**PROIECT**

**CLINICĂ VETERINARĂ**

Student: Staicu Octavian-Florin

Grupa 406 Master Inginerie Software

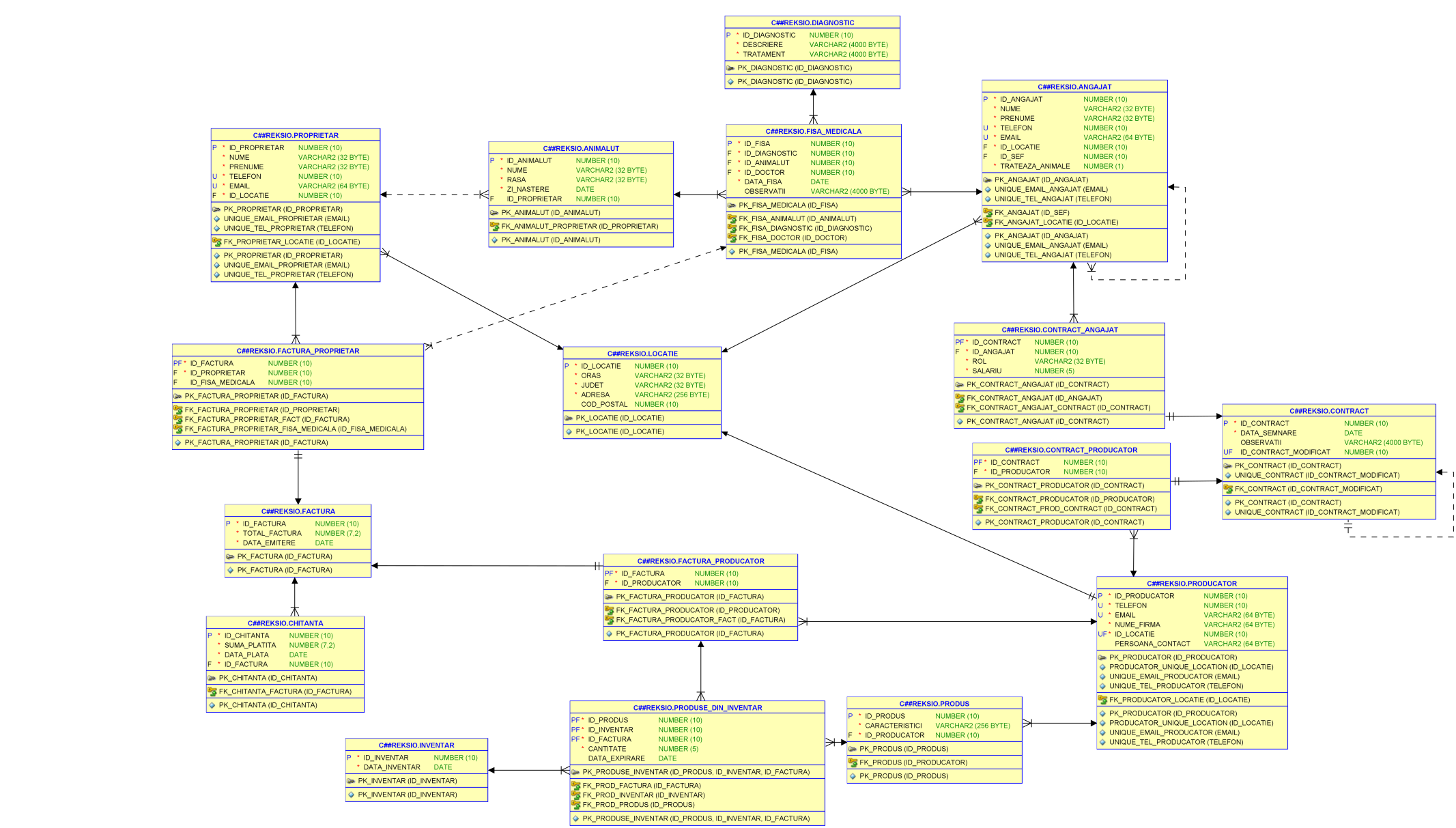
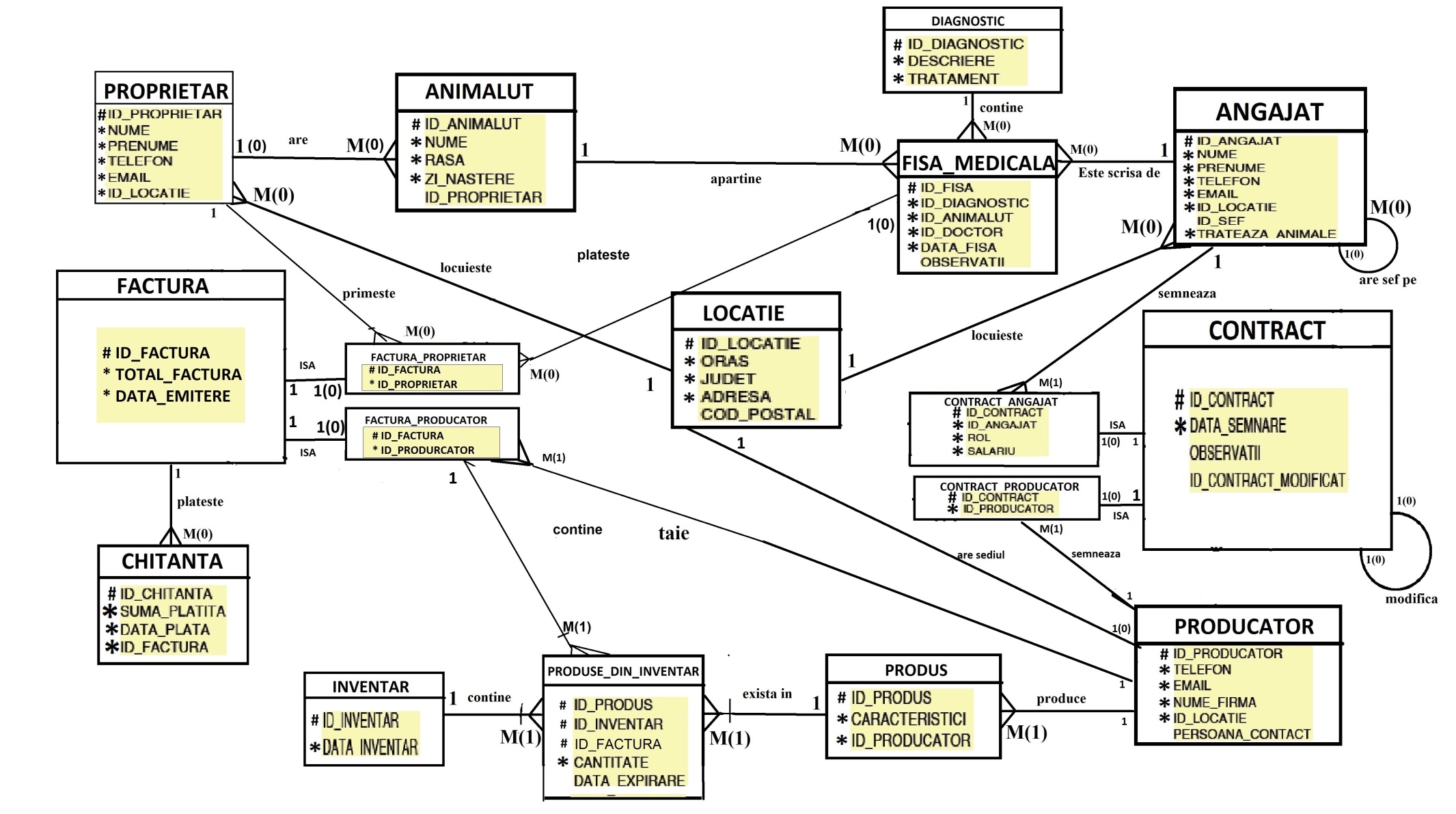
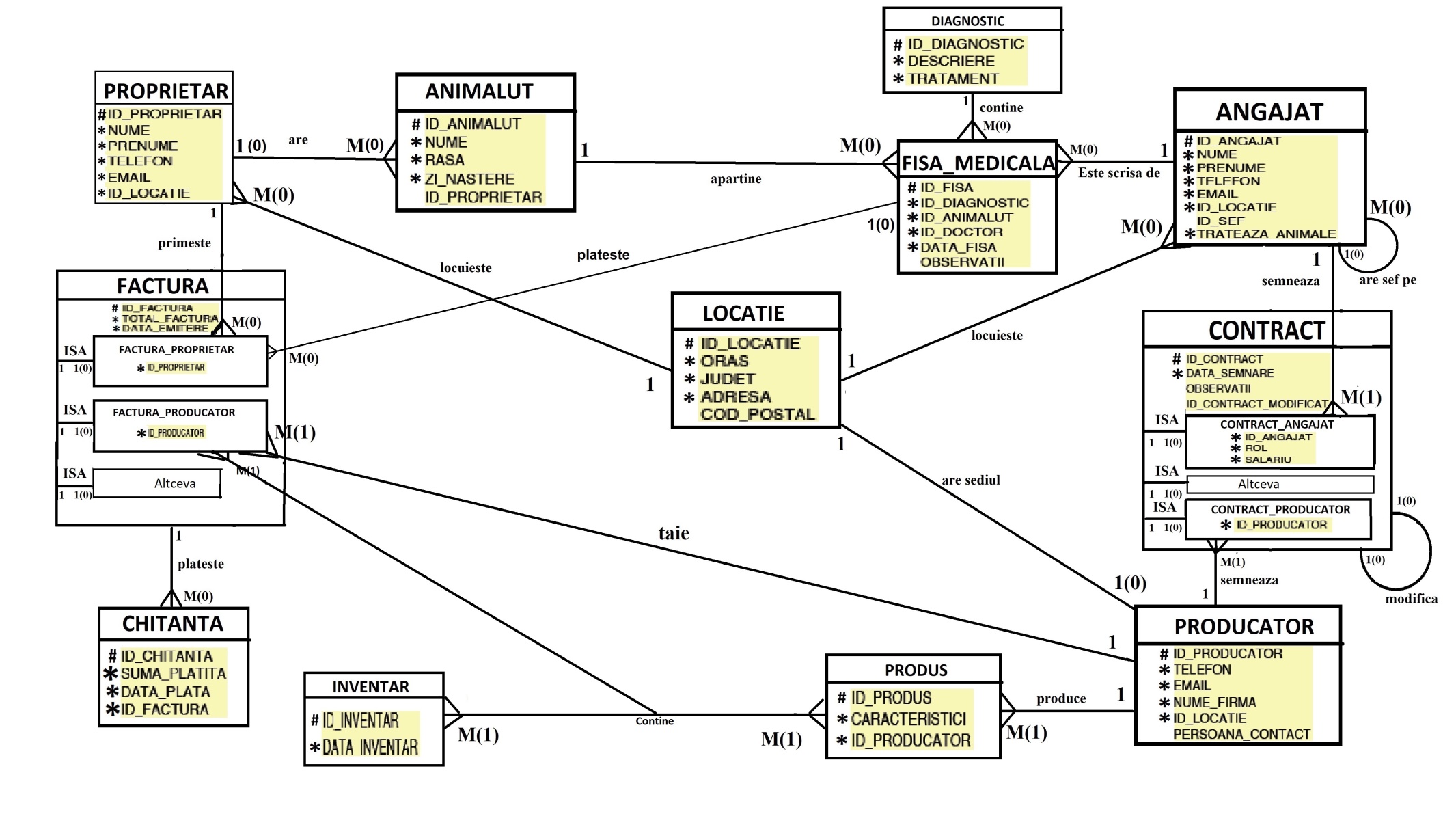
***Clinică veterinară***

Acest proiect se adresează unei clinici veterinare

# Descrierea Regulilor de Bussines

* Un proprietar are mai multe animale. Când un animăluț este tratat la clinică, i se face o nouă fișă medicală (relație de tip 3 între animal, diagnostic și un angajat care are voie să trateze animale).  Pe fișa medicală apare un diagnostic, pentru care se știe tratamentul (sau se adaugă în tabela diagnostic). Un animăluț se poate găsi în baza de date fără să aibă o fișă medicală (spre exemplu, dacă doar s-a microcipat)
* Angajații au unul sau mai multe contracte de muncă (în caz că un angajat vrea să negocieze ulterior ore suplimentare sau măriri de salariu etc., se face un nou contract, care, eventual, poate modifica un contract anterior). Doar unii angajați au voie să trateze animale (precizat in atributul din tabela angajat – ex. doar doctorii și asistentele etc.; întrucât au mai multe contracte, am ales să includ acest atribut pentru a evita o căutare prin toate contractele semnate pentru a verifica dacă există unul care să îi permită să trateze animăluțe ).
* Clinica are mai multe produse: vaccinuri, medicamente, aparatură medicală, mâncare de animale, dezinfectanți etc. Produsele sunt produse de producători, care încheie contracte cu clinica. Produsele se vor găsi în inventarele care se fac o dată la o perioadă de timp îndelungată.
* Clinica emite facturi către proprietarii de animăluțe. Aceștia le pot achita eșalonat, deci există mai multe chitanțe pentru o singură factură. Pentru o fisa medicala se pot emite mai multe facturi.
* Producătorii de produse emit și ei facturi, pe care clinica trebuie să le plătească (la fel, eșalonat).

***Pe următoarele două pagini se află diagrama entitate relație și diagrama conceptuală.***



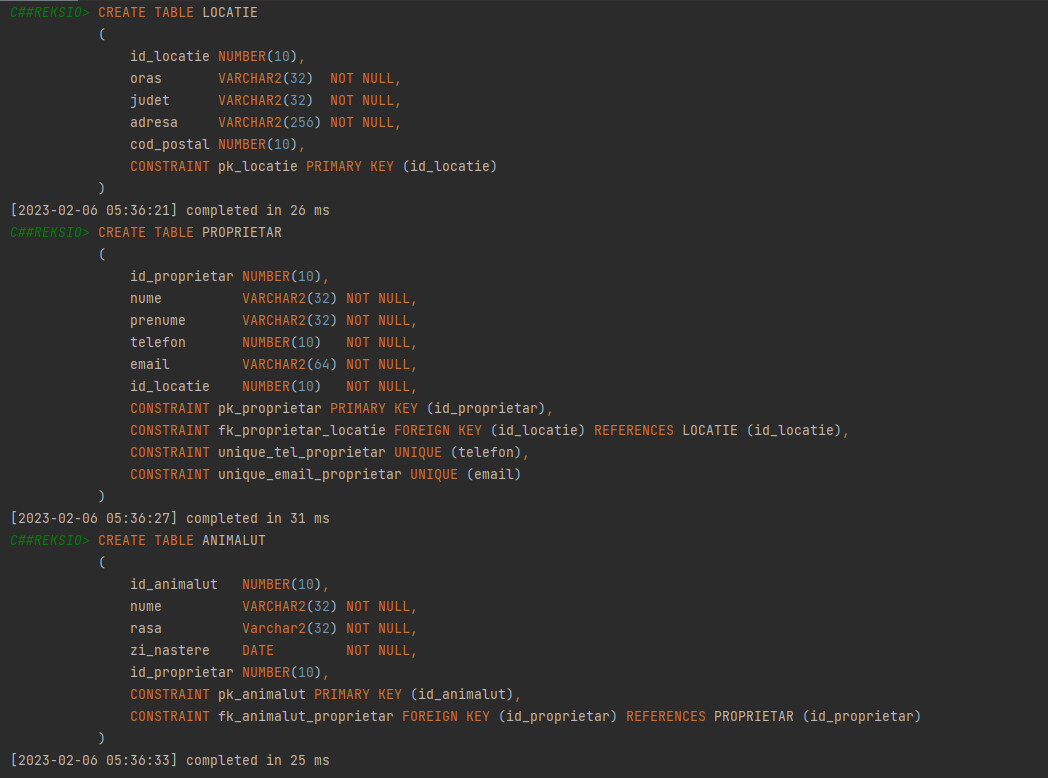
-- Ex. 5. Diagramele pe care le-am folosit le-am conceput de la inceput in FN3.

-- Fn1 ar fi fost incalcat daca spre exemplu ar fi existat in tabela proprietar antributul animalute (ar fi existat un atribut cu valori multiple).

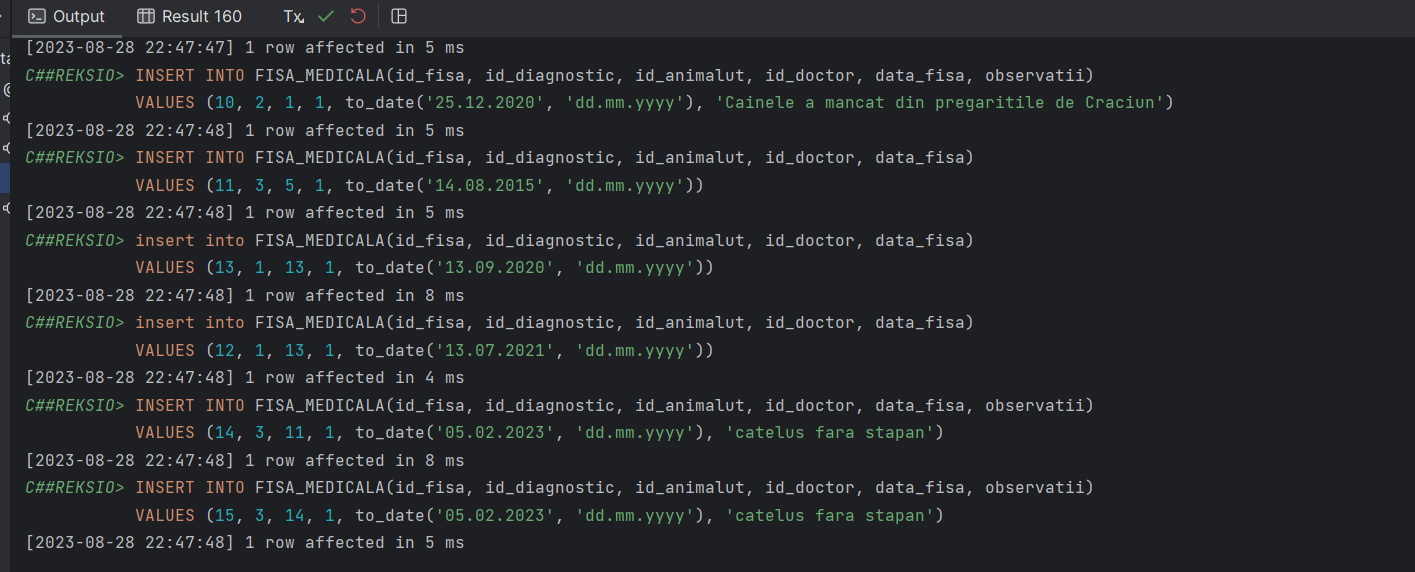
-- Fn2 ar fi fost incalcat spre exemplu daca tabela animalut ar fi avut atributele tabelei diagnostic. Astfel, ar fi trebuit sa duplicam intrarile animalutului (nume, rasa, zi\_nastere, id\_proprietar) ori de cate ori se punea un nou daignostic. Cu alte cuvinte, ar fi existat o asa zisa cheia primara compusa din id\_proprietar si id\_animalut, iar fn2 spune ca orice atribut care nu face parte din cheia primara trebuie sa depinda de intreaga cheie (spre exemplu, rasa unui catel nu depinda de id\_proprietar).

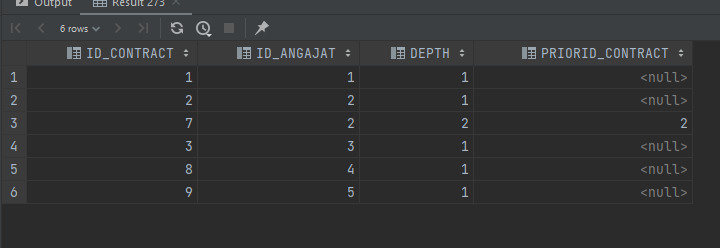
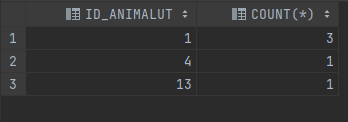
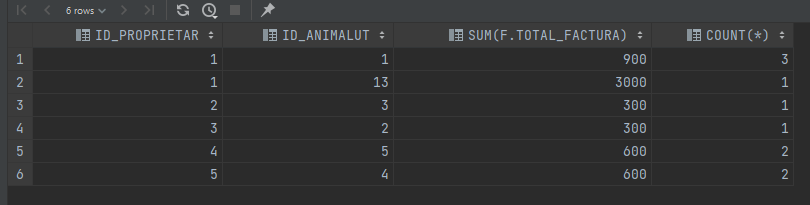
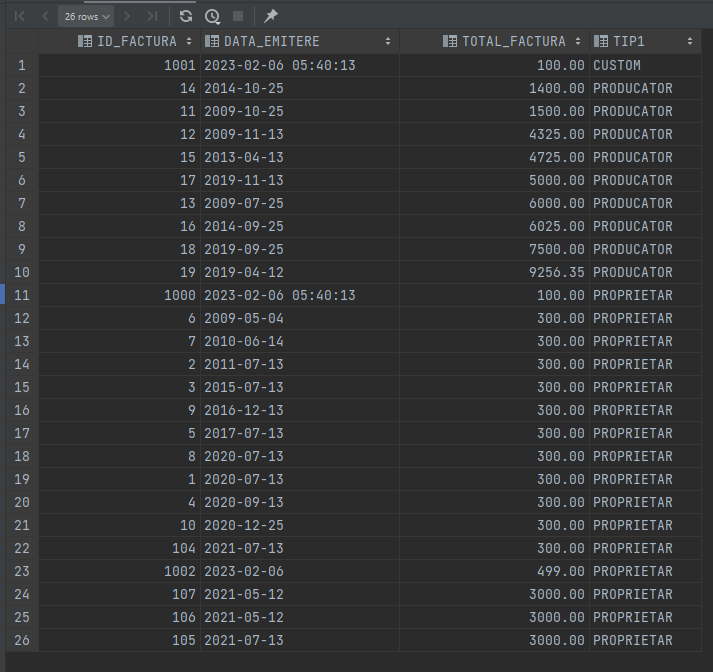
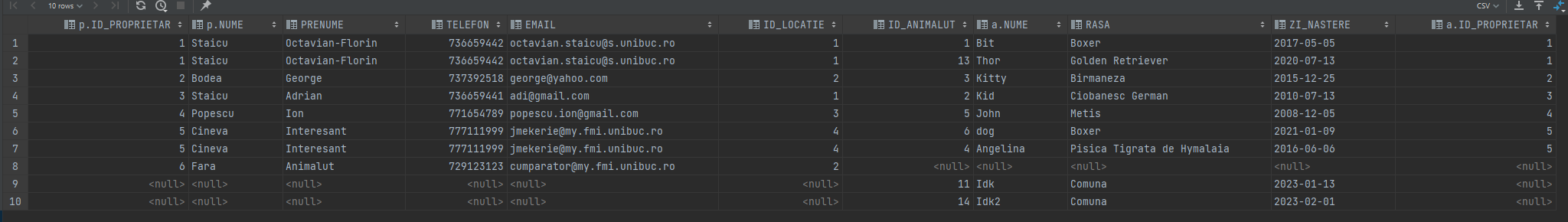
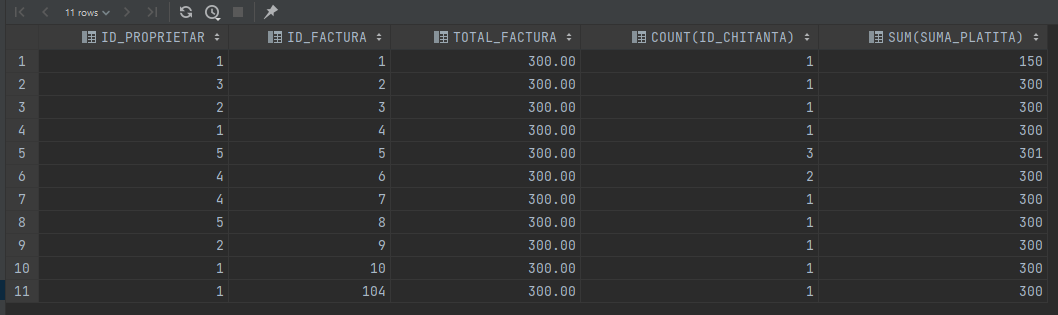
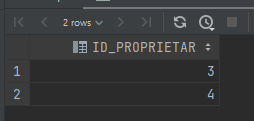
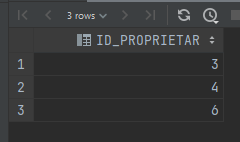
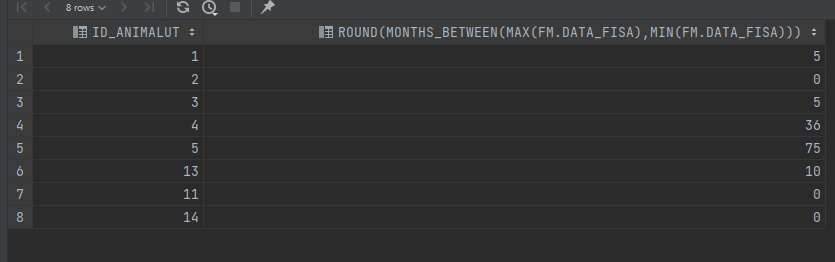
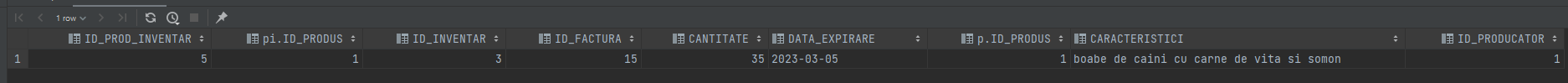
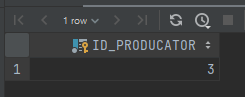
-- Fn3 ar fi fost incalcat spre exemplu daca in tabela proprietar am fi avut si toate atributele tabelei locatie, caci asa am avea dependenta tranzitiva intre acele atribute si id-ul locatiei (pt fiecare proprietar, am avea de fiecare data valori duplicate pt oras, judet, adresa, cod\_postal).

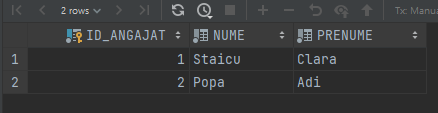
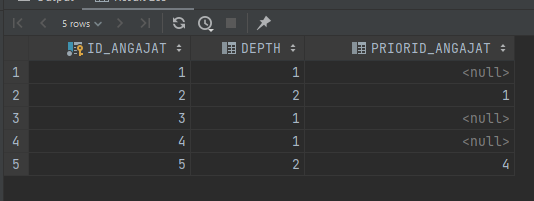
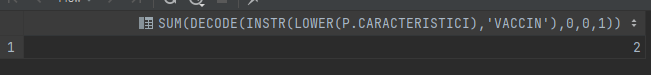
--6  
CREATE TABLE LOCATIE  
(  
 id\_locatie NUMBER(10),  
 oras VARCHAR2(32) NOT NULL,  
 judet VARCHAR2(32) NOT NULL,  
 adresa VARCHAR2(256) NOT NULL,  
 cod\_postal NUMBER(10),  
 CONSTRAINT pk\_locatie PRIMARY KEY (id\_locatie)  
);  
  
CREATE TABLE PROPRIETAR  
(  
 id\_proprietar NUMBER(10),  
 nume VARCHAR2(32) NOT NULL,  
 prenume VARCHAR2(32) NOT NULL,  
 telefon NUMBER(10) NOT NULL,  
 email VARCHAR2(64) NOT NULL,  
 id\_locatie NUMBER(10) NOT NULL,  
 CONSTRAINT pk\_proprietar PRIMARY KEY (id\_proprietar),  
 CONSTRAINT fk\_proprietar\_locatie FOREIGN KEY (id\_locatie) REFERENCES LOCATIE (id\_locatie),  
 CONSTRAINT unique\_tel\_proprietar UNIQUE (telefon),  
 CONSTRAINT unique\_email\_proprietar UNIQUE (email)  
);  
  
CREATE TABLE ANIMALUT  
(  
 id\_animalut NUMBER(10),  
 nume VARCHAR2(32) NOT NULL,  
 rasa Varchar2(32) NOT NULL,  
 zi\_nastere DATE NOT NULL,  
 id\_proprietar NUMBER(10),  
 CONSTRAINT pk\_animalut PRIMARY KEY (id\_animalut),  
 CONSTRAINT fk\_animalut\_proprietar FOREIGN KEY (id\_proprietar) REFERENCES PROPRIETAR (id\_proprietar)  
);  
  
CREATE TABLE DIAGNOSTIC  
(  
 id\_diagnostic NUMBER(10),  
 descriere VARCHAR2(4000) NOT NULL,  
 tratament VARCHAR2(4000) NOT NULL,  
 CONSTRAINT pk\_diagnostic PRIMARY KEY (id\_diagnostic)  
);  
  
CREATE TABLE ANGAJAT  
(  
 id\_angajat NUMBER(10),  
 nume VARCHAR2(32) NOT NULL,  
 prenume VARCHAR2(32) NOT NULL,  
 telefon NUMBER(10) NOT NULL,  
 email VARCHAR2(64) NOT NULL,  
 id\_locatie NUMBER(10) NOT NULL,  
 id\_sef NUMBER(10),  
 trateaza\_animale NUMBER(1) NOT NULL,  
 CONSTRAINT pk\_angajat PRIMARY KEY (id\_angajat),  
 CONSTRAINT fk\_angajat\_locatie FOREIGN KEY (id\_locatie) REFERENCES LOCATIE (id\_locatie),  
 CONSTRAINT fk\_angajat FOREIGN KEY (id\_sef) REFERENCES ANGAJAT (id\_angajat),  
 CONSTRAINT unique\_tel\_angajat UNIQUE (telefon),  
 CONSTRAINT unique\_email\_angajat UNIQUE (email)  
);  
  
CREATE TABLE FISA\_MEDICALA  
(  
 id\_fisa NUMBER(10),  
 id\_diagnostic NUMBER(10) NOT NULL,  
 id\_animalut NUMBER(10) NOT NULL,  
 id\_doctor NUMBER(10) NOT NULL,  
 data\_fisa DATE NOT NULL,  
 observatii VARCHAR2(4000),  
 CONSTRAINT pk\_fisa\_medicala PRIMARY KEY (id\_fisa),  
 CONSTRAINT fk\_fisa\_diagnostic FOREIGN KEY (id\_diagnostic) REFERENCES DIAGNOSTIC (id\_diagnostic),  
 CONSTRAINT fk\_fisa\_animalut FOREIGN KEY (id\_animalut) REFERENCES ANIMALUT (id\_animalut),  
 CONSTRAINT fk\_fisa\_doctor FOREIGN KEY (id\_doctor) REFERENCES ANGAJAT (id\_angajat)  
);  
  
CREATE TABLE PRODUCATOR  
(  
 id\_producator NUMBER(10),  
 telefon NUMBER(10) NOT NULL,  
 email VARCHAR2(64) NOT NULL,  
 nume\_firma VARCHAR2(64) NOT NULL,  
 id\_locatie NUMBER(10) NOT NULL,  
 persoana\_contact VARCHAR2(64),  
 CONSTRAINT pk\_producator PRIMARY KEY (id\_producator),  
 CONSTRAINT fk\_producator\_locatie FOREIGN KEY (id\_locatie) REFERENCES LOCATIE (id\_locatie),  
 CONSTRAINT producator\_unique\_location UNIQUE (id\_locatie),  
 CONSTRAINT unique\_tel\_producator UNIQUE (telefon),  
 CONSTRAINT unique\_email\_producator UNIQUE (email)  
);  
  
CREATE TABLE CONTRACT  
(  
 id\_contract NUMBER(10),  
 data\_semnare DATE NOT NULL,  
 observatii VARCHAR2(4000),  
 id\_contract\_modificat NUMBER(10),  
 CONSTRAINT pk\_contract PRIMARY KEY (id\_contract),  
 CONSTRAINT fk\_contract FOREIGN KEY (id\_contract\_modificat) REFERENCES CONTRACT (id\_contract),  
 CONSTRAINT unique\_contract UNIQUE (id\_contract\_modificat)  
);  
  
CREATE TABLE CONTRACT\_ANGAJAT  
(  
 id\_contract NUMBER(10),  
 id\_angajat NUMBER(10) NOT NULL,  
 rol VARCHAR2(32) NOT NULL,  
 salariu NUMBER(5) NOT NULL,  
 CONSTRAINT pk\_contract\_angajat PRIMARY KEY (id\_contract),  
 CONSTRAINT fk\_contract\_angajat\_contract FOREIGN KEY (id\_contract) REFERENCES CONTRACT (id\_contract),  
 CONSTRAINT fk\_contract\_angajat FOREIGN KEY (id\_angajat) REFERENCES ANGAJAT (id\_angajat)  
);  
  
CREATE TABLE CONTRACT\_PRODUCATOR  
(  
 id\_contract NUMBER(10),  
 id\_producator NUMBER(10) NOT NULL,  
 CONSTRAINT pk\_contract\_producator PRIMARY KEY (id\_contract),  
 CONSTRAINT fk\_contract\_prod\_contract FOREIGN KEY (id\_contract) REFERENCES CONTRACT (id\_contract),  
 CONSTRAINT fk\_contract\_producator FOREIGN KEY (id\_producator) REFERENCES PRODUCATOR (id\_producator)  
);  
  
CREATE TABLE FACTURA  
(  
 id\_factura NUMBER(10),  
 total\_factura NUMBER(7, 2) NOT NULL,  
 data\_emitere DATE NOT NULL,  
 CONSTRAINT pk\_factura PRIMARY KEY (id\_factura)  
);  
  
CREATE TABLE FACTURA\_PROPRIETAR  
(  
 id\_factura NUMBER(10),  
 id\_proprietar NUMBER(10) NOT NULL,  
 id\_fisa\_medicala NUMBER(10),  
 CONSTRAINT pk\_factura\_proprietar PRIMARY KEY (id\_factura),  
 CONSTRAINT fk\_factura\_proprietar FOREIGN KEY (id\_proprietar) REFERENCES PROPRIETAR (id\_proprietar),  
 CONSTRAINT fk\_factura\_proprietar\_fact FOREIGN KEY (id\_factura) REFERENCES FACTURA (id\_factura),  
 CONSTRAINT fk\_factura\_proprietar\_fisa\_medicala FOREIGN KEY (id\_fisa\_medicala) REFERENCES FISA\_MEDICALA (id\_fisa)  
);  
  
CREATE TABLE FACTURA\_PRODUCATOR  
(  
 id\_factura NUMBER(10),  
 id\_producator NUMBER(10) NOT NULL,  
 CONSTRAINT pk\_factura\_producator PRIMARY KEY (id\_factura),  
 CONSTRAINT fk\_factura\_producator FOREIGN KEY (id\_producator) REFERENCES PRODUCATOR (id\_producator),  
 CONSTRAINT fk\_factura\_producator\_fact FOREIGN KEY (id\_factura) REFERENCES FACTURA (id\_factura)  
);  
  
CREATE TABLE CHITANTA  
(  
 id\_chitanta NUMBER(10),  
 suma\_platita NUMBER(7, 2) NOT NULL,  
 data\_plata DATE NOT NULL,  
 id\_factura NUMBER(10) NOT NULL,  
 CONSTRAINT pk\_chitanta PRIMARY KEY (id\_chitanta),  
 CONSTRAINT fk\_chitanta\_factura FOREIGN KEY (id\_factura) references FACTURA (id\_factura)  
);  
  
  
CREATE TABLE PRODUS  
(  
 id\_produs NUMBER(10),  
 caracteristici VARCHAR2(256) NOT NULL,  
 id\_producator NUMBER(10) NOT NULL,  
 CONSTRAINT pk\_produs PRIMARY KEY (id\_produs),  
 CONSTRAINT fk\_produs FOREIGN KEY (id\_producator) REFERENCES PRODUCATOR (id\_producator)  
);  
  
CREATE TABLE INVENTAR  
(  
 id\_inventar NUMBER(10),  
 data\_inventar DATE NOT NULL,  
 CONSTRAINT pk\_inventar PRIMARY KEY (id\_inventar)  
);  
  
CREATE TABLE PRODUSE\_DIN\_INVENTAR  
(  
 id\_produs NUMBER(10) NOT NULL,  
 id\_inventar NUMBER(10) NOT NULL,  
 id\_factura NUMBER(10) NOT NULL,  
 cantitate NUMBER(5) NOT NULL,  
 data\_expirare DATE,  
 CONSTRAINT pk\_produse\_inventar PRIMARY KEY (id\_produs, id\_inventar, id\_factura),  
 CONSTRAINT fk\_prod\_inventar FOREIGN KEY (id\_inventar) REFERENCES INVENTAR (id\_inventar),  
 CONSTRAINT fk\_prod\_produs FOREIGN KEY (id\_produs) REFERENCES PRODUS (id\_produs),  
 CONSTRAINT fk\_prod\_factura FOREIGN KEY (id\_factura) REFERENCES FACTURA\_PRODUCATOR (id\_factura)  
);  
  
commit;

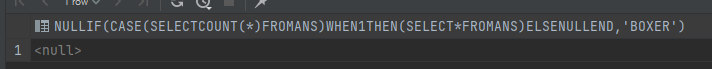


--5  
INSERT INTO LOCATIE(id\_locatie, oras, judet, adresa)  
VALUES (1, 'Bucuresti', 'Sector 2', 'soseaua colentina nr 2');  
INSERT INTO LOCATIE(id\_locatie, oras, judet, adresa, cod\_postal)  
VALUES (2, 'Bucuresti', 'Sector 1', 'soseaua Aviatorilor nr 1', 12345);  
INSERT INTO LOCATIE(id\_locatie, oras, judet, adresa)  
VALUES (3, 'Pantelimon', 'Ilfov', 'soseaua dn2');  
INSERT INTO LOCATIE(id\_locatie, oras, judet, adresa)  
VALUES (4, 'Cluj-Napoca', 'Cluj', 'soseaua jmekera');  
INSERT INTO LOCATIE(id\_locatie, oras, judet, adresa)  
VALUES (5, 'Iasi', 'Iasi', 'soseaua industriei');  
INSERT INTO LOCATIE(id\_locatie, oras, judet, adresa)  
VALUES (6, 'Timisoara', 'Timis', 'soseaua intreprinderii');  
INSERT INTO LOCATIE(id\_locatie, oras, judet, adresa)  
VALUES (7, 'Constanta', 'Constanta', 'soseaua antreprenorilor');  
INSERT INTO LOCATIE(id\_locatie, oras, judet, adresa)  
VALUES (8, 'Giurgiu', 'Giurgiu', 'soseaua giurgiului');  
INSERT INTO LOCATIE(id\_locatie, oras, judet, adresa)  
VALUES (9, 'Severin', 'Mehedinti', 'traian');  
  
INSERT INTO PROPRIETAR(id\_proprietar, nume, prenume, email, telefon, id\_locatie)  
VALUES (1, 'Staicu', 'Octavian-Florin', 'octavian.staicu@s.unibuc.ro', 0736659442, 1);  
INSERT INTO PROPRIETAR(id\_proprietar, nume, prenume, email, telefon, id\_locatie)  
VALUES (2, 'Bodea', 'George', 'george@yahoo.com', 0737392518, 2);  
INSERT INTO PROPRIETAR(id\_proprietar, nume, prenume, email, telefon, id\_locatie)  
VALUES (3, 'Staicu', 'Adrian', 'adi@gmail.com', 0736659441, 1);  
INSERT INTO PROPRIETAR(id\_proprietar, nume, prenume, email, telefon, id\_locatie)  
VALUES (4, 'Popescu', 'Ion', 'popescu.ion@gmail.com', 0771654789, 3);  
INSERT INTO PROPRIETAR(id\_proprietar, nume, prenume, email, telefon, id\_locatie)  
VALUES (5, 'Cineva', 'Interesant', 'jmekerie@my.fmi.unibuc.ro', 0777111999, 4);  
INSERT INTO PROPRIETAR(id\_proprietar, nume, prenume, email, telefon, id\_locatie)  
VALUES (6, 'Fara', 'Animalut', 'cumparator@my.fmi.unibuc.ro', 0729123123, 2);  
  
INSERT INTO ANIMALUT(id\_animalut, nume, rasa, zi\_nastere, id\_proprietar)  
VALUES (1, 'Bit', 'Boxer', *to\_date*('05/05/2017', 'DD/MM/YYYY'), 1);  
INSERT INTO ANIMALUT(id\_animalut, nume, rasa, zi\_nastere, id\_proprietar)  
VALUES (2, 'Kid', 'Ciobanesc German', *to\_date*('13/07/2010', 'DD/MM/YYYY'), 3);  
INSERT INTO ANIMALUT(id\_animalut, nume, rasa, zi\_nastere, id\_proprietar)  
VALUES (3, 'Kitty', 'Birmaneza', *to\_date*('25/12/2015', 'DD/MM/YYYY'), 2);  
INSERT INTO ANIMALUT(id\_animalut, nume, rasa, zi\_nastere, id\_proprietar)  
VALUES (4, 'Angelina', 'Pisica Tigrata de Hymalaia', *to\_date*('06/06/2016', 'DD/MM/YYYY'), 5);  
INSERT INTO ANIMALUT(id\_animalut, nume, rasa, zi\_nastere, id\_proprietar)  
VALUES (5, 'John', 'Metis', *to\_date*('05/12/2008', 'DD/MM/YYYY'), 4);  
insert into ANIMALUT(id\_animalut, nume, rasa, zi\_nastere, id\_proprietar)  
VALUES (6, 'dog', 'Boxer', *to\_date*('09.01.2021', 'dd.mm.yyyy'), 5);  
insert into ANIMALUT(id\_animalut, nume, rasa, zi\_nastere, id\_proprietar)  
VALUES (13, 'Thor', 'Golden Retriever', *to\_date*('13.07.2020', 'dd.mm.yyyy'), 1);  
insert into ANIMALUT(id\_animalut, nume, rasa, zi\_nastere)  
VALUES (11, 'Idk', 'Comuna', *to\_date*('13.01.2023', 'dd.mm.yyyy'));  
insert into ANIMALUT(id\_animalut, nume, rasa, zi\_nastere)  
VALUES (14, 'Idk2', 'Comuna', *to\_date*('01.02.2023', 'dd.mm.yyyy'));  
  
INSERT INTO DIAGNOSTIC(id\_diagnostic, descriere, tratament)  
VALUES (1, 'Vaccinare Leptospiroza', 'Vaccin Leptospiroza');  
INSERT INTO DIAGNOSTIC(id\_diagnostic, descriere, tratament)  
VALUES (2, 'Enterocolita', 'Enteroguarg');  
INSERT INTO DIAGNOSTIC(id\_diagnostic, descriere, tratament)  
VALUES (3, 'Rana deschisa', 'Spray Cicatrizant');  
INSERT INTO DIAGNOSTIC(id\_diagnostic, descriere, tratament)  
VALUES (4, 'durere abdominala', 'ecografie');  
INSERT INTO DIAGNOSTIC(id\_diagnostic, descriere, tratament)  
VALUES (5, 'capusa', 'eliminare capusa');  
  
INSERT INTO ANGAJAT(id\_angajat, nume, prenume, telefon, email, id\_locatie, trateaza\_animale)  
VALUES (1, 'Staicu', 'Clara', 0722012983, 'dr.staicu-clara@clinicatareanimale.ro', 1, 1);  
INSERT INTO ANGAJAT(id\_angajat, nume, prenume, telefon, email, id\_sef, id\_locatie, trateaza\_animale)  
VALUES (2, 'Popa', 'Adi', 0723145867, 'asistent.popa-adi@clinicatareanimale.ro', 1, 2, 1);  
INSERT INTO ANGAJAT(id\_angajat, nume, prenume, telefon, email, id\_locatie, trateaza\_animale)  
VALUES (3, 'Spalatorul', 'Pop', 0711012070, 'spalatorul.pop@gmail.com', 3, 0);  
INSERT INTO ANGAJAT(id\_angajat, nume, prenume, telefon, email, id\_locatie, trateaza\_animale)  
VALUES (4, 'Ana', 'Maria', 0723012982, 'dr.ana-maria@clinicatareanimale.ro', 1, 1);  
INSERT INTO ANGAJAT(id\_angajat, nume, prenume, telefon, email, id\_sef, id\_locatie, trateaza\_animale)  
VALUES (5, 'Numinescu', 'George', 0723145868, 'asistent.numinescu-george@clinicatareanimale.ro', 4, 2, 1);  
  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa, observatii)  
VALUES (1, 2, 1, 1, *to\_date*('13.07.2020', 'dd.mm.yyyy'), 'Stapanul a dat caineul sa manance tort');  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa)  
VALUES (2, 1, 2, 1, *to\_date*('13.07.2011', 'dd.mm.yyyy'));  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa)  
VALUES (3, 1, 3, 2, *to\_date*('13.07.2016', 'dd.mm.yyyy'));  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa, observatii)  
VALUES (4, 3, 1, 2, *to\_date*('13.09.2020', 'dd.mm.yyyy'), 'Rana deschisa in zona botului');  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa, observatii)  
VALUES (5, 1, 4, 1, *to\_date*('13.07.2017', 'dd.mm.yyyy'), 'Una din cele mai frumoase specimene');  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa)  
VALUES (6, 2, 5, 1, *to\_date*('04.05.2009', 'dd.mm.yyyy'));  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa, observatii)  
VALUES (7, 1, 5, 2, *to\_date*('14.06.2010', 'dd.mm.yyyy'), 'Vaccinare anuala');  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa)  
VALUES (8, 1, 4, 2, *to\_date*('13.07.2020', 'dd.mm.yyyy'));  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa, observatii)  
VALUES (9, 3, 3, 1, *to\_date*('13.12.2016', 'dd.mm.yyyy'), 'S-a taiat intr-o conserva de peste');  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa, observatii)  
VALUES (10, 2, 1, 1, *to\_date*('25.12.2020', 'dd.mm.yyyy'), 'Cainele a mancat din pregaritile de Craciun');  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa)  
VALUES (11, 3, 5, 1, *to\_date*('14.08.2015', 'dd.mm.yyyy'));  
insert into FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa)  
VALUES (13, 1, 13, 1, *to\_date*('13.09.2020', 'dd.mm.yyyy'));  
insert into FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa)  
VALUES (12, 1, 13, 1, *to\_date*('13.07.2021', 'dd.mm.yyyy'));  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa, observatii)  
VALUES (14, 3, 11, 1, *to\_date*('05.02.2023', 'dd.mm.yyyy'), 'catelus fara stapan');  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa, observatii)  
VALUES (15, 3, 14, 1, *to\_date*('05.02.2023', 'dd.mm.yyyy'), 'catelus fara stapan');  
  
INSERT INTO PRODUCATOR(id\_producator, nume\_firma, telefon, email, id\_locatie)  
VALUES (1, 'Mancare Catei SA', 0767582563, 'comenzi@mancarecatei.ro', 5);  
INSERT INTO PRODUCATOR(id\_producator, nume\_firma, telefon, email, persoana\_contact, id\_locatie)  
VALUES (2, 'Aparatura Animale SRL', 0798712365, 'aparatus.ion@aparaturaanimale.ro', 'Aparatus Ion', 6);  
INSERT INTO PRODUCATOR(id\_producator, nume\_firma, telefon, email, persoana\_contact, id\_locatie)  
VALUES (3, 'Vaccinam cu Iubire', 0712345678, 'bussines@vacciniubire.ro', 'Ana Vacix', 7);  
INSERT INTO PRODUCATOR(id\_producator, nume\_firma, telefon, email, persoana\_contact, id\_locatie)  
VALUES (4, 'Producator 4', 0700000000, 'producator4@bussines.ro', 'Producator 4 contact', 8);  
INSERT INTO PRODUCATOR(id\_producator, nume\_firma, telefon, email, persoana\_contact, id\_locatie)  
VALUES (5, 'Producator 5', 0711111111, 'bussines@producator5.ro', 'contact prod 5', 9);  
  
INSERT INTO CONTRACT(id\_contract, data\_semnare)  
VALUES (1, *to\_date*('02.01.2005', 'dd.mm.yyyy'));  
INSERT INTO CONTRACT(id\_contract, data\_semnare)  
VALUES (2, *to\_date*('13.07.2004', 'dd.mm.yyyy'));  
INSERT INTO CONTRACT(id\_contract, data\_semnare)  
VALUES (3, *to\_date*('09.09.2005', 'dd.mm.yyyy'));  
INSERT INTO CONTRACT(id\_contract, data\_semnare)  
VALUES (4, *to\_date*('01.05.2007', 'dd.mm.yyyy'));  
INSERT INTO CONTRACT(id\_contract, data\_semnare)  
VALUES (5, *to\_date*('02.06.2008', 'dd.mm.yyyy'));  
INSERT INTO CONTRACT(id\_contract, data\_semnare)  
VALUES (6, *to\_date*('02.01.2005', 'dd.mm.yyyy'));  
INSERT INTO CONTRACT(id\_contract, data\_semnare, observatii, id\_contract\_modificat)  
VALUES (7, *to\_date*('28.12.2020', 'dd.mm.yyyy'), 'Promovare', 2);  
INSERT INTO CONTRACT(id\_contract, data\_semnare)  
VALUES (8, *to\_date*('02.01.2020', 'dd.mm.yyyy'));  
INSERT INTO CONTRACT(id\_contract, data\_semnare)  
VALUES (9, *to\_date*('02.01.2021', 'dd.mm.yyyy'));  
INSERT INTO CONTRACT(id\_contract, data\_semnare)  
VALUES (10, *to\_date*('02.07.2022', 'dd.mm.yyyy'));  
INSERT INTO CONTRACT(id\_contract, data\_semnare)  
VALUES (11, *to\_date*('13.01.2023', 'dd.mm.yyyy'));  
  
INSERT INTO CONTRACT\_ANGAJAT(id\_contract, id\_angajat, rol, salariu)  
VALUES (1, 1, 'doctor veterinar', 6000);  
INSERT INTO CONTRACT\_ANGAJAT(id\_contract, id\_angajat, rol, salariu)  
VALUES (2, 2, 'asistent veterinar', 3000);  
INSERT INTO CONTRACT\_ANGAJAT(id\_contract, id\_angajat, rol, salariu)  
VALUES (3, 3, 'paznic', 1500);  
INSERT INTO CONTRACT\_ANGAJAT(id\_contract, id\_angajat, rol, salariu)  
VALUES (7, 2, 'doctor veterinar', 6000);  
INSERT INTO CONTRACT\_ANGAJAT(id\_contract, id\_angajat, rol, salariu)  
VALUES (8, 4, 'doctor veterinar', 6000);  
INSERT INTO CONTRACT\_ANGAJAT(id\_contract, id\_angajat, rol, salariu)  
VALUES (9, 5, 'asistent veterinar', 3000);  
  
INSERT INTO CONTRACT\_PRODUCATOR(id\_contract, id\_producator)  
VALUES (4, 1);  
INSERT INTO CONTRACT\_PRODUCATOR(id\_contract, id\_producator)  
VALUES (5, 2);  
INSERT INTO CONTRACT\_PRODUCATOR(id\_contract, id\_producator)  
VALUES (6, 3);  
INSERT INTO CONTRACT\_PRODUCATOR(id\_contract, id\_producator)  
VALUES (10, 4);  
INSERT INTO CONTRACT\_PRODUCATOR(id\_contract, id\_producator)  
VALUES (11, 5);  
  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (1, 300, *to\_date*('13.07.2020', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (2, 300, *to\_date*('13.07.2011', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (3, 300, *to\_date*('13.07.2015', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (4, 300, *to\_date*('13.09.2020', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (5, 300, *to\_date*('13.07.2017', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (6, 300, *to\_date*('04.05.2009', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (7, 300, *to\_date*('14.06.2010', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (8, 300, *to\_date*('13.07.2020', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (9, 300, *to\_date*('13.12.2016', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (10, 300, *to\_date*('25.12.2020', 'dd.mm.yyyy'));  
insert into FACTURA(id\_factura, total\_factura, data\_emitere)  
VALUES (104, 300, *to\_date*('13.07.2021', 'dd.mm.yyyy'));  
insert into FACTURA(id\_factura, total\_factura, data\_emitere)  
VALUES (105, 3000, *to\_date*('13.07.2021', 'dd.mm.yyyy'));  
insert into FACTURA(id\_factura, total\_factura, data\_emitere)  
VALUES (106, 3000, *to\_date*('12.05.2021', 'dd.mm.yyyy'));  
insert into FACTURA(id\_factura, total\_factura, data\_emitere)  
VALUES (107, 3000, *to\_date*('12.05.2021', 'dd.mm.yyyy'));  
Insert into FACTURA(id\_factura, total\_factura, data\_emitere)  
VALUES (1000, 100, *sysdate*);  
Insert into FACTURA(id\_factura, total\_factura, data\_emitere)  
VALUES (1001, 100, *sysdate*);  
Insert into FACTURA(id\_factura, total\_factura, data\_emitere)  
VALUES (1002, 499, *to\_date*('06.02.2023', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (11, 1500, *to\_date*('25.10.2009', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (12, 4325, *to\_date*('13.11.2009', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (13, 6000, *to\_date*('25.07.2009', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (14, 1400, *to\_date*('25.10.2014', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (15, 4725, *to\_date*('13.04.2013', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (16, 6025, *to\_date*('25.09.2014', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (17, 5000, *to\_date*('13.11.2019', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (18, 7500, *to\_date*('25.09.2019', 'dd.mm.yyyy'));  
INSERT INTO FACTURA(ID\_FACTURA, TOTAL\_FACTURA, DATA\_EMITERE)  
VALUES (19, 9256.35, *to\_date*('12.04.2019', 'dd.mm.yyyy'));  
  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (1, 1, 1);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (2, 3, 2);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (3, 2, 3);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (4, 1, 4);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (5, 5, 5);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (6, 4, 6);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (7, 4, 7);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (8, 5, 8);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (9, 2, 9);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (10, 1, 10);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (104, 1, 11);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (105, 1, 12);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (106, 2, 13);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (107, 2, 13);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar)  
VALUES (1000, 5);  
INSERT INTO FACTURA\_PROPRIETAR(id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (1002, 5, 15);  
  
INSERT INTO FACTURA\_PRODUCATOR(id\_factura, id\_producator)  
VALUES (11, 1);  
INSERT INTO FACTURA\_PRODUCATOR(id\_factura, id\_producator)  
VALUES (12, 2);  
INSERT INTO FACTURA\_PRODUCATOR(id\_factura, id\_producator)  
VALUES (13, 3);  
INSERT INTO FACTURA\_PRODUCATOR(id\_factura, id\_producator)  
VALUES (14, 1);  
INSERT INTO FACTURA\_PRODUCATOR(id\_factura, id\_producator)  
VALUES (15, 2);  
INSERT INTO FACTURA\_PRODUCATOR(id\_factura, id\_producator)  
VALUES (16, 3);  
INSERT INTO FACTURA\_PRODUCATOR(id\_factura, id\_producator)  
VALUES (17, 1);  
INSERT INTO FACTURA\_PRODUCATOR(id\_factura, id\_producator)  
VALUES (18, 2);  
INSERT INTO FACTURA\_PRODUCATOR(id\_factura, id\_producator)  
VALUES (19, 3);  
  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (1, 150, *to\_date*('13.07.2020', 'dd.mm.yyyy'), 1);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (2, 300, *to\_date*('13.07.2011', 'dd.mm.yyyy'), 2);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (3, 300, *to\_date*('13.08.2015', 'dd.mm.yyyy'), 3);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (4, 300, *to\_date*('03.01.2021', 'dd.mm.yyyy'), 4);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (5, 150, *to\_date*('13.07.2017', 'dd.mm.yyyy'), 5);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (6, 75.25, *to\_date*('13.08.2017', 'dd.mm.yyyy'), 5);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (7, 75.75, *to\_date*('13.09.2017', 'dd.mm.yyyy'), 5);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (8, 100, *to\_date*('13.07.2009', 'dd.mm.yyyy'), 6);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (9, 200, *to\_date*('17.08.2009', 'dd.mm.yyyy'), 6);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (10, 300, *to\_date*('13.07.2010', 'dd.mm.yyyy'), 7);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (11, 300, *to\_date*('13.09.2020', 'dd.mm.yyyy'), 8);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (12, 300, *to\_date*('13.12.2016', 'dd.mm.yyyy'), 9);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (13, 300, *to\_date*('15.01.2021', 'dd.mm.yyyy'), 10);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (14, 1500, *to\_date*('25.10.2009', 'dd.mm.yyyy'), 11);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (15, 4325, *to\_date*('25.10.2009', 'dd.mm.yyyy'), 12);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (16, 3000, *to\_date*('25.07.2009', 'dd.mm.yyyy'), 13);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (17, 3000, *to\_date*('25.09.2009', 'dd.mm.yyyy'), 13);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (18, 1400, *to\_date*('25.10.2014', 'dd.mm.yyyy'), 14);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (19, 4725, *to\_date*('13.05.2013', 'dd.mm.yyyy'), 15);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (20, 3025, *to\_date*('25.09.2014', 'dd.mm.yyyy'), 16);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (21, 3000, *to\_date*('01.11.2014', 'dd.mm.yyyy'), 16);  
INSERT INTO CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (22, 4500, *to\_date*('25.10.2019', 'dd.mm.yyyy'), 18);  
insert into CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (104, 300, *to\_date*('13.07.2021', 'dd.mm.yyyy'), 104);  
insert into CHITANTA(id\_chitanta, suma\_platita, data\_plata, id\_factura)  
VALUES (106, 600, *to\_date*('13.07.2021', 'dd.mm.yyyy'), 106);  
  
  
INSERT INTO PRODUS(id\_produs, caracteristici, id\_producator)  
VALUES (1, 'boabe de caini cu carne de vita si somon', 1);  
INSERT INTO PRODUS(id\_produs, caracteristici, id\_producator)  
VALUES (2, 'boabe de pisici cu ulei de peste', 1);  
INSERT INTO PRODUS(id\_produs, caracteristici, id\_producator)  
VALUES (3, 'aparat radiografie', 2);  
INSERT INTO PRODUS(id\_produs, caracteristici, id\_producator)  
VALUES (4, 'stetoscop', 2);  
INSERT INTO PRODUS(id\_produs, caracteristici, id\_producator)  
VALUES (5, 'masa operatii', 2);  
INSERT INTO PRODUS(id\_produs, caracteristici, id\_producator)  
VALUES (6, 'vaccin leptospiroza', 3);  
INSERT INTO PRODUS(id\_produs, caracteristici, id\_producator)  
VALUES (7, 'vaccin hexavalent', 3);  
  
INSERT INTO INVENTAR(id\_inventar, data\_inventar)  
VALUES (1, *to\_date*('05.01.2010', 'dd.mm.yyyy'));  
INSERT INTO INVENTAR(id\_inventar, data\_inventar)  
VALUES (2, *to\_date*('05.01.2015', 'dd.mm.yyyy'));  
INSERT INTO INVENTAR(id\_inventar, data\_inventar)  
VALUES (3, *to\_date*('05.01.2020', 'dd.mm.yyyy'));  
INSERT INTO INVENTAR(id\_inventar, data\_inventar)  
VALUES (4, *to\_date*('05.01.2021', 'dd.mm.yyyy'));  
INSERT INTO INVENTAR(id\_inventar, data\_inventar)  
VALUES (5, *to\_date*('05.01.2022', 'dd.mm.yyyy'));  
  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (1, 1, 11, 10, *to\_date*('05.01.2011', 'dd.mm.yyyy'));  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (2, 1, 12, 15, *to\_date*('05.01.2011', 'dd.mm.yyyy'));  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (1, 2, 13, 20, *to\_date*('05.01.2016', 'dd.mm.yyyy'));  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (2, 2, 14, 25, *to\_date*('05.01.2016', 'dd.mm.yyyy'));  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (1, 3, 15, 35, *to\_date*('05.03.2023', 'dd.mm.yyyy'));  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (2, 3, 16, 30, *to\_date*('05.01.2021', 'dd.mm.yyyy'));  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate)  
VALUES (3, 1, 17, 1);  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate)  
VALUES (3, 2, 18, 1);  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate)  
VALUES (3, 3, 19, 2);  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate)  
VALUES (4, 1, 11, 5);  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate)  
VALUES (4, 2, 12, 10);  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate)  
VALUES (4, 3, 13, 15);  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate)  
VALUES (5, 1, 14, 1);  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate)  
VALUES (5, 2, 15, 1);  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate)  
VALUES (5, 3, 16, 2);  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (6, 1, 17, 100, *to\_date*('13.07.2011', 'dd.mm.yyyy'));  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (7, 1, 18, 150, *to\_date*('13.07.2011', 'dd.mm.yyyy'));  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (6, 2, 19, 100, *to\_date*('13.07.2011', 'dd.mm.yyyy'));  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (7, 2, 11, 150, *to\_date*('13.07.2011', 'dd.mm.yyyy'));  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (6, 3, 12, 234, *to\_date*('13.07.2011', 'dd.mm.yyyy'));  
INSERT INTO PRODUSE\_DIN\_INVENTAR(id\_produs, id\_inventar, id\_factura, cantitate, data\_expirare)  
VALUES (7, 3, 13, 512, *to\_date*('13.07.2011', 'dd.mm.yyyy'));  
  
  
commit;

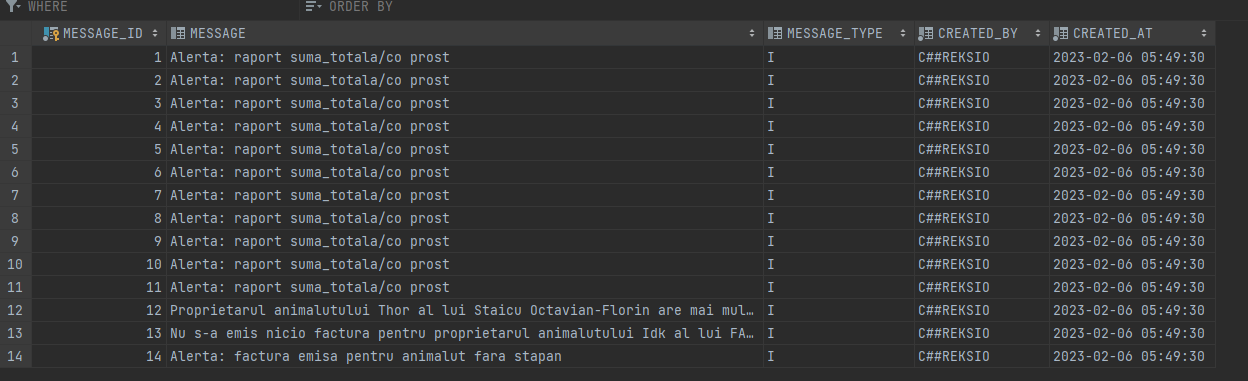


--7 SQL  
  
-- 7.1  
-- afisati pt fiecare contract de angajat, al catelea contract aditional este si ce contract modifica  
SELECT ctr.id\_contract, ctr.id\_angajat, LEVEL depth, PRIOR id\_contract  
FROM (select c.id\_contract, c.id\_contract\_modificat, a.id\_angajat  
 from CONTRACT\_ANGAJAT a,  
 CONTRACT c  
 where c.id\_contract = a.id\_contract) ctr  
START WITH ctr.ID\_CONTRACT\_MODIFICAT IS NULL  
CONNECT BY PRIOR ctr.id\_contract = ctr.id\_contract\_modificat;  
  
-- 7.2  
-- pt fiecare animalut care a fost tratat in anul 2020, afisati nr de consultatii  
select fm.id\_animalut, *count*(\*)  
from ANIMALUT a,  
 FISA\_MEDICALA fm  
where a.id\_animalut = fm.id\_animalut  
 and *TO\_CHAR*(fm.DATA\_FISA, 'YYYY') like '2020'  
group by fm.id\_animalut;  
  
-- 7.3  
-- sa se afiseze pt fiecare animalut, suma total cheltuita pe consultari  
-- a caror factura a fost emisa in aceeasi zi cu consultatia pe numele proprietarului  
-- si numarul acestora  
select a.id\_proprietar, a.id\_animalut, *sum*(f.total\_factura), *count*(\*)  
from FACTURA\_PROPRIETAR fp,  
 FACTURA f,  
 PROPRIETAR p,  
 ANIMALUT a  
where fp.id\_factura = f.id\_factura  
 and fp.id\_proprietar = p.id\_proprietar  
 and p.id\_proprietar = a.id\_proprietar  
 and fp.id\_fisa\_medicala in (select fm.id\_fisa  
 from FISA\_MEDICALA fm  
 where fm.id\_animalut = a.id\_animalut  
 and f.data\_emitere = fm.data\_fisa)  
group by a.id\_proprietar, a.id\_animalut  
order by id\_proprietar;  
  
-- 7.4  
-- afisati fiecare factura ce tip are cu majuscule  
-- daca nu are niciun tip(nu e nici producator, nici proprietar), scrie custom  
with f as (select f.id\_factura, f.data\_emitere, f.total\_factura, 'proprietar' tip  
 from FACTURA f,  
 FACTURA\_PROPRIETAR fp  
 where f.id\_factura = fp.id\_factura  
 union  
 select f.id\_factura, f.data\_emitere, f.total\_factura, 'producator' tip  
 from FACTURA f,  
 FACTURA\_PRODUCATOR fp  
 where f.id\_factura = fp.id\_factura)  
select fi.id\_factura, fi.data\_emitere, fi.total\_factura, *UPPER*(*NVL*(f.tip, 'custom')) tip1  
from f,  
 FACTURA fi  
where fi.id\_factura = f.id\_factura(+)  
order by tip1, fi.total\_factura, fi.data\_emitere;  
  
-- 7.5  
-- afisati toti proprietarii si toti cainii acestora  
-- inclusiv proprietarii fara caini si cainii fara stapan  
select \*  
from PROPRIETAR p  
 FULL OUTER JOIN ANIMALUT a ON p.id\_proprietar = a.id\_proprietar  
order by p.id\_proprietar;  
  
-- 7.6  
-- pt fiecare factura cu suma sub 1000ron, afisati totalul  
-- lor, cate chitante au fost eliberate si suma totala a acestora  
-- precum si proprietarul  
select p.id\_proprietar, f.id\_factura, total\_factura, *count*(id\_chitanta), *sum*(suma\_platita)  
from FACTURA f,  
 CHITANTA c,  
 FACTURA\_PROPRIETAR fp,  
 PROPRIETAR p  
where f.id\_factura = c.id\_factura  
 and fp.id\_factura = f.id\_factura  
 and p.id\_proprietar = fp.id\_proprietar  
having total\_factura < 1000  
group by p.id\_proprietar, f.id\_factura, total\_factura;  
  
-- 7.7  
-- proprietarii care au platite toate facturile (si au minim o factura)  
-- operatorul division cu count  
select p.id\_proprietar  
from PROPRIETAR p,  
 FACTURA f,  
 FACTURA\_PROPRIETAR fp  
where p.id\_proprietar = fp.id\_proprietar  
 and f.id\_factura = fp.id\_factura  
group by p.id\_proprietar  
having *COUNT*(\*) = (select *count*(\*)  
 from FACTURA f1,  
 FACTURA\_PROPRIETAR fp1  
 where f1.id\_factura = fp1.id\_factura  
 and fp1.id\_proprietar = p.id\_proprietar  
 and f1.total\_factura <= (select *sum*(c.suma\_platita)  
 from CHITANTA c  
 where c.id\_factura = f1.id\_factura));  
  
-- 7.8  
-- divison: proprietarii pt care nu exista facturi neplatite  
-- inclusiv cei care nu au facturi  
select p.id\_proprietar  
from PROPRIETAR p  
where NOT *EXISTS*(  
 select 1  
 from FACTURA f,  
 FACTURA\_PROPRIETAR fp  
 where f.id\_factura = fp.id\_factura  
 and fp.id\_proprietar = p.id\_proprietar  
 and not *exists*(  
 select 1  
 from FACTURA f1  
 where f1.id\_factura = f.id\_factura  
 and f.total\_factura >= (select *sum*(c.suma\_platita) from CHITANTA c where c.id\_factura = f1.id\_factura)  
 )  
 );  
  
-- 7.9  
-- sa se afiseze pentru fiecare animalut cate luni au trecut intre prima si ultima fise medicala  
select a.id\_animalut, *round*(*MONTHS\_BETWEEN*(*max*(fm.data\_fisa), *min*(fm.data\_fisa)))  
from ANIMALUT a,  
 FISA\_MEDICALA fm  
where a.id\_animalut = fm.id\_animalut  
group by a.id\_animalut;  
  
-- 7.10  
-- afisati produse care expira luna viitoare din ultimul inventar  
select \*  
from PRODUSE\_DIN\_INVENTAR pi,  
 PRODUS p  
where p.id\_produs = pi.id\_produs  
 and pi.id\_inventar = 3  
 and *TO\_CHAR*(*Add\_months*(*sysdate*, 1), 'yyyy-mm') like *TO\_CHAR*(pi.DATA\_EXPIRARE, 'yyyy-mm');  
  
-- 7.11  
-- afiseaza producatorul cu cea mai mare suma pe care o avem in total  
select id\_producator  
from PRODUCATOR p  
where id\_producator in (select fp.id\_producator  
 from FACTURA\_PRODUCATOR fp,  
 FACTURA f  
 where fp.id\_factura = f.id\_factura  
 group by fp.id\_producator  
 having *sum*(f.TOTAL\_FACTURA) = (select *max*(s)  
 from (select *sum*(f1.total\_factura) s  
 from factura f1,  
 FACTURA\_PRODUCATOR fp1  
 where f1.id\_factura = fp1.id\_factura  
 group by fp1.id\_producator)));   
  


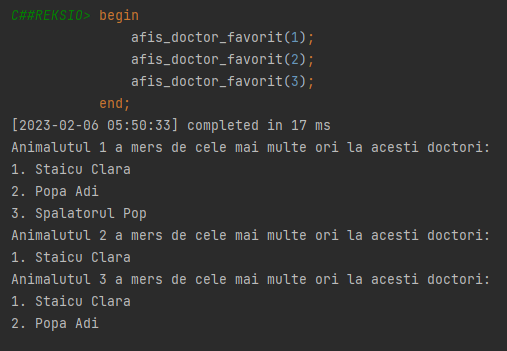
-- 7.12  
-- afisati toti angajatii care au consultat animalutul 1  
select distinct j.id\_angajat, j.nume, j.prenume  
from ANGAJAT j,  
 FISA\_MEDICALA fm,  
 ANIMALUT a  
where a.id\_animalut = fm.id\_animalut  
 and j.id\_angajat = fm.id\_doctor  
 and a.id\_animalut = 1;  
  
-- 7.13  
-- afisati pt fiecare contract de angajat, al catelea in ierarhia manageriala este  
SELECT ctr.id\_angajat, LEVEL depth, PRIOR id\_angajat  
from angajat ctr  
START WITH ctr.id\_sef IS NULL  
CONNECT BY PRIOR ctr.id\_angajat = ctr.id\_sef;  
  
-- 7.14 afisati cate tipuri de vaccin au fost vreodata pe stoc  
select *sum*(*Decode*(*INSTR*(*LOWER*(p.caracteristici), 'vaccin'), 0, 0, 1))  
from produs p;  
  
-- 7.15 returnati null daca boxer e cea mai consultata rasa, altfel rasa  
-- daca sunt mai multe rase cu acelasi nr de tratamente maxim, sa se afiseze null  
with c as (select rasa, *count*(\*) nr  
 from ANIMALUT  
 group by rasa),  
 ans as (select c.rasa from c where c.nr = (select *max*(c2.nr) from c c2))  
select *NULLIF*(  
 case (select *count*(\*) from ans)  
 when 1 then (select \* from ans)  
 else null  
 end, 'Boxer')  
from dual;



-- 8  
CREATE TABLE MESAJE  
(  
 message\_id NUMBER,  
 message VARCHAR2(255),  
 message\_type VARCHAR2(1),  
 created\_by VARCHAR2(40) NOT NULL,  
 created\_at DATE NOT NULL,  
 CONSTRAINT pk\_message PRIMARY KEY (message\_id),  
 CHECK (message\_type = 'E' OR message\_type = 'W' OR message\_type = 'I')  
);  
  
CREATE SEQUENCE seq\_err START WITH 1;

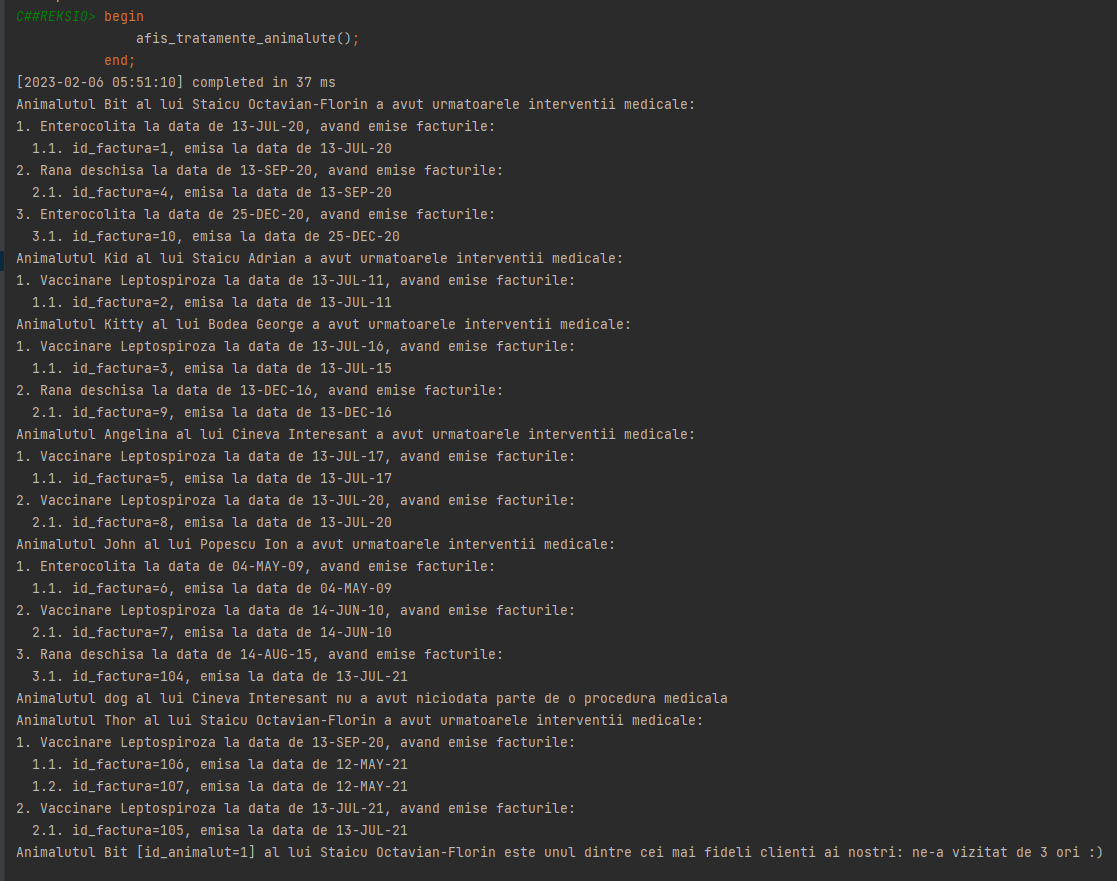


-- 9 PL/SQL  
  
-- 9.1  
-- pentru un animalut dat, afisati doctorii la care acesta a mers cel mai des (de cele mai multe ori)  
-- utilizand doua tipuri de colectii invatate  
CREATE OR REPLACE PROCEDURE *afis\_doctor\_favorit*(id\_animal ANIMALUT.id\_animalut%type)  
 IS  
 TYPE tab\_ind IS TABLE OF PLS\_INTEGER INDEX BY PLS\_INTEGER;  
 TYPE tab\_imb IS TABLE OF FISA\_MEDICALA.id\_doctor%type;  
 cnt tab\_ind := tab\_ind();  
 doctors tab\_imb := tab\_imb();  
 index\_max pls\_integer;  
 id\_doctor pls\_integer;  
 ok binary\_integer := 0;  
 id pls\_integer;  
 num ANGAJAT.nume%type;  
 prenum ANGAJAT.prenume%type;  
 co pls\_integer := 0;  
BEGIN  
 select fm.id\_doctor bulk collect  
 into doctors  
 from FISA\_MEDICALA fm  
 where fm.id\_animalut = id\_animal;  
  
 id\_doctor := doctors.FIRST;  
 loop  
 exit when id\_doctor is null;  
 begin  
 cnt(id\_doctor) := cnt(id\_doctor) + 1;  
 if ok = 1 then  
 if cnt(id\_doctor) > cnt(index\_max) then  
 index\_max := id\_doctor;  
 end if;  
 end if;  
 EXCEPTION  
 when NO\_DATA\_FOUND then  
 cnt(id\_doctor) := 1;  
 if ok = 0 then  
 ok := 1;  
 index\_max := id\_doctor;  
 end if;  
 end;  
 id\_doctor := doctors.*NEXT*(id\_doctor);  
 end loop;  
 if ok = 0 then  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || id\_animal || ' nu are nicio fisa medicala in sistem');  
 else  
 id := cnt.FIRST;  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || id\_animal || ' a mers de cele mai multe ori la acesti doctori:');  
 loop  
 exit when id is null;  
 if cnt(id) = cnt(index\_max) then  
 select a.nume, a.prenume  
 into num, prenum  
 from ANGAJAT a  
 where a.id\_angajat = id;  
 co := co + 1;  
 DBMS\_OUTPUT.*PUT\_LINE*(co || '. ' || num || ' ' || prenum);  
 end if;  
 id := cnt.*NEXT*(id);  
 end loop;  
 end if;  
END;  
/  
  
begin  
 *afis\_doctor\_favorit*(1);  
 *afis\_doctor\_favorit*(2);  
 *afis\_doctor\_favorit*(3);  
end;  
  
/

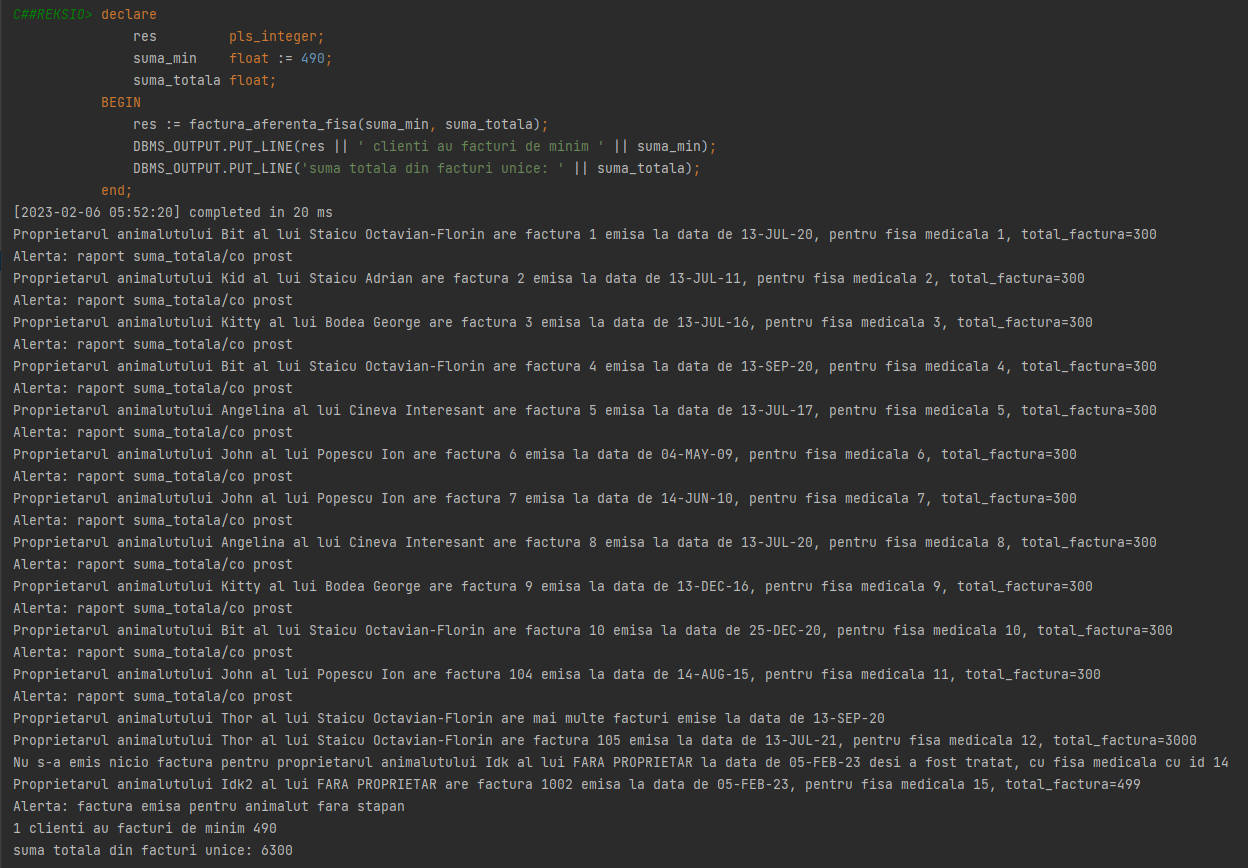


-- 9.2  
-- pentru fiecare animalut cu proprietar, afisati toate fisele medicale ale acestuia, data la care au fost diagnosticati  
-- precum si facturile corespunzatoare acelor fise  
-- utilizand minim 2 tipuri de cursoare invatate, dintre care minim 1 sa fie parametrizat  
CREATE OR REPLACE PROCEDURE *afis\_tratamente\_animalute* IS  
 TYPE refcursor IS REF CURSOR;  
 cursor mycursor(id\_anim IN FISA\_MEDICALA.id\_animalut%type) is select diag.descriere,  
 fm.data\_fisa,  
 cursor (select fp.id\_factura, f.data\_emitere  
 from FACTURA\_PROPRIETAR fp,  
 FACTURA f  
 where fp.id\_factura = f.id\_factura  
 and fp.id\_fisa\_medicala = fm.id\_fisa)  
 from FISA\_MEDICALA fm,  
 DIAGNOSTIC diag  
 where fm.id\_diagnostic = diag.id\_diagnostic  
 and fm.id\_animalut = id\_anim;  
 v\_cursor refcursor;  
 diag DIAGNOSTIC.descriere%type;  
 data\_diagnostic FISA\_MEDICALA.data\_fisa%type;  
 id\_factura FACTURA\_PROPRIETAR.id\_factura%type;  
 data\_factura FACTURA.data\_emitere%type;  
 id\_animalut\_max ANIMALUT.id\_animalut%type;  
 nume\_animalut\_max ANIMALUT.nume%type;  
 nume\_proprietar\_max PROPRIETAR.nume%type;  
 prenume\_proprietar\_max PROPRIETAR.prenume%type;  
 ok\_factura binary\_integer;  
 co\_max pls\_integer := 0;  
 faliment EXCEPTION;  
BEGIN  
 for i in (select a.id\_animalut, a.nume nume\_animalut, p.nume nume\_proprietar, p.prenume prenume\_proprietar  
 from ANIMALUT a,  
 PROPRIETAR p  
 where a.id\_proprietar = p.id\_proprietar)  
 loop  
 open mycursor(i.id\_animalut);  
 fetch mycursor into diag, data\_diagnostic, v\_cursor;  
 if mycursor%notfound then  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || i.nume\_animalut || ' al lui ' || i.nume\_proprietar || ' ' ||  
 i.prenume\_proprietar || ' nu a avut niciodata parte de o procedura medicala');  
 DBMS\_OUTPUT.*NEW\_LINE*();  
 else  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || i.nume\_animalut || ' al lui ' || i.nume\_proprietar || ' ' ||  
 i.prenume\_proprietar || ' a avut urmatoarele interventii medicale:');  
 loop  
 exit when mycursor%notfound;  
 DBMS\_OUTPUT.*PUT\_LINE*(mycursor%rowcount || '. ' || diag || ' la data de ' || data\_diagnostic ||  
 ', avand emise facturile:');  
 ok\_factura := 0;  
 loop  
 fetch v\_cursor into id\_factura, data\_factura;  
 exit when v\_cursor%notfound;  
 ok\_factura := 1;  
 DBMS\_OUTPUT.*PUT\_LINE*(' ' || mycursor%rowcount || '.' || v\_cursor%rowcount || '. id\_factura=' ||  
 id\_factura || ', emisa la data de ' || data\_factura);  
 end loop;  
  
 if ok\_factura = 0 then  
 DBMS\_OUTPUT.*PUT\_LINE*(' Nicio factura emisa pentru aceasta fisa medicala');  
 end if;  
  
 fetch mycursor into diag, data\_diagnostic, v\_cursor;  
 end loop;  
 DBMS\_OUTPUT.*NEW\_LINE*();  
  
 if mycursor%rowcount > co\_max then  
 co\_max := mycursor%rowcount;  
 id\_animalut\_max := i.id\_animalut;  
 nume\_animalut\_max := i.nume\_animalut;  
 nume\_proprietar\_max := i.nume\_proprietar;  
 prenume\_proprietar\_max := i.prenume\_proprietar;  
 end if;  
 end if;  
 close mycursor;  
 end loop;  
 if co\_max = 0 then  
 raise faliment;  
 end if;  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || nume\_animalut\_max || ' [id\_animalut=' || id\_animalut\_max || '] al lui ' ||  
 nume\_proprietar\_max || ' ' || prenume\_proprietar\_max ||  
 ' este unul dintre cei mai fideli clienti ai nostri: ne-a vizitat de ' || co\_max || ' ori :)');  
EXCEPTION  
 when faliment then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu avem niciun animalut care s-a tratat la noi...');  
 insert into mesaje(message\_id, message, message\_type, created\_by, created\_at)  
 values (seq\_err.NEXTVAL, 'Nu avem niciun animalut care s-a tratat la noi...', 'I', *USER*, *sysdate*);  
 commit;  
END;  
/  
  
begin  
 *afis\_tratamente\_animalute*();  
end;  
/

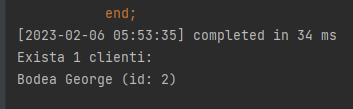
-- pt a da trigger la exceptia faliment  
-- drop table FISA\_MEDICALA cascade constraints purge  
-- apoi re create table FISA\_MEDICALA



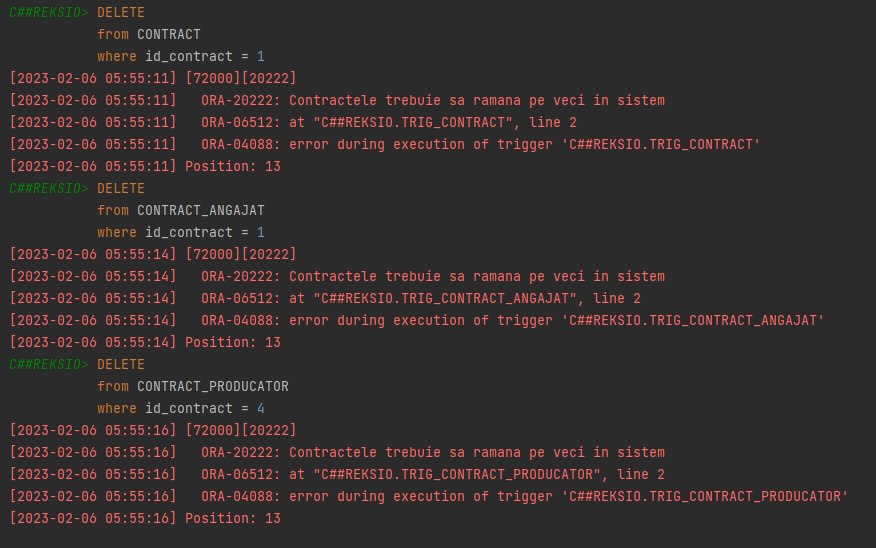
-- 9.3  
-- pentru fiecare fisa medicala a fiecarui animalut, afisati factura corespunzatoare si suma acesteia.  
-- Pt o suma minima primita ca parametru, returnati cate facturi au minim aceasta suma, iar prin parametru  
-- intoarceti suma tuturor facturilor. Daca sunt mai multe facturi pe aceeasi fisa medicala, ignorati-le si  
-- afisati un mesaj de alerta. Afisati de asemenea mesaje de alerta daca la fiecare moment in care raportul  
-- dintre suma totala a facturilor si nr acestora este mai mica decat suma minima primita ca parametru.  
-- Afisati totodata si un mesaj de alerta in caz ca se gasesc facturi emise pentru un animalut fara stapan  
-- (probabil ca a fost emisa pentru o persoana care a gasit un catelus pe strada ce avea nevoie de ingrijire,  
-- dar care nu a putut sa il adopte),  
-- utilizand minim o comanda sql care sa foloseasca 3 din tabelele definite si definind minim 2 exceptii  
CREATE OR REPLACE FUNCTION *factura\_aferenta\_fisa*(suma\_min IN float, suma\_totala OUT float) return pls\_integer  
 IS  
 id FACTURA\_PROPRIETAR.id\_factura%type;  
 pret FACTURA.total\_factura%type;  
 co pls\_integer := 0;  
 co\_tot pls\_integer := 0;  
 factura\_fara\_stapan EXCEPTION;  
 raport\_pret\_consultatii\_prost EXCEPTION;  
BEGIN  
 suma\_totala := 0;  
 for i in (select p.id\_proprietar, p.nume propnum, p.prenume, a.nume animnume, fm.data\_fisa, fm.id\_fisa  
 from FISA\_MEDICALA fm,  
 ANIMALUT a,  
 PROPRIETAR p  
 where fm.id\_animalut = a.id\_animalut  
 and a.id\_proprietar = p.id\_proprietar(+))  
 loop  
 BEGIN  
 select fp.id\_factura, f.total\_factura  
 into id, pret  
 from FACTURA\_PROPRIETAR fp,  
 FACTURA f  
 where fp.id\_factura = f.id\_factura  
 and i.id\_fisa = fp.id\_fisa\_medicala;  
 DBMS\_OUTPUT.*PUT\_LINE*('Proprietarul animalutului ' || i.animnume || ' al lui ' ||  
 *nvl*(i.propnum, 'FARA') || ' ' || *nvl*(i.prenume, 'PROPRIETAR') || ' are factura ' ||  
 id || ' emisa la data de ' || i.data\_fisa || ', pentru fisa medicala ' ||  
 i.id\_fisa || ', total\_factura=' || pret);  
  
 if i.id\_proprietar is null then  
 raise factura\_fara\_stapan;  
 end if;  
  
 suma\_totala := suma\_totala + pret;  
 co\_tot := co\_tot + 1;  
 if pret >= suma\_min then  
 co := co + 1;  
 end if;  
  
 if suma\_totala / co\_tot < suma\_min then  
 raise raport\_pret\_consultatii\_prost;  
 end if;  
 EXCEPTION  
 when NO\_DATA\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu s-a emis nicio factura pentru proprietarul animalutului ' || i.animnume ||  
 ' al lui ' || *nvl*(i.propnum, 'FARA') || ' ' || *nvl*(i.prenume, 'PROPRIETAR') ||  
 ' la data de ' || i.data\_fisa ||  
 ' desi a fost tratat, cu fisa medicala cu id ' || i.id\_fisa);  
 insert into mesaje(message\_id, message, message\_type, created\_by, created\_at)  
 values (seq\_err.NEXTVAL,  
 'Nu s-a emis nicio factura pentru proprietarul animalutului ' || i.animnume ||  
 ' al lui ' || *nvl*(i.propnum, 'FARA') || ' ' || *nvl*(i.prenume, 'PROPRIETAR') ||  
 ' la data de ' || i.data\_fisa ||  
 ' desi a fost tratat, cu fisa medicala cu id ' || i.id\_fisa, 'I', *USER*, *sysdate*);  
 when TOO\_MANY\_ROWS then  
 DBMS\_OUTPUT.*PUT\_LINE*('Proprietarul animalutului ' || i.animnume || ' al lui ' ||  
 *nvl*(i.propnum, 'FARA') || ' ' || *nvl*(i.prenume, 'PROPRIETAR') ||  
 ' are mai multe facturi emise la data de ' || i.data\_fisa);  
 insert into mesaje(message\_id, message, message\_type, created\_by, created\_at)  
 values (seq\_err.NEXTVAL, 'Proprietarul animalutului ' || i.animnume || ' al lui ' ||  
 *nvl*(i.propnum, 'FARA') || ' ' || *nvl*(i.prenume, 'PROPRIETAR') ||  
 ' are mai multe facturi emise la data de ' || i.data\_fisa, 'I', *USER*,  
 *sysdate*);  
 when factura\_fara\_stapan then  
 DBMS\_OUTPUT.*PUT\_LINE*('Alerta: factura emisa pentru animalut fara stapan');  
 insert into mesaje(message\_id, message, message\_type, created\_by, created\_at)  
 values (seq\_err.NEXTVAL, 'Alerta: factura emisa pentru animalut fara stapan', 'I', *USER*, *sysdate*);  
 when raport\_pret\_consultatii\_prost then  
 DBMS\_OUTPUT.*PUT\_LINE*('Alerta: raport suma\_totala/co prost');  
 insert into mesaje(message\_id, message, message\_type, created\_by, created\_at)  
 values (seq\_err.NEXTVAL, 'Alerta: raport suma\_totala/co prost', 'I', *USER*, *sysdate*);  
 END;  
 end loop;  
 commit;  
 return co;  
end;  
/  
  
declare  
 res pls\_integer;  
 suma\_min float := 490;  
 suma\_totala float;  
BEGIN  
 res := *factura\_aferenta\_fisa*(suma\_min, suma\_totala);  
 DBMS\_OUTPUT.*PUT\_LINE*(res || ' clienti au facturi de minim ' || suma\_min);  
 DBMS\_OUTPUT.*PUT\_LINE*('suma totala din facturi unice: ' || suma\_totala);  
end;  
/



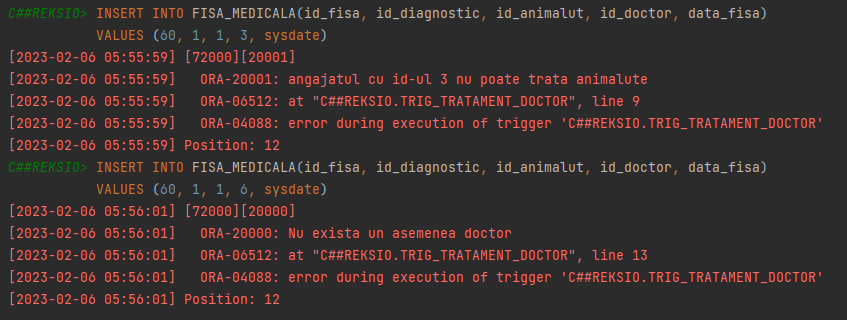
--9 suplimentar (folosit in pcahetul 2)  
--Intoarcem id-ul clientilor cu cea mai mare suma achitata (nu datorata) prin chitante  
--Daca sunt mai multi, ii intoarcem pe toti  
CREATE or REPLACE TYPE tab\_imb is TABLE OF NUMBER(10);  
/  
  
CREATE or REPLACE FUNCTION *topclienti*(top\_clienti out tab\_imb) return pls\_integer  
 IS  
 TYPE refcursor IS REF CURSOR;  
 cursor mycursor is select f.id\_proprietar,  
 cursor (select c.suma\_platita  
 from CHITANTA c  
 where c.id\_factura = f.id\_factura)  
 from FACTURA\_PROPRIETAR f,  
 PROPRIETAR p  
 where f.id\_proprietar = p.id\_proprietar  
 order by p.id\_proprietar;  
 v\_cursor refcursor;  
 id FACTURA\_PROPRIETAR.id\_factura%type := null;  
 suma pls\_integer := 0;  
 total\_suma pls\_integer := 0;  
 max\_suma pls\_integer := 0;  
 last\_id PROPRIETAR.id\_proprietar%type := null;  
 co pls\_integer := 0;  
BEGIN  
 top\_clienti := tab\_imb();  
 open mycursor;  
 loop  
 fetch mycursor into id, v\_cursor;  
 if last\_id is null then  
 last\_id := id;  
 else  
 if mycursor%notfound or id <> last\_id then  
 if total\_suma > max\_suma then  
 top\_clienti.DELETE;  
 top\_clienti.EXTEND;  
 co := 1;  
 top\_clienti(1) := last\_id;  
 max\_suma := total\_suma;  
 elsif total\_suma = max\_suma then  
 top\_clienti.EXTEND;  
 co := co + 1;  
 top\_clienti(co) := last\_id;  
 end if;  
 last\_id := id;  
 total\_suma := 0;  
 end if;  
 end if;  
 exit when mycursor%notfound;  
 loop  
 fetch v\_cursor into suma;  
 exit when v\_cursor%notfound;  
 total\_suma := total\_suma + suma;  
 end loop;  
 end loop;  
 close mycursor;  
 return co;  
  
END;  
/  
--functia returneaza nr de clienti, iar parametrul intoarce colectia  
--vom scrie un bloc anonim mai complex, pt a verifica corectitudinea functiei  
declare  
 co pls\_integer;  
 id NUMBER(10);  
 mytab tab\_imb;  
 num PROPRIETAR.nume%type;  
 prenum PROPRIETAR.prenume%type;  
begin  
 co := *TOPCLIENTI*(mytab);  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista ' || co || ' clienti:');  
  
 id := mytab.FIRST;  
 loop  
 exit when id is null;  
 select nume, prenume  
 into num, prenum  
 from PROPRIETAR  
 where id\_proprietar = mytab(id);  
 DBMS\_OUTPUT.*PUT\_LINE*(num || ' ' || prenum || ' (id: ' || mytab(id) || ')');  
 id := mytab.*NEXT*(id);  
 end loop;  
end;  
/



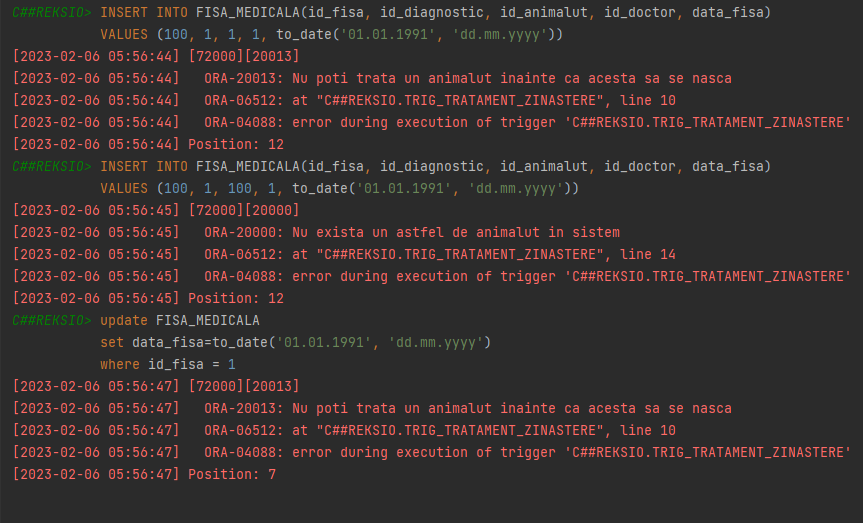
-- 9.4 trigger lmd la nivel de comanda  
CREATE OR REPLACE TRIGGER trig\_contract  
 BEFORE DELETE  
 on CONTRACT  
BEGIN  
 *RAISE\_APPLICATION\_ERROR*(-20222, 'Contractele trebuie sa ramana pe veci in sistem');  
END;  
/  
  
CREATE OR REPLACE TRIGGER trig\_contract\_angajat  
 BEFORE DELETE  
 on CONTRACT\_ANGAJAT  
BEGIN  
 *RAISE\_APPLICATION\_ERROR*(-20222, 'Contractele trebuie sa ramana pe veci in sistem');  
END;  
/  
  
CREATE OR REPLACE TRIGGER trig\_contract\_producator  
 BEFORE DELETE  
 on CONTRACT\_PRODUCATOR  
BEGIN  
 *RAISE\_APPLICATION\_ERROR*(-20222, 'Contractele trebuie sa ramana pe veci in sistem');  
END;  
/  
  
DELETE  
from CONTRACT  
where id\_contract = 1;  
DELETE  
from CONTRACT\_ANGAJAT  
where id\_contract = 1;  
DELETE  
from CONTRACT\_PRODUCATOR  
where id\_contract = 4;



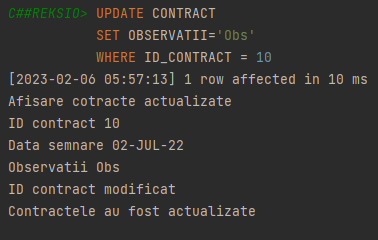
-- 9.5 triggeri lmd la nivel de linie  
CREATE OR REPLACE TRIGGER trig\_tratament\_doctor  
 BEFORE INSERT OR UPDATE of id\_doctor  
 on FISA\_MEDICALA  
 FOR EACH ROW  
DECLARE  
 ok ANGAJAT.trateaza\_animale%type;  
BEGIN  
 select trateaza\_animale  
 into ok  
 from ANGAJAT  
 where id\_angajat = :NEW.id\_doctor;  
 if ok <> 1 then  
 *RAISE\_APPLICATION\_ERROR*(-20001, 'angajatul cu id-ul ' || :NEW.id\_doctor || ' nu poate trata animalute');  
 end if;  
EXCEPTION  
 when NO\_DATA\_FOUND then  
 *RAISE\_APPLICATION\_ERROR*(-20000, 'Nu exista un asemenea doctor');  
end;  
/  
  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa)  
VALUES (60, 1, 1, 3, *sysdate*);  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa)  
VALUES (60, 1, 1, 6, *sysdate*);



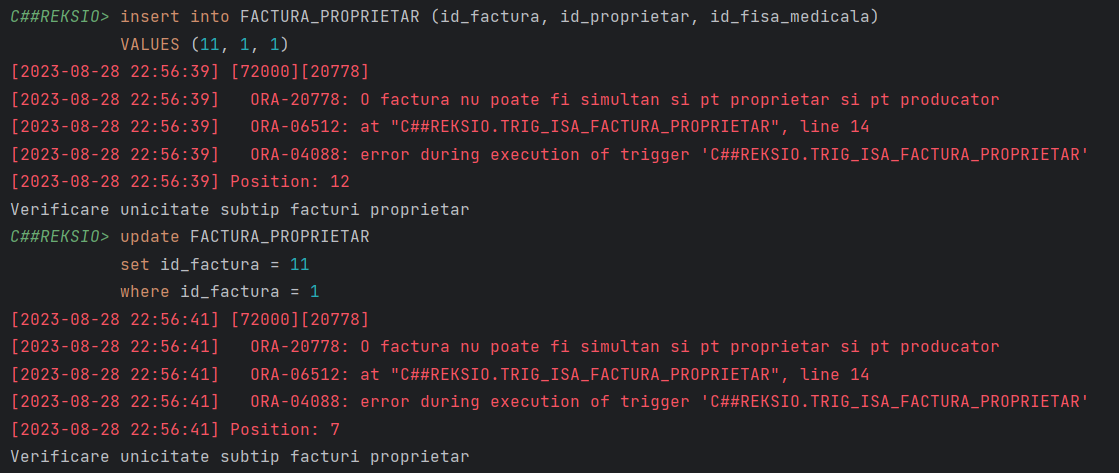
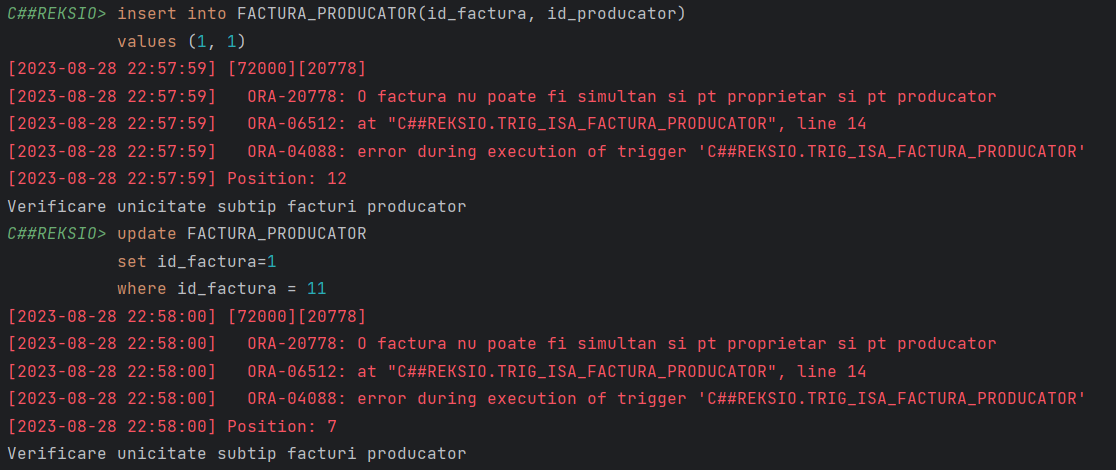
CREATE OR REPLACE TRIGGER trig\_tratament\_zinastere  
 BEFORE INSERT OR UPDATE  
 on FISA\_MEDICALA  
 FOR EACH ROW  
DECLARE  
 zinastere ANIMALUT.zi\_nastere%type;  
BEGIN  
 select zi\_nastere  
 into zinastere  
 from ANIMALUT  
 where id\_animalut = :NEW.id\_animalut;  
  
 if :NEW.data\_fisa < zinastere then  
 *RAISE\_APPLICATION\_ERROR*(-20013, 'Nu poti trata un animalut inainte ca acesta sa se nasca');  
 end if;  
EXCEPTION  
 when NO\_DATA\_FOUND then  
 *RAISE\_APPLICATION\_ERROR*(-20000, 'Nu exista un astfel de animalut in sistem');  
end;  
/  
  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa)  
VALUES (100, 1, 1, 1, *to\_date*('01.01.1991', 'dd.mm.yyyy'));  
INSERT INTO FISA\_MEDICALA(id\_fisa, id\_diagnostic, id\_animalut, id\_doctor, data\_fisa)  
VALUES (100, 1, 100, 1, *to\_date*('01.01.1991', 'dd.mm.yyyy'));  
update FISA\_MEDICALA  
set data\_fisa=*to\_date*('01.01.1991', 'dd.mm.yyyy')  
where id\_fisa = 1;



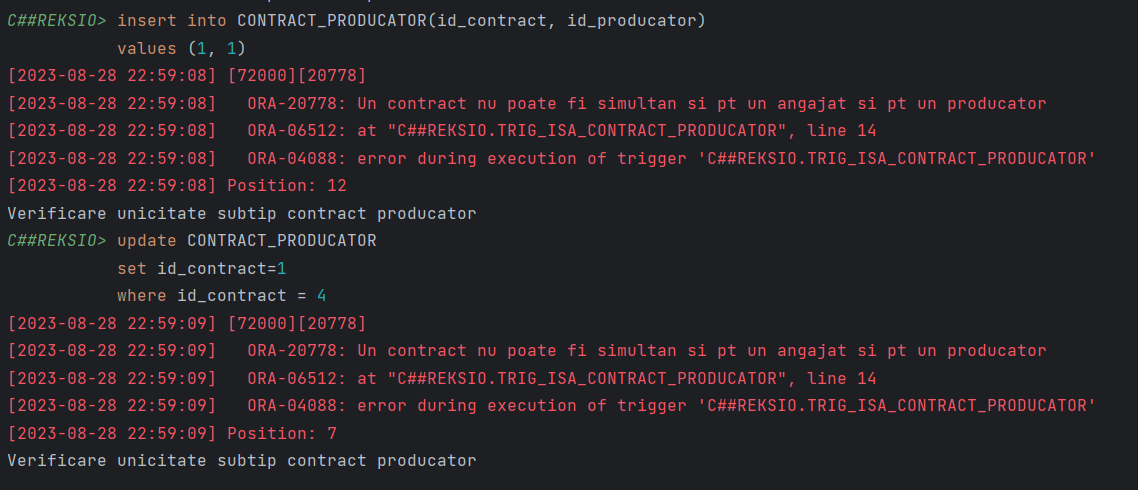
-- compound trigger  
CREATE OR REPLACE TRIGGER trig\_update\_contracte  
 FOR UPDATE  
 ON CONTRACT COMPOUND TRIGGER  
  
BEFORE STATEMENT IS  
BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('Afisare cotracte actualizate');  
END BEFORE STATEMENT;  
  
 AFTER EACH ROW IS  
 BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('ID contract ' || :new.id\_contract);  
 DBMS\_OUTPUT.*PUT\_LINE*('Data semnare ' || :new.data\_semnare);  
 DBMS\_OUTPUT.*PUT\_LINE*('Observatii ' || :new.observatii);  
 DBMS\_OUTPUT.*PUT\_LINE*('ID contract modificat ' || :new.id\_contract\_modificat);  
 END AFTER EACH ROW;  
  
 AFTER STATEMENT IS  
 BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('Contractele au fost actualizate');  
 END AFTER STATEMENT;  
 END trig\_update\_contracte;  
/  
  
UPDATE CONTRACT  
SET OBSERVATII='Obs'  
WHERE ID\_CONTRACT = 10;



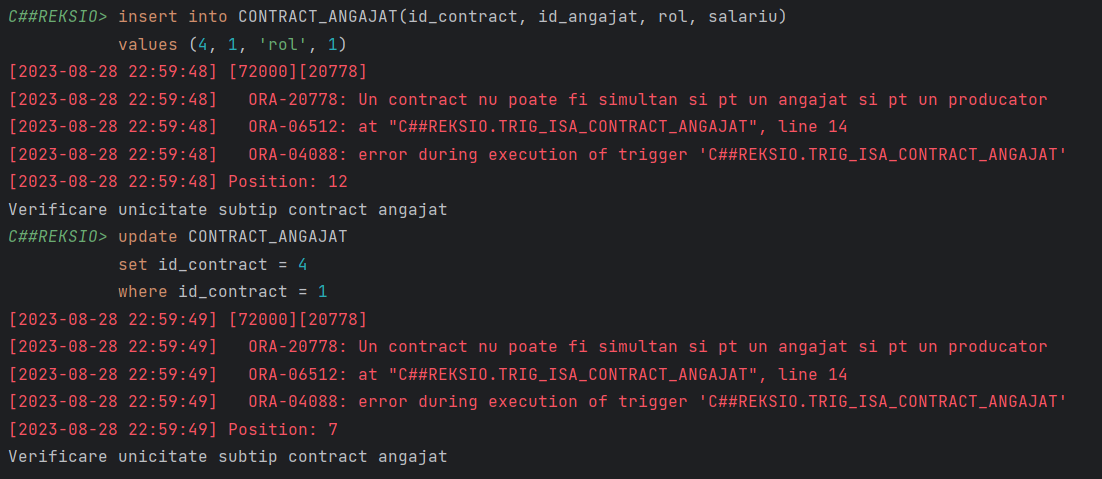
CREATE OR REPLACE TRIGGER trig\_isa\_factura\_proprietar  
 FOR INSERT OR UPDATE  
 ON FACTURA\_PROPRIETAR COMPOUND TRIGGER  
  
BEFORE STATEMENT IS  
BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('Verificare unicitate subtip facturi proprietar');  
END BEFORE STATEMENT;  
 BEFORE EACH ROW IS  
 Begin  
 declare  
 co pls\_integer := 0;  
 begin  
 select *count*(\*) into co from FACTURA\_PRODUCATOR fp where fp.id\_factura = :NEW.id\_factura;  
 if co > 0 then  
 *RAISE\_APPLICATION\_ERROR*(-20778, 'O factura nu poate fi simultan si pt proprietar si pt producator');  
 end if;  
 end;  
 end before each row;  
  
 AFTER EACH ROW IS  
 BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('ID factura ' || :new.id\_factura || ' este acum de tip proprietar');  
 END AFTER EACH  
 ROW;  
  
 AFTER STATEMENT IS  
 BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('Facturile au fost actualizare');  
 END AFTER STATEMENT;  
 END trig\_isa\_factura\_proprietar;  
/  
  
insert into FACTURA\_PROPRIETAR (id\_factura, id\_proprietar, id\_fisa\_medicala)  
VALUES (11, 1, 1);  
update FACTURA\_PROPRIETAR  
set id\_factura = 11  
where id\_factura = 1;

  
  
CREATE OR REPLACE TRIGGER trig\_isa\_factura\_producator  
 FOR INSERT OR UPDATE  
 ON FACTURA\_PRODUCATOR COMPOUND TRIGGER  
  
BEFORE STATEMENT IS  
BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('Verificare unicitate subtip facturi producator');  
END BEFORE STATEMENT;  
 BEFORE EACH ROW IS  
 Begin  
 declare  
 co pls\_integer := 0;  
 begin  
 select *count*(\*) into co from FACTURA\_PROPRIETAR fp where fp.id\_factura = :NEW.id\_factura;  
 if co > 0 then  
 *RAISE\_APPLICATION\_ERROR*(-20778, 'O factura nu poate fi simultan si pt proprietar si pt producator');  
 end if;  
 end;  
 end before each row;  
  
 AFTER EACH ROW IS  
 BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('ID factura ' || :new.id\_factura || ' este acum de tip producator');  
 END AFTER EACH  
 ROW;  
  
 AFTER STATEMENT IS  
 BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('Facturile au fost actualizare');  
 END AFTER STATEMENT;  
 END trig\_isa\_factura\_producator;  
/  
  
insert into FACTURA\_PRODUCATOR(id\_factura, id\_producator)  
values (1, 1);  
update FACTURA\_PRODUCATOR  
set id\_factura=1  
where id\_factura = 11;  
  


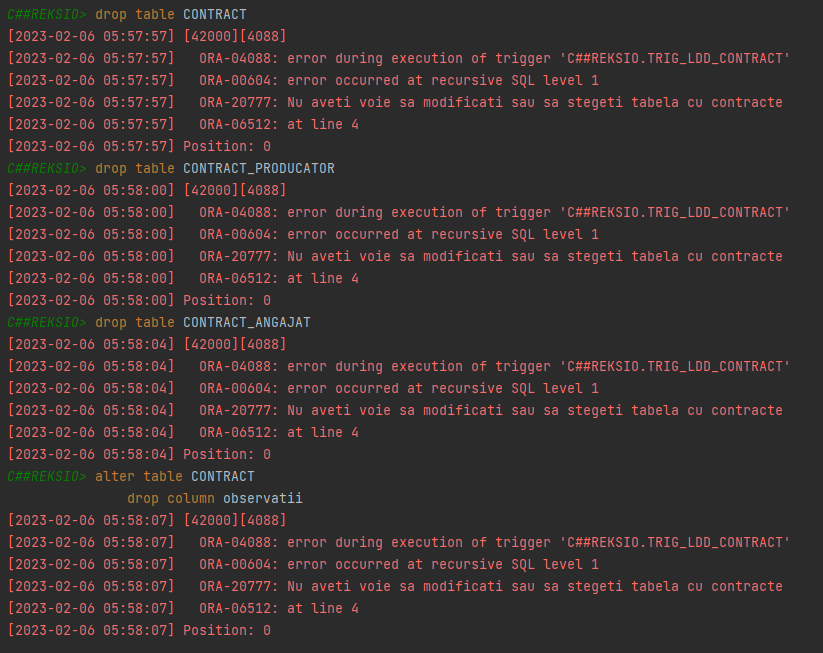
CREATE OR REPLACE TRIGGER trig\_isa\_contract\_producator  
 FOR INSERT OR UPDATE  
 ON CONTRACT\_PRODUCATOR COMPOUND TRIGGER  
  
BEFORE STATEMENT IS  
BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('Verificare unicitate subtip contract producator');  
END BEFORE STATEMENT;  
 BEFORE EACH ROW IS  
 Begin  
 declare  
 co pls\_integer := 0;  
 begin  
 select *count*(\*) into co from CONTRACT\_ANGAJAT cp where cp.id\_contract = :NEW.id\_contract;  
 if co > 0 then  
 *RAISE\_APPLICATION\_ERROR*(-20778,  
 'Un contract nu poate fi simultan si pt un angajat si pt un producator');  
 end if;  
 end;  
 end before each row;  
  
 AFTER EACH ROW IS  
 BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('ID contract ' || :new.id\_contract || ' este acum de tip producator');  
 END AFTER EACH  
 ROW;  
  
 AFTER STATEMENT IS  
 BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('Contractele au fost actualizare');  
 END AFTER STATEMENT;  
 END trig\_isa\_contract\_producator;  
/  
  
insert into CONTRACT\_PRODUCATOR(id\_contract, id\_producator)  
values (1, 1);  
update CONTRACT\_PRODUCATOR  
set id\_contract=1  
where id\_contract = 4;

  
  
CREATE OR REPLACE TRIGGER trig\_isa\_contract\_angajat  
 FOR INSERT OR UPDATE  
 ON CONTRACT\_ANGAJAT COMPOUND TRIGGER  
  
BEFORE STATEMENT IS  
BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('Verificare unicitate subtip contract angajat');  
END BEFORE STATEMENT;  
 BEFORE EACH ROW IS  
 Begin  
 declare  
 co pls\_integer := 0;  
 begin  
 select *count*(\*) into co from CONTRACT\_PRODUCATOR cp where cp.id\_contract = :NEW.id\_contract;  
 if co > 0 then  
 *RAISE\_APPLICATION\_ERROR*(-20778,  
 'Un contract nu poate fi simultan si pt un angajat si pt un producator');  
 end if;  
 end;  
 end before each row;  
  
 AFTER EACH ROW IS  
 BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('ID contract ' || :new.id\_contract || ' este acum de tip angajat');  
 END AFTER EACH  
 ROW;  
  
 AFTER STATEMENT IS  
 BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('Contractele au fost actualizare');  
 END AFTER STATEMENT;  
 END trig\_isa\_contract\_angajat;  
/

insert into CONTRACT\_ANGAJAT(id\_contract, id\_angajat, rol, salariu)  
values (4, 1, 'rol', 1);  
update CONTRACT\_ANGAJAT  
set id\_contract = 4  
where id\_contract = 1;



-- 9.6 trigger ldd  
CREATE OR REPLACE TRIGGER trig\_ldd\_contract  
 BEFORE DROP OR ALTER  
 on schema  
begin  
 if *lower*(ORA\_DICT\_OBJ\_NAME) = *lower*('Contract') or *lower*(ORA\_DICT\_OBJ\_NAME) = *lower*('Contract\_angajat') or  
 *lower*(ORA\_DICT\_OBJ\_NAME) = *lower*('Contract\_producator') then  
 *RAISE\_APPLICATION\_ERROR*(-20777, 'Nu aveti voie sa modificati sau sa stegeti tabela cu contracte');  
 end if;  
end;  
/  
  
drop table CONTRACT;  
drop table CONTRACT\_PRODUCATOR;  
drop table CONTRACT\_ANGAJAT;  
alter table CONTRACT  
 drop column observatii;



-- 9.7 pachet  
CREATE OR REPLACE PACKAGE package\_clinica\_vet\_tav13  
AS  
 PROCEDURE *afis\_doctor\_favorit*(id\_animal ANIMALUT.id\_animalut%type);  
 PROCEDURE *afis\_tratamente\_animalute*;  
 FUNCTION *factura\_aferenta\_fisa*(suma\_min IN float, suma\_totala OUT float) return pls\_integer;  
 TYPE tab\_imb is TABLE OF NUMBER(10);  
 FUNCTION *topclienti*(top\_clienti out tab\_imb) return pls\_integer;  
END package\_clinica\_vet\_tav13;  
/  
  
CREATE OR REPLACE PACKAGE BODY package\_clinica\_vet\_tav13  
AS  
 PROCEDURE *afis\_doctor\_favorit*(id\_animal ANIMALUT.id\_animalut%type)  
 IS  
 TYPE tab\_ind IS TABLE OF PLS\_INTEGER INDEX BY PLS\_INTEGER;  
 TYPE tab\_imb IS TABLE OF FISA\_MEDICALA.id\_doctor%type;  
 cnt tab\_ind := tab\_ind();  
 doctors tab\_imb := tab\_imb();  
 index\_max pls\_integer;  
 id\_doctor pls\_integer;  
 ok binary\_integer := 0;  
 id pls\_integer;  
 num ANGAJAT.nume%type;  
 prenum ANGAJAT.prenume%type;  
 co pls\_integer := 0;  
 BEGIN  
 select fm.id\_doctor bulk collect  
 into doctors  
 from FISA\_MEDICALA fm  
 where fm.id\_animalut = id\_animal;  
  
 id\_doctor := doctors.FIRST;  
 loop  
 exit when id\_doctor is null;  
 begin  
 cnt(id\_doctor) := cnt(id\_doctor) + 1;  
 if ok = 1 then  
 if cnt(id\_doctor) > cnt(index\_max) then  
 index\_max := id\_doctor;  
 end if;  
 end if;  
 EXCEPTION  
 when NO\_DATA\_FOUND then  
 cnt(id\_doctor) := 1;  
 if ok = 0 then  
 ok := 1;  
 index\_max := id\_doctor;  
 end if;  
 end;  
 id\_doctor := doctors.*NEXT*(id\_doctor);  
 end loop;  
 if ok = 0 then  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || id\_animal || ' nu are nicio fisa medicala in sistem');  
 else  
 id := cnt.FIRST;  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || id\_animal || ' a mers de cele mai multe ori la acesti doctori:');  
 loop  
 exit when id is null;  
 if cnt(id) = cnt(index\_max) then  
 select a.nume, a.prenume  
 into num, prenum  
 from ANGAJAT a  
 where a.id\_angajat = id;  
 co := co + 1;  
 DBMS\_OUTPUT.*PUT\_LINE*(co || '. ' || num || ' ' || prenum);  
 end if;  
 id := cnt.*NEXT*(id);  
 end loop;  
 end if;  
 END *afis\_doctor\_favorit*;  
  
 PROCEDURE *afis\_tratamente\_animalute* IS  
 TYPE refcursor IS REF CURSOR;  
 cursor mycursor(id\_anim IN FISA\_MEDICALA.id\_animalut%type) is select diag.descriere,  
 fm.data\_fisa,  
 cursor (select fp.id\_factura, f.data\_emitere  
 from FACTURA\_PROPRIETAR fp,  
 FACTURA f  
 where fp.id\_factura = f.id\_factura  
 and fp.id\_fisa\_medicala = fm.id\_fisa)  
 from FISA\_MEDICALA fm,  
 DIAGNOSTIC diag  
 where fm.id\_diagnostic = diag.id\_diagnostic  
 and fm.id\_animalut = id\_anim;  
 v\_cursor refcursor;  
 diag DIAGNOSTIC.descriere%type;  
 data\_diagnostic FISA\_MEDICALA.data\_fisa%type;  
 id\_factura FACTURA\_PROPRIETAR.id\_factura%type;  
 data\_factura FACTURA.data\_emitere%type;  
 id\_animalut\_max ANIMALUT.id\_animalut%type;  
 nume\_animalut\_max ANIMALUT.nume%type;  
 nume\_proprietar\_max PROPRIETAR.nume%type;  
 prenume\_proprietar\_max PROPRIETAR.prenume%type;  
 ok\_factura binary\_integer;  
 co\_max pls\_integer := 0;  
 faliment EXCEPTION;  
 BEGIN  
 for i in (select a.id\_animalut, a.nume nume\_animalut, p.nume nume\_proprietar, p.prenume prenume\_proprietar  
 from ANIMALUT a,  
 PROPRIETAR p  
 where a.id\_proprietar = p.id\_proprietar)  
 loop  
 open mycursor(i.id\_animalut);  
 fetch mycursor into diag, data\_diagnostic, v\_cursor;  
 if mycursor%notfound then  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || i.nume\_animalut || ' al lui ' || i.nume\_proprietar || ' ' ||  
 i.prenume\_proprietar || ' nu a avut niciodata parte de o procedura medicala');  
 DBMS\_OUTPUT.*NEW\_LINE*();  
 else  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || i.nume\_animalut || ' al lui ' || i.nume\_proprietar || ' ' ||  
 i.prenume\_proprietar || ' a avut urmatoarele interventii medicale:');  
 loop  
 exit when mycursor%notfound;  
 DBMS\_OUTPUT.*PUT\_LINE*(mycursor%rowcount || '. ' || diag || ' la data de ' || data\_diagnostic ||  
 ', avand emise facturile:');  
 ok\_factura := 0;  
 loop  
 fetch v\_cursor into id\_factura, data\_factura;  
 exit when v\_cursor%notfound;  
 ok\_factura := 1;  
 DBMS\_OUTPUT.*PUT\_LINE*(' ' || mycursor%rowcount || '.' || v\_cursor%rowcount ||  
 '. id\_factura=' ||  
 id\_factura || ', emisa la data de ' || data\_factura);  
 end loop;  
  
 if ok\_factura = 0 then  
 DBMS\_OUTPUT.*PUT\_LINE*(' Nicio factura emisa pentru aceasta fisa medicala');  
 end if;  
  
 fetch mycursor into diag, data\_diagnostic, v\_cursor;  
 end loop;  
 DBMS\_OUTPUT.*NEW\_LINE*();  
  
 if mycursor%rowcount > co\_max then  
 co\_max := mycursor%rowcount;  
 id\_animalut\_max := i.id\_animalut;  
 nume\_animalut\_max := i.nume\_animalut;  
 nume\_proprietar\_max := i.nume\_proprietar;  
 prenume\_proprietar\_max := i.prenume\_proprietar;  
 end if;  
 end if;  
 close mycursor;  
 end loop;  
 if co\_max = 0 then  
 raise faliment;  
 end if;  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || nume\_animalut\_max || ' [id\_animalut=' || id\_animalut\_max || '] al lui ' ||  
 nume\_proprietar\_max || ' ' || prenume\_proprietar\_max ||  
 ' este unul dintre cei mai fideli clienti ai nostri: ne-a vizitat de ' || co\_max ||  
 ' ori :)');  
 EXCEPTION  
 when faliment then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu avem niciun animalut care s-a tratat la noi...');  
 commit;  
 END *afis\_tratamente\_animalute*;  
  
 FUNCTION *factura\_aferenta\_fisa*(suma\_min IN float, suma\_totala OUT float) return pls\_integer  
 IS  
 id FACTURA\_PROPRIETAR.id\_factura%type;  
 pret FACTURA.total\_factura%type;  
 co pls\_integer := 0;  
 co\_tot pls\_integer := 0;  
 factura\_fara\_stapan EXCEPTION;  
 raport\_pret\_consultatii\_prost EXCEPTION;  
 BEGIN  
 suma\_totala := 0;  
 for i in (select p.id\_proprietar, p.nume propnum, p.prenume, a.nume animnume, fm.data\_fisa, fm.id\_fisa  
 from FISA\_MEDICALA fm,  
 ANIMALUT a,  
 PROPRIETAR p  
 where fm.id\_animalut = a.id\_animalut  
 and a.id\_proprietar = p.id\_proprietar(+))  
 loop  
 BEGIN  
 select fp.id\_factura, f.total\_factura  
 into id, pret  
 from FACTURA\_PROPRIETAR fp,  
 FACTURA f  
 where fp.id\_factura = f.id\_factura  
 and i.id\_fisa = fp.id\_fisa\_medicala;  
 DBMS\_OUTPUT.*PUT\_LINE*('Proprietarul animalutului ' || i.animnume || ' al lui ' ||  
 *nvl*(i.propnum, 'FARA') || ' ' || *nvl*(i.prenume, 'PROPRIETAR') ||  
 ' are factura ' ||  
 id || ' emisa la data de ' || i.data\_fisa || ', pentru fisa medicala ' ||  
 i.id\_fisa || ', total\_factura=' || pret);  
  
 if i.id\_proprietar is null then  
 raise factura\_fara\_stapan;  
 end if;  
  
 suma\_totala := suma\_totala + pret;  
 co\_tot := co\_tot + 1;  
 if pret >= suma\_min then  
 co := co + 1;  
 end if;  
  
 if suma\_totala / co\_tot < suma\_min then  
 raise raport\_pret\_consultatii\_prost;  
 end if;  
 EXCEPTION  
 when NO\_DATA\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu s-a emis nicio factura pentru proprietarul animalutului ' ||  
 i.animnume ||  
 ' al lui ' || *nvl*(i.propnum, 'FARA') || ' ' ||  
 *nvl*(i.prenume, 'PROPRIETAR') ||  
 ' la data de ' || i.data\_fisa ||  
 ' desi a fost tratat, cu fisa medicala cu id ' || i.id\_fisa);  
 insert into mesaje(message\_id, message, message\_type, created\_by, created\_at)  
 values (seq\_err.NEXTVAL,  
 'Nu s-a emis nicio factura pentru proprietarul animalutului ' || i.animnume ||  
 ' al lui ' || *nvl*(i.propnum, 'FARA') || ' ' || *nvl*(i.prenume, 'PROPRIETAR') ||  
 ' la data de ' || i.data\_fisa ||  
 ' desi a fost tratat, cu fisa medicala cu id ' || i.id\_fisa, 'I', *USER*, *sysdate*);  
 when TOO\_MANY\_ROWS then  
 DBMS\_OUTPUT.*PUT\_LINE*('Proprietarul animalutului ' || i.animnume || ' al lui ' ||  
 *nvl*(i.propnum, 'FARA') || ' ' || *nvl*(i.prenume, 'PROPRIETAR') ||  
 ' are mai multe facturi emise la data de ' || i.data\_fisa);  
 insert into mesaje(message\_id, message, message\_type, created\_by, created\_at)  
 values (seq\_err.NEXTVAL, 'Proprietarul animalutului ' || i.animnume || ' al lui ' ||  
 *nvl*(i.propnum, 'FARA') || ' ' || *nvl*(i.prenume, 'PROPRIETAR') ||  
 ' are mai multe facturi emise la data de ' || i.data\_fisa, 'I', *USER*,  
 *sysdate*);  
 when factura\_fara\_stapan then  
 DBMS\_OUTPUT.*PUT\_LINE*('Alerta: factura emisa pentru animalut fara stapan');  
 insert into mesaje(message\_id, message, message\_type, created\_by, created\_at)  
 values (seq\_err.NEXTVAL, 'Alerta: factura emisa pentru animalut fara stapan', 'I', *USER*,  
 *sysdate*);  
 when raport\_pret\_consultatii\_prost then  
 DBMS\_OUTPUT.*PUT\_LINE*('Alerta: raport suma\_totala/co prost');  
 insert into mesaje(message\_id, message, message\_type, created\_by, created\_at)  
 values (seq\_err.NEXTVAL, 'Alerta: raport suma\_totala/co prost', 'I', *USER*, *sysdate*);  
 END;  
 end loop;  
 commit;  
 return co;  
 end *factura\_aferenta\_fisa*;  
  
 FUNCTION *topclienti*(top\_clienti out tab\_imb) return pls\_integer  
 IS  
 TYPE refcursor IS REF CURSOR;  
 cursor mycursor is select f.id\_proprietar,  
 cursor (select c.suma\_platita  
 from CHITANTA c  
 where c.id\_factura = f.id\_factura)  
 from FACTURA\_PROPRIETAR f,  
 PROPRIETAR p  
 where f.id\_proprietar = p.id\_proprietar  
 order by p.id\_proprietar;  
 v\_cursor refcursor;  
 id FACTURA\_PROPRIETAR.id\_factura%type := null;  
 suma pls\_integer := 0;  
 total\_suma pls\_integer := 0;  
 max\_suma pls\_integer := 0;  
 last\_id PROPRIETAR.id\_proprietar%type := null;  
 co pls\_integer := 0;  
 BEGIN  
 top\_clienti := tab\_imb();  
 open mycursor;  
 loop  
 fetch mycursor into id, v\_cursor;  
 if last\_id is null then  
 last\_id := id;  
 else  
 if mycursor%notfound or id <> last\_id then  
 if total\_suma > max\_suma then  
 top\_clienti.DELETE;  
 top\_clienti.EXTEND;  
 co := 1;  
 top\_clienti(1) := last\_id;  
 max\_suma := total\_suma;  
 elsif total\_suma = max\_suma then  
 top\_clienti.EXTEND;  
 co := co + 1;  
 top\_clienti(co) := last\_id;  
 end if;  
 last\_id := id;  
 total\_suma := 0;  
 end if;  
 end if;  
 exit when mycursor%notfound;  
 loop  
 fetch v\_cursor into suma;  
 exit when v\_cursor%notfound;  
 total\_suma := total\_suma + suma;  
 end loop;  
 end loop;  
 close mycursor;  
 return co;  
  
 END *topclienti*;  
  
END package\_clinica\_vet\_tav13;  
/  
  
begin  
 package\_clinica\_vet\_tav13.*afis\_doctor\_favorit*(1);  
 DBMS\_OUTPUT.*PUT\_LINE*('........................................................................................');  
  
 package\_clinica\_vet\_tav13.*afis\_tratamente\_animalute*();  
 DBMS\_OUTPUT.*PUT\_LINE*('........................................................................................');  
  
 declare  
 co pls\_integer;  
 sum\_tot float;  
 begin  
 co := package\_clinica\_vet\_tav13.*factura\_aferenta\_fisa*(490, sum\_tot);  
 DBMS\_OUTPUT.*PUT\_LINE*(co || ' clienti au facturi de minim ' || 490);  
 DBMS\_OUTPUT.*PUT\_LINE*('suma totala din facturi unice: ' || sum\_tot);  
 end;  
 DBMS\_OUTPUT.*PUT\_LINE*('........................................................................................');  
  
 declare  
 co pls\_integer;  
 id NUMBER(10);  
 mytab package\_clinica\_vet\_tav13.tab\_imb;  
 num PROPRIETAR.nume%type;  
 prenum PROPRIETAR.prenume%type;  
 begin  
 co := package\_clinica\_vet\_tav13.*TOPCLIENTI*(mytab);  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista ' || co || ' clienti:');  
  
 id := mytab.FIRST;  
 loop  
 exit when id is null;  
 select nume, prenume  
 into num, prenum  
 from PROPRIETAR  
 where id\_proprietar = mytab(id);  
 DBMS\_OUTPUT.*PUT\_LINE*(num || ' ' || prenum || ' (id: ' || mytab(id) || ')');  
 id := mytab.*NEXT*(id);  
 end loop;  
 end;  
end;  
/

[2023-02-06 05:58:56] completed in 52 ms

Animalutul 1 a mers de cele mai multe ori la acesti doctori:

1. Staicu Clara

2. Popa Adi

3. Spalatorul Pop

........................................................................................

Animalutul Bit al lui Staicu Octavian-Florin a avut urmatoarele interventii medicale:

1. Enterocolita la data de 13-JUL-20, avand emise facturile:

1.1. id\_factura=1, emisa la data de 13-JUL-20

2. Rana deschisa la data de 13-SEP-20, avand emise facturile:

2.1. id\_factura=4, emisa la data de 13-SEP-20

3. Enterocolita la data de 25-DEC-20, avand emise facturile:

3.1. id\_factura=10, emisa la data de 25-DEC-20

Animalutul Kid al lui Staicu Adrian a avut urmatoarele interventii medicale:

1. Vaccinare Leptospiroza la data de 13-JUL-11, avand emise facturile:

1.1. id\_factura=2, emisa la data de 13-JUL-11

Animalutul Kitty al lui Bodea George a avut urmatoarele interventii medicale:

1. Vaccinare Leptospiroza la data de 13-JUL-16, avand emise facturile:

1.1. id\_factura=3, emisa la data de 13-JUL-15

2. Rana deschisa la data de 13-DEC-16, avand emise facturile:

2.1. id\_factura=9, emisa la data de 13-DEC-16

Animalutul Angelina al lui Cineva Interesant a avut urmatoarele interventii medicale:

1. Vaccinare Leptospiroza la data de 13-JUL-17, avand emise facturile:

1.1. id\_factura=5, emisa la data de 13-JUL-17

2. Vaccinare Leptospiroza la data de 13-JUL-20, avand emise facturile:

2.1. id\_factura=8, emisa la data de 13-JUL-20

Animalutul John al lui Popescu Ion a avut urmatoarele interventii medicale:

1. Enterocolita la data de 04-MAY-09, avand emise facturile:

1.1. id\_factura=6, emisa la data de 04-MAY-09

2. Vaccinare Leptospiroza la data de 14-JUN-10, avand emise facturile:

2.1. id\_factura=7, emisa la data de 14-JUN-10

3. Rana deschisa la data de 14-AUG-15, avand emise facturile:

3.1. id\_factura=104, emisa la data de 13-JUL-21

Animalutul dog al lui Cineva Interesant nu a avut niciodata parte de o procedura medicala

Animalutul Thor al lui Staicu Octavian-Florin a avut urmatoarele interventii medicale:

1. Vaccinare Leptospiroza la data de 13-SEP-20, avand emise facturile:

1.1. id\_factura=106, emisa la data de 12-MAY-21

1.2. id\_factura=107, emisa la data de 12-MAY-21

2. Vaccinare Leptospiroza la data de 13-JUL-21, avand emise facturile:

2.1. id\_factura=105, emisa la data de 13-JUL-21

Animalutul Bit [id\_animalut=1] al lui Staicu Octavian-Florin este unul dintre cei mai fideli clienti ai nostri: ne-a vizitat de 3 ori :)

........................................................................................

Proprietarul animalutului Bit al lui Staicu Octavian-Florin are factura 1 emisa la data de 13-JUL-20, pentru fisa medicala 1, total\_factura=300

Alerta: raport suma\_totala/co prost

Proprietarul animalutului Kid al lui Staicu Adrian are factura 2 emisa la data de 13-JUL-11, pentru fisa medicala 2, total\_factura=300

Alerta: raport suma\_totala/co prost

Proprietarul animalutului Kitty al lui Bodea George are factura 3 emisa la data de 13-JUL-16, pentru fisa medicala 3, total\_factura=300

Alerta: raport suma\_totala/co prost

Proprietarul animalutului Bit al lui Staicu Octavian-Florin are factura 4 emisa la data de 13-SEP-20, pentru fisa medicala 4, total\_factura=300

Alerta: raport suma\_totala/co prost

Proprietarul animalutului Angelina al lui Cineva Interesant are factura 5 emisa la data de 13-JUL-17, pentru fisa medicala 5, total\_factura=300

Alerta: raport suma\_totala/co prost

Proprietarul animalutului John al lui Popescu Ion are factura 6 emisa la data de 04-MAY-09, pentru fisa medicala 6, total\_factura=300

Alerta: raport suma\_totala/co prost

Proprietarul animalutului John al lui Popescu Ion are factura 7 emisa la data de 14-JUN-10, pentru fisa medicala 7, total\_factura=300

Alerta: raport suma\_totala/co prost

Proprietarul animalutului Angelina al lui Cineva Interesant are factura 8 emisa la data de 13-JUL-20, pentru fisa medicala 8, total\_factura=300

Alerta: raport suma\_totala/co prost

Proprietarul animalutului Kitty al lui Bodea George are factura 9 emisa la data de 13-DEC-16, pentru fisa medicala 9, total\_factura=300

Alerta: raport suma\_totala/co prost

Proprietarul animalutului Bit al lui Staicu Octavian-Florin are factura 10 emisa la data de 25-DEC-20, pentru fisa medicala 10, total\_factura=300

Alerta: raport suma\_totala/co prost

Proprietarul animalutului John al lui Popescu Ion are factura 104 emisa la data de 14-AUG-15, pentru fisa medicala 11, total\_factura=300

Alerta: raport suma\_totala/co prost

Proprietarul animalutului Thor al lui Staicu Octavian-Florin are mai multe facturi emise la data de 13-SEP-20

Proprietarul animalutului Thor al lui Staicu Octavian-Florin are factura 105 emisa la data de 13-JUL-21, pentru fisa medicala 12, total\_factura=3000

Nu s-a emis nicio factura pentru proprietarul animalutului Idk al lui FARA PROPRIETAR la data de 05-FEB-23 desi a fost tratat, cu fisa medicala cu id 14

Proprietarul animalutului Idk2 al lui FARA PROPRIETAR are factura 1002 emisa la data de 05-FEB-23, pentru fisa medicala 15, total\_factura=499

Alerta: factura emisa pentru animalut fara stapan

1 clienti au facturi de minim 490

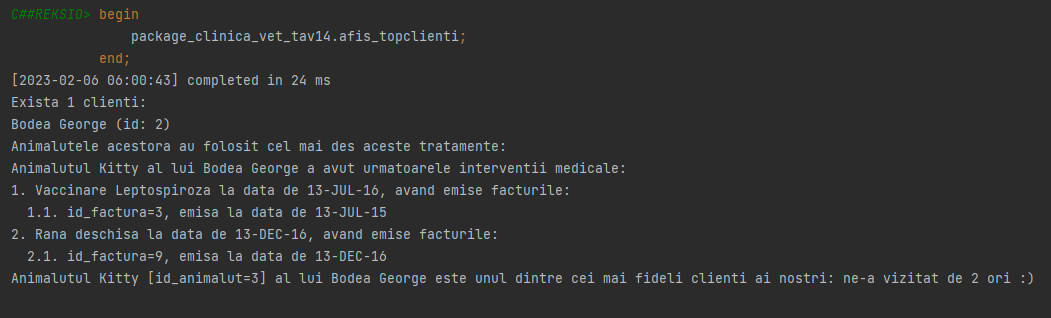
suma totala din facturi unice: 6300

........................................................................................

Exista 1 clienti:

Bodea George (id: 2)

-- Suplimentar: pachet care demonstreaza cum pot fi utilizate mai multe din procedurile anterioare  
-- unitar, pentru a afisa tratamentele princ care au trecut clientii fidel  
CREATE OR REPLACE PACKAGE package\_clinica\_vet\_tav14 AS  
 TYPE tab\_imb is TABLE OF NUMBER(10);  
 top\_clienti tab\_imb := tab\_imb();  
 FUNCTION *topclienti* RETURN PLS\_INTEGER;  
 PROCEDURE *afis\_topclienti*;  
 PROCEDURE *afis\_tratamente\_animalute*;  
 TYPE refcursor IS REF CURSOR;  
 type rec is record  
 (  
 nume\_animalut ANIMALUT.nume%type,  
 nume\_proprietar PROPRIETAR.nume%type,  
 prenume\_proprietar PROPRIETAR.prenume%type  
 );  
END package\_clinica\_vet\_tav14;  
/  
  
CREATE OR REPLACE PACKAGE BODY package\_clinica\_vet\_tav14 AS  
  
 FUNCTION *topclienti* RETURN PLS\_INTEGER  
 IS  
 cursor mycursor is select f.id\_proprietar,  
 cursor (select c.suma\_platita  
 from CHITANTA c  
 where c.id\_factura = f.id\_factura)  
 from FACTURA\_PROPRIETAR f,  
 PROPRIETAR p  
 where f.id\_proprietar = p.id\_proprietar  
 order by p.id\_proprietar;  
 v\_cursor refcursor;  
 id FACTURA\_PROPRIETAR.id\_factura%type := null;  
 suma pls\_integer := 0;  
 total\_suma pls\_integer := 0;  
 max\_suma pls\_integer := 0;  
 last\_id PROPRIETAR.id\_proprietar%type := null;  
 co pls\_integer := 0;  
 BEGIN  
 open mycursor;  
 loop  
 fetch mycursor into id, v\_cursor;  
 if last\_id is null then  
 last\_id := id;  
 else  
 if id <> last\_id or mycursor%notfound then  
 if total\_suma > max\_suma then  
 top\_clienti.DELETE;  
 top\_clienti.EXTEND;  
 co := 1;  
 top\_clienti(1) := last\_id;  
 max\_suma := total\_suma;  
 elsif total\_suma = max\_suma then  
 top\_clienti.EXTEND;  
 co := co + 1;  
 top\_clienti(co) := last\_id;  
 end if;  
 last\_id := id;  
 total\_suma := 0;  
 end if;  
 end if;  
 exit when mycursor%notfound;  
 loop  
 fetch v\_cursor into suma;  
 exit when v\_cursor%notfound;  
 total\_suma := total\_suma + suma;  
 end loop;  
 end loop;  
 close mycursor;  
 return co;  
  
 END *topclienti*;  
  
 PROCEDURE *afis\_tratamente\_animalute* IS  
 TYPE refcursor IS REF CURSOR;  
 cursor mycursor(id\_anim IN FISA\_MEDICALA.id\_animalut%type) is select diag.descriere,  
 fm.data\_fisa,  
 cursor (select fp.id\_factura, f.data\_emitere  
 from FACTURA\_PROPRIETAR fp,  
 FACTURA f  
 where fp.id\_factura = f.id\_factura  
 and fp.id\_fisa\_medicala = fm.id\_fisa)  
 from FISA\_MEDICALA fm,  
 DIAGNOSTIC diag  
 where fm.id\_diagnostic = diag.id\_diagnostic  
 and fm.id\_animalut = id\_anim;  
 v\_cursor refcursor;  
 diag DIAGNOSTIC.descriere%type;  
 data\_diagnostic FISA\_MEDICALA.data\_fisa%type;  
 id\_factura FACTURA\_PROPRIETAR.id\_factura%type;  
 data\_factura FACTURA.data\_emitere%type;  
 id\_animalut\_max ANIMALUT.id\_animalut%type;  
 nume\_animalut\_max ANIMALUT.nume%type;  
 nume\_proprietar\_max PROPRIETAR.nume%type;  
 prenume\_proprietar\_max PROPRIETAR.prenume%type;  
 ok\_factura binary\_integer;  
 co\_max pls\_integer := 0;  
 faliment EXCEPTION;  
 BEGIN  
 for i in (select a.id\_animalut, a.nume nume\_animalut, p.nume nume\_proprietar, p.prenume prenume\_proprietar  
 from ANIMALUT a,  
 PROPRIETAR p  
 where a.id\_proprietar = p.id\_proprietar  
 and p.id\_proprietar in (select \* from table (top\_clienti)) )  
 loop  
 open mycursor(i.id\_animalut);  
 fetch mycursor into diag, data\_diagnostic, v\_cursor;  
 if mycursor%notfound then  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || i.nume\_animalut || ' al lui ' || i.nume\_proprietar || ' ' ||  
 i.prenume\_proprietar || ' nu a avut niciodata parte de o procedura medicala');  
 DBMS\_OUTPUT.*NEW\_LINE*();  
 else  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || i.nume\_animalut || ' al lui ' || i.nume\_proprietar || ' ' ||  
 i.prenume\_proprietar || ' a avut urmatoarele interventii medicale:');  
 loop  
 exit when mycursor%notfound;  
 DBMS\_OUTPUT.*PUT\_LINE*(mycursor%rowcount || '. ' || diag || ' la data de ' || data\_diagnostic ||  
 ', avand emise facturile:');  
 ok\_factura := 0;  
 loop  
 fetch v\_cursor into id\_factura, data\_factura;  
 exit when v\_cursor%notfound;  
 ok\_factura := 1;  
 DBMS\_OUTPUT.*PUT\_LINE*(' ' || mycursor%rowcount || '.' || v\_cursor%rowcount ||  
 '. id\_factura=' ||  
 id\_factura || ', emisa la data de ' || data\_factura);  
 end loop;  
  
 if ok\_factura = 0 then  
 DBMS\_OUTPUT.*PUT\_LINE*(' Nicio factura emisa pentru aceasta fisa medicala');  
 end if;  
  
 fetch mycursor into diag, data\_diagnostic, v\_cursor;  
 end loop;  
 DBMS\_OUTPUT.*NEW\_LINE*();  
  
 if mycursor%rowcount > co\_max then  
 co\_max := mycursor%rowcount;  
 id\_animalut\_max := i.id\_animalut;  
 nume\_animalut\_max := i.nume\_animalut;  
 nume\_proprietar\_max := i.nume\_proprietar;  
 prenume\_proprietar\_max := i.prenume\_proprietar;  
 end if;  
 end if;  
 close mycursor;  
 end loop;  
 if co\_max = 0 then  
 raise faliment;  
 end if;  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutul ' || nume\_animalut\_max || ' [id\_animalut=' || id\_animalut\_max || '] al lui ' ||  
 nume\_proprietar\_max || ' ' || prenume\_proprietar\_max ||  
 ' este unul dintre cei mai fideli clienti ai nostri: ne-a vizitat de ' || co\_max ||  
 ' ori :)');  
 EXCEPTION  
 when faliment then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu avem niciun animalut care s-a tratat la noi...');  
 END *afis\_tratamente\_animalute*;  
  
 PROCEDURE *afis\_topclienti* IS  
 co pls\_integer;  
 id NUMBER(10);  
 num PROPRIETAR.nume%type;  
 prenum PROPRIETAR.prenume%type;  
 begin  
 co := package\_clinica\_vet\_tav14.*TOPCLIENTI*;  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista ' || co || ' clienti:');  
 id := top\_clienti.FIRST;  
 loop  
 exit when id is null;  
 select nume, prenume  
 into num, prenum  
 from PROPRIETAR  
 where id\_proprietar = top\_clienti(id);  
 DBMS\_OUTPUT.*PUT\_LINE*(num || ' ' || prenum || ' (id: ' || top\_clienti(id) || ')');  
 id := top\_clienti.NEXT(id);  
 end loop;  
 DBMS\_OUTPUT.*PUT\_LINE*('Animalutele acestora au folosit cel mai des aceste tratamente:');  
 package\_clinica\_vet\_tav14.*AFIS\_TRATAMENTE\_ANIMALUTE*();  
 end *afis\_topclienti*;  
  
END package\_clinica\_vet\_tav14;  
/  
  
begin  
 package\_clinica\_vet\_tav14.*afis\_topclienti*;  
end;  
/



-- Inca un pachet suplimentar, cu doua utilitare folositoare  
-- Primul subprogram din pachet ne permit sa ordonam pacientii la primirea in cabinet astfel incat sa reducem timpul  
-- mediu de asteptare. Al doilea subprogram afiseaza datoriile ramase de achitat ale clientilor  
CREATE OR REPLACE PACKAGE package\_clinica\_vet\_utilitare  
IS  
  
 type t\_idx is table of pls\_integer index by pls\_integer;  
 type array is VARRAY(1000) of pls\_integer;  
 type myrec\_datorii is record  
 (  
 id\_prop PROPRIETAR.id\_proprietar%type,  
 nume\_prop PROPRIETAR.nume%type,  
 prenume\_prop PROPRIETAR.prenume%type,  
 id\_fac FACTURA\_PROPRIETAR.id\_factura%type,  
 total\_fac FACTURA.total\_factura%type,  
 suma\_platita CHITANTA.suma\_platita%type  
 );  
 type datorii\_imb is table of myrec\_datorii;  
 FUNCTION *timp\_minim\_de\_asteptare*(v in out array) RETURN t\_idx;  
 PROCEDURE *afis\_datorii*;  
END package\_clinica\_vet\_utilitare;  
/  
  
  
CREATE OR REPLACE PACKAGE BODY package\_clinica\_vet\_utilitare  
IS  
  
 FUNCTION *timp\_minim\_de\_asteptare*(v in out array) RETURN t\_idx  
 Is --Primind un vector cu timpul in care dureaza fiecare procedura medicala pt oamenii din acea zi  
 --se stabileste in ce ordine vor intra in cabinet, ai timpul mediu de asteptare sa fie minim  
 poz\_min pls\_integer;  
 sol t\_idx;  
 co pls\_integer := 0;  
 BEGIN  
 if v.FIRST is null then  
 raise NO\_DATA\_FOUND;  
 end if;  
 loop  
 poz\_min := 1;  
 for i in v.FIRST..v.LAST  
 loop  
 if v(i) < v(poz\_min) then  
 poz\_min := i;  
 end if;  
 end loop;  
 exit when v(poz\_min) = 1001;  
 co := co + 1;  
 sol(poz\_min) := co;  
 v(poz\_min) := 1001;  
 end loop;  
 return sol;  
 EXCEPTION  
 when NO\_DATA\_FOUND then  
 return sol;  
 END *timp\_minim\_de\_asteptare*;  
  
 PROCEDURE *afis\_datorii* is  
 my\_t datorii\_imb;  
 s pls\_integer;  
 s\_dat pls\_integer := 0;  
 idx pls\_integer;  
 last\_prop PROPRIETAR.id\_proprietar%type := null;  
 Begin  
  
 select p.id\_proprietar,  
 p.nume,  
 p.prenume,  
 fp.id\_factura,  
 f.total\_factura,  
 *sum*(c.suma\_platita) bulk collect  
 into my\_t  
 from PROPRIETAR p,  
 FACTURA\_PROPRIETAR fp,  
 FACTURA f,  
 CHITANTA c  
 where p.id\_proprietar = fp.id\_proprietar  
 and fp.id\_factura = f.id\_factura  
 and f.id\_factura = c.id\_factura(+)  
 group by p.id\_proprietar, p.nume, p.prenume, fp.id\_factura, f.total\_factura  
 order by p.id\_proprietar;  
 idx := my\_t.FIRST;  
 loop  
 if last\_prop is null then  
 if idx is not null then  
 last\_prop := my\_t(idx).id\_prop;  
 DBMS\_OUTPUT.*PUT\_LINE*(  
 'Proprietarul ' || my\_t(idx).nume\_prop || ' ' || my\_t(idx).prenume\_prop || ' (id:' ||  
 my\_t(idx).id\_prop || '):');  
 end if;  
 else  
 if idx is null or my\_t(idx).id\_prop <> last\_prop then  
 DBMS\_OUTPUT.*PUT\_LINE*('Total de plata: ' || s\_dat);  
 DBMS\_OUTPUT.*NEW\_LINE*();  
 s\_dat := 0;  
 if idx is not null then  
 last\_prop := my\_t(idx).id\_prop;  
 DBMS\_OUTPUT.*PUT\_LINE*(  
 'Proprietarul ' || my\_t(idx).nume\_prop || ' ' || my\_t(idx).prenume\_prop ||  
 ' (id:' ||  
 my\_t(idx).id\_prop || '):');  
 end if;  
 end if;  
 end if;  
 exit when idx is null;  
 s := my\_t(idx).total\_fac - *nvl*(my\_t(idx).suma\_platita, 0);  
 s\_dat := s\_dat + s;  
 if s > 0 then  
 DBMS\_OUTPUT.*PUT\_LINE*('La factura ' || my\_t(idx).id\_fac || ' mai are de platit: ' || s);  
 end if;  
 idx := my\_t.*NEXT*(idx);  
 end loop;  
  
 end *afis\_datorii*;  
  
END package\_clinica\_vet\_utilitare;  
/  
  
begin  
 package\_clinica\_vet\_utilitare.*afis\_datorii*;  
end;  
/  
  
DECLARE  
 v package\_clinica\_vet\_utilitare.array := package\_clinica\_vet\_utilitare.array(7, 1, 2, 5, 3);  
 sol package\_clinica\_vet\_utilitare.t\_idx;  
BEGIN  
  
 sol := package\_clinica\_vet\_utilitare.*timp\_minim\_de\_asteptare*(v);  
 if sol.FIRST is not null then  
 for i in sol.FIRST..sol.LAST  
 loop  
 DBMS\_OUTPUT.*PUT\_LINE*(i || '. ' || sol(i));  
 end loop;  
 else  
 DBMS\_OUTPUT.*PUT\_LINE*('Niciun Client Programat');  
 end if;  
end;  
/

