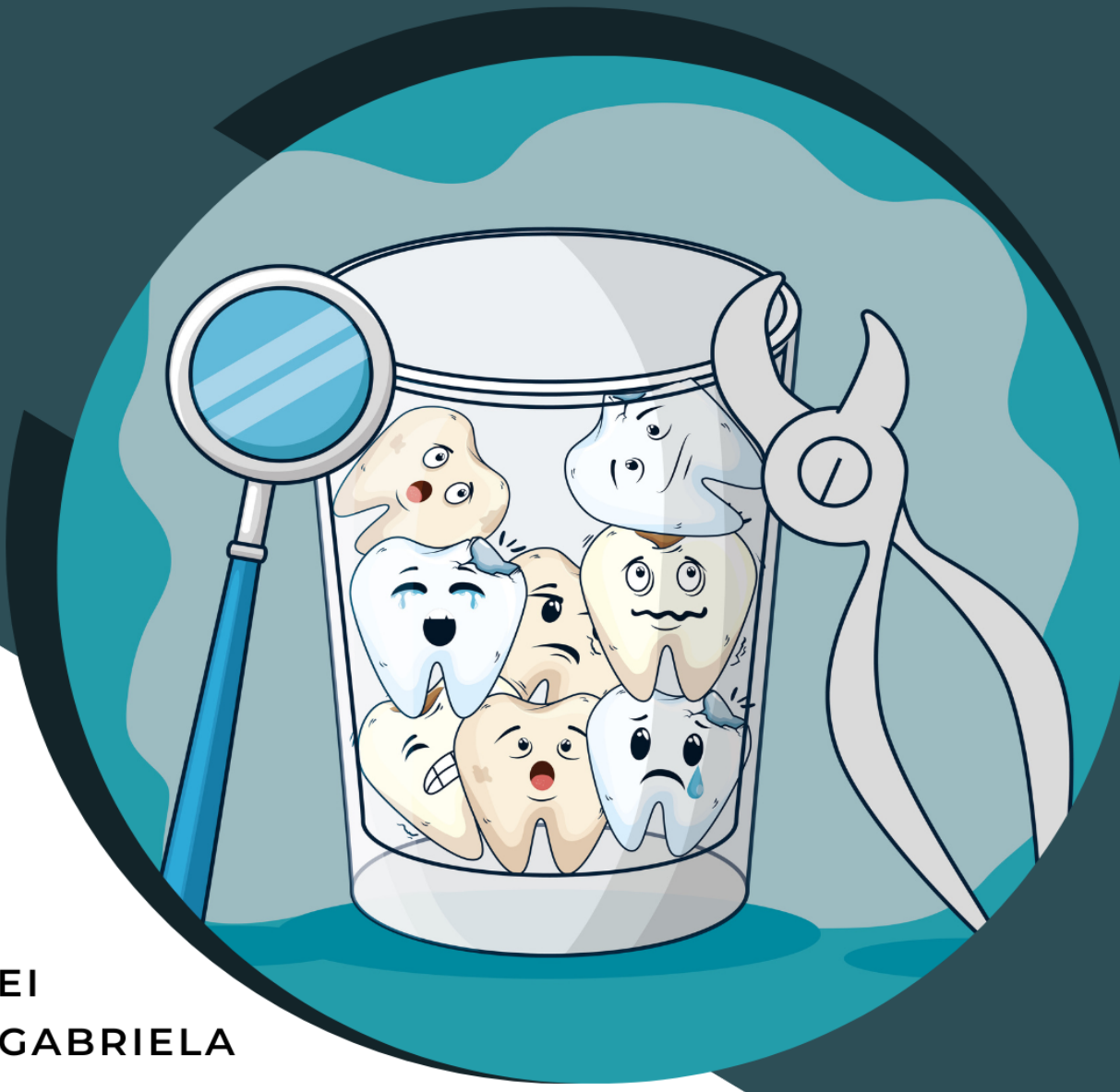
PROIECT SGBD

DENTAL CLINIC



ASAVOAEI
BIANCA GABRIELA

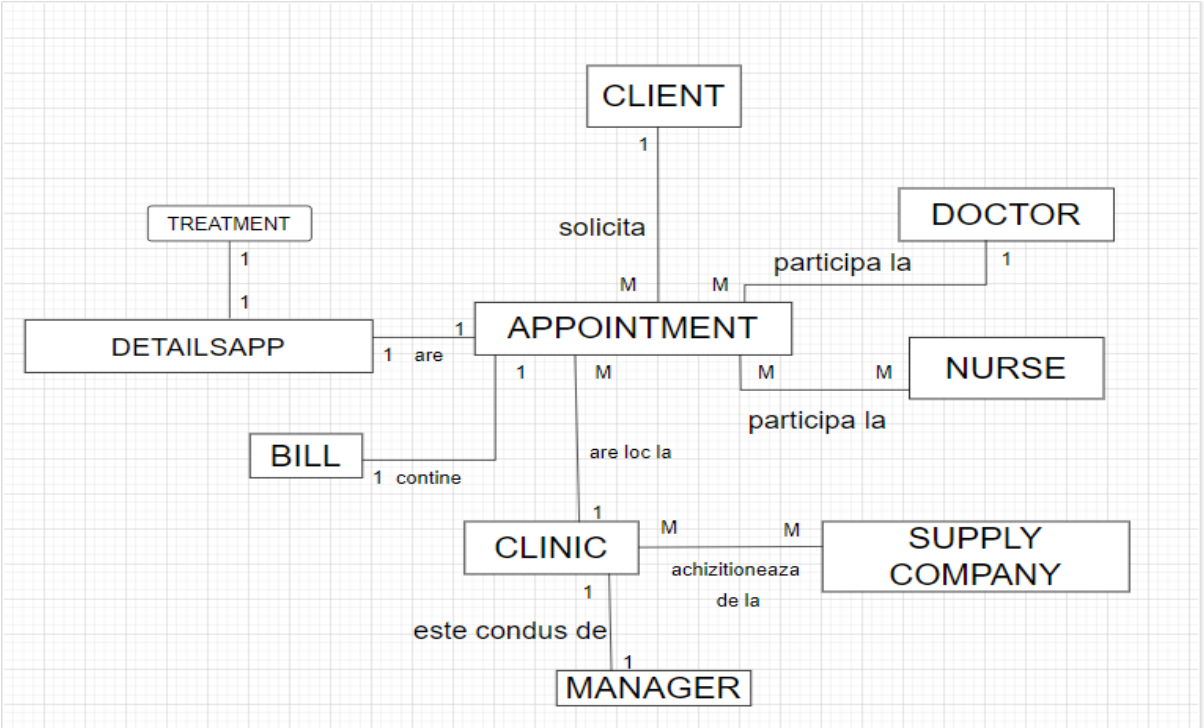
GRUPA 231

1. Prezentați pe scurt baza de date (utilitatea ei).

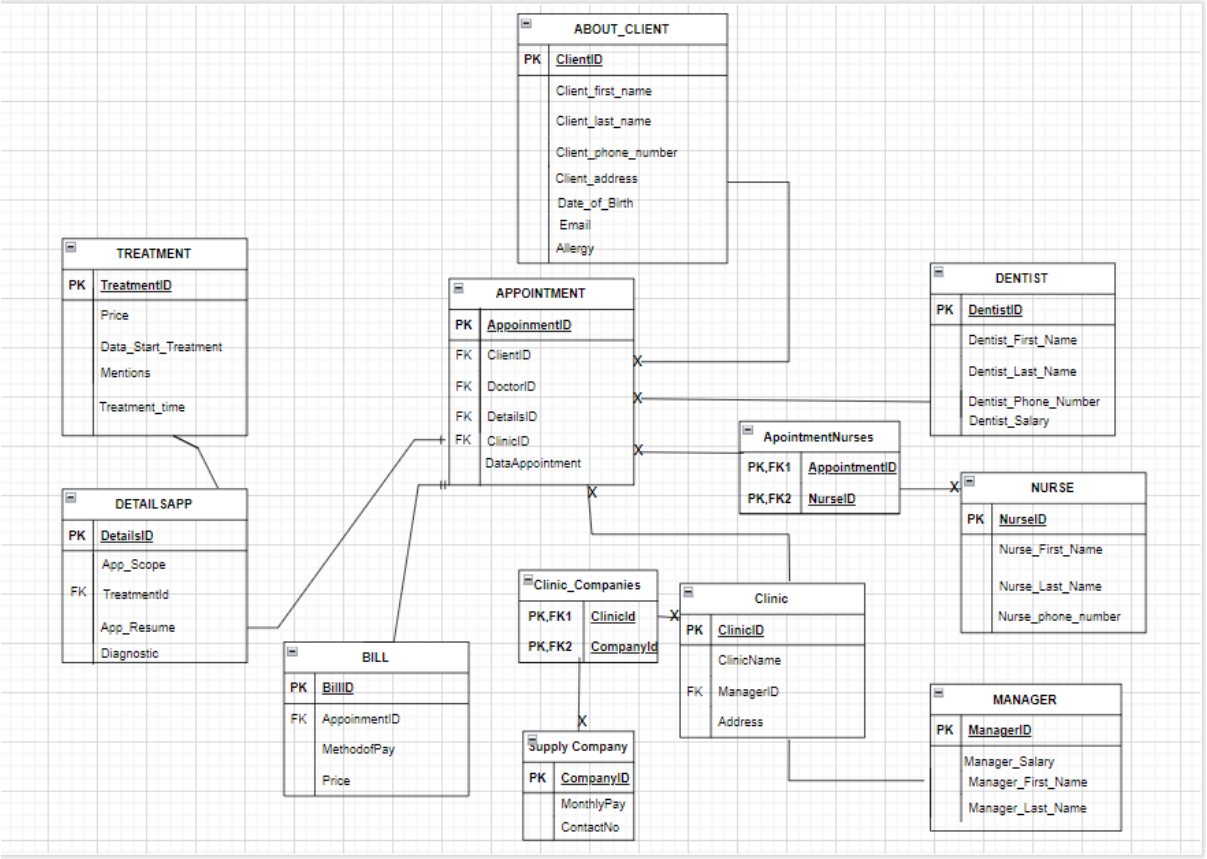
Ca pacient într-un cabinet de stomatologie, pentru mine a fost important cât timp petreceam așteptând să intru în cabinet. Bineînțeles, acesta este un vis pe care toți îl avem, de a ajunge la doctor și atunci să ne fie onorată programarea, dar de foarte multe ori ajungem, printr-o comunicare deficitară, să pierdem ore bune așteptând. Această bază de date reprezintă pentru mine un prim plan pentru o aplicație care să ajute pacientul, dar și dentistul, să-și eficientizeze timpul.

Baza de date păstrează informații relevante pentru dosarul medical al pacientului, programarea unei întâlniri, vizualizarea personalului și a activității sale, dar și a numeroaselor subcereri de informații.

2. Realizați diagrama entitate-relație (ERD).



3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.



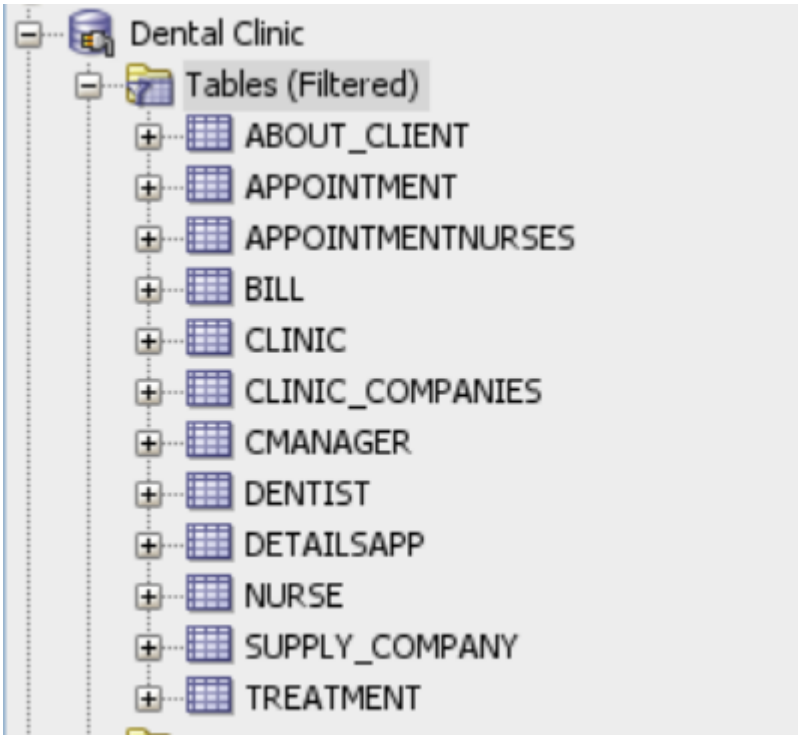
4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).

```
Create table Treatment (
    treatmentId int not null,
    price int,
    data_start_treatment date,
    treatment_time int,-- exemple 1 an si jumatate, 3 luni etc
    mentions varchar(150),
```

```
 constraint treatment_pk PRIMARY KEY (treatmentID)
 );

Create table About_Client (
    clientId int not null,
    client_first_name varchar(20),
    client_last_name varchar(20),
    email varchar(40),
    client_phone_number varchar(15),
    client_address varchar(50),
    data_of_birth date,
    allergy varchar(50),
    constraint client_pk primary key (clientID)
    );


Create table Nurse (
    nurseId int not null,
    nurse_first_name varchar(20),
    nurse_last_name varchar(20),
    nurse_phone_number varchar(15),
    constraint nurse_pk primary key (nurseId)
    );

Create table Dentist (
   dentistId int not null,
   dentist_first_name varchar(20),
   dentist_last_name varchar(20),
   dentist_salary int,
   dentist_phone_number varchar(20),
   constraint dentist_pk primary key (dentistId)
   );

Create table Supply_Company (
    companyId int not null,
    montly_pay int,
    contact_no varchar(15),
    constraint company_pk primary key (companyID)
```

```
);

Create table Clinic_Companies(
    ClinicId int not null,
    CompanyId int not null,
    constraint clinic_keys_pk primary key (clinicId, companyId),
    constraint clinik_fk foreign key (clinicId) references Clinic(clinicId),
        constraint companny_fk foreign key (companyId) references
Supply_Company(companyId)
    );

Create table CManager (
    managerId int not null,
    manager_first_name varchar(20),
    manager_last_name varchar(20),
    manager_salary int,
    constraint manager_pk primary key (managerId)
    );

Create table Clinic (
    clinicId int not null,
    managerId int not null,
    clinic_name varchar(30),
    address varchar(50),
    constraint clinic_pk primary key (clinicId),
        constraint manager_fk foreign key (managerId) references
CManager(managerId)
    );

Create table Bill (
    billId int not null,
    appointmentId int not null,
    method_of_pay varchar(20),
     constraint check_method_of_pay check (method_of_pay in ('cash',
'card', 'instalment')),
    constraint bill_pk primary key (billId)
    );
```

```
Create table Appointment (
    appointmentId int not null,
    clientId int not null,
    dentistId int not null,
    clinicId int not null,
    detailsId int not null,
    data_appointment date,
    constraint appointment_pk primary key(appointmentId),
        constraint client_fk foreign key (clientId) references About_Client(clientId),
        constraint dentist_fk foreign key (dentistId) references Dentist(dentistId),
    constraint clinic_fk foreign key (clinicId) references Clinic(clinicId),
        constraint details_fk foreign key (detailsId) references DetailsApp(detailsId)
    );

Create Table DetailsApp (
    detailsId int not null,
    treatmentId int not null,
    app_scope varchar(20),
    app_resume varchar(200),
    diagnostic varchar(50),
    constraint details_pk primary key(detailsId),
        constraint treatment_fk foreign key (treatmentId) references Treatment(treatmentId),
    constraint app_scope check (app_scope in ('checkup', 'control'))
    );

Create table AppointmentNurses(
    appointmentId int not null,
    nurseId int not null,
    constraint keys_pk primary key (appointmentId, nurseId),
        constraint app_fk foreign key (appointmentId) references Appointment(appointmentId),
        constraint nursse_fk foreign key (nurseId) references Nurse(nurseId) );
```



5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

Insert into About_Client Values(1, 'Gabriela', 'Asavoaei','bianca.asavoaei@yahoo.com', '0771002933', 'Vaslui, Vaslui, str.Bucuresti, bl.441', '06-NOV-2001', 'no allergy');
Insert into About_Client Values(2, 'Elizabeth', 'Clerance','eli_clerance@yahoo.com', '0799002933', 'Bucuresti, Sector 2, Str. Partiturii 13', '18-NOV-1998', 'no allergy');
Insert into About_Client Values(3, 'Vasile', 'Popa', '-', '0744562772', 'Bucuresti, Sector 4, Str. Locotenent Maior, nr. 12', '06-NOV-1975', 'silicon');
Insert into About_Client Values(4, 'Ernest', 'Apetrei','-', '0771087633', 'Ploiesti, str. Bucuresti, nr 14', '18-APR-1991', 'no allergy');

Insert into About_Client Values(5, 'Mihaela', 'Dava','-', '0741062933', 'Bucuresti, Sector 4, Str. Vasile Lupu, nr 6', '04-JUL-1969', 'no allergy');
Insert into About_client VALUES(6, 'Ana', 'Iova', 'ana_iova@gmail.com', '0744328976', 'Bucuresti, Sector 5, Str. 3 carari', '07-NOV-2004', 'silicon');
Insert into About_client VALUES(7, 'Marcel', 'Castor', 'marcel.c987@gmail.com', '0754387292', 'Bucuresti, Sector 3, Str. Vladimir Putin, nr 3', '12-APR-2004', 'no allergy');
Insert into About_client VALUES(8, 'Georgiana', 'Asavoaei', 'georgiana_geo18@yahoo.com', '0874553942', 'Vaslui, str. Bucuresti, bl. 441', '18-AUG-1998', 'coloranti A, B');
Insert into About_client VALUES(9, 'Andrei', 'Eduard', 'andrei_lol@yahoo.com', '0747554433', 'Bucuresti, Sector 2, str. Valea Seaca, nr 19', '14-FEB-1970', 'no allergy');
Insert into About_client VALUES(10, 'Antonia', 'Popovici', 'antonia_popovici@gmail.com', '0794229422', 'Bucuresti, Sector 2, str. Partiturii, nr 102', '13-APR-2008', 'no allergy');
Insert into About_client VALUES(11, 'Corina', 'Gherasim', 'corina.gherasim@gmail.com', '0733427891', 'Bucuresti, Sector 3, Str. 2 Petale, nr. 8', '01-JAN-2010', 'no allergy');
Insert into About_client VALUES(12, 'Sebi', 'Apostol', 'sebi.apostol@gmail.com', '0733482289', 'Bucuresti, Sector 2, Str. Westminister, nr. 9', '02-MAR-2009', 'no allergy');

Insert into Appointment VALUES(1, 1, 3, 4, 1, '15-MAY-2022');
Insert into Appointment VALUES(2, 3, 3, 5, 2, '18-SEPT-2022');
Insert into Appointment VALUES(3, 5, 1, 1, 3, '18-SEPT-2022');
Insert into Appointment VALUES(4, 2, 2, 2, 5, '19-SEPT-2022');
Insert into Appointment VALUES(5, 4, 5, 3, 4, '21-SEPT-2022');
Insert into Appointment VALUES(6, 6, 3, 3, 3, '23-SEPT-2022');
Insert into Appointment VALUES(7, 10, 3, 4, 2, '19-FEB-2022');
Insert into Appointment VALUES(8, 1, 2, 4, 6, '15-JUL-2022');
Insert into Appointment VALUES(9, 1, 2, 4, 7, '02-SEP-2022');
Insert into Appointment VALUES(10, 6, 4, 2, 3, '03-SEP-2022');
Insert into Appointment VALUES(11, 11, 3, 2, 4, '04-SEP-2022');
Insert into Appointment VALUES(12, 12, 3, 4, 8, '23-OCT-2022');
Insert into Appointment VALUES(13, 12, 3, 4, 9, '15-OCT-2022');
Insert into Appointment VALUES(14, 12, 3, 3, 10, '16-OCT-2022');

```
Insert into Appointment VALUES(15, 6, 3, 2, 11, '17-OCT-2022');


Insert into  AppointmentNurses VALUES(1, 2);
Insert into  AppointmentNurses VALUES(1, 3);
Insert into  AppointmentNurses VALUES(1, 4);
Insert into  AppointmentNurses VALUES(2, 1);
Insert into  AppointmentNurses VALUES(2, 2);
Insert into  AppointmentNurses VALUES(3, 5);
Insert into  AppointmentNurses VALUES(3, 3);
Insert into  AppointmentNurses VALUES(4, 1);
Insert into  AppointmentNurses VALUES(4, 5);
Insert into  AppointmentNurses VALUES(5, 4);


Insert into Bill VALUES(1, 1, 'cash');
Insert into Bill VALUES(2, 4, 'instalment');
Insert into Bill VALUES(3, 2, 'card');
Insert into Bill VALUES(4, 5, 'cash');
Insert into Bill VALUES(5, 3, 'cash');
Insert into Bill VALUES(6, 7, 'instalment');
Insert into Bill VALUES(7, 8, 'instalment');
Insert into Bill VALUES(8, 9, 'instalment');
Insert into Bill VALUES(9, 11, 'instalment');
Insert into Bill VALUES(10, 10, 'instalment');
Insert into Bill VALUES(12, 12, 'instalment');
Insert into Bill VALUES(11, 11, 'instalment');
Insert into Bill VALUES(13, 13, 'instalment');
Insert into Bill VALUES(14, 14, 'instalment');
Insert into Bill VALUES(15, 15, 'instalment');


Insert into Clinic VALUES(1, 1, 'Regina Elizabeta', 'Bucuresti, Sector 3,
Str. Valea Lupului, nr 4');
Insert into Clinic VALUES(2, 3, 'Regina Maria', 'Bucuresti, Sector 1, Str.
Capitan Andrei, nr 18');
Insert into Clinic VALUES(3, 2, 'Regina Elena', 'Bucuresti, Sector 4, Str.
Ferdinand 1, nr 2');
```

Insert into Clinic VALUES(4, 5, 'Regina Consuela', 'Bucuresti, Sector 3, Str. Narcisele, nr 19');
Insert into Clinic VALUES(5, 4, 'Regina Alberta', 'Bucuresti, Sector 6, Str. 7 Drumuri, nr 15');

Insert into Clinic_Companies VALUES (1, 2);
Insert into Clinic_Companies VALUES (1, 3);
Insert into Clinic_Companies VALUES (1, 4);
Insert into Clinic_Companies VALUES (1, 5);
Insert into Clinic_Companies VALUES (2, 2);
Insert into Clinic_Companies VALUES (2, 1);
Insert into Clinic_Companies VALUES (3, 5);
Insert into Clinic_Companies VALUES (4, 2);
Insert into Clinic_Companies VALUES (4, 1);
Insert into Clinic_Companies VALUES (5, 4);

Insert into CManager VALUES(1, 'John', 'Travolta', 5000);
Insert into CManager VALUES(2, 'Alessandra', 'Mich', 5000);
Insert into CManager VALUES(3, 'Antonio', 'Sincarenco', 6500);
Insert into CManager VALUES(4, 'Alex', 'Russo', 5500);
Insert into CManager VALUES(5, 'Ana', 'Carolina', 5000);

Insert into DENTIST Values (1, 'Costel', 'Mircel', '0799147791', 12000);
Insert into DENTIST Values (2, 'Costel', 'Hazam', '0799122791', 11000);
Insert into DENTIST Values (3, 'Antonia', 'Hazam', '0797537791', 12000);
Insert into DENTIST Values (4, 'Ela', 'Casanova', '0744147791', 10000);
Insert into DENTIST Values (5, 'Matei', 'Tolescu', '0795547791', 15000);

Insert into DetailsApp VALUES(1, 3, 'control', 'Pacientul a venit la control pentru verificarea plombelor si detartraj. Recomandare: fara mancare picanta sau sosuri timp de 48 de ore.', '-');
Insert into DetailsApp VALUES(2, 1, 'checkup', 'Pacientul a venit pentru un consult + curatare. A fost luata amprenta.', 'Dinti usor patati, anomalie maxilar');
Insert into DetailsApp VALUES(3, 5, 'control', 'Aparat dentar ceramic. Recomandare: fara alune, ciocolata sau mancare tare 3 zile. Revenire la control in 15 zile', 'Anomalie maxilar');

Insert into DetailsApp VALUES(4, 2, 'control', 'Revenire la control pentru amprenta luata pe 15 martie.', 'Dinti incalecati');
Insert into DetailsApp VALUES(5, 3, 'checkup', 'Pacientul a venit pentru consult.', 'Dinti sanatosi');
Insert into DetailsApp VALUES(6, 9, 'checkup', 'Pacientul continua si tratamentul de data trecuta', 'Maxilar usor in fata');
Insert into DetailsApp VALUES(7, 9, 'checkup', 'Pacientul continua si tratamentul de data trecuta', 'Maxilar usor in fata');
Insert into DetailsApp VALUES(8, 8, 'checkup', 'Aparat dentar clasic pentru dinti nealiniati', 'Dinti aliniati');
Insert into DetailsApp VALUES(9, 8, 'checkup', 'Pacientul are aparat dentar clasic', 'Muscatura inegala');
Insert into DetailsApp VALUES(10, 11, 'control', 'Montat aparat dentar', 'Anomalie maxilar');
Insert into DetailsApp VALUES(11, 12, 'control', 'Montat aparat dentar ceramic', 'Anomalie maxilar');

Insert into Nurse VALUES (1, 'Cassandra', 'Banks' , '0723418799');
Insert into Nurse VALUES(2, 'Ana', 'Owen', '0776542987');
Insert into Nurse VALUES(3, 'Karla', 'Coll', '0747554941');
Insert into Nurse VALUES(4, 'Elizabeth', 'July', '0789654321');
Insert into Nurse VALUES(5, 'Ana', 'Maria', '0712345679');

Insert into Supply_Company VALUES (1, 1500, '0765437210');
Insert into Supply_Company VALUES (2, 1500, '0765137210');
Insert into Supply_Company VALUES (3, 2500, '0765437990');
Insert into Supply_Company VALUES (4, 1000, '0787437210');
Insert into Supply_Company VALUES (5, 150, '0765537210');

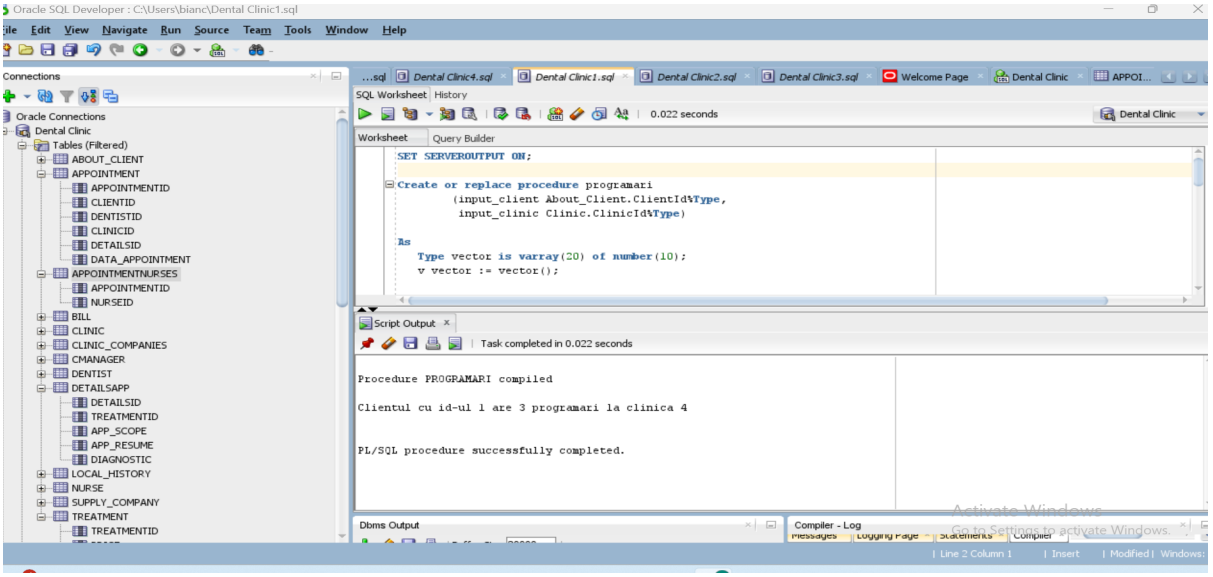
Insert into Treatment VALUES (1, 1500, '15-FEB-2022', 36, 'Dinti pozionati incorect. Aparat dentar clasic');
Insert into Treatment VALUES (2, 1500, '01-MAR-2022', 24, 'Muscatura gresita. Maxilar puternic orientat in fata. Aparat dentar clasic si aparat dentar lingual');
Insert into Treatment VALUES (3, 100, '15-FEB-2022', 0 , 'Detartraj dentar + periaj');

Insert into Treatment VALUES (4, 450, '21-MAY-2022', 0, 'Detartraj + periaj + air-flow');
Insert into Treatment VALUES (5, 3500, '18-SEP-2022', 18, 'Aparat dentar ceramic');
Insert into Treatment VALUES (6, 20, '18-SEP-2022', 0, 'Control perfect');
Insert into Treatment VALUES (7, 3200, '30-NOV-2022', 24, 'Aparat dentar clasic. Muscatura indreptata spre fata');
Insert into Treatment VALUES (8, 250, '11-NOV-2022', 2, 'Plomba masele maxilar inferior. Recomandare: fara alune, ciocolata sau mancaruri tari');
Insert into Treatment VALUES (9, 150, '11-NOV-2022', 4, 'Gutiera noaptea');
Insert into Treatment VALUES (10, 210, '15-NOV-2022', 2, 'Plomba 3 masele maxilar superior. Recomandare: fara alune, ciocolata sau mancaruri tari');
Insert into Treatment VALUES(11, 2500, '19-JAN-2022', 18, 'Montare aparat dentar. Recomandare: fara alune, dulce');
Insert into Treatment VALUES(12, 2500, '20-JAN-2022', 24, 'Montare aparat dentar. Recomandare: fara alune, dulce');

6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze două tipuri diferite de colecții studiate. Apelați subprogramul.

Pentru un client dat (id-ul clientului) și o clinică dată (id-ul clinicii), aflați câte programări a avut clientul la clinica respectivă.

```
SET SERVEROUTPUT ON;

Create or replace procedure programari
        (input_client About_Client.ClientId%Type,
         input_clinic Clinic.ClinicId%Type)

As
   Type vector is varray(20) of number(10);
   v vector := vector();
   Type tablou_indexat IS TABLE of Appointment%Rowtype Index by Binary_Integer;
   t tablou_indexat;
   dim_t integer;
   ind integer;

BEGIN
   ind := 0;
   Delete from Bill;
   Delete from appointmentnurses;

   Delete
   From Appointment
   Where ClientId = input_client
   Returning AppointmentId, ClientId, DentistId, ClinicId, DetailsId, Data_Appointment
   Bulk Collect into t;

   for i in 1..t.count() loop
       if t(i).ClinicId = input_clinic then
               v.extend();
               v(i) := i;
       end if;
    end loop;
    DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul ' || input_client || ' are ' || v.count || '
programari la clinica ' || input_clinic);

    t.delete;
    ROLLBACK;

END programari;
/
execute programari(1,4);
/
```



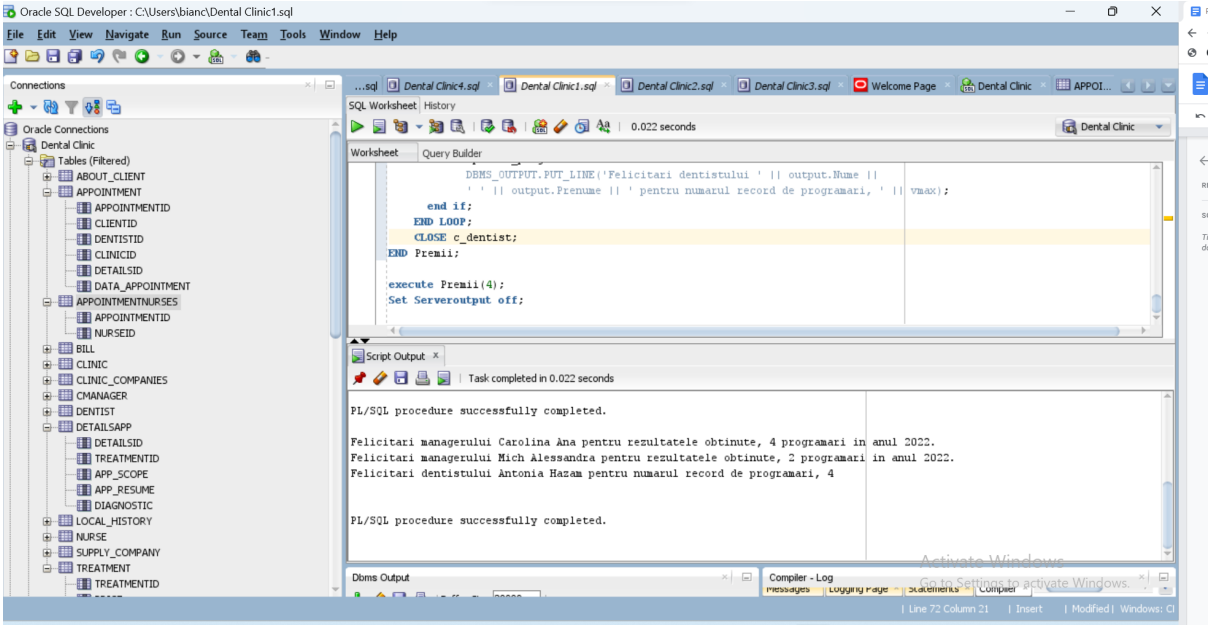
7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat. Apelați subprogramul.

La finalul anului 2022, vor avea loc Premiile BestDent. Obțineți numele managerilor a căror clinici au avut peste un număr de programări (număr dat) și numele dentistului cu cele mai multe programări.

```
Set Serveroutput on;
CREATE or REPLACE PROCEDURE Premii
   (nr_appointments Integer)

AS
   TYPE detalii_output IS RECORD (
   Nume Dentist.Dentist_First_Name%TYPE,
   Prenume Dentist.Dentist_Last_Name%TYPE,
   Nr_programari Int
   );
```

```
output detalii_output;

--- primul cursor parametrizat: imi parcurge managerii a caror clinici au
avut peste un nr de programari dat
CURSOR c_manager (nr_appointments int) IS
        SELECT  m.Manager_first_name,  m.Manager_last_name,
        COUNT(a.appointmentId)
          From CManager m, Clinic c, Appointment a
          Where m.managerid = c.managerId
          and a.clinicId = c.clinicId
          Group by  m.Manager_first_name, m.Manager_last_name
          Having COUNT(a.appointmentId) > nr_appointments;

 --- al doilea cursor neparametrizat: imi selecteaza dentistul numele si
nr de programari
CURSOR c_dentist IS
            Select   d.Dentist_first_name,   d.Dentist_last_name,
            Count(a.appointmentID)
              from Dentist d, Appointment a
              where d.dentistId = a.dentistId
              group by d.Dentist_first_name, d.Dentist_last_name;

vmax integer;
Prenume dentist.dentist_first_name%TYPE;
Nume dentist.dentist_last_name%TYPE;
Nr_programari int;


BEGIN

vmax := 0;
Open c_manager(nr_appointments);
LOOP
  FETCH c_manager into output;
  EXIT WHEN c_manager%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE('Felicitari  managerului  '  ||
output.Prenume || ' ' || output.Nume
```

```
                                || ' pentru rezultatele obtinute, ' || output.Nr_programari
|| ' programari in anul 2022.' );

   END LOOP;
   Close c_manager;

   Open c_dentist;
   LOOP
      FETCH c_dentist into  output;
      EXIT WHEN c_dentist%NOTFOUND;
      if output.nr_programari > vmax then
           vmax := output.nr_programari;
     end if;
   END LOOP;
   Close c_dentist;


   Open c_dentist;
   LOOP
      FETCH c_dentist into output;
      EXIT WHEN c_dentist%NOTFOUND;
      if output.nr_programari = vmax then
           DBMS_OUTPUT.PUT_LINE('Felicitari dentistului ' || output.Nume
||
            ' ' || output.Prenume || ' pentru numarul record de programari, ' ||
vmax);
     end if;
   END LOOP;
   CLOSE c_dentist;
END Premii;

execute Premii(4);
Set Serveroutput off;
```



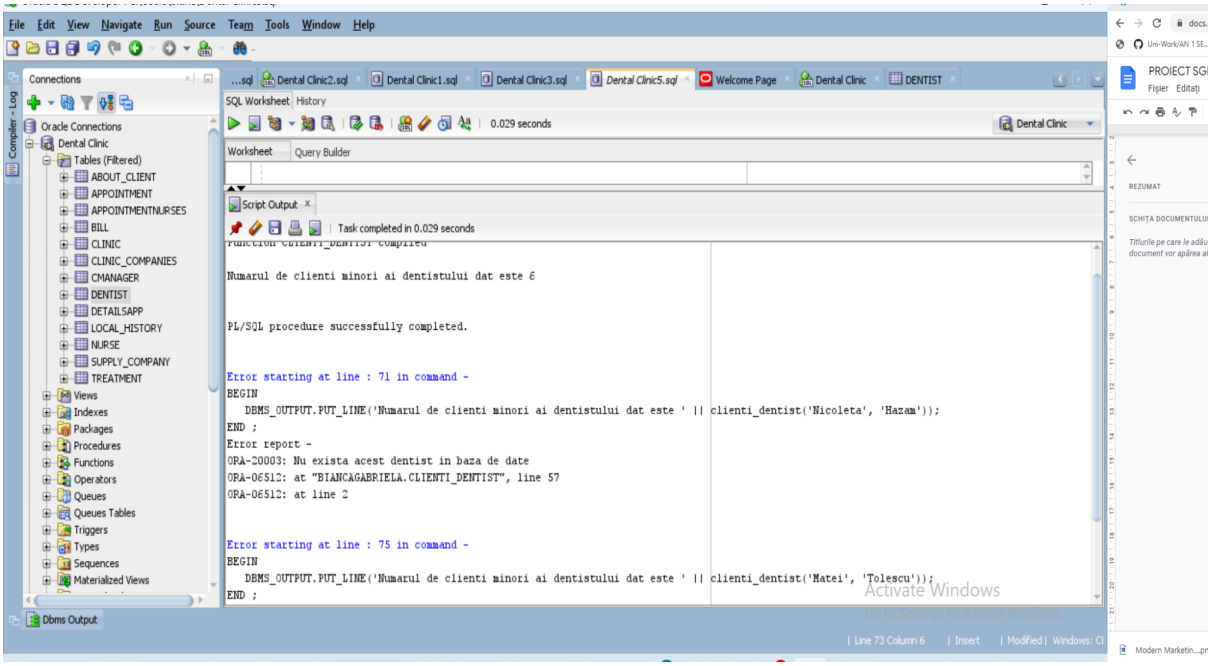
8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Afișați numărul clienților minori (în momentul intervenției) a unui dentist dat (nume , prenume).

```
SET SERVEROUTPUT ON;
SET VERIFY OFF;

CREATE or REPLACE FUNCTION clienti_dentist
                (i_prenume dentist.dentist_first_name%TYPE, i_nume
dentist.dentist_last_name%TYPE)

RETURN NUMBER AS
   id_dentist dentist.dentistid%TYPE;
   id_client_cur about_client.clientId%TYPE;
   nr_clienti number;
```

```
   DENTISTNOTFOUND exception;

   d_cur int;

   CURSOR c_dentist IS
      select dentistId
      from dentist
            where dentist_last_name =i_nume   and dentist_first_name =
i_prenume;

   CURSOR c_client IS
      SELECT c.clientId
      From About_Client c, Appointment a, Dentist d
       Where d.dentist_first_name = i_prenume and d.dentist_last_name
= i_nume
      and d.dentistId = a.dentistId and a.clientId = c.clientId and
      months_between( a.data_appointment, c.data_of_birth)/12  < 18;

BEGIN
   --- vedem daca exista acest doctor in baza de date
   id_dentist := -1;
   nr_clienti := 0;
   open c_dentist;
   loop
   Fetch c_dentist into d_cur;
   exit when c_dentist%notfound;
      id_dentist := d_cur;
   end loop;

   close c_dentist;

   if id_dentist = -1 then
       raise DENTISTNOTFOUND;
   end if;

   open c_client;
   loop
      fetch c_client into id_client_cur;
```

```
        exit when c_client%NOTFOUND;
            nr_clienti := nr_clienti + 1;
    end loop;
    close c_client;

    if nr_clienti = 0 then
        raise NO_DATA_FOUND;
    else
         Return nr_clienti;
    end if;

EXCEPTION
      WHEN DENTISTNOTFOUND then
            RAISE_APPLICATION_ERROR(-20003, 'Nu exista acest dentist
in baza de date');
     WHEN NO_DATA_FOUND then
            RAISE_APPLICATION_ERROR(-20004, 'Nu exista clienti minori
sau programari pentru doctorul dat');

END clienti_dentist;

-- doctor care are clienti minori
BEGIN
     DBMS_OUTPUT.PUT_LINE('Numarul de clienti minori ai dentistului
dat este ' || clienti_dentist('Antonia', 'Hazam'));
END ;

-- doctor care nu exista
BEGIN
     DBMS_OUTPUT.PUT_LINE('Numarul de clienti minori ai dentistului
dat este ' || clienti_dentist('Nicoleta', 'Hazam'));
END ;

-- doctor care exista si  care nu are clienti minori
BEGIN
     DBMS_OUTPUT.PUT_LINE('Numarul de clienti minori ai dentistului
dat este ' || clienti_dentist('Matei', 'Tolescu'))
END ;
```



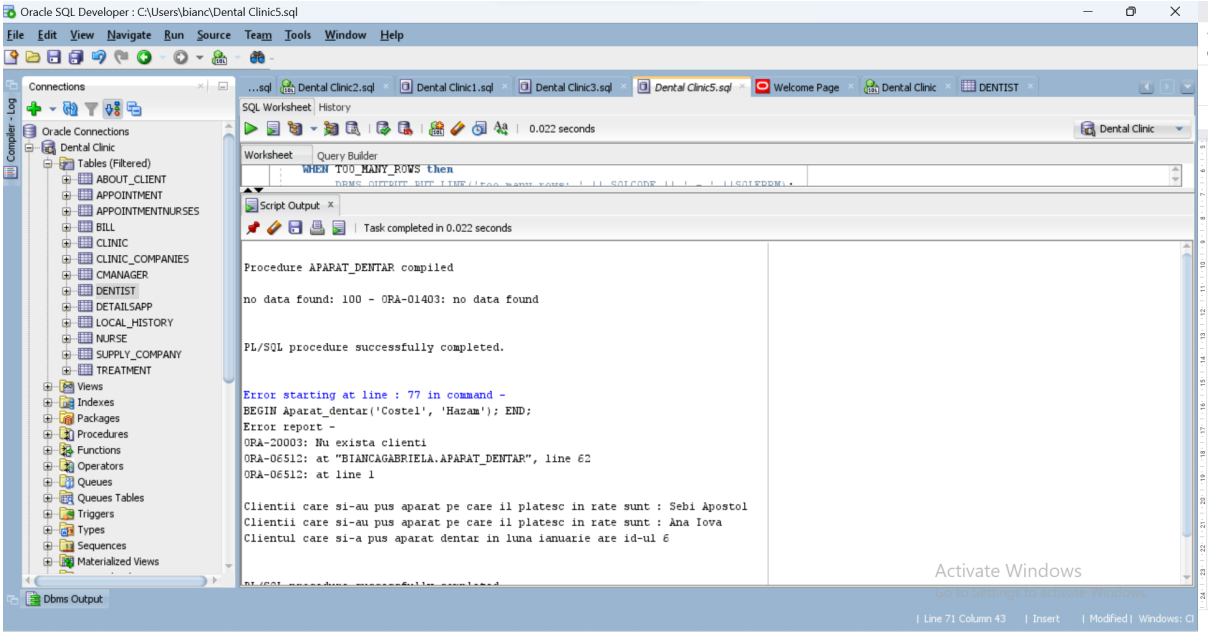
9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Pentru un dentist dat (prenume, nume), afișați clienții care plătesc în rate aparatul dentar și clientul care și-a pus aparat în luna ianuarie.

```
SET SERVEROUTPUT ON;

CREATE or REPLACE procedure Aparat_dentar
        (prenume_dentist dentist.dentist_first_name%Type, nume_dentist dentist.dentist_last_name%type)
```

```
AS
   Prenume About_Client.Client_First_name%TYPE;
   Nume About_Client.Client_Last_name%TYPE;
   ExistaDentist Dentist.dentistID%Type;
   client_cautat About_client.ClientId%Type;
   NODENTIST exception;
   NOCLIENTS exception;
   ok Number;

     -- pentru dentistul dat cursorul va trece prin clientii care platesc in rate
pentru aparat dentar

   CURSOR cauta_clienti(prenume_dentist varchar, nume_dentist varchar) IS
      Select ac.Client_first_name, ac.Client_last_name
      From About_Client ac, Appointment a, DetailsApp da, Bill b, Dentist d
            Where a.ClientId = ac.ClientId and da.DetailsId = a.DetailsId and
b.AppointmentId = a.AppointmentId
        and d.Dentist_first_name = prenume_dentist and d.Dentist_last_name =
nume_dentist and  a.DentistId= d.DentistId
           and b.method_of_Pay = 'instalment' and lower( da.app_resume) like
'%aparat dentar%'
      group by (ac.Client_first_name, ac.Client_last_name);


Begin

   ExistaDentist := -1;
  Select dentistId into ExistaDentist
  From Dentist
     Where dentist_first_name = prenume_dentist and dentist_last_name =
nume_dentist;



  ok := 0;
  open cauta_clienti(prenume_dentist, nume_dentist);
  loop
     Fetch cauta_clienti into Prenume, Nume;
     Exit when cauta_clienti%NOTFOUND;
        ok := 1;
```

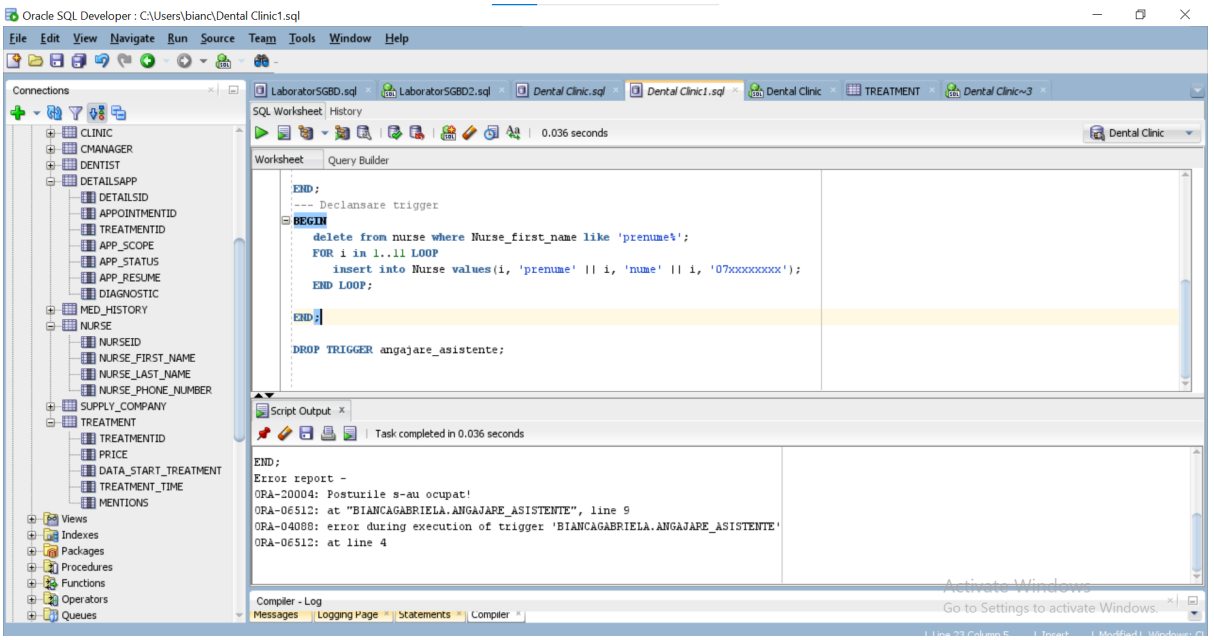
```
            DBMS_OUTPUT.PUT_LINE('Clientii care si-au pus aparat pe care il
platesc in rate sunt : ' || Prenume || ' ' || Nume);
      --- cautam clientul care a inceput tratamentul in luna ianuarie
         select ac.ClientId into client_cautat
         from About_Client ac, Appointment a, DetailsApp da, Treatment t
              where ac.Client_first_name = Prenume and ac.Client_last_name =
Nume
                and ac.ClientId = a.ClientId and a.DetailsId = da.DetailsId and
t.TreatmentId = da.TreatmentId
         and  extract(month from t.data_start_treatment) = 1;

   end loop;
   close cauta_clienti;

   if ok = 0 then
      raise NOCLIENTS;
   end if;

     DBMS_OUTPUT.PUT_LINE('Clientul care si-a pus aparat dentar in luna
ianuarie are id-ul ' || client_cautat);

EXCEPTION
   WHEN NO_DATA_FOUND then
              DBMS_OUTPUT.PUT_LINE('no data found: ' || SQLCODE || ' - '
||SQLERRM);
   WHEN NODENTIST then
          RAISE_APPLICATION_ERROR(-20001, 'Dentistul cautat nu se afla in
baza de date');
   WHEN NOCLIENTS then
       RAISE_APPLICATION_ERROR(-20003, 'Nu exista clienti');
   WHEN TOO_MANY_ROWS then
            DBMS_OUTPUT.PUT_LINE('too many rows: ' || SQLCODE || ' - '
||SQLERRM);

END Aparat_dentar;

-- doctor existent
execute Aparat_dentar('Antonia', 'Hazam');
-- doctor neexistent ---> no data found
execute Aparat_dentar('Vladimir', 'Putin');
execute Aparat_dentar('Costel', 'Hazam');
```



10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

```
CREATE or REPLACE TRIGGER angajare_asistente
    BEFORE INSERT ON Nurse

DECLARE
    nr_nurses int := 0;
BEGIN
   select Count(*) into nr_nurses
   From Nurse;

   if nr_nurses > 5 then
            RAISE_APPLICATION_ERROR(-20004, 'Posturile s-au
ocupat!');
   end if;

END;
--- Declansare trigger
BEGIN
  delete from nurse where Nurse_first_name like 'prenume%';
```

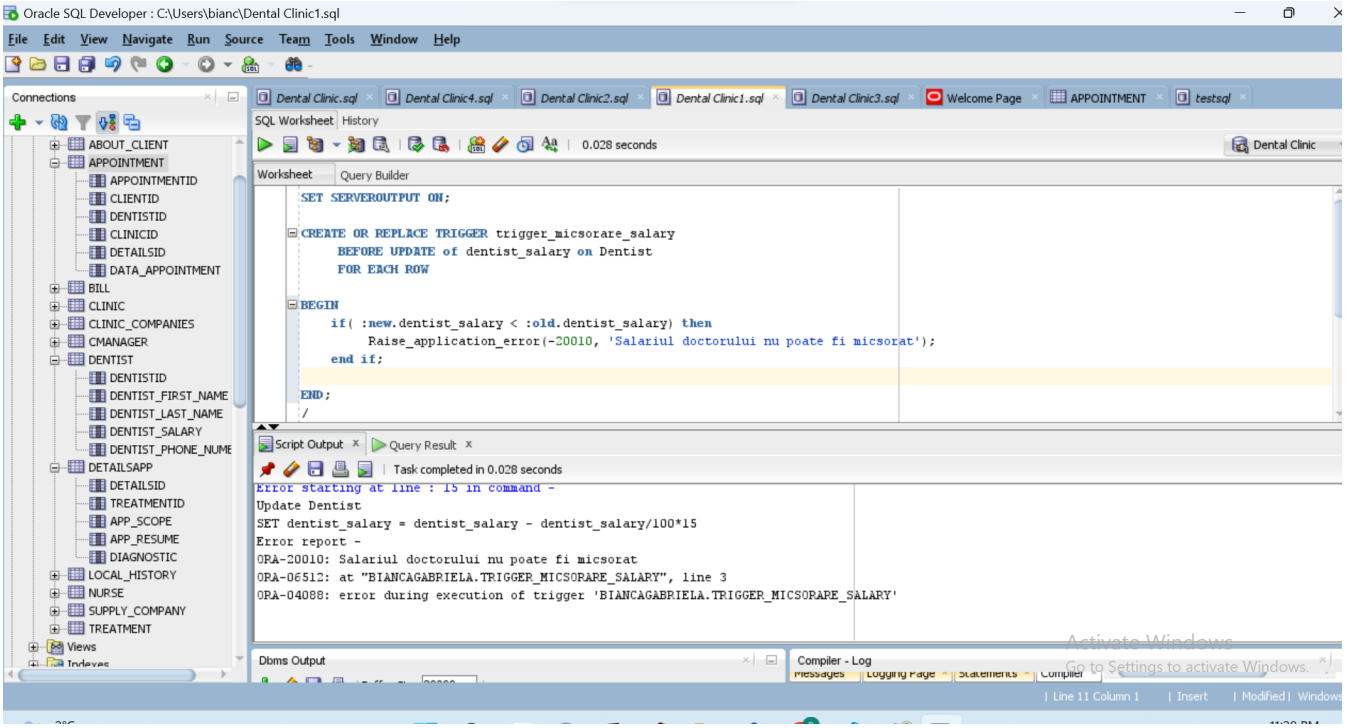
```
  FOR i in 1..11 LOOP
          insert into Nurse values(i, 'prenume' || i, 'nume' || i,
'07xxxxxxxx');
  END LOOP;

END;

DROP TRIGGER angajare_asistente;
```



11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

```
 SET SERVEROUTPUT ON;

CREATE OR REPLACE TRIGGER trigger_micsorare_salary
    BEFORE UPDATE of dentist_salary on Dentist
    FOR EACH ROW

BEGIN
   if( :new.dentist_salary < :old.dentist_salary) then
        Raise_application_error(-20010, 'Salariul doctorului nu poate
fi micsorat');
```

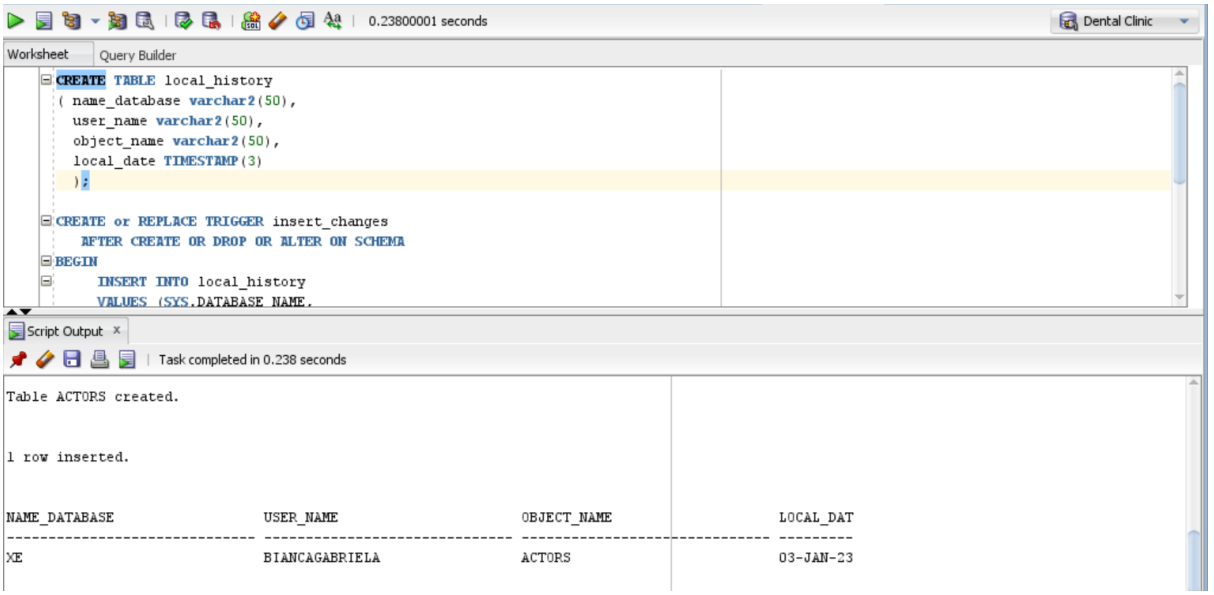
```
    end if;

END;
/
Update Dentist
SET dentist_salary = dentist_salary - dentist_salary/100*15;

SET SERVEROUTPUT OFF;
Drop trigger trigger_micsorare_salary;
```



12. Definiți un trigger de tip LDD. Declanșați trigger-ul.

```
  CREATE TABLE local_history
( name_database varchar2(50),
  user_name varchar2(50),
  object_name varchar2(50),
  local_date TIMESTAMP(3)
  );

CREATE or REPLACE TRIGGER insert_changes
  AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
```

```
    INSERT INTO local_history
    VALUES (SYS.DATABASE_NAME,
    SYS.LOGIN_USER, SYS.DICTIONARY_OBJ_NAME,
    SYSTIMESTAMP(3));
END;

CREATE TABLE actors (id_actor number(4), nume varchar(20));

INSERT INTO actors VALUES (1, 'DiCaprio');

SELECT * from local_history;
```



13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PACKAGE proiect_dentalClinic AS
                    PROCEDURE programari (input_client
About_Client.ClientId%Type, input_clinic Clinic.ClinicId%Type);
    PROCEDURE Premii(nr_appointments Integer);
                    FUNCTION clienti_dentist (i_prenume
dentist.dentist_first_name%TYPE,                i_nume
dentist.dentist_last_name%TYPE)RETURN NUMBER;
                    PROCEDURE Aparat_dentar (prenume_dentist
dentist.dentist_first_name%Type,                nume_dentist
dentist.dentist_last_name%type);
```

```
END proiect_dentalClinic;
/
CREATE OR REPLACE PACKAGE BODY proiect_dentalClinic AS
  ---Exercitiul 6: pentru un ClientId dat si un ClinicId dat, aflati cate
programari a avut clientul
  --- la clinica respectiva

  PROCEDURE programari
        (input_client About_Client.ClientId%Type,
         input_clinic Clinic.ClinicId%Type)

As
   Type vector is varray(20) of number(10);
   v vector := vector();
    Type tablou_indexat IS TABLE of Appointment%Rowtype Index
by Binary_Integer;
   t tablou_indexat;
   dim_t integer;
   ind integer;

BEGIN

   ind := 0;
   Delete from Bill;
   Delete from appointmentnurses;

   Delete
   From Appointment
   Where ClientId = input_client
    Returning AppointmentId, ClientId, DentistId, ClinicId, DetailsId,
Data_Appointment
   Bulk Collect into t;


   for i in 1..t.count() loop
       if t(i).ClinicId = input_clinic then
               v.extend();
               v(i) := i;
```

```
      end if;
   end loop;
    DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul ' || input_client || '
are ' || v.count || ' programari la clinica ' || input_clinic);

   t.delete;
   ROLLBACK;

END programari;

    ---Exercitiul 7: aflati numele managerilor a caror clinici au avut
peste un nr (dat) de programari si
   -- numele dentistului cu cele mai multe programari

   PROCEDURE Premii
   (nr_appointments Integer)

AS
   TYPE detalii_output IS RECORD (
   Nume Dentist.Dentist_First_Name%TYPE,
   Prenume Dentist.Dentist_Last_Name%TYPE,
   Nr_programari Int
   );

   output detalii_output;

    --- primul cursor parametrizat: imi parcurge managerii a caror
clinici au avut peste un nr de programari dat
   CURSOR c_manager (nr_appointments int) IS
SELECT        m.Manager_first_name,        m.Manager_last_name,
COUNT(a.appointmentId)
   From CManager m, Clinic c, Appointment a
   Where m.managerid = c.managerId
   and a.clinicId = c.clinicId
   Group by  m.Manager_first_name, m.Manager_last_name
   Having COUNT(a.appointmentId) > nr_appointments;
```

```
--- al doilea cursor neparametrizat: imi selecteaza dentistul numele si nr de programari
   CURSOR c_dentist IS
Select d.Dentist_first_name, d.Dentist_last_name, Count(a.appointmentID)
   from Dentist d, Appointment a
   where d.dentistId = a.dentistId
   group by d.Dentist_first_name, d.Dentist_last_name;

   vmax integer;
   Prenume dentist.dentist_first_name%TYPE;
   Nume dentist.dentist_last_name%TYPE;
   Nr_programari int;


BEGIN

   vmax := 0;
   Open c_manager(nr_appointments);
   LOOP
      FETCH c_manager into output;
      EXIT WHEN c_manager%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE('Felicitari managerului ' || output.Prenume || ' ' || output.Nume
                                    || ' pentru rezultatele obtinute, ' || output.Nr_programari || ' programari in anul 2022.' );

   END LOOP;
   Close c_manager;

   Open c_dentist;
   LOOP
      FETCH c_dentist into output;
      EXIT WHEN c_dentist%NOTFOUND;
      if output.nr_programari > vmax then
          vmax := output.nr_programari;
     end if;
   END LOOP;
```

```
   Close c_dentist;


   Open c_dentist;
   LOOP
      FETCH c_dentist into output;
      EXIT WHEN c_dentist%NOTFOUND;
      if output.nr_programari = vmax then
                  DBMS_OUTPUT.PUT_LINE('Felicitari  dentistului  ' ||
output.Nume ||
                  ' ' || output.Prenume || ' pentru numarul record de
programari, ' || vmax);
     end if;
   END LOOP;
   CLOSE c_dentist;
END Premii;

    ---Exercitiul 8: numarul de clienti minori ai unui dentist dat

    FUNCTION clienti_dentist
              (i_prenume  dentist.dentist_first_name%TYPE,  i_nume
dentist.dentist_last_name%TYPE)

RETURN NUMBER AS
   id_dentist dentist.dentistid%TYPE;
   id_client_cur about_client.clientId%TYPE;
   nr_clienti number;
   DENTISTNOTFOUND exception;

   d_cur int;

   CURSOR c_dentist IS
      select dentistId
      from dentist
       where dentist_last_name =i_nume  and dentist_first_name =
i_prenume;

   CURSOR c_client IS
```

```
        SELECT c.clientId
        From About_Client c, Appointment a, Dentist d
                    Where d.dentist_first_name = i_prenume and
d.dentist_last_name = i_nume
        and d.dentistId = a.dentistId and a.clientId = c.clientId and
         months_between( a.data_appointment, c.data_of_birth)/12 <
18;

BEGIN
   --- vedem daca exista acest doctor in baza de date
   id_dentist := -1;
   nr_clienti := 0;
   open c_dentist;
   loop
   Fetch c_dentist into d_cur;
   exit when c_dentist%notfound;
      id_dentist := d_cur;
   end loop;

   close c_dentist;

   if id_dentist = -1 then
       raise DENTISTNOTFOUND;
   end if;

   open c_client;
   loop
      fetch c_client into id_client_cur;
      exit when c_client%NOTFOUND;
         nr_clienti := nr_clienti + 1;
   end loop;
   close c_client;

   if nr_clienti = 0 then
      raise NO_DATA_FOUND;
   else
       Return nr_clienti;
   end if;
```

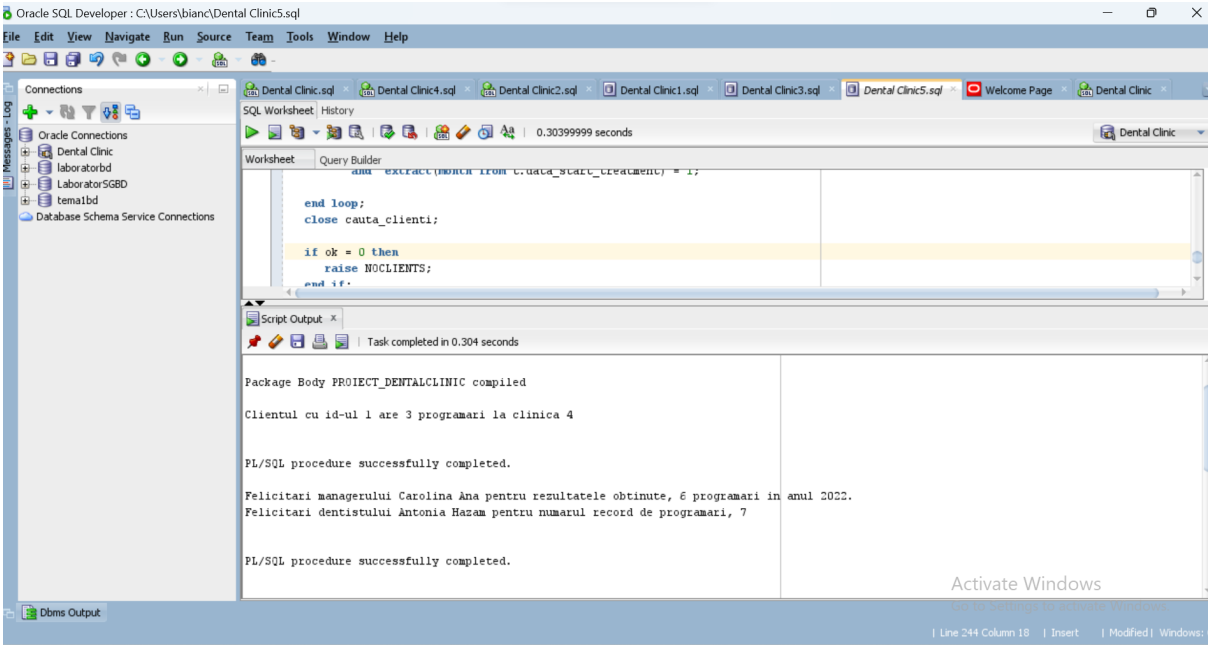
```
EXCEPTION
     WHEN DENTISTNOTFOUND then
            RAISE_APPLICATION_ERROR(-20003, 'Nu exista acest
dentist in baza de date');

    WHEN NO_DATA_FOUND then
           RAISE_APPLICATION_ERROR(-20004, 'Nu exista clienti
minori sau programari pentru doctorul dat');

END clienti_dentist;

          ---Exercitiul 9:     Pentru un dentist dat (prenume, nume),
afi?a?i clien?ii care pl?tesc
       --în rate aparatul dentar ?i clientul care ?i-a pus aparat în luna
ianuarie.

      procedure Aparat_dentar
                  (prenume_dentist    dentist.dentist_first_name%Type,
nume_dentist dentist.dentist_last_name%type)

AS
   Prenume About_Client.Client_First_name%TYPE;
   Nume About_Client.Client_Last_name%TYPE;
   ExistaDentist Dentist.dentistID%Type;
   client_cautat About_client.ClientId%Type;
   NODENTIST exception;
   NOCLIENTS exception;
   ok Number;

    -- pentru dentistul dat cursorul va trece prin clientii care platesc
in rate pentru aparat dentar

     CURSOR cauta_clienti(prenume_dentist varchar, nume_dentist
varchar) IS
       Select ac.Client_first_name, ac.Client_last_name
        From About_Client ac, Appointment a, DetailsApp da, Bill b,
Dentist d
```

```
        Where a.ClientId = ac.ClientId and da.DetailsId = a.DetailsId
and b.AppointmentId = a.AppointmentId
       and d.Dentist_first_name = 'Antonia' and d.Dentist_last_name
= 'Hazam' and  a.DentistId= d.DentistId
                  and  b.method_of_Pay  =  'instalment'  and  lower(
da.app_resume) like '%aparat dentar%'
       group by (ac.Client_first_name, ac.Client_last_name);


Begin

  -- Daca nu exista clinica
  ExistaDentist := -1;
  Select dentistId into ExistaDentist
  From Dentist
          Where  dentist_first_name  =  prenume_dentist  and
dentist_last_name = nume_dentist;

  if ExistaDentist = -1 then
       raise NODENTIST;
  end if;

  ok := 0;
  open cauta_clienti(prenume_dentist, nume_dentist);
  loop
    Fetch cauta_clienti into Prenume, Nume;
    Exit when cauta_clienti%NOTFOUND;
       ok := 1;
          DBMS_OUTPUT.PUT_LINE('Clientii care si-au pus aparat
pe care il platesc in rate sunt : ' || Prenume || ' ' || Nume);
    --- cautam clientul care a inceput tratamentul in luna ianuarie
       select ac.ClientId into client_cautat
             from About_Client ac, Appointment a, DetailsApp da,
Treatment t
                    where ac.Client_first_name = Prenume and
ac.Client_last_name = Nume
          and ac.ClientId = a.ClientId and a.DetailsId = da.DetailsId
and t.TreatmentId = da.TreatmentId
```

```
        and  extract(month from t.data_start_treatment) = 1;

   end loop;
   close cauta_clienti;

   if ok = 0 then
      raise NOCLIENTS;
   end if;

    DBMS_OUTPUT.PUT_LINE('Clientul care si-a pus aparat dentar
in luna ianuarie are id-ul ' || client_cautat);

EXCEPTION
   WHEN NO_DATA_FOUND then
        DBMS_OUTPUT.PUT_LINE('no data found: ' || SQLCODE || '
- ' ||SQLERRM);
   WHEN NODENTIST then
         RAISE_APPLICATION_ERROR(-20001, 'Dentistul cautat nu
se afla in baza de date');
   WHEN NOCLIENTS then
       RAISE_APPLICATION_ERROR(-20003, 'Nu exista clienti');
   WHEN TOO_MANY_ROWS then
        DBMS_OUTPUT.PUT_LINE('too many rows: ' || SQLCODE ||
' - ' ||SQLERRM);

END Aparat_dentar;

END proiect_dentalClinic;


execute proiect_dentalClinic.programari(1,4);
execute proiect_dentalClinic.premii(4);

BEGIN
       DBMS_OUTPUT.PUT_LINE('Numarul de clienti minori ai
dentistului dat este ' || proiect_dentalClinic.clienti_dentist('Antonia',
'Hazam'));
END ;
```

execute proiect_dentalClinic.aparat_dentar('Antonia', 'Hazam');

SET SERVEROUTPUT OFF;



14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

CREATE OR REPLACE PACKAGE proiect_dentalClinic2

AS

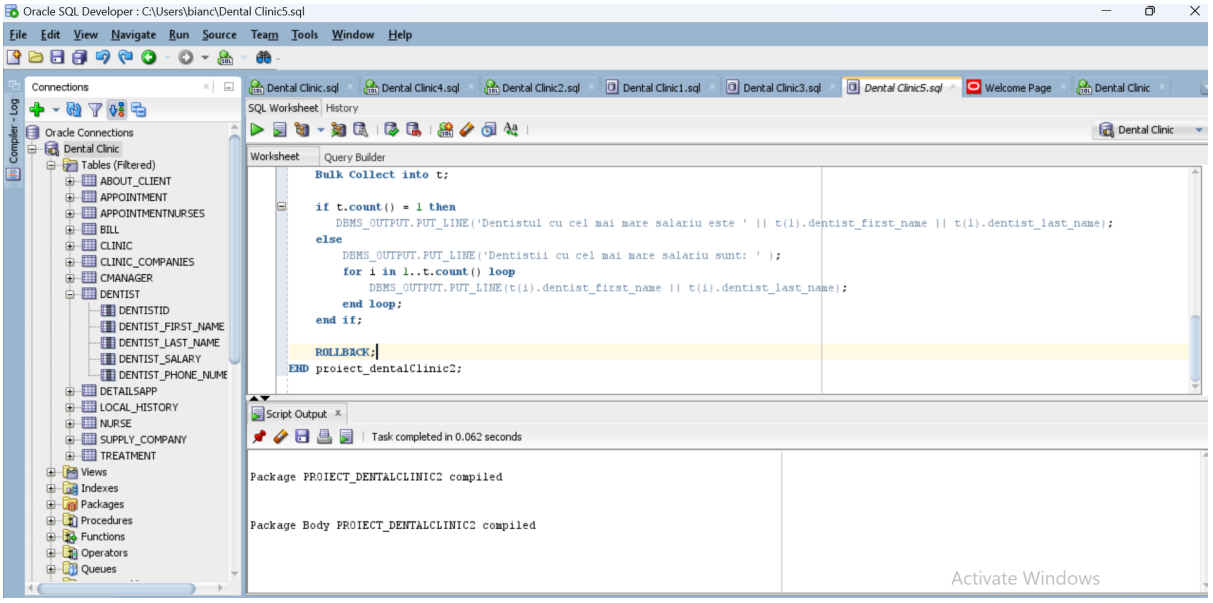
--Ex 6

Type vector is varray(20) of number(10);

Type tablou_indexat IS TABLE of Dentist%Rowtype Index by Binary_Integer;

--Ex 7

```
    TYPE detalii_output IS RECORD (
        Prenume Dentist.Dentist_First_Name%TYPE,
        Nume Dentist.Dentist_Last_Name%TYPE,
        Nr_programari Int
        );

    ---Exercitiul 8
END proiect_dentalClinic2;

CREATE OR REPLACE PACKAGE BODY proiect_dentalClinic2

AS
   t tablou_indexat;

Begin
   Delete
   From Dentist
   Where dentist_salary = ( select max(dentist_salary) from dentist)
            Returning dentistId, dentist_first_name, dentist_last_name,
dentist_salary, dentist_phone_number
   Bulk Collect into t;

   if t.count() = 1 then
      DBMS_OUTPUT.PUT_LINE('Dentistul cu cel mai mare salariu este '
|| t(1).dentist_first_name || t(1).dentist_last_name);
   else
       DBMS_OUTPUT.PUT_LINE('Dentistii cu cel mai mare salariu sunt:
' );
      for i in 1..t.count() loop
                        DBMS_OUTPUT.PUT_LINE(t(i).dentist_first_name  ||
t(i).dentist_last_name);
      end loop;
   end if;

   ROLLBACK;
END proiect_dentalClinic2;
```


Oracle SQL Developer : C:\Users\bianc\Dental Clinic5.sql
File Edit View Navigate Run Source Team Tools Window Help
Connections
Oracle Connections
Dental Clinic
Tables (Filtered)
ABOUT_CLIENT
APPOINTMENT
APPOINTMENTNURSES
BILL
CLINIC
CLINIC_COMPANIES
CMANAGER
DENTIST
DENTISTID
DENTIST_FIRST_NAME
DENTIST_LAST_NAME
DENTIST_SALARY
DENTIST_PHONE_NUME
DETAILSAPP
LOCAL_HISTORY
NURSE
SUPPLY_COMPANY
TREATMENT
Views
Indexes
Packages
Procedures
Functions
Operators
Queues
Dental Clinic.sql
Dental Clinic4.sql
Dental Clinic2.sql
Dental Clinic1.sql
Dental Clinic3.sql
Dental Clinic5.sql
Welcome Page
Dental Clinic
SQL Worksheet History
Worksheet Query Builder
Bulk Collect into t;
if t.count() = 1 then
DBMS_OUTPUT.PUT_LINE('Dentistul cu cel mai mare salariu este ' || t(1).dentist_first_name || t(1).dentist_last_name);
else
DBMS_OUTPUT.PUT_LINE('Dentistii cu cel mai mare salariu sunt: ');
for i in 1..t.count() loop
DBMS_OUTPUT.PUT_LINE(t(i).dentist_first_name || t(i).dentist_last_name);
end loop;
end if;
ROLLBACK;
END proiect_dentalClinic2;
Script Output
Task completed in 0.062 seconds
Package PROIECT_DENTALCLINIC2 compiled
Package Body PROIECT_DENTALCLINIC2 compiled
Activate Windows