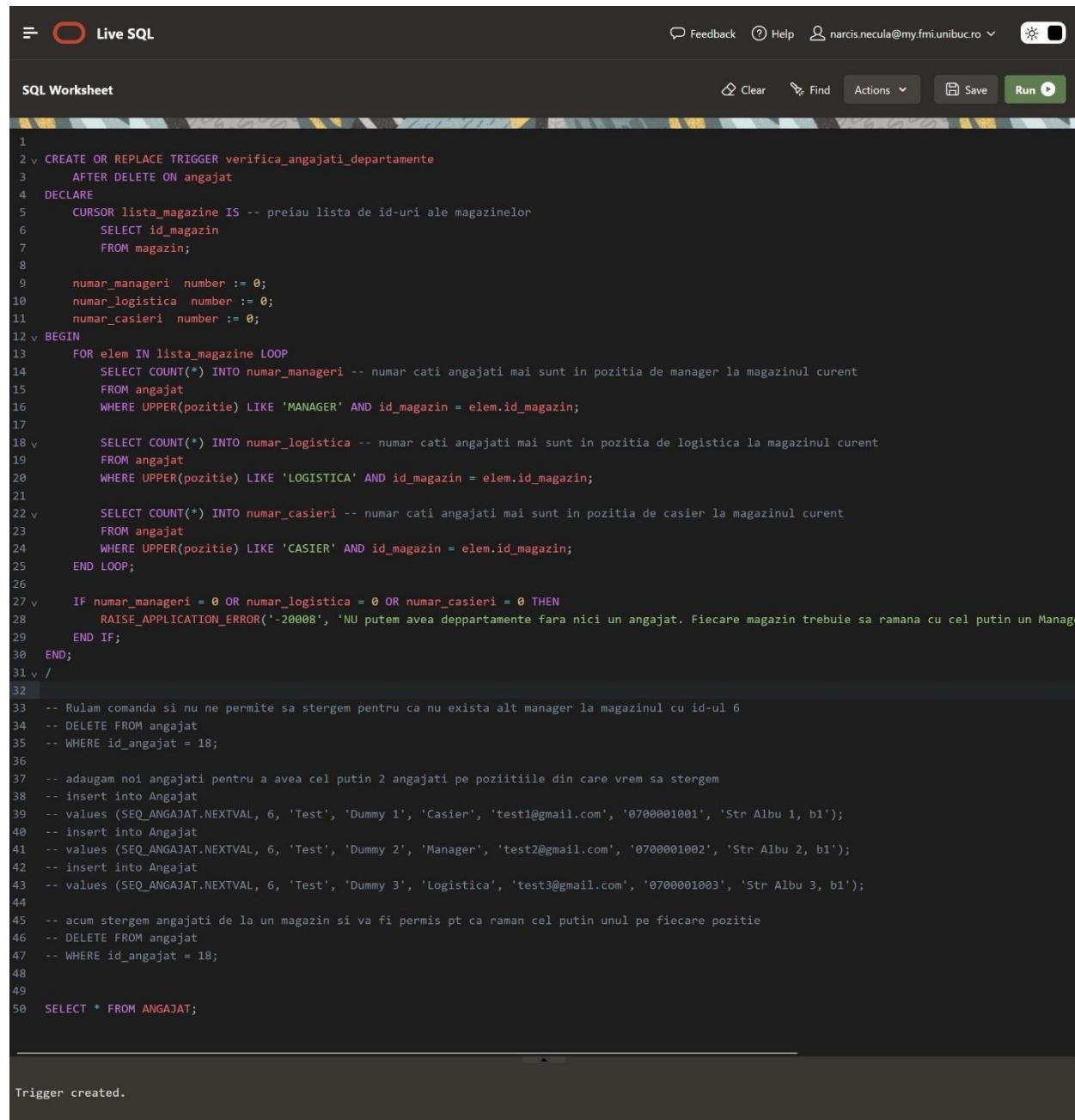
```

```
-- values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 3', 'Logistica', 'test3@gmail.com',
'0700001003', 'Str Albu 3, b1');

-- acum stergem angajati de la un magazin si va fi permis pt ca raman cel putin unul
pe fiecare pozitie
-- DELETE FROM angajat
-- WHERE id_angajat = 18;

SELECT * FROM ANGAJAT;
```

### *Crearea triggerului:*



The screenshot shows a SQL worksheet interface with the following details:

- Header:** Live SQL, Feedback, Help, Email (narcis.necula@my.fmi.unibuc.ro), and a brightness slider.
- Toolbar:** Clear, Find, Actions (dropdown), Save, and Run.
- Code Area:**

```

1 CREATE OR REPLACE TRIGGER verifica_angajati_departamente
2      AFTER DELETE ON angajat
3      DECLARE
4          CURSOR lista_magazine IS -- preiau lista de id-uri ale magazinelor
5              SELECT id_magazin
6                  FROM magazin;
7
8          numar_manageri number := 0;
9          numar_logistica number := 0;
10         numar_casieri number := 0;
11
12        BEGIN
13            FOR elem IN lista_magazine LOOP
14                SELECT COUNT(*) INTO numar_manageri -- numar cati angajati mai sunt in pozitia de manager la magazinul curent
15                FROM angajat
16                WHERE UPPER(pozitie) LIKE 'MANAGER' AND id_magazin = elem.id_magazin;
17
18            SELECT COUNT(*) INTO numar_logistica -- numar cati angajati mai sunt in pozitia de logistica la magazinul curent
19            FROM angajat
20            WHERE UPPER(pozitie) LIKE 'LOGISTICA' AND id_magazin = elem.id_magazin;
21
22            SELECT COUNT(*) INTO numar_casieri -- numar cati angajati mai sunt in pozitia de casier la magazinul curent
23            FROM angajat
24            WHERE UPPER(pozitie) LIKE 'CASIER' AND id_magazin = elem.id_magazin;
25        END LOOP;
26
27        IF numar_manageri = 0 OR numar_logistica = 0 OR numar_casieri = 0 THEN
28            RAISE_APPLICATION_ERROR('-20008', 'NU putem avea departamente fara nici un angajat. Fiecare magazin trebuie sa ramana cu cel putin un Manager');
29        END IF;
30    END;
31    /
32
33    -- Rulam comanda si nu ne permite sa stergem pentru ca nu exista alt manager la magazinul cu id-ul 6
34    -- DELETE FROM angajat
35    -- WHERE id_angajat = 18;
36
37    -- adaugam noi angajati pentru a avea cel putin 2 angajati pe pozitiile din care vrem sa stergem
38    -- insert into Angajat
39    -- values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 1', 'Casier', 'test1@gmail.com', '0700001001', 'Str Albu 1, b1');
40    -- insert into Angajat
41    -- values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 2', 'Manager', 'test2@gmail.com', '0700001002', 'Str Albu 2, b1');
42    -- insert into Angajat
43    -- values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 3', 'Logistica', 'test3@gmail.com', '0700001003', 'Str Albu 3, b1');
44
45    -- acum stergem angajati de la un magazin si va fi permis pt ca raman cel putin unul pe fiecare pozitie
46    -- DELETE FROM angajat
47    -- WHERE id_angajat = 18;
48
49
50    SELECT * FROM ANGAJAT;
```
- Status Bar:** Trigger created.

*Triggerul se activează și nu permite:*

The screenshot shows a SQL worksheet interface with the following details:

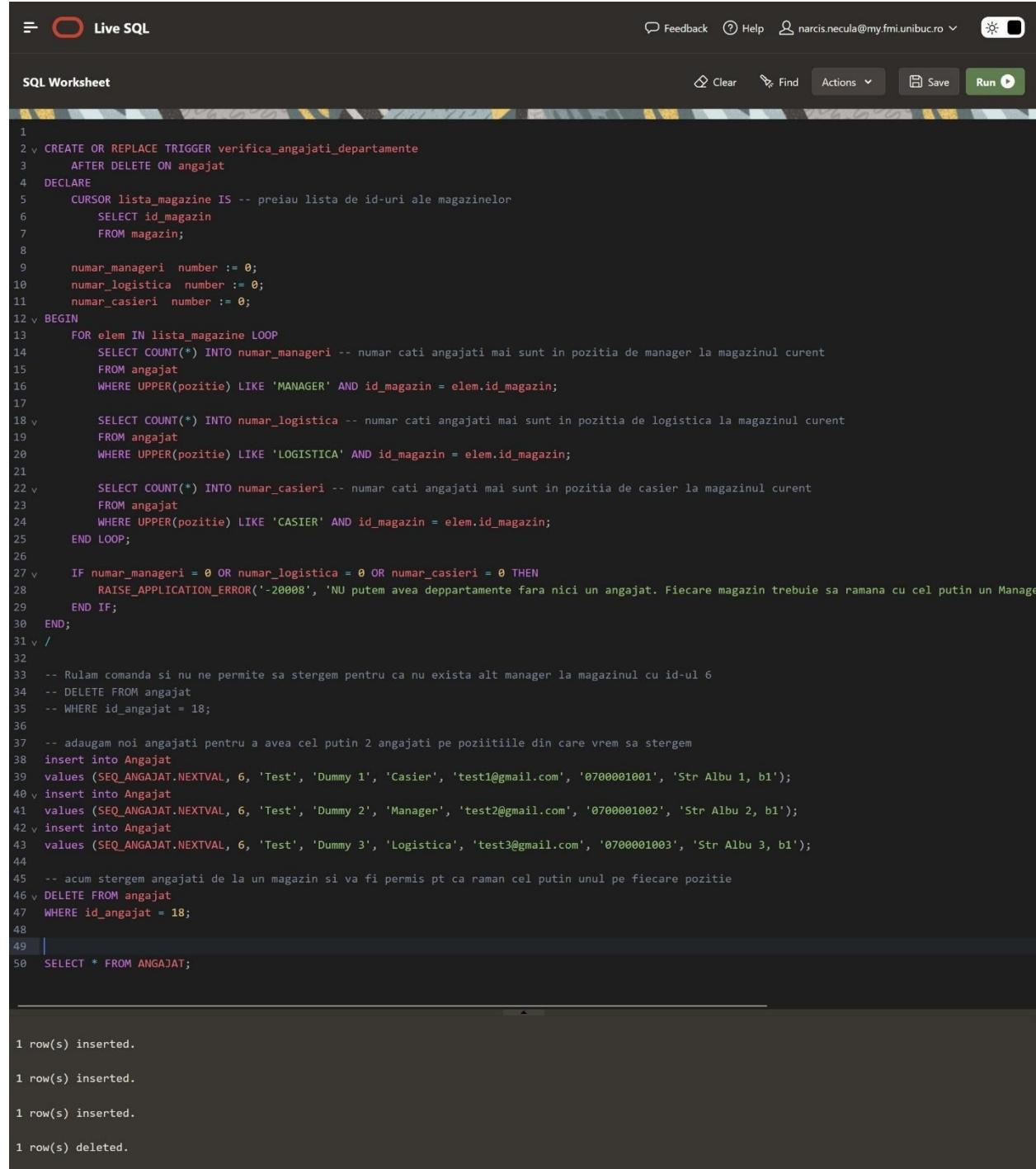
- Header:** Live SQL, Feedback, Help, Email (narcis.necula@my.fmi.unibuc.ro), Light/Dark mode switch.
- Toolbar:** Clear, Find, Actions (dropdown), Save, Run.
- SQL Worksheet Content:**

```

1  CREATE OR REPLACE TRIGGER verifica_angajati_departamente
2      AFTER DELETE ON angajat
3  DECLARE
4      CURSOR lista_magazine IS -- preiau lista de id-uri ale magazinelor
5          SELECT id_magazin
6              FROM magazin;
7
8      numar_manageri number := 0;
9      numar_logistica number := 0;
10     numar_casieri number := 0;
11
12 BEGIN
13     FOR elem IN lista_magazine LOOP
14         SELECT COUNT(*) INTO numar_manageri -- numar cati angajati mai sunt in pozitia de manager la magazinul curent
15         FROM angajat
16         WHERE UPPER(pozitie) LIKE 'MANAGER' AND id_magazin = elem.id_magazin;
17
18         SELECT COUNT(*) INTO numar_logistica -- numar cati angajati mai sunt in pozitia de logistica la magazinul curent
19         FROM angajat
20         WHERE UPPER(pozitie) LIKE 'LOGISTICA' AND id_magazin = elem.id_magazin;
21
22         SELECT COUNT(*) INTO numar_casieri -- numar cati angajati mai sunt in pozitia de casier la magazinul curent
23         FROM angajat
24         WHERE UPPER(pozitie) LIKE 'CASIER' AND id_magazin = elem.id_magazin;
25     END LOOP;
26
27     IF numar_manageri = 0 OR numar_logistica = 0 OR numar_casieri = 0 THEN
28         RAISE_APPLICATION_ERROR('-20008', 'NU putem avea deparamente fara nici un angajat. Fiecare magazin trebuie sa ramana cu cel putin un Manager');
29     END IF;
30 END;
31 /
32
33 -- Rulam comanda si nu ne permite sa stergem pentru ca nu exista alt manager la magazinul cu id-ul 6
34 DELETE FROM angajat
35 WHERE id_angajat = 18;
36
37 -- adaugam noi angajati pentru a avea cel putin 2 angajati pe pozitiile din care vrem sa stergem
38 -- insert into Angajat
39 -- values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 1', 'Casier', 'test1@gmail.com', '0700001001', 'Str Albu 1, b1');
40 -- insert into Angajat
41 -- values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 2', 'Manager', 'test2@gmail.com', '0700001002', 'Str Albu 2, b1');
42 -- insert into Angajat
43 -- values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 3', 'Logistica', 'test3@gmail.com', '0700001003', 'Str Albu 3, b1');
44
45 -- acum stergem angajati de la un magazin si va fi permis pt ca raman cel putin unul pe fiecare pozitie
46 -- DELETE FROM angajat
47 -- WHERE id_angajat = 18;
48
49 |
50 SELECT * FROM ANGAJAT;
```
- Output Area:**

ORA-20008: NU putem avea deparamente fara nici un angajat. Fiecare magazin trebuie sa ramana cu cel putin un Manager, un angajat in logistica si un casier. ORA-06512: at "SQL\_YNSWEFXACCLFGFKDBXFDNPB8.VERIFICA\_ANGAJATI\_DEPARTAMENTE", line 25  
ORA-06512: at "SYS.DBMS\_SQL", line 1721

*Triggerul permite efectuarea stergerii:*



The screenshot shows a SQL worksheet interface with the following details:

- Header:** Live SQL, Feedback, Help, Email (narcis.necula@my.fmi.unibuc.ro), and a brightness slider.
- Toolbar:** Clear, Find, Actions (dropdown), Save, and Run.
- SQL Worksheet Content:**

```

1 CREATE OR REPLACE TRIGGER verifica_angajati_departamente
2      AFTER DELETE ON angajat
3      DECLARE
4          CURSOR lista_magazine IS -- preiau lista de id-uri ale magazinelor
5              SELECT id_magazin
6                  FROM magazin;
7
8          numar_manageri number := 0;
9          numar_logistica number := 0;
10         numar_casieri number := 0;
11
12    BEGIN
13        FOR elem IN lista_magazine LOOP
14            SELECT COUNT(*) INTO numar_manageri -- numar cati angajati mai sunt in pozitia de manager la magazinul curent
15            FROM angajat
16            WHERE UPPER(pozitie) LIKE 'MANAGER' AND id_magazin = elem.id_magazin;
17
18        SELECT COUNT(*) INTO numar_logistica -- numar cati angajati mai sunt in pozitia de logistica la magazinul curent
19        FROM angajat
20        WHERE UPPER(pozitie) LIKE 'LOGISTICA' AND id_magazin = elem.id_magazin;
21
22        SELECT COUNT(*) INTO numar_casieri -- numar cati angajati mai sunt in pozitia de casier la magazinul curent
23        FROM angajat
24        WHERE UPPER(pozitie) LIKE 'CASIER' AND id_magazin = elem.id_magazin;
25    END LOOP;
26
27    IF numar_manageri = 0 OR numar_logistica = 0 OR numar_casieri = 0 THEN
28        RAISE_APPLICATION_ERROR('-20008', 'NU putem avea deppartamente fara nici un angajat. Fiecare magazin trebuie sa ramana cu cel putin un Manager');
29    END IF;
30 END;
31 /
32
33 -- Rulam comanda si nu ne permite sa stergem pentru ca nu exista alt manager la magazinul cu id-ul 6
34 -- DELETE FROM angajat
35 -- WHERE id_angajat = 18;
36
37 -- adaugam noi angajati pentru a avea cel putin 2 angajati pe pozitiile din care vrem sa stergem
38 insert into Angajat
39 values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 1', 'Casier', 'test1@gmail.com', '0700001001', 'Str Albu 1, b1');
40
41 insert into Angajat
42 values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 2', 'Manager', 'test2@gmail.com', '0700001002', 'Str Albu 2, b1');
43
44 insert into Angajat
45 values (SEQ_ANGAJAT.NEXTVAL, 6, 'Test', 'Dummy 3', 'Logistica', 'test3@gmail.com', '0700001003', 'Str Albu 3, b1');
46
47 -- acum stergem angajati de la un magazin si va fi permis pt ca raman cel putin unul pe fiecare pozitie
48 DELETE FROM angajat
49 WHERE id_angajat = 18;
50
51
52 SELECT * FROM ANGAJAT;

```
- Execution Results:**

```

1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) deleted.

```

## Cerinta 11:

*Enunț: LMD la nivel de linie , nu permite inserarea unei promoții dacă există una activă, dacă promoția există cu aceeași valoare o extindem, dacă există o promoție dar valoarea acesteia e mai mică se mărește la noua valoare, dacă există și noua valoare e mai mică se păstrează valoarea mai mare.*

```

CREATE OR REPLACE PROCEDURE actualizare_data_promotiei(noua_data_sfarsit
promotie.data_sfarsit%TYPE, id_promotie_de_actualizat promotie.id_promotie%TYPE)
AS PRAGMA AUTONOMOUS_TRANSACTION; -- ca sa putem modifica datele in trigger.
BEGIN
    UPDATE promotie
    SET data_sfarsit = noua_data_sfarsit
    WHERE id_promotie = id_promotie_de_actualizat;
    COMMIT;
END;
/
CREATE OR REPLACE PROCEDURE sterge_promotie(id_promotie_de_sters
promotie.id_promotie%TYPE) AS
BEGIN
    DELETE FROM promotie
    WHERE id_promotie = id_promotie_de_sters;
END;
/
CREATE OR REPLACE TRIGGER promotie_activa
    BEFORE INSERT ON promotie
    FOR EACH ROW
DECLARE
    discount_existent promotie.procent_discount%type;
    exista_promotie_activa number := 0;
    id_promotie_existentă promotie.id_promotie%type;
BEGIN
    SELECT COUNT(*) INTO exista_promotie_activa -- cautam promociile active
    FROM promotie
    WHERE data_sfarsit >= SYSDATE AND :NEW.id_produs = id_produs;

    IF exista_promotie_activa >= 1 THEN
        SELECT procent_discount, id_promotie INTO discount_existent,
        id_promotie_existentă -- preluam date despre acestea
        FROM promotie
        WHERE data_sfarsit >= SYSDATE AND id_produs = :NEW.id_produs;

        IF :NEW.procent_discount < discount_existent THEN
            RAISE_APPLICATION_ERROR(-20005, 'Exista promotie activa mai mare.
Promotia activa este de ' || discount_existent);

            ELSIF :NEW.procent_discount = discount_existent THEN
                actualizare_data_promotiei(:NEW.data_sfarsit, id_promotie_existentă);
                RAISE_APPLICATION_ERROR(-20006, 'Exista promotie activa cu acelsi
discount. Aceasta a fost actualizata cu noua data de sfarsit.');

            ELSIF :NEW.procent_discount > discount_existent THEN
                sterge_promotie(id_promotie_existentă);
        END IF;
    END IF;
END;
/

```

```

        dbms_output.put_line('0 promotie mai mica a fost stearsa si cea
noua i-a luat locul.');
    END IF;
END IF;
/
-- insereaza fara declansarea unei erori pentru ca nu exista promotii active
-- INSERT INTO Promotie
-- VALUES ( SEQ_PROMOTIE.NEXTVAL, 10, SYSDATE, TO_DATE('01-06-2023','dd-mm-yyyy'),
10.00);
-- Se actualizeaza data de sfarsit
INSERT INTO Promotie
VALUES ( SEQ_PROMOTIE.NEXTVAL, 10, SYSDATE, TO_DATE('05-06-2023','dd-mm-yyyy'),
10.00);

select * from promotie;

```

*Crearea triggerului:*

```

12 CREATE OR REPLACE PROCEDURE sterge_promotie(id_promotie_de_sters promotie.id_promotie%TYPE) AS
13 BEGIN
14     DELETE FROM promotie
15     WHERE id_promotie = id_promotie_de_sters;
16 END;
17 /
18
19 CREATE OR REPLACE TRIGGER promotie_activa
20     BEFORE INSERT ON promotie
21     FOR EACH ROW
22 DECLARE
23     discount_existent promotie.procent_discount%type;
24     exista_promotie_activa number := 0;
25     id_promotie_existenta promotie.id_promotie%type;
26 BEGIN
27     SELECT COUNT(*) INTO exista_promotie_activa -- cautam promotiile active
28     FROM promotie
29     WHERE data_sfarsit >= SYSDATE AND :NEW.id_produs = id_produs;
30
31 IF exista_promotie_activa >= 1 THEN
32     SELECT procent_discount, id_promotie INTO discount_existent, id_promotie_existenta -- preluam date despre acestea
33     FROM promotie
34     WHERE data_sfarsit >= SYSDATE AND id_produs = :NEW.id_produs;
35
36 IF :NEW.procent_discount < discount_existent THEN
37     RAISE_APPLICATION_ERROR(-20005, 'Exista promotie activa mai mare. Promotia activa este de ' || discount_existent);
38
39 ELSIF :NEW.procent_discount = discount_existent THEN
40     actualizare_data_promotiei(:NEW.data_sfarsit, id_promotie_existenta);
41     RAISE_APPLICATION_ERROR(-20006, 'Exista promotie activa cu acelsi discount. Aceasta a fost actualizata cu noua data de sfarsit.');
42
43 ELSIF :NEW.procent_discount > discount_existent THEN
44     sterge_promotie(id_promotie_existenta);
45     dbms_output.put_line('0 promotie mai mica a fost stearsa si cea noua i-a luat locul.');
46 END IF;
47 END IF;
48 END;
49 /
50
51 -- insereaza fara declansarea unei erori pentru ca nu exista promotii active
52 -- INSERT INTO Promotie
53 -- VALUES ( SEQ_PROMOTIE.NEXTVAL, 10, SYSDATE, TO_DATE('01-06-2023','dd-mm-yyyy'), 10.00);
54
55 -- Se actualizeaza data de sfarsit
56 INSERT INTO Promotie
57 VALUES ( SEQ_PROMOTIE.NEXTVAL, 10, SYSDATE, TO_DATE('05-06-2023','dd-mm-yyyy'), 10.00);

58
59 select * from promotie;

```

Procedure created.

Procedure created.

Trigger created.

## Rularea triggerului și activarea:

The screenshot shows a SQL worksheet interface with the following details:

- Header:** Live SQL, Feedback, Help, Email (narcis.necula@my.fmi.unibuc.ro), Actions, Save, Run.
- SQL Worksheet Content:**

```

1 CREATE OR REPLACE PROCEDURE actualizare_data_promotiei(noua_data_sfarsit promotie.data_sfarsit%TYPE, id_promotie_de_actualizat promotie.id_promotie%TYPE)
2 AS PRAGMA AUTONOMOUS_TRANSACTION; -- ca sa putem modifica datele in trigger.
3 BEGIN
4     UPDATE promotie
5         SET data_sfarsit = noua_data_sfarsit
6         WHERE id_promotie = id_promotie_de_actualizat;
7     COMMIT;
8 END;
9 /
10
11 CREATE OR REPLACE PROCEDURE sterge_promotie(id_promotie_de_sters promotie.id_promotie%TYPE) AS
12 BEGIN
13     DELETE FROM promotie
14     WHERE id_promotie = id_promotie_de_sters;
15 END;
16 /
17
18 CREATE OR REPLACE TRIGGER promotie_activa
19     BEFORE INSERT ON promotie
20     FOR EACH ROW
21     DECLARE
22         discount_existent promotie.procent_discount%type;
23         exista_promotie_activa number := 0;
24         id_promotie_existenta promotie.id_promotie%type;
25     BEGIN
26         SELECT COUNT(*) INTO exista_promotie_activa -- cautam promotiile active
27             FROM promotie
28             WHERE data_sfarsit >= SYSDATE AND :NEW.id_produs = id_produs;
29
30         IF exista_promotie_activa >= 1 THEN
31             SELECT procent_discount, id_promotie INTO discount_existent, id_promotie_existenta -- preluam date despre acestea
32                 FROM promotie
33                 WHERE data_sfarsit >= SYSDATE AND id_produs = :NEW.id_produs;
34
35         IF :NEW.procent_discount < discount_existent THEN
36             RAISE_APPLICATION_ERROR(-20005, 'Există o promovare activă mai mare. Promovarea activă este de ' || discount_existent);
37
38         ELSIF :NEW.procent_discount = discount_existent THEN
39             actualizare_data_promotiei(:NEW.data_sfarsit, id_promotie_existenta);
40             RAISE_APPLICATION_ERROR(-20006, 'Există o promovare activă cu același discount. Aceasta a fost actualizată cu noua data de sfarsit.');
41
42         ELSIF :NEW.procent_discount > discount_existent THEN
43             sterge_promotie(id_promotie_existenta);
44             dbms_output.put_line('O promovare mai mică a fost stărișă și cea nouă i-a luat locul.');
45         END IF;
46     END IF;
47     END IF;
48 END;
49 /
50
51 -- inserează fără declansarea unei erori pentru că nu există promovare active
52 -- INSERT INTO Promovare
53 -- VALUES ( SEQ_PROMOTIE.NEXTVAL, 10, SYSDATE, TO_DATE('01-06-2023', 'dd-mm-yyyy'), 10.00);
54
55 -- Se actualizează data de sfarsit
56 INSERT INTO Promovare
57 VALUES ( SEQ_PROMOTIE.NEXTVAL, 10, SYSDATE, TO_DATE('05-06-2023', 'dd-mm-yyyy'), 10.00);
58
59 select * from promotie;

```
- Output:**

```

ORA-20006: Există o promovare activă cu același discount. Aceasta a fost actualizată cu noua data de sfarsit. ORA-06512: at
"SQL_YNSWEFXACCLFGFKDBXFDNPARB.PROMOTIE_ACTIVĂ", line 20
ORA-06512: at "SYS.DBMS_SQL", line 1721

```

## Cerinta 12:

*Enunț: Creez un tabel de log-uri în care să rețin dacă s-au făcut modificari asupra structurii, cine le-a făcut și dacă au reușit sau nu.*

*Restrângerile sunt că nu se pot face modificări asupra structurii bazei de date fără să fi admin, și nici în intervalul de operare a magazinelor (L-V: 07-23, S: 08-23, D: 09-20).*

```

CREATE TABLE log_modificari (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
);

-- Creez un tabel pentru loguri sa stim ce erori au aparut si ce modificari facurte
de cine au fost realizate asupra schemei bazei de date.
CREATE OR REPLACE PROCEDURE adaugare_log(mesaj varchar2) IS
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
    INSERT INTO log_modificari
        VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME,
        SYSDATE, mesaj);
    commit;
END;
/

CREATE OR REPLACE TRIGGER limitare_acces_ore_lucratoare
    BEFORE CREATE OR ALTER OR DROP ON SCHEMA
BEGIN

    IF SYS.LOGIN_USER <> 'NARCIS1' THEN -- numele admin, numai el are permisiunea de
    modificare a structurii
        adaugare_log('Administratorul este singurul care poate modifica structura
        bazei de date.');
        RAISE_APPLICATION_ERROR(-20009, 'Nu aveti permisiunile necesare pentru
        aceasta actiune.');
    ELSE
        -- verific programul fiecarei zile sa nu fie suprapus cu ora si ziua curenta
        IF (((TO_CHAR(SYSDATE, 'D') BETWEEN 2 AND 6) AND (TO_CHAR(SYSDATE, 'HH24')
        BETWEEN 7 AND 23))
            OR ((TO_CHAR(SYSDATE, 'D') = 7) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 8 AND
        23))
            OR ((TO_CHAR(SYSDATE, 'D') =1) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 9 AND
        20))) THEN
            adaugare_log('Nu se pot face modificari in timpul programului de
            operare.');
            RAISE_APPLICATION_ERROR(-20010, 'Nu se pot face modificari in timpul
            programului de operare.');
        END IF;
    END IF;

```

```
adaugare_log('Baza de date a fost modificata.');
dbms_output.put_line('Modificari realizate!');
END;
/
select * from log_modificari;

-- pentru testarea tuturor erorilor am incercat sa adaug acest tabel, am modificat
paraetrii putin pentru obtinerea tuturor erorilor posibile din ss

CREATE TABLE dummy (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
);
```

*(Nu am rulat în SQL-Live pentru că aveam probleme cu SYS valori și am rulat într-o bază de date locală la un coleg, de asta diferă screen-shot-urile)*

*Am rulat cu ADMIN fiind singurul user cu permisiuni, deci nu am drepturi*

```

Worksheet  Query Builder
CREATE TABLE log_modificari (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
);

-- Creez un tabel pentru loguri sa stim ce erori au aparut si ce modificari facute de cine au fost
CREATE OR REPLACE PROCEDURE adaugare_log(mesaj varchar2) IS
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
    INSERT INTO log_modificari
    VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE, mesaj);
    commit;
END;
/


CREATE OR REPLACE TRIGGER limitare_acces_ore_lucratoare
    BEFORE CREATE OR ALTER OR DROP ON SCHEMA
BEGIN

    IF SYS.LOGIN_USER <> 'ADMIN' THEN
        adaugare_log('Administratorul este singurul care poate modifica structura bazei de date.');
        RAISE_APPLICATION_ERROR(-20009, 'Nu aveti permisiunile necesare pentru aceasta actiune.');
    ELSE
        IF ((TO_CHAR(SYSDATE, 'D') BETWEEN 2 AND 6) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 7 AND 23))
        OR ((TO_CHAR(SYSDATE, 'D') = 7) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 8 AND 23))
        OR ((TO_CHAR(SYSDATE, 'D') =1) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 9 AND 20)) THEN
            adaugare_log('Nu se pot face modificarile in timpul programului de operare.');
            RAISE_APPLICATION_ERROR(-20010, 'Nu se pot face modificarile in timpul programului de operare.');
        END IF;
    END IF;

    adaugare_log('Baza de date a fost modifitata.');
    dbms_output.put_line('Modificari realizate!');
END;
/


select * from log_modificari;

CREATE TABLE dummy (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
);

```

Script Output | Query Result | Task completed in 0.06 seconds

Error starting at line : 43 in command -

```

CREATE TABLE dummy (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
)
Error report -
ORA-00604: error occurred at recursive SQL level 1
ORA-20009: Nu aveti permisiunile necesare pentru aceasta actiune.
ORA-06512: at line 5
00604. 00000 - "error occurred at recursive SQL level %s"
*Cause: An error occurred while processing a recursive SQL statement
(a statement applying to internal dictionary tables).
*Action: If the situation described in the next error on the stack
can be corrected, do so; otherwise contact Oracle Support.

```

Am făcut modificări la ora 18:05 care este în timpul programului de operare

The screenshot shows the Oracle SQL Developer interface. The main area is a 'Worksheet' tab where a complex PL/SQL script is being written. The script includes several CREATE statements for tables and a trigger, as well as a PROCEDURE for logging changes. A specific line of code in the trigger section is highlighted in yellow. Below the worksheet, the 'Script Output' tab is open, displaying error messages related to the execution of the script.

```

CREATE TABLE log_modificari (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
);

-- Creez un tabel pentru loguri sa stim ce erori au aparut si ce modificari facute de cine au fost
CREATE OR REPLACE PROCEDURE adaugare_log(mesaj varchar2) IS
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
    INSERT INTO log_modificari
    VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE, mesaj);
    commit;
END;
/

CREATE OR REPLACE TRIGGER limitare_acces_ore_lucratoare
    BEFORE CREATE OR ALTER OR DROP ON SCHEMA
BEGIN

    IF SYS.LOGIN_USER <> 'NARCIS1' THEN
        adaugare_log('Administratorul este singurul care poate modifica structura bazei de date.');
        RAISE_APPLICATION_ERROR(-20009, 'Nu aveti permisiunile necesare pentru aceasta actiune.');
    ELSE
        IF (((TO_CHAR(SYSDATE, 'D') BETWEEN 2 AND 6) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 7 AND 23)
        OR ((TO_CHAR(SYSDATE, 'D') = 7) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 8 AND 23))
        OR ((TO_CHAR(SYSDATE, 'D') = 1) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 9 AND 20))) THEN
            adaugare_log('Nu se pot face modificarile in timpul programului de operare.');
            RAISE_APPLICATION_ERROR(-20010, 'Nu se pot face modificarile in timpul programului de operare');
        END IF;
    END IF;

    adaugare_log('Baza de date a fost modificata.');
    dbms_output.put_line('Modificari realizate!');
END;
/

select * from log_modificari;

CREATE TABLE dummy (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
);

```

Script Output x Query Result x

Error starting at line : 43 in command -

```

CREATE TABLE dummy (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
)

```

Error report -

ORA-00604: error occurred at recursive SQL level 1  
ORA-20010: Nu se pot face modificarile in timpul programului de operare.  
ORA-06512: at line 11  
00604. 00000 - "error occurred at recursive SQL level %s"  
\*Cause: An error occurred while processing a recursive SQL statement  
(a statement applying to internal dictionary tables).  
\*Action: If the situation described in the next error on the stack  
can be corrected, do so; otherwise contact Oracle Support.

Am modificat ora de operare să pot să fac modificări asupra bazei de date:

```

Worksheet Query Builder
CREATE TABLE log_modificari (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
);

-- Creez un tabel pentru loguri sa stim ce erori au aparut si ce modificari facute de cine au fost
CREATE OR REPLACE PROCEDURE adaugare_log(mesaj varchar2) IS
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
    INSERT INTO log_modificari
    VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE, mesaj);
    commit;
END;
/

CREATE OR REPLACE TRIGGER limitare_acces_ore_lucratoare
    BEFORE CREATE OR ALTER OR DROP ON SCHEMA
BEGIN

    IF SYS.LOGIN_USER <> 'NARCIS1' THEN
        adaugare_log('Administratorul este singurul care poate modifica structura bazei de date.');
        RAISE_APPLICATION_ERROR(-20009, 'Nu aveti permisiunile necesare pentru aceasta actiune.');
    ELSE
        IF ((TO_CHAR(SYSDATE, 'D') BETWEEN 2 AND 6) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 7 AND 10))
        OR ((TO_CHAR(SYSDATE, 'D') = 7) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 8 AND 23))
        OR ((TO_CHAR(SYSDATE, 'D') = 1) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 9 AND 20))) THEN
            adaugare_log('Nu se pot face modificari in timpul programului de operare.');
            RAISE_APPLICATION_ERROR(-20010, 'Nu se pot face modificari in timpul programului de operare');
        END IF;
    END IF;

    adaugare_log('Baza de date a fost modificata.');
    dbms_output.put_line('Modificari realizate!');
END;
/

select * from log_modificari;

CREATE TABLE dummy (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
);

```

Script Output X | Query Result X | Task completed in 0.046 seconds

Table DUMMY created.

Dovada intrărilor în log-uri:

```

Worksheet Query Builder
CREATE TABLE log_modificari (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
);

-- Creez un tabel pentru loguri sa stim ce erori au aparut si ce modificari facute de cine au fost
CREATE OR REPLACE PROCEDURE adaugare_log(mesaj varchar2) IS
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
    INSERT INTO log_modificari
    VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE, mesaj);
    commit;
END;
/

CREATE OR REPLACE TRIGGER limitare_acces_ore_lucratoare
    BEFORE CREATE OR ALTER OR DROP ON SCHEMA
BEGIN

    IF SYS.LOGIN_USER <> 'NARCIS1' THEN
        adaugare_log('Administratorul este singurul care poate modifica structura bazei de date.');
        RAISE_APPLICATION_ERROR(-20009, 'Nu aveti permisiunile necesare pentru aceasta actiune.');
    ELSE
        IF (((TO_CHAR(SYSDATE, 'D') BETWEEN 2 AND 6) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 7 AND 10))
            OR ((TO_CHAR(SYSDATE, 'D') = 7) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 8 AND 23))
            OR ((TO_CHAR(SYSDATE, 'D') =1) AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 9 AND 20))) THEN
            adaugare_log('Nu se pot face modificarile in timpul programului de operare.');
            RAISE_APPLICATION_ERROR(-20010, 'Nu se pot face modificarile in timpul programului de operare');
        END IF;
    END IF;

    adaugare_log("Baza de date a fost modificata.");
    dbms_output.put_line('Modificari realizate!');
END;
/

select * from log_modificari;

CREATE TABLE dummy (
    nume_utilizator varchar2(100),
    nume_bd varchar2(100),
    eveniment varchar2(100),
    nume_object varchar2(100),
    data DATE,
    mesaj varchar2(100)
);

```

Script Output | Query Result | All Rows Fetched: 5 in 0.002 seconds

NUME_UTILIZATOR	NUME_BD	EVENIMENT	NUME_OBJECT	DATA	MESAJ
1 NARCIS1	XE	CREATE	ADAUGARE LOG	23-MAY-23	Nu se pot face modificarile in timpul programului de operare.
2 NARCIS1	XE	CREATE	DUMMY	23-MAY-23	Nu se pot face modificarile in timpul programului de operare.
3 NARCIS1	XE	CREATE	DUMMY	23-MAY-23	Administratorul este singurul care poate modifica structura bazei de date.
4 NARCIS1	XE	CREATE	DUMMY	23-MAY-23	Nu se pot face modificarile in timpul programului de operare.
5 NARCIS1	XE	CREATE	DUMMY	23-MAY-23	Baza de date a fost modificata.

### Cerinta 13:

Enunț: Cream un packet care să rulez toate celelalte exerciții (fără 4 și 5).

```

CREATE OR REPLACE TYPE tabel_de_returnat_mariri IS OBJECT ( nume varchar(100),
                                                               salariu_vechi number(10),
                                                               salariu_nou number(10),
                                                               pozitie varchar(100));
/
CREATE OR REPLACE TYPE tabel_mariri IS TABLE OF tabel_de_returnat_mariri;
/

CREATE OR REPLACE PACKAGE supermarket_management IS
    -----
    PROCEDURE cel_mai_bine_vandut_produs;

    -----
    PROCEDURE calcularea_magazinului_cu_plata_cea_mai_buna;

    -----
    FUNCTION angajati_acordare_marire ( nume_magazin magazin.nume%type,
                                         procent_de_marire number)
        RETURN tabel_mariri;

    -----
    PROCEDURE clienti_afectati_de_rechemare ( furnizor_afectat furnizor.nume%type,
                                                element_rechemat
                                                produs.descriere%type);

END supermarket_management;
/


CREATE OR REPLACE PACKAGE BODY supermarket_management IS
    -----
    PROCEDURE cel_mai_bine_vandut_produs
    IS
        TYPE tabel_indexat IS TABLE OF continut_comanda.cantitate%type INDEX BY
        PLS_INTEGER;
        TYPE tabel_imbricat IS TABLE OF continut_comanda.id_produs%type;

        nr_unitati_vandute tabel_indexat; -- numarul de unitati vandute din fiecare
        produs_vandut
        produse_vandute tabel_imbricat := tabel_imbricat(); --Idurile produselor
        vandute

        cantitate_maxima number := 0;
        i number := 0; -- indice de parcurgere
        nume_produs produs.nume%type;
    BEGIN
        FOR vanzare IN ( SELECT id_produs, cantitate
                           FROM continut_comanda) LOOP
            i := 1;

```

```

        WHILE i <= produse_vandute.COUNT AND produse_vandute(i) != vanzare.id_produs LOOP -- Cautam daca exista deja inregistrare de vanzare a acestui produs
            i := i + 1;
        END LOOP;

        IF i = produse_vandute.COUNT + 1 THEN -- In caz ca nu am gasit produsul deja adaugat il adaugam in lista de produse vandute
            produse_vandute.EXTEND;
            produse_vandute(produse_vandute.COUNT) := vanzare.id_produs;
            nr_unitati_vandute(produse_vandute.COUNT) := vanzare.cantitate;

            IF nr_unitati_vandute(produse_vandute.COUNT) > cantitate_maxima THEN -- Calculam cantitatea maxima in timp real pt eficienta
                cantitate_maxima :=
                nr_unitati_vandute(produse_vandute.COUNT);
            END IF;
            ELSE -- in cazul unui produs deja inregistrat ca vandut doar adaugam cantitatea noua la cea existenta
                nr_unitati_vandute(i) := nr_unitati_vandute(i) + vanzare.cantitate;

            IF nr_unitati_vandute(i) > cantitate_maxima THEN-- Calculam cantitatea maxima in timp real pt eficienta
                cantitate_maxima := nr_unitati_vandute(i);
            END IF;
            END IF;
        END LOOP;

        IF cantitate_maxima = 0 THEN
            dbms_output.put_line('Nu avem vanzari inregistrate');
        ELSE
            FOR i IN 1..produse_vandute.COUNT LOOP -- Cautam numele produsului cu cantitate maxima vanduta si ii afisam un mesaj intuitiv
                IF nr_unitati_vandute(i) = cantitate_maxima THEN
                    SELECT nume INTO nume_produs
                    FROM produs
                    WHERE id_produs = produse_vandute(i);

                    dbms_output.put_line('Produs cel mai bine vandut: ' || nume_produs || ' in cantitate de: ' || cantitate_maxima || ' unitati');
                END IF;
            END LOOP;
        END IF;
    END cel_mai_bine_vandut_produs;

----- cerinta 7
PROCEDURE calcularea_magazinului_cu_plata_cea_mai_buna
IS
    TYPE lista_id IS TABLE OF angajat.id_magazin%type;
    CURSOR salariile_medii IS -- creez cursor pentru aflarea salariilor medii oferite de fiecare magazin
        SELECT AVG(salariu), id_magazin
        FROM informatii_salariale JOIN (SELECT id_angajat, id_magazin
                                         FROM angajat) USING
        (id_angajat)
        GROUP BY id_magazin;

```

```

media_curenta number;
magazin_curent angajat.id_magazin%type;
maxim_medie_salarii number := 0;
lista_magazine lista_id := lista_id();
nume_magazin magazin.nume%type;
BEGIN
    OPEN salariile_medii;
    LOOP
        FETCH salariile_medii into media_curenta, magazin_curent; --
parcurg datele din cursor una cate una
        EXIT WHEN salariile_medii%NOTFOUND;

        IF media_curenta > maxim_medie_salarii THEN -- aflu media maxima
si salvez toate magazinele care o ofera
            maxim_medie_salarii := media_curenta;
            lista_magazine.DELETE;
            lista_magazine.extend;
            lista_magazine(lista_magazine.COUNT) := magazin_curent;
        ELSIF media_curenta = maxim_medie_salarii THEN
            lista_magazine.extend;
            lista_magazine(lista_magazine.COUNT) := magazin_curent;
        END IF;
    END LOOP;
    CLOSE salariile_medii;

    IF lista_magazine.COUNT = 0 THEN -- in cazul in care nu avem date sause
intampla ceva la preluarea datelor.
        dbms_output.put_line('NU avem date despre salariile oferite');
    ELSE

        FOR i IN 1..lista_magazine.COUNT LOOP -- preiau numele magazinului bazat
pe id-ul salvat mai devreme
            SELECT nume INTO nume_magazin
            FROM magazin
            WHERE id_magazin = lista_magazine(i);

            dbms_output.put_line('Cele mai bune salarii sunt oferite de ' ||
nume_magazin || ' ');
        END LOOP;
    END IF;

END calcularea_magazinului_cu_plata_cea_mai_buna;

----- cerinta 8
FUNCTION angajati_acordare_marire(nume_magazin magazin.nume%type,
procent_de_marire number)
RETURN tabel_mariri IS

nr_magazine_numite_la_fel number := 0;
lista_mariri tabel_mariri := tabel_mariri();

TOO_MANY_STORES EXCEPTION;
NO_STORE_FOUND EXCEPTION;
BEGIN
    SELECT COUNT(id_magazin) INTO nr_magazine_numite_la_fel
    FROM magazin

```

```

        WHERE ((UPPER(numel) LIKE UPPER(numel_magazin)) OR (UPPER(adresa) LIKE
        UPPER(numel_magazin))); -- Verific daca magazinul introdus are acelasi nume cu alt
        magazin din lista

                                -- sau daca
adresa sau numele introdus este valid

        IF nr_magazine_numite_la_fel > 1 THEN -- in cazul in care avem mai multe
        magazine cu acelasi nume trebuie introdusa adresa
            raise TOO_MANY_STORES;
        ELSIF nr_magazine_numite_la_fel = 0 THEN -- in cazul in care numele sau
        adresa introdusa nu au fost gasite
            raise NO_STORE_FOUND;
        END IF;

        FOR linie IN (  SELECT prenume || ' ' || nume_familie as nume,
                                salariu as salariu_vechi,
                                (salariu + (salariu * procent_de_marire / 100))
as salariu_nou,
                                pozitie,
                                a.id_angajat -- preluam datele ce trebuie
returnate
                                FROM angajat a JOIN informatii_salariale i_s ON (a.id_angajat
= i_s.id_angajat)
                                JOIN magazin m ON (m.id_magazin =
a.id_magazin) -- folosim 3 tabele - magazin. informatii_saalriale si angajat
                                WHERE ( (UPPER(m.numel) LIKE UPPER(numel_magazin)) OR
--verific daca numele sauza adresa se potrivesc si conditia de accordare a maririi
                                (UPPER(m.adresa) LIKE UPPER(numel_magazin)))
AND
                                i_s.ultima_marire NOT BETWEEN (SYSDATE-365)
AND SYSDATE) LOOP
            lista_mariri.extend;
            lista_mariri(lista_mariri.COUNT) := tabel_de_returnat_mariri(linie.numel,
linie.salariu_vechi, linie.salariu_nou, linie.pozitie); -- adaugam in lista de
returnat

            UPDATE informatii_salariale -- facem schimarea salariului cu cel marit
            SET salariu = linie.salariu_nou,
                ultima_marire = SYSDATE
            WHERE id_angajat = linie.id_angajat;
        END LOOP;

        IF lista_mariri.COUNT = 0 THEN -- in cazul in care pt magazinul ales nu exista
        angajati care primesc marire
            raise NO_DATA_FOUND;
        END IF;

        RETURN lista_mariri;

EXCEPTION -- definim exceptiile
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Nu avem angajati care se califica
pentru mariri');
    WHEN TOO_MANY_STORES THEN
        RAISE_APPLICATION_ERROR(-20003, 'Prea multe magazine cu acelasi nume,
introduceti adresa si incercati din nou');

```

```

WHEN NO_STORE_FOUND THEN
    RAISE_APPLICATION_ERROR(-20002, 'Nu am gasit magazin cu acest nume sau
adresa');
WHEN others THEN
    RAISE_APPLICATION_ERROR(-20001, SQLERRM);

END angajati_acordare_marire;

----- cerinta 9
PROCEDURE clienti_afectati_de_rechemare ( furnizor_afectat furnizor.nume%type,
                                            element_rechemat
produs.descriere%type) IS
    TYPE client_afectat IS TABLE OF VARCHAR(1000);

    lista_clienti_afectati client_afectat := client_afectat(); -- Definim lista ce
va fi afisata
    id_furnizor_afectat furnizor.id_furnizor%type;

    NO_AFFECTED_CLIENTS EXCEPTION; -- declar exceptia care nu e predefinita
BEGIN
    SELECT id_furnizor INTO id_furnizor_afectat -- Verificam daca furnizorul
inclus exista / este unic
        FROM furnizor
        WHERE UPPER(nume) LIKE UPPER(furnizor_afectat);

    SELECT 'Date de contact client - email: ' || email || ', numarul de telefon: '
|| telefon || ' va trebui contactat.' BULK COLLECT INTO lista_clienti_afectati -- Cream textul de afisat cu datele clientului
        FROM client
        WHERE id_client IN (SELECT id_client -- preluam toate comenziile care contin
produse afectate
                                FROM comanda
                                WHERE id_comanda IN (SELECT id_comanda
                                FROM
continut_comanda
                                WHERE
id_produs IN (SELECT id_produs -- preluam toate produsele care contin elementul
rechemat pt ca asta le afecteaza
                                FROM produs
                                WHERE (UPPER(descriere) LIKE '%' || UPPER(element_rechemat) ||
'%')
                                AND id_produs IN (SELECT id_produs
                                FROM produs_furnizor
                                WHERE id_furnizor =
id_furnizor_afectat))));

    IF lista_clienti_afectati.COUNT = 0 THEN
        RAISE NO_AFFECTED_CLIENTS;
    END IF;

    FOR i IN 1..lista_clienti_afectati.COUNT LOOP
        dbms_output.put_line(lista_clienti_afectati(i));
    END LOOP;

```

```

        END LOOP;

        EXCEPTION -- definim exceptiile
            WHEN NO_DATA_FOUND THEN
                RAISE_APPLICATION_ERROR(-20004, 'Nu exista furnizori afectati pentru
produsele vandute.');?>
            WHEN TOO_MANY_ROWS THEN
                RAISE_APPLICATION_ERROR(-20003, 'Furnizorul nu poate fi identificat
pentru ca exista mai multi cu acelasi nume.');
            WHEN NO_AFFECTED_CLIENTS THEN
                RAISE_APPLICATION_ERROR(-20002, 'Nici un client nu a fost afectat de
aceste produse!');?>
            WHEN others THEN
                RAISE_APPLICATION_ERROR(-20001, SQLERRM);
        END clienti_afectati_de_rechemare;

    END supermarket_management;
/
DECLARE
    lista_afisare tabel_mariri := tabel_mariri();
BEGIN
    dbms_output.put_line('Cerinta 6');
    supermarket_management.cel_mai_bine_vandut_produs;

    dbms_output.put_line('Cerinta 7');
    supermarket_management.calcularea_magazinului_cu_plata_cea_mai_buna;

    dbms_output.put_line('Cerinta 8');
    lista_afisare := supermarket_management.angajati_acordare_marire('Str. V. olt,
139', 10);
    FOR i IN 1..lista_afisare.COUNT LOOP
        dbms_output.put_line(i || '.' || lista_afisare(i).nume || ' -- Salariul
vechi: ' || lista_afisare(i).salariu_vechi || ' -- Salariul nou: ' ||
lista_afisare(i).salariu_nou || ' -- Pozitia: ' || lista_afisare(i).pozitie);
    END LOOP;

    dbms_output.put_line('Cerinta 9');
    supermarket_management.clienti_afectati_de_rechemare('PILOS', 'paine');
END;

```

SQL Worksheet

Clear Find Actions Save Run

```

174         raise NO_DATA_FOUND;
175     END IF;
176
177     RETURN lista_mariri;
178
179 v   EXCEPTION -- definim exceptiile
180     WHEN NO_DATA_FOUND THEN
181         RAISE_APPLICATION_ERROR(-20004, 'Nu avem angajati care se califica pentru mariri');
182 v     WHEN TOO_MANY_STORES THEN
183         RAISE_APPLICATION_ERROR(-20003, 'Prea multe magazine cu acelasi nume, introduceti adresa si incercati din nou');
184 v     WHEN NO_STORE_FOUND THEN
185         RAISE_APPLICATION_ERROR(-20002, 'Nu am gasit magazin cu acest nume sau adresa');
186 v     WHEN others THEN
187         RAISE_APPLICATION_ERROR(-20001, SQLERRM);
188
189 END angajati_acordare_marire;
190
191 ----- cerinta 9
192 v   PROCEDURE clienti_afectati_de_rechemare ( furnizor_afectat furnizor.nume%type,
193                                         element_rechemat produs.descriere%type) IS
194     TYPE client_afectat IS TABLE OF VARCHAR(1000);
195
196     lista_clienti_afectati client_afectat := client_afectat(); -- Definim lista ce va fi afisata
197     id_furnizor_afectat furnizor.id_furnizor%type;
198
199     NO_AFFECTED_CLIENTS EXCEPTION; -- declar exceptia care nu e predefinita
200 v   BEGIN
201     SELECT id_furnizor INTO id_furnizor_afectat -- Verificam daca furnizorul inclus exista / este unic
202     FROM furnizor
203     WHERE UPPER(nume) LIKE UPPER(furnizor_afectat);
204
205     SELECT 'Date de contact client - email: ' || email || ', numarul de telefon: ' || telefon || ' va trebui contactat.' BULK COLLECT INTO lista
206     FROM client
207     WHERE id_client IN (SELECT id_client -- preluam toate comenzile care contin produse afectate
208                           FROM comanda
209                           WHERE id_comanda IN (SELECT id_comanda
210                                     FROM continut_comanda
211                                     WHERE id_produs IN (SELECT id_produs -- preluam toate produsele care contin elementul rechemat pt c
212                                               FROM produs
213                                               WHERE (UPPER(descriere) LIKE '%' || UPPER(element_rechemat) || '%')
214                                                 AND id_produs IN (SELECT id_produs
215                                                   FROM produs_furnizor
216                                                   WHERE id_furnizor = id_furnizor_afectat)));
217
218     IF lista_clienti_afectati.COUNT = 0 THEN
219         RAISE NO_AFFECTED_CLIENTS;
220     END IF;
221
222     FOR i IN 1..lista_clienti_afectati.COUNT LOOP
223         dbms_output.put_line(lista_clienti_afectati(i));
224     END LOOP;
225
226     EXCEPTION -- definim exceptiile
227       WHEN NO_DATA_FOUND THEN
228           RAISE_APPLICATION_ERROR(-20004, 'Nu exista furnizori afectati pentru produsele vandute.');// suprascriu exceptiile cu mesaje custom
229 v       WHEN TOO_MANY_ROWS THEN
230           RAISE_APPLICATION_ERROR(-20003, 'Furnizorul nu poate fi identificat pentru ca exista mai multi cu acelasi nume.');// suprascriu exceptiile
231 v       WHEN NO_AFFECTED_CLIENTS THEN
232           RAISE_APPLICATION_ERROR(-20002, 'Nici un client nu a fost afectat de aceste produse!'); -- definesc exceptia nou declarata
233 v       WHEN others THEN
234           RAISE_APPLICATION_ERROR(-20001, SQLERRM);
235     END clienti_afectati_de_rechemare;
236
237 END supermarket_management;
238 v /
239
240 DECLARE
241   lista_afisare tabel_mariri := tabel_mariri();
242 v   BEGIN
243     dbms_output.put_line('Cerinta 6');
244     supermarket_management.cel_mai_bine_vandut_produs;
245
246     dbms_output.put_line('Cerinta 7');
247     supermarket_management.calcularea_magazinului_cu_plata_cea_mai_buna;
248
249     dbms_output.put_line('Cerinta 8');
250     lista_afisare := supermarket_management.angajati_acordare_marire('Str. V. olt, 139', 10);
251 v     FOR i IN 1..lista_afisare.COUNT LOOP
252       dbms_output.put_line(i || '.' || lista_afisare(i).nume || ' -- Salariul vechi: ' || lista_afisare(i).salariu_vechi || ' -- Salariul nou: ' || lista_afisare(i).salariu_nou);
253     END LOOP;
254
255     dbms_output.put_line('Cerinta 9');
256     supermarket_management.clienti_afectati_de_rechemare('PILOS', 'paine');
257   END;
258 /

```

Type created.

Type created.

Package created.

Package Body created.

Statement processed.

Cerinta 6  
Produs cel mai bine vandut: Painea Campionilor in cantitate de: 36 unitati

Cerinta 7  
Cele mai bune salarii sunt oferite de Cora

Cerinta 8  
1. Andrei Pascu -- Salariul vechi: 2000 -- Salariul nou: 2200 -- Pozitia: Casier

Cerinta 9  
Date de contact client - email: constantin.corban@gmail.com, numarul de telefon: 0700000028 va trebui contactat.  
Date de contact client - email: bogdan.maftei@gmail.com, numarul de telefon: 0700000030 va trebui contactat.

### Cerință 14:

*Enunț: Pentru un furnizor și o listă de produse date se vor face următoarele acțiuni automat:*

- Se va introduce noul furnizor și toate produsele;
- Se vor realiza legăturile și intrările în tabelul de legătură;
- Pentru produsele care au specificat o reducere se va introduce o reducere cu valoarea specificată până la data specificată.

```

CREATE OR REPLACE PACKAGE cerinta_14 IS
    -- tip custom de date pentru datele de intrare initiale pe care le vom procesa
    TYPE produs_input IS RECORD ( nume_produs produs.nume%TYPE,
                                    descriere produs.descriere%TYPE,
                                    cantitate produs.cantitate%TYPE,
                                    pret produs.pret%TYPE,
                                    are_reducere boolean,
                                    procent_reducere promotie.PROCENT_DISCOUNT%TYPE,
                                    data_sfarsit_reducere promotie.data_sfarsit%TYPE);

    TYPE lista_produse_input IS TABLE OF produs_input;

    -- al doilea tip custom de date care vor fi procesate în lista de reduceri
    TYPE discount IS RECORD( id_produs produs.id_produs%TYPE,
                             data_sfarsit_reducere
                             promotie.data_sfarsit%TYPE,
                             procent_reducere
                             promotie.PROCENT_DISCOUNT%TYPE);

    -- procedura care va fi apelata de noi initial
    PROCEDURE main ( nume_furnizor furnizor.nume%TYPE,
                      email_furnizor furnizor.email%TYPE,
                      telefon_furnizor furnizor.telefon%TYPE,
                      adresa_furnizor furnizor.adresa%TYPE,
                      lista_produse lista_produse_input);

    FUNCTION procesare_date_furnizor( nume_furnizor furnizor.nume%TYPE,
                                       email_furnizor
                                       furnizor.email%TYPE,
                                       telefon_furnizor
                                       furnizor.telefon%TYPE,
                                       adresa_furnizor
                                       furnizor.adresa%TYPE)
        RETURN furnizor.id_furnizor%TYPE;

    FUNCTION introducere_produs( nume_produs produs.nume%TYPE,
                                 descriere_produs
                                 produs.descriere%TYPE,
                                 pret_produs produs.pret%TYPE,
                                 cantitate_produs produs.cantitate%TYPE,
                                 id_furnizor_nou
                                 furnizor.id_furnizor%TYPE)
        RETURN produs.id_produs%TYPE;

    PROCEDURE adauga_reducere (promotie_noua discount);

```

```

END cerinta_14;
/

CREATE OR REPLACE PACKAGE BODY cerinta_14 IS

    -- adaug furnizorul cu datele introduse si returnez id-ul furnizorului nou creat
    FUNCTION procesare_date_furnizor (nume_furnizor furnizor.nume%TYPE,
                                       email_furnizor
furnizor.email%TYPE,
                                       telefon_furnizor
furnizor.telefon%TYPE,
                                       adresa_furnizor
furnizor.adresa%TYPE)
        RETURN furnizor.id_furnizor%TYPE IS
            nou_furnizor furnizor.id_furnizor%TYPE;
    BEGIN
        INSERT INTO furnizor
        VALUES (SEQ_FURNIZOR.NEXTVAL, nume_furnizor, email_furnizor, telefon_furnizor,
adresa_furnizor)
        RETURNING id_furnizor INTO nou_furnizor;

        RETURN nou_furnizor;
    END procesare_date_furnizor;

    -- introduc produsul curent si relatia produs-furnizor in tabelul
corespunzator
    FUNCTION introducere_produs( nume_produs produs.nume%TYPE,
                                  descriere_produs
produs.descriere%TYPE,
                                  pret_produs produs.pret%TYPE,
                                  cantitate_produs produs.cantitate%TYPE,
                                  id_furnizor_nou
furnizor.id_furnizor%TYPE)
        RETURN produs.id_produs%TYPE IS
            id_produs_nou produs.id_produs%TYPE;
    BEGIN
        -- adaug produsul nou
        INSERT INTO produs
        VALUES(SEQ_PRODUS.NEXTVAL, nume_produs, descriere_produs, pret_produs,
cantitate_produs)
        RETURNING id_produs INTO id_produs_nou;

        -- si relatia dintre produs si furnizor
        INSERT INTO produs_furnizor
        VALUES (id_furnizor_nou, id_produs_nou);

        RETURN id_produs_nou;
    END introducere_produs;

    PROCEDURE adauga_reducere (promotie_noua discount) IS
    BEGIN
        INSERT INTO promotie
        VALUES (SEQ_PROMOTIE.NEXTVAL, promotie_noua.id_produs, SYSDATE,
promotie_noua.data_sfarsit_reducere, promotie_noua.procent_reducere);
    END adauga_reducere;

```

```

PROCEDURE main ( nume_furnizor furnizor.nume%TYPE,
                email_furnizor furnizor.email%TYPE,
                telefon_furnizor furnizor.telefon%TYPE,
                adresa_furnizor furnizor.adresa%TYPE,
                lista_produse lista_produse_input) IS

    id_furnizor_nou furnizor.id_furnizor%TYPE;
    id_produs_nou produs.id_produs%TYPE;
    promotie_noua discount;

BEGIN
    -- Adaugam noul furnizor si returnam id-ul acestuia pentru mai tarziu
    id_furnizor_nou := procesare_date_furnizor(nume_furnizor,
email_furnizor, telefon_furnizor, adresa_furnizor);

    -- Parcurgem fiecare produs de adaugat
    FOR i IN lista_produse.FIRST..lista_produse.LAST LOOP
        -- adaug produsul nou
        id_produs_nou := introducere_produs(lista_produse(i).nume_produs,
lista_produse(i).descriere, lista_produse(i).pret, lista_produse(i).cantitate,
id_furnizor_nou);

        -- verific daca are discount si daca da atunci adaug discountul
        IF lista_produse(i).are_reducere = true THEN
            promotie_noua.id_produs := id_produs_nou;
            promotie_noua.data_sfarsit_reducere :=
lista_produse(i).data_sfarsit_reducere;
            promotie_noua.procent_reducere := lista_produse(i).procent_reducere;

            adauga_reducere(promotie_noua);
        END IF;
    END LOOP;

    END main;
END cerinta_14;
/

```

```

DECLARE
    lista_produse cerinta_14.lista_produse_input :=
cerinta_14.lista_produse_input( cerinta_14.produs_input('Produs 1', 'Descriere produs
1', 10, 7, true, 25, TO_DATE('26-07-2023','dd-mm-yyyy')),

                                            cerinta_14.produs_input('Produs 2',
'Descriere produs 2', 20, 5, true, 25, TO_DATE('26-07-2023','dd-mm-yyyy')),

                                            cerinta_14.produs_input('Produs
3', 'Descriere produs 3', 30, 23, false, NULL, NULL),

                                            cerinta_14.produs_input('Produs
4', 'Descriere produs 4', 40, 55, false, NULL, NULL));
BEGIN
    cerinta_14.main('Test_furnizor4', 'Test_furnizor4@gmail.com', '0700010034',
'Str. Business 104', lista_produse);
END;
/

```

Crearea pachetului

```

38 PROCEDURE adauga_reducere (promotie_noua discount);
39
40 END cerinta_14;
41 v /
42
43 CREATE OR REPLACE PACKAGE BODY cerinta_14 IS
44
45 -- adaug furnizorul cu datele introduce si returnez id-ul furnizorului nou creat
46 FUNCTION procesare_date_furnizor (nume_furnizor furnizor.nume%TYPE,
47                                     email_furnizor furnizor.email%TYPE,
48                                     telefon_furnizor furnizor.telefon%TYPE,
49                                     adresa_furnizor furnizor.adresa%TYPE)
50 RETURN furnizor.id_furnizor%TYPE IS
51     nou_furnizor furnizor.id_furnizor%TYPE;
52 BEGIN
53     INSERT INTO furnizor
54         VALUES (SEQ_FURNIZOR.NEXTVAL, nume_furnizor, email_furnizor, telefon_furnizor, adresa_furnizor)
55         RETURNING id_furnizor INTO nou_furnizor;
56
57     RETURN nou_furnizor;
58 END procesare_date_furnizor;
59
60 -- introduc produsul curent si relatia produs-furnizor in tabelul corespunzator
61 FUNCTION introducere_produs( nume_produs produs.nume%TYPE,
62                               descriere_produs produs.descriere%TYPE,
63                               pret_produs produs.pret%TYPE,
64                               cantitate_produs produs.cantitate%TYPE,
65                               id_furnizor_nou furnizor.id_furnizor%TYPE)
66 RETURN produs.id_produs%TYPE IS
67     id_produs_nou produs.id_produs%TYPE;
68 BEGIN
69     -- adaug produsul nou
70     INSERT INTO produs
71         VALUES(SEQ_PRODUS.NEXTVAL, nume_produs, descriere_produs, pret_produs, cantitate_produs)
72         RETURNING id_produs INTO id_produs_nou;
73
74     -- si relatia dintre produs si furnizor
75     INSERT INTO produs_furnizor
76         VALUES (id_furnizor_nou, id_produs_nou);
77
78     RETURN id_produs_nou;
79 END introducere_produs;
80
81
82 PROCEDURE adauga_reducere (promotie_noua discount) IS
83 BEGIN
84     INSERT INTO promotie
85         VALUES (SEQ_PROMOTIE.NEXTVAL, promotie_noua.id_produs, SYSDATE, promotie_noua.data_sfarsit_reducere, promotie_noua.procent_reducere);
86 END adauga_reducere;
87
88
89 PROCEDURE main ( nume_furnizor furnizor.nume%TYPE,
90                   email_furnizor furnizor.email%TYPE,
91                   telefon_furnizor furnizor.telefon%TYPE,
92                   adresa_furnizor furnizor.adresa%TYPE,
93                   lista_produse lista_produse_input) IS
94
95     id_furnizor_nou furnizor.id_furnizor%TYPE;
96     id_produs_nou produs.id_produs%TYPE;
97     promotie_noua discount;
98 BEGIN
99     -- Adaugam noul furnizor si returnam id-ul acestuia pentru mai tarziu
100    id_furnizor_nou := procesare_date_furnizor(nume_furnizor, email_furnizor, telefon_furnizor, adresa_furnizor);
101
102    -- Parcurgem fiecare produs de adaugat
103    FOR i IN lista_produse.FIRST..lista_produse.LAST LOOP
104        -- adaug produsul nou
105        id_produs_nou := introducere_produs(lista_produse(i).nume_produs, lista_produse(i).descriere, lista_produse(i).pret, lista_produse(i).cantitate);
106
107        -- verific daca are discount si daca da atunci adaug discountul
108        IF lista_produse(i).are_reducere = true THEN
109            promotie_noua.id_produs := id_produs_nou;
110            promotie_noua.data_sfarsit_reducere := lista_produse(i).data_sfarsit_reducere;
111            promotie_noua.procent_reducere := lista_produse(i).procent_reducere;
112
113            adauga_reducere(promotie_noua);
114        END IF;
115    END LOOP;
116
117    END main;
118 END cerinta_14;
119 v /
120
121
```

**SQL Worksheet**

Clear Find Actions Save Run

ID_FURNIZOR	NUME	EMAIL	TELEFON	ADRESA
1	Pilos	pilos@contact.com	0700000031	Str. Business 43
2	Star	star@contact.com	0700000032	Str. Business 44
3	Milbona	milbona@contact.com	0700000033	Str. Business 45
4	Dole	dole@contact.com	0700000034	Str. Business 46
5	Coca-Cola	coca.cola@contact.com	0700000035	Str. Business 47
6	Uniliver	uniliver@contact.com	0700000036	Str. Business 48

Download CSV

6 rows selected.

ID_PRODUS	NUME	DESCRISEREA	PRET	CANTITATE
1	Banane	Banane dole pret per KG	6.99	100
2	Rosii	Rosii pret per KG	4.49	100
3	Rosii cherry	Pret la bucata	12	100
4	Paine graham	Paine cu faina integrala	7.35	100
5	Pinea Campionilor	Paine cu multe seminte	9.25	100
6	Mango	Mango pret per bucata	8.1	100
7	Lapte Pilos 1.5	Lapte 1.5% grasime	4.99	100
8	Lapte Pilos 3.5	Lapte 3.5%	5.79	100
9	Teamea Pilos	Teamea de vaca, per kg	23.99	100
10	Sushi	sushi autentic	26.99	100
11	Sushi XL	Sushi XL autentic	46.99	100
12	Gogosi 6 Buc	Gogosi gem	9	100

Download CSV

12 rows selected.

ID_PROMOTIE	ID_PRODUS	DATA_INCEPUT	DATA_SFARSIT	PROCENT_DISCOUNT
1	3	16-FEB-22	26-FEB-22	10
2	5	01-APR-21	16-APR-21	33
3	6	11-FEB-22	11-MAR-22	25
4	7	09-MAY-22	29-MAY-22	5
5	2	26-FEB-21	28-MAR-21	33.33
6	12	02-MAR-21	12-MAR-21	15
7	9	11-APR-21	21-APR-21	60
8	10	08-JUL-22	18-SEP-22	75
9	10	01-OCT-22	29-NOV-22	25

Download CSV

9 rows selected.

ID_FURNIZOR	ID_PRODUS
1	1
1	2
1	3
1	4
1	5
2	2
2	4
2	5
2	6
2	3
3	6

Tabelele înainte de rulare

*Tabelele după rulare*

**SQL Worksheet**

Clear Find Actions Save Run

**Table 1: FURNIZOR**

ID_FURNIZOR	NUME	EMAIL	TELEFON	ADRESA
1	Pilos	pilos@contact.com	0700000031	Str. Business 43
2	Star	star@contact.com	0700000032	Str. Business 44
3	Milbona	milbona@contact.com	0700000033	Str. Business 45
4	Dole	dole@contact.com	0700000034	Str. Business 46
5	Coca-Cola	coca.col@contact.com	0700000035	Str. Business 47
6	Uniliver	uniliver@contact.com	0700000036	Str. Business 48
7	Test_furnizor4	Test_furnizor4@gmail.com	0700010034	Str. Business 104

[Download CSV](#)

7 rows selected.

**Table 2: PRODUS**

ID_PRODUS	NUME	DESCRISEREA	PRET	CANTITATE
1	Banane	Banane dole pret per KG	6.99	100
2	Rosii	Rosii pret per KG	4.49	100
3	Rosii cherry	Pret la bucata	12	100
4	Paine graham	Paine cu faina integrala	7.35	100
5	Painea Campionilor	Paine cu multe seminte	9.25	100
6	Mango	Mango pret per bucata	8.1	100
7	Lapte Pilos 1.5	Lapte 1.5% grasime	4.99	100
8	Lapte Pilos 3.5	Lapte 3.5%	5.79	100
9	Telemea Pilos	Telemea de vaca, per kg	23.99	100
10	Sushi	sushi autentic	26.99	100
11	Sushi XL	Sushi XL autentic	46.99	100
12	Gogosi 6 Buc	Gogosi gem	9	100
13	Produs 1	Descriere produs 1	7	10
14	Produs 2	Descriere produs 2	5	20
15	Produs 3	Descriere produs 3	23	30
16	Produs 4	Descriere produs 4	55	40

[Download CSV](#)

16 rows selected.

**Table 3: PROMOTIE**

ID_PROMOTIE	ID_PRODUS	DATA_INCEPUT	DATA_SFARSIT	PROCENT_DISCOUNT
1	3	16-FEB-22	26-FEB-22	10
2	5	01-APR-21	16-APR-21	33
3	6	11-FEB-22	11-MAR-22	25
4	7	09-MAY-22	29-MAY-22	5
5	2	26-FEB-21	28-MAR-21	33.33
6	12	02-MAR-21	12-MAR-21	15
7	9	11-APR-21	21-APR-21	60
8	10	08-JUL-22	18-SEP-22	75
9	10	01-OCT-22	29-NOV-22	25
10	13	23-MAY-23	26-JUL-23	25
11	14	23-MAY-23	26-JUL-23	25

[Download CSV](#)

11 rows selected.

**Table 4: RELATION**

ID_FURNIZOR	ID_PRODUS
1	1
1	2
1	3
1	4