BLOOD BANKS
DATABASE MANAGEMENT

St. Hazaparu Daria

Facultatea de Matematica si Informatica

Sisteme de gestiune a bazelor de date

Proiect final – documentatie

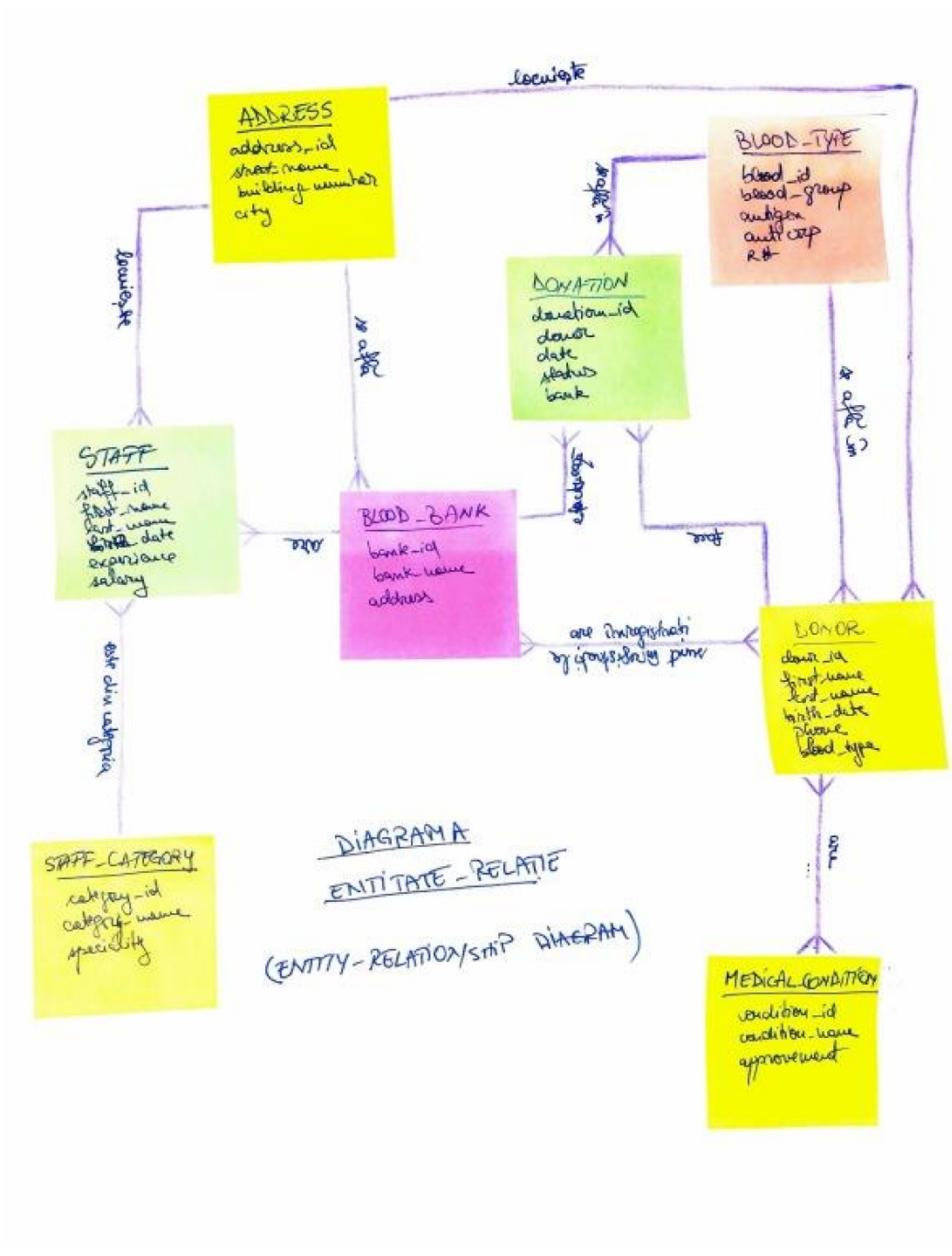
Prof. Gabriela Mihai

1)

Pentru acest proiect, am ales sa reprezint baza de date a unui sistem de banci destinate donatiilor de sange. Diagrama este compusa din 8 entitati si 2 tabele associative dupa cum urmeaza in imaginile de mai jos (imaginile 1 si 2).

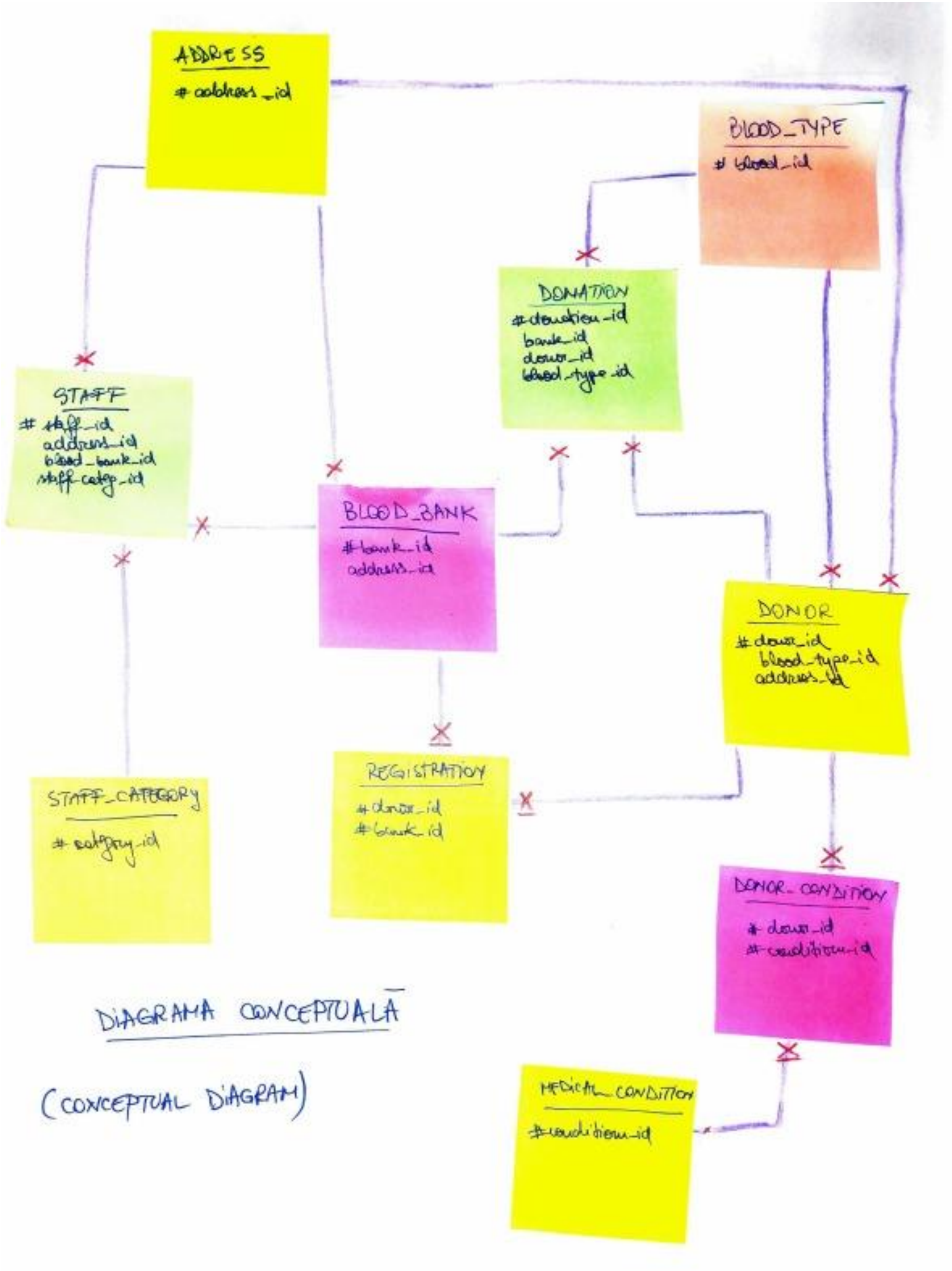
Baza de date contine mai multe **banci** aflate la **adrese** diferite. Adresele au cate un id si contin atat adresele bancilor, cat si ale angajatilor si ale donatorilor. La aceste banci lucreaza **angajati** care fac parte din **categorii de angajati** (doctori, asistente medicale, rezidenti in diferite specializari); lucreaza la o singura banca. Toti angajatii sunt si ei **donatori**. La banca sunt *inregistrati* donatori, care la randul lor pot fi inregistrati la mai multe banci. Un donator poate dona si la alte banci, la care nu este inregistrat. **Donatia** este inregistrata cu un id, donator, banca la care a fost efectuata colectarea si **grupa de sange. (**Donatorul are si el scris in detaliile lui grupa de sange.) Exista si conditii pentru a fi potrivit pentru donarea de sange, astfel ca exista **conditii medicale** despre care se stie daca se incadreaza in limite. Un donator poate avea *mai multe conditii* medicale.

2)


ADDRESS
address_id
street_name
building_number
city
BLOOD_TYPE
blood_id
blood_group
antigen
anticorp
RH
DONATION
donation_id
donor
date
status
bank
STAFF
staff_id
first_name
last_name
birth_date
experience
salary
BLOOD_BANK
bank_id
bank_name
address
DONOR
donor_id
first_name
last_name
birth_date
phone
blood_type
STAFF_CATEGORY
category_id
category_name
speciality
MEDICAL_CONDITION
condition_id
condition_name
approvement
locuieste
locuieste
se afla
se afla
se afla in
stocheaza
face
are
are inregistrati
este din categoria
are
DIAGRAMA
ENTITATE - RELATIE
(ENTITY-RELATIONSHIP DIAGRAM)

Imaginea 2

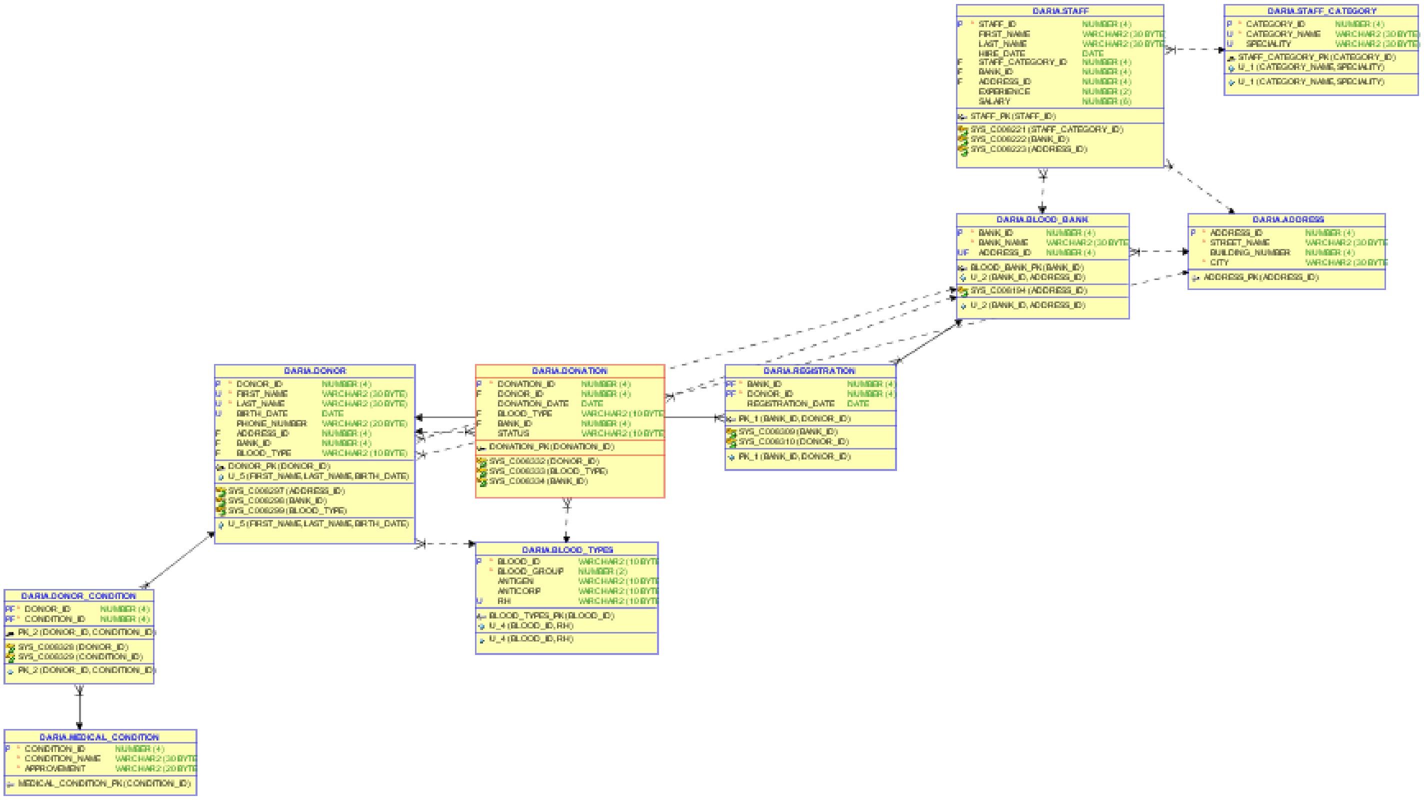
3)


ADDRESS
# address_id
BLOOD_TYPE
# blood_id
DONATION
# donation_id
bank_id
donor_id
blood_type_id
STAFF
# staff_id
address_id
blood_bank_id
staff-categ_id
BLOOD_BANK
# bank_id
address_id
DONOR
# donor_id
blood_type_id
address_id
STAFF_CATEGORY
# category_id
REGISTRATION
# donor_id
# bank_id
DONOR_CONDITION
# donor_id
# condition_id
MEDICAL_CONDITION
# condition_id
DIAGRAMA CONCEPTUALĂ
(CONCEPTUAL DIAGRAM)

Imaginea 3

4)

Dupa ce am implementat tabelele, diagrama generata arata ca in imaginea de mai jos (3).

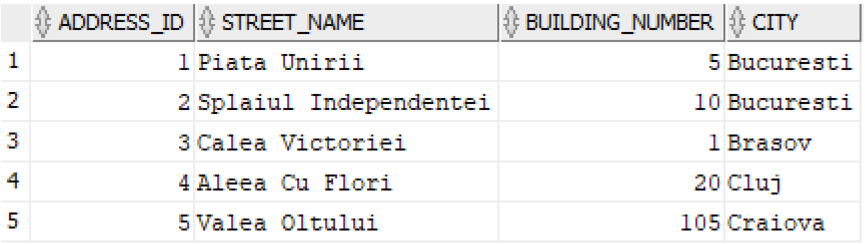

DARIA.STAFF
P * STAFF_ID NUMBER (4)
FIRST_NAME VARCHAR2 (30 BYTE)
LAST_NAME VARCHAR2 (30 BYTE)
HIRE_DATE DATE
F STAFF_CATEGORY_ID NUMBER (4)
F BANK_ID NUMBER (4)
F ADDRESS_ID NUMBER (4)
EXPERIENCE NUMBER (2)
SALARY NUMBER (6)
STAFF_PK (STAFF_ID)
SYS_C008221 (STAFF_CATEGORY_ID)
SYS_C008222 (BANK_ID)
SYS_C008223 (ADDRESS_ID)
DARIA.STAFF_CATEGORY
P * CATEGORY_ID NUMBER (4)
U * CATEGORY_NAME VARCHAR2 (30 BYTE)
U SPECIALITY VARCHAR2 (30 BYTE)
STAFF_CATEGORY_PK (CATEGORY_ID)
U_1 (CATEGORY_NAME, SPECIALITY)
U_1 (CATEGORY_NAME, SPECIALITY)
DARIA.BLOOD_BANK
P * BANK_ID NUMBER (4)
* BANK_NAME VARCHAR2 (30 BYTE)
UF ADDRESS_ID NUMBER (4)
BLOOD_BANK_PK (BANK_ID)
U_2 (BANK_ID, ADDRESS_ID)
SYS_C008194 (ADDRESS_ID)
U_2 (BANK_ID, ADDRESS_ID)
DARIA.ADDRESS
P * ADDRESS_ID NUMBER (4)
* STREET_NAME VARCHAR2 (30 BYTE)
BUILDING_NUMBER NUMBER (4)
* CITY VARCHAR2 (30 BYTE)
ADDRESS_PK (ADDRESS_ID)
DARIA.DONOR
P * DONOR_ID NUMBER (4)
U * FIRST_NAME VARCHAR2 (30 BYTE)
U * LAST_NAME VARCHAR2 (30 BYTE)
U BIRTH_DATE DATE
PHONE_NUMBER VARCHAR2 (20 BYTE)
F ADDRESS_ID NUMBER (4)
F BANK_ID NUMBER (4)
F BLOOD_TYPE VARCHAR2 (10 BYTE)
DONOR_PK (DONOR_ID)
U_5 (FIRST_NAME, LAST_NAME, BIRTH_DATE)
SYS_C008297 (ADDRESS_ID)
SYS_C008298 (BANK_ID)
SYS_C008299 (BLOOD_TYPE)
U_5 (FIRST_NAME, LAST_NAME, BIRTH_DATE)
DARIA.DONATION
P * DONATION_ID NUMBER (4)
F DONOR_ID NUMBER (4)
DONATION_DATE DATE
F BLOOD_TYPE VARCHAR2 (10 BYTE)
F BANK_ID NUMBER (4)
STATUS VARCHAR2 (10 BYTE)
DONATION_PK (DONATION_ID)
SYS_C008332 (DONOR_ID)
SYS_C008333 (BLOOD_TYPE)
SYS_C008334 (BANK_ID)
DARIA.REGISTRATION
PF * BANK_ID NUMBER (4)
PF * DONOR_ID NUMBER (4)
REGISTRATION_DATE DATE
PK_1 (BANK_ID, DONOR_ID)
SYS_C008309 (BANK_ID)
SYS_C008310 (DONOR_ID)
PK_1 (BANK_ID, DONOR_ID)
DARIA.BLOOD_TYPES
P * BLOOD_ID VARCHAR2 (10 BYTE)
* BLOOD_GROUP NUMBER (2)
ANTIGEN VARCHAR2 (10 BYTE)
ANTICORP VARCHAR2 (10 BYTE)
U RH VARCHAR2 (10 BYTE)
BLOOD_TYPES_PK (BLOOD_ID)
U_4 (BLOOD_ID, RH)
U_4 (BLOOD_ID, RH)
DARIA.DONOR_CONDITION
PF * DONOR_ID NUMBER (4)
PF * CONDITION_ID NUMBER (4)
PK_2 (DONOR_ID, CONDITION_ID)
SYS_C008328 (DONOR_ID)
SYS_C008329 (CONDITION_ID)
PK_2 (DONOR_ID, CONDITION_ID)
DARIA.MEDICAL_CONDITION
P * CONDITION_ID NUMBER (4)
* CONDITION_NAME VARCHAR2 (30 BYTE)
* APPROVEMENT VARCHAR2 (20 BYTE)
MEDICAL_CONDITION_PK (CONDITION_ID)

Imaginea 3

5)

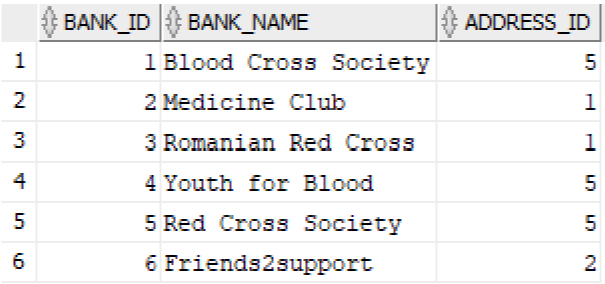
In continuare sunt imagini care demonstreaza ca am populat tabelele definite mai sus.

Address



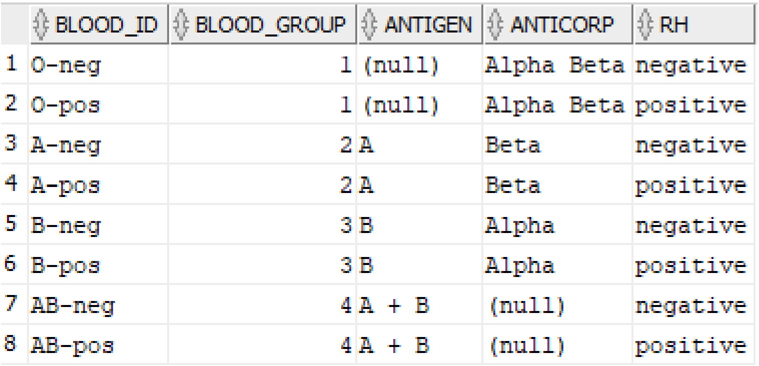
	ADDRESS_ID	STREET_NAME	BUILDING_NUMBER	CITY
1	1	Piata Unirii	5	Bucuresti
2	2	Splaiul Independentei	10	Bucuresti
3	3	Calea Victoriei	1	Brasov
4	4	Aleea Cu Flori	20	Cluj
5	5	Valea Oltului	105	Craiova

Blood_bank



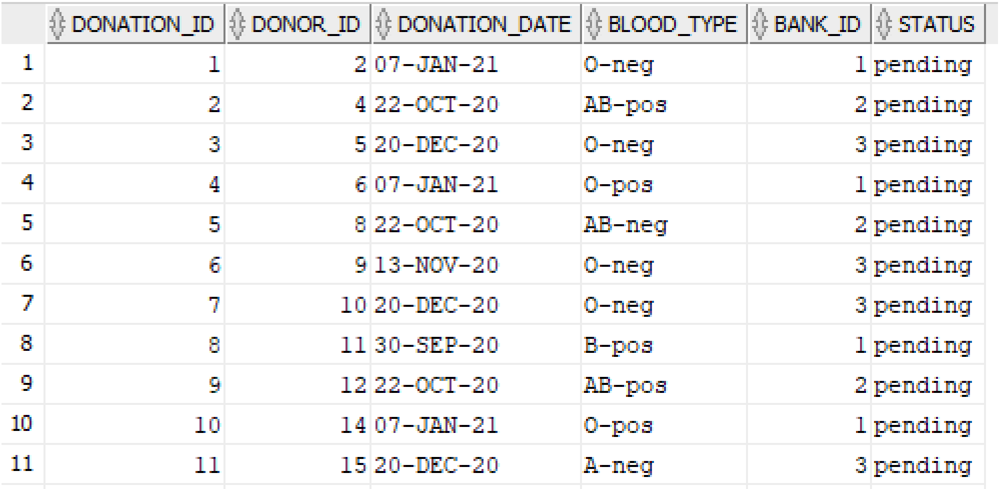
	BANK_ID	BANK_NAME	ADDRESS_ID
1	1	Blood Cross Society	5
2	2	Medicine Club	1
3	3	Romanian Red Cross	1
4	4	Youth for Blood	5
5	5	Red Cross Society	5
6	6	Friends2support	2

Blood_types



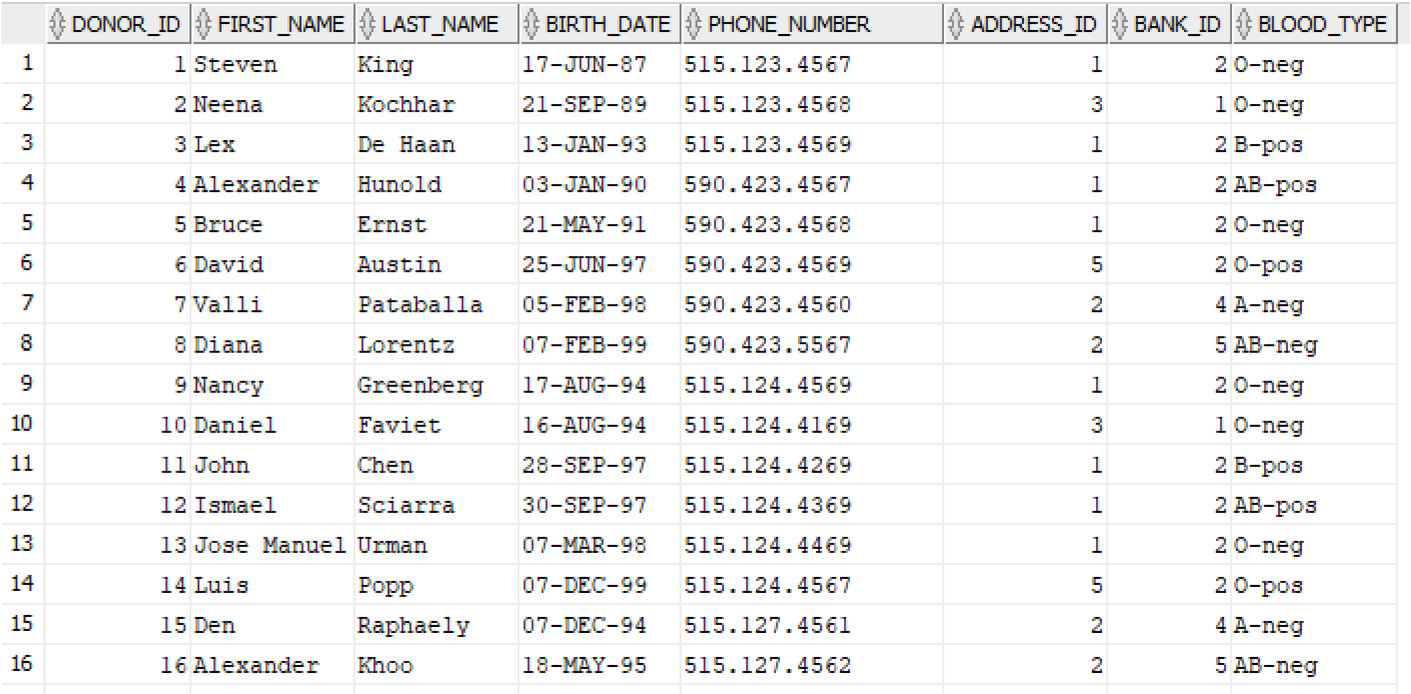
	BLOOD_ID	BLOOD_GROUP	ANTIGEN	ANTICORP	RH
1	O-neg	1	(null)	Alpha Beta	negative
2	O-pos	1	(null)	Alpha Beta	positive
3	A-neg	2	A	Beta	negative
4	A-pos	2	A	Beta	positive
5	B-neg	3	B	Alpha	negative
6	B-pos	3	B	Alpha	positive
7	AB-neg	4	A + B	(null)	negative
8	AB-pos	4	A + B	(null)	positive

Donation



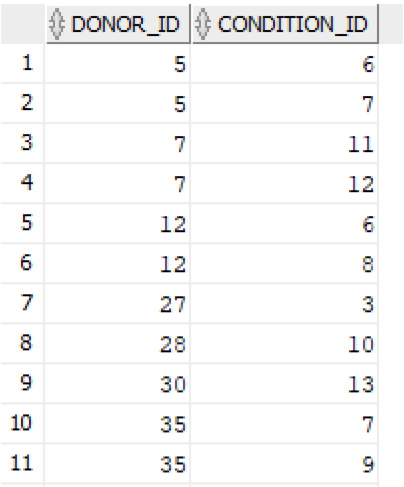
	DONATION_ID	DONOR_ID	DONATION_DATE	BLOOD_TYPE	BANK_ID	STATUS
1	1	2	07-JAN-21	O-neg	1	pending
2	2	4	22-OCT-20	AB-pos	2	pending
3	3	5	20-DEC-20	O-neg	3	pending
4	4	6	07-JAN-21	O-pos	1	pending
5	5	8	22-OCT-20	AB-neg	2	pending
6	6	9	13-NOV-20	O-neg	3	pending
7	7	10	20-DEC-20	O-neg	3	pending
8	8	11	30-SEP-20	B-pos	1	pending
9	9	12	22-OCT-20	AB-pos	2	pending
10	10	14	07-JAN-21	O-pos	1	pending
11	11	15	20-DEC-20	A-neg	3	pending

Donor



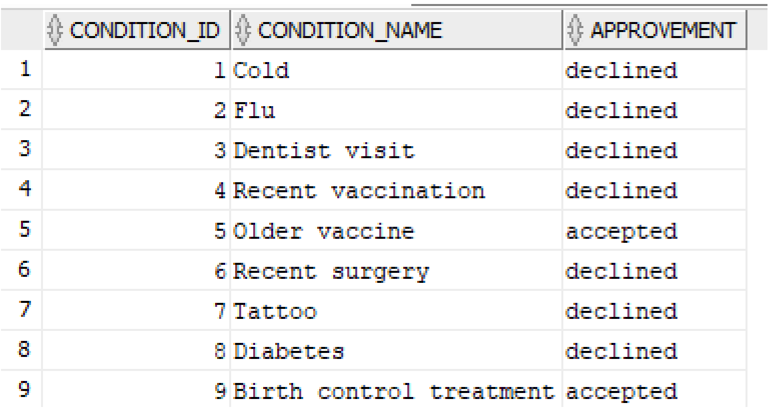
	DONOR_ID	FIRST_NAME	LAST_NAME	BIRTH_DATE	PHONE_NUMBER	ADDRESS_ID	BANK_ID	BLOOD_TYPE
1	1	Steven	King	17-JUN-87	515.123.4567	1	2	O-neg
2	2	Neena	Kochhar	21-SEP-89	515.123.4568	3	1	O-neg
3	3	Lex	De Haan	13-JAN-93	515.123.4569	1	2	B-pos
4	4	Alexander	Hunold	03-JAN-90	590.423.4567	1	2	AB-pos
5	5	Bruce	Ernst	21-MAY-91	590.423.4568	1	2	O-neg
6	6	David	Austin	25-JUN-97	590.423.4569	5	2	O-pos
7	7	Valli	Pataballa	05-FEB-98	590.423.4560	2	4	A-neg
8	8	Diana	Lorentz	07-FEB-99	590.423.5567	2	5	AB-neg
9	9	Nancy	Greenberg	17-AUG-94	515.124.4569	1	2	O-neg
10	10	Daniel	Faviet	16-AUG-94	515.124.4169	3	1	O-neg
11	11	John	Chen	28-SEP-97	515.124.4269	1	2	B-pos
12	12	Ismael	Sciarra	30-SEP-97	515.124.4369	1	2	AB-pos
13	13	Jose Manuel	Urman	07-MAR-98	515.124.4469	1	2	O-neg
14	14	Luis	Popp	07-DEC-99	515.124.4567	5	2	O-pos
15	15	Den	Raphaely	07-DEC-94	515.127.4561	2	4	A-neg
16	16	Alexander	Khoo	18-MAY-95	515.127.4562	2	5	AB-neg

Donor_condition



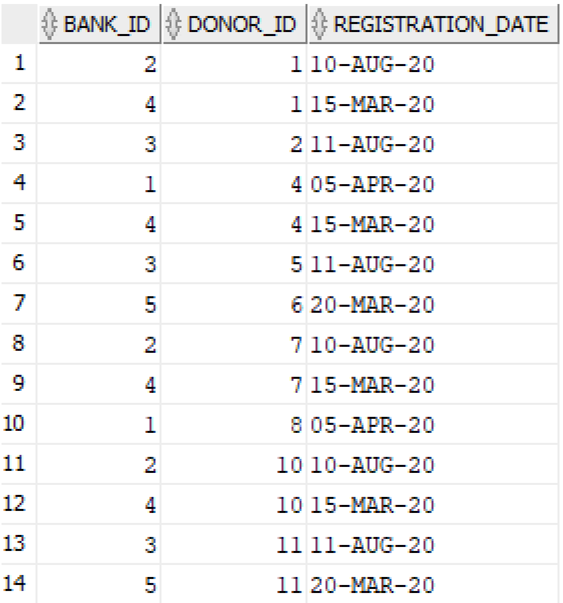
	DONOR_ID	CONDITION_ID
1	5	6
2	5	7
3	7	11
4	7	12
5	12	6
6	12	8
7	27	3
8	28	10
9	30	13
10	35	7
11	35	9

Medical_condition



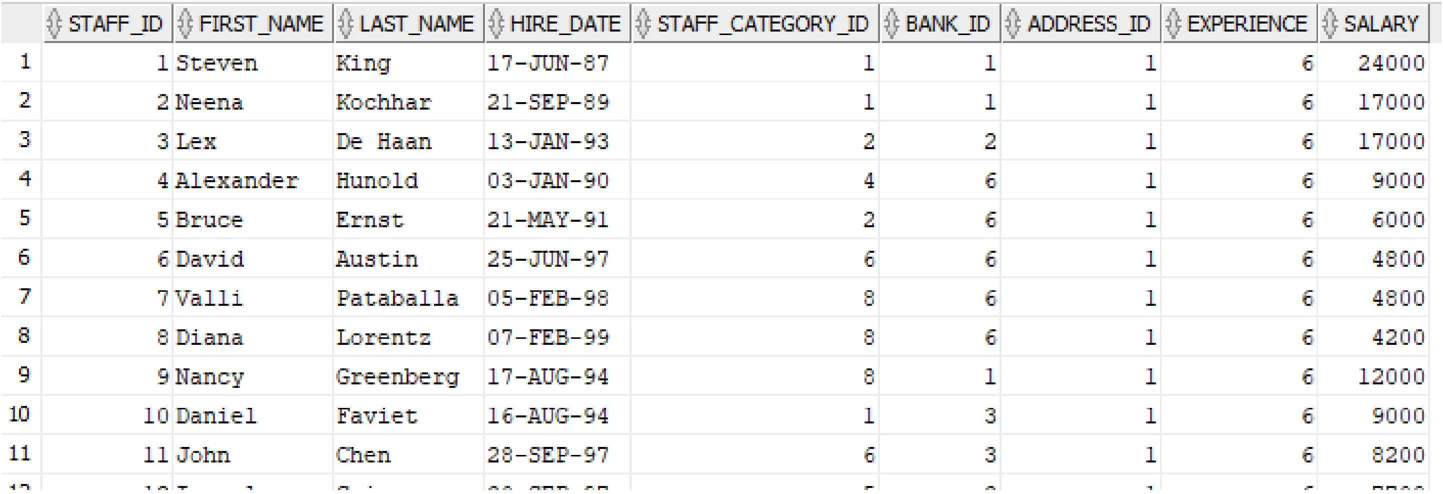
	CONDITION_ID	CONDITION_NAME	APPROVEMENT
1	1	Cold	declined
2	2	Flu	declined
3	3	Dentist visit	declined
4	4	Recent vaccination	declined
5	5	Older vaccine	accepted
6	6	Recent surgery	declined
7	7	Tattoo	declined
8	8	Diabetes	declined
9	9	Birth control treatment	accepted

Registration



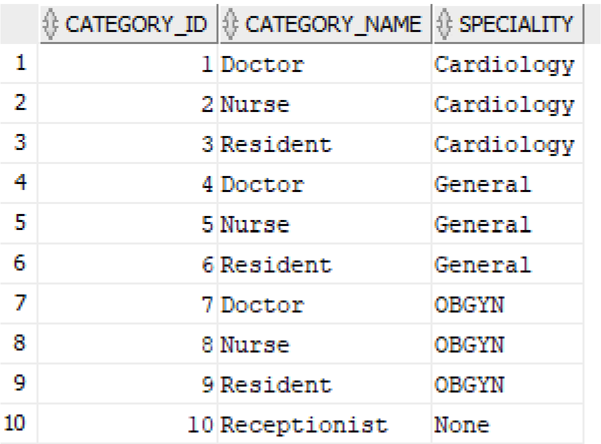
	BANK_ID	DONOR_ID	REGISTRATION_DATE
1	2	1	10-AUG-20
2	4	1	15-MAR-20
3	3	2	11-AUG-20
4	1	4	05-APR-20
5	4	4	15-MAR-20
6	3	5	11-AUG-20
7	5	6	20-MAR-20
8	2	7	10-AUG-20
9	4	7	15-MAR-20
10	1	8	05-APR-20
11	2	10	10-AUG-20
12	4	10	15-MAR-20
13	3	11	11-AUG-20
14	5	11	20-MAR-20

Staff



	STAFF_ID	FIRST_NAME	LAST_NAME	HIRE_DATE	STAFF_CATEGORY_ID	BANK_ID	ADDRESS_ID	EXPERIENCE	SALARY
1	1	Steven	King	17-JUN-87	1	1	1	6	24000
2	2	Neena	Kochhar	21-SEP-89	1	1	1	6	17000
3	3	Lex	De Haan	13-JAN-93	2	2	1	6	17000
4	4	Alexander	Hunold	03-JAN-90	4	6	1	6	9000
5	5	Bruce	Ernst	21-MAY-91	2	6	1	6	6000
6	6	David	Austin	25-JUN-97	6	6	1	6	4800
7	7	Valli	Pataballa	05-FEB-98	8	6	1	6	4800
8	8	Diana	Lorentz	07-FEB-99	8	6	1	6	4200
9	9	Nancy	Greenberg	17-AUG-94	8	1	1	6	12000
10	10	Daniel	Faviet	16-AUG-94	1	3	1	6	9000
11	11	John	Chen	28-SEP-97	6	3	1	6	8200

Staff_category



	CATEGORY_ID	CATEGORY_NAME	SPECIALITY
1	1	Doctor	Cardiology
2	2	Nurse	Cardiology
3	3	Resident	Cardiology
4	4	Doctor	General
5	5	Nurse	General
6	6	Resident	General
7	7	Doctor	OBGYN
8	8	Nurse	OBGYN
9	9	Resident	OBGYN
10	10	Receptionist	None

6) Subprogram care utilizeaza tipuri de colectii sudiate

Sa se creeze un tabel care contine cate o lista de angajati pentru fiecare banca. Folosind o procedura stocata ce primeste ca parametru adresa id introdusa de la tastatura, sa se afiseze specializarile care se pot gasi la adresa data.

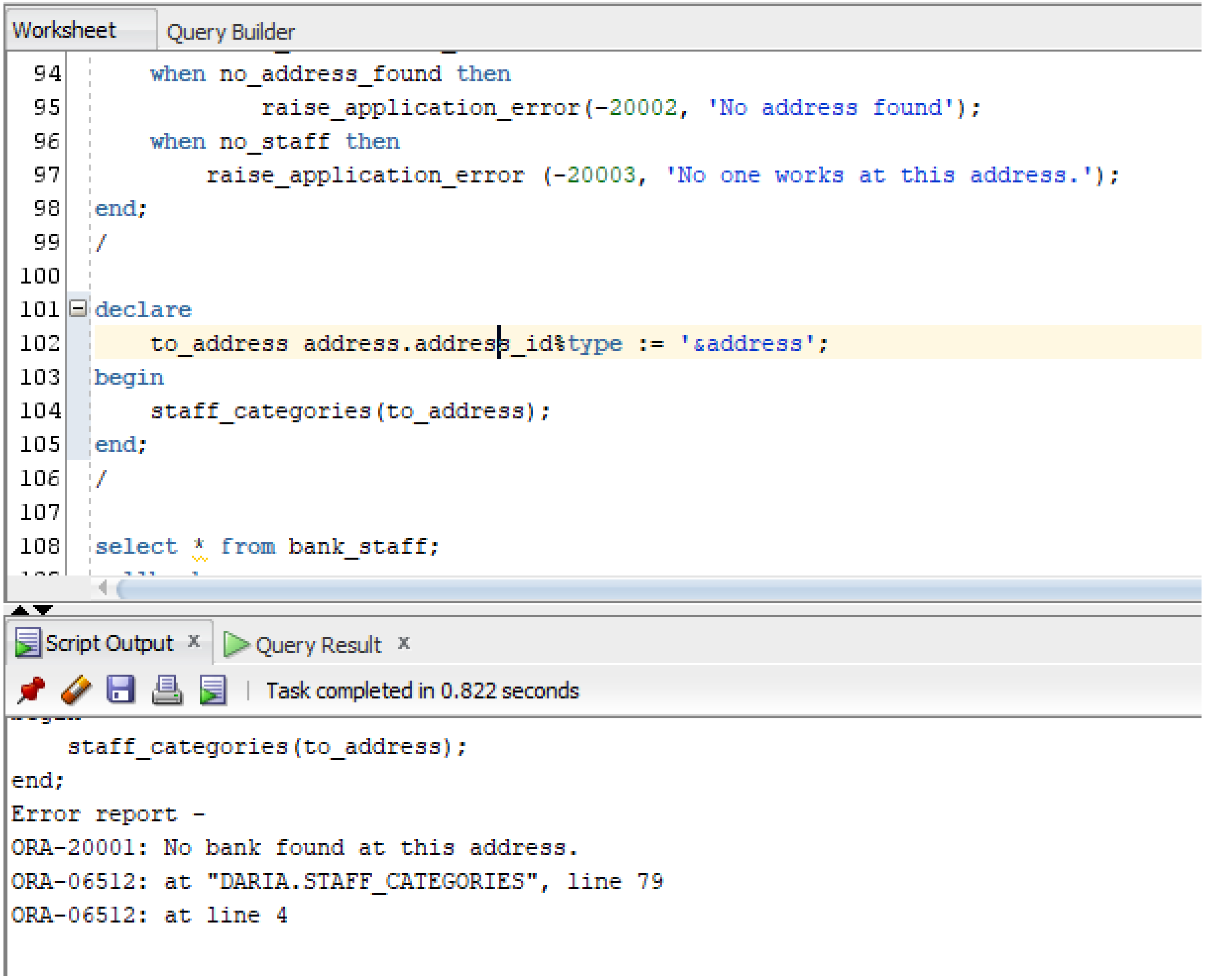
Pentru rezolvarea cerintei, am utilizat tabele (nested table) pentru a retine lista angajatilor (si a o adauga in tabelul bank_staff), pentru a o transforma in lista de id uri de categorii de angajati, iar pe baza acestui tabel, pentru a crea un tabel care tine o singura data fiecare specializare.

Am pus in evidenta faptul ca se poate crea un tabel in care o coloanal poate fi o lista si faptul ca tabelele sunt un bun instrument atunci cand avem mai multe variabile care pot fi identice si trebuie afisate o singura data.

```
create or replace procedure staff_categories (
    to_address address.address_id%type
) as
    nr_address number(4);
    lista_emp number_list := number_list();
    lista_bank number_list := number_list();
    lista_spec char_list := char_list();
    spec STAFF_CATEGORY.SPECIALITY%type;
    ind number(4) := 0;
    is_in_list boolean;
    no_bank_found exception;
    no_address_found exception;
    no_staff exception;
begin
    select count(*) into nr_address
    from address
    where address_id = to_address;

    if nr_address = 0 then
        raise no_address_found;
    end if;

    select bank_id
    bulk collect into lista_bank
    from blood_bank
    where address_id = to_address;

    if lista_bank.count() = 0 then
        raise no_bank_found;
    end if;
```

```
for i in lista_bank.first..lista_bank.last
loop
    select staff_category_id
    bulk collect into lista_emp
    from staff
    where bank_id = lista_bank(i);

    insert into bank_staff
    values (lista_bank(i), lista_emp);

    if lista_emp.count() = 0 then
        dbms_output.put_line('No one works at this bank -> ' || lista_bank(i));
    else
        for j in lista_emp.first..lista_emp.last
        loop
            select speciality into spec
            from staff_category
            where category_id = lista_emp(j);

            is_in_list := true;
            if lista_spec.count() <> 0 then
                for k in lista_spec.first..lista_spec.last
                loop
                    --dbms_output.put_line(lista_spec(k));
                    if lista_spec(k) = spec then
                        is_in_list := false;
                    end if;
                end loop;
            end if;
                if is_in_list = true then
                                    ind := ind + 1;
                        lista_spec.extend();
                        lista_spec(ind) := spec;
                    end if;
            end loop;
        end if;

    end loop;

    for i in lista_spec.first..lista_spec.last
    loop
        if lista_spec(i) <> 'None' then
            dbms_output.put_line(lista_spec(i));
        end if;
    end loop;
exception
    when no_bank_found then
        raise_application_error (-20001, 'No bank found at this address.');
    when no_address_found then
            raise_application_error(-20002, 'No address found');
    when no_staff then
        raise_application_error (-20003, 'No one works at this address.');
end;
/
```

Cu adresa 4

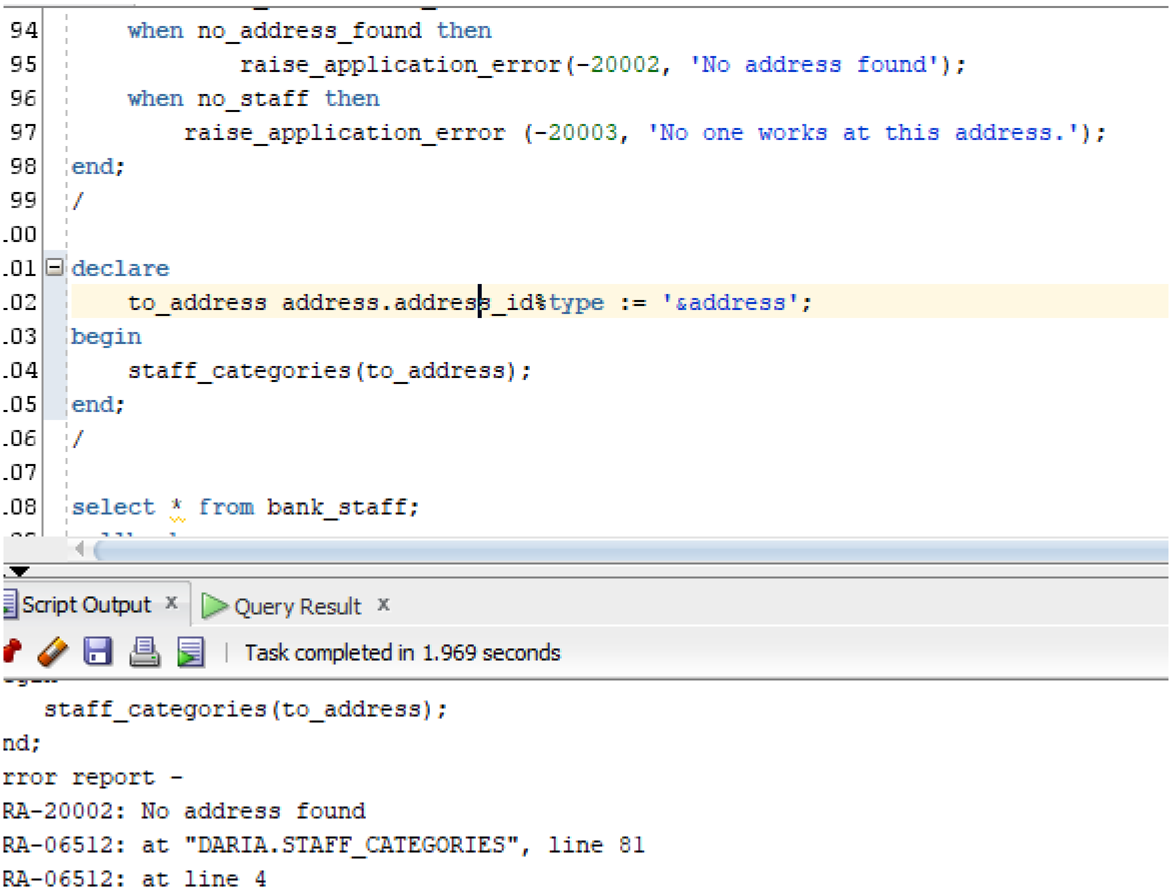


```
    when no_address_found then
            raise_application_error(-20002, 'No address found');
    when no_staff then
        raise_application_error (-20003, 'No one works at this address.');
end;
/

declare
    to_address address.address_id%type := '&address';
begin
    staff_categories(to_address);
end;
/

select * from bank_staff;
```

Task completed in 0.822 seconds

```
    staff_categories(to_address);
end;
Error report -
ORA-20001: No bank found at this address.
ORA-06512: at "DARIA.STAFF_CATEGORIES", line 79
ORA-06512: at line 4
```

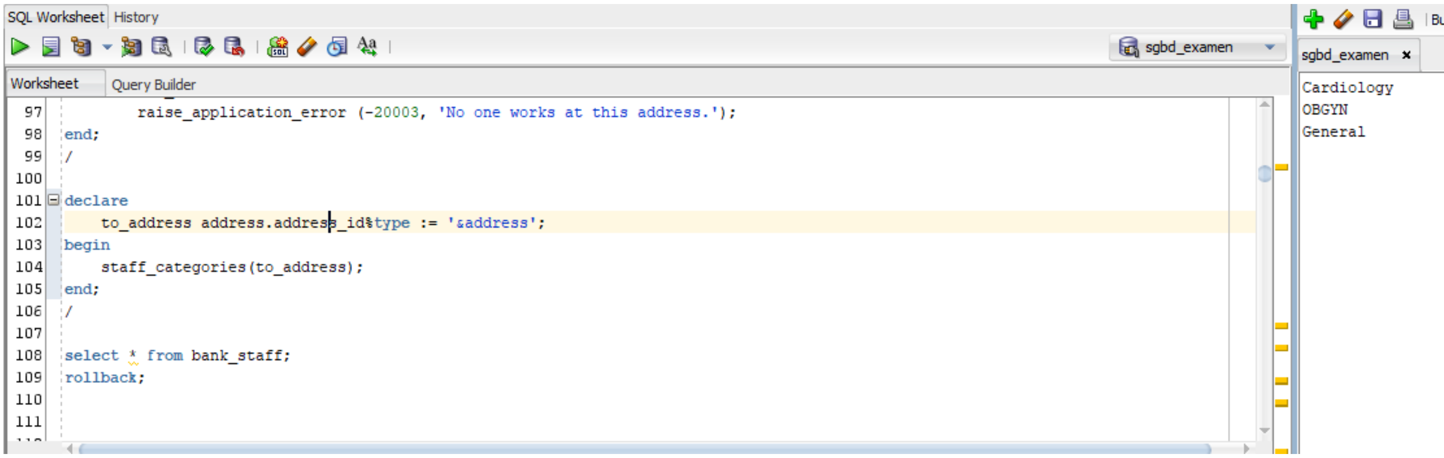
Cu adresa 9



```
    when no_address_found then
            raise_application_error(-20002, 'No address found');
    when no_staff then
        raise_application_error (-20003, 'No one works at this address.');
end;
/

declare
    to_address address.address_id%type := '&address';
begin
    staff_categories(to_address);
end;
/

select * from bank_staff;
```

```
   staff_categories(to_address);
nd;
rror report -
RA-20002: No address found
RA-06512: at "DARIA.STAFF_CATEGORIES", line 81
RA-06512: at line 4
```

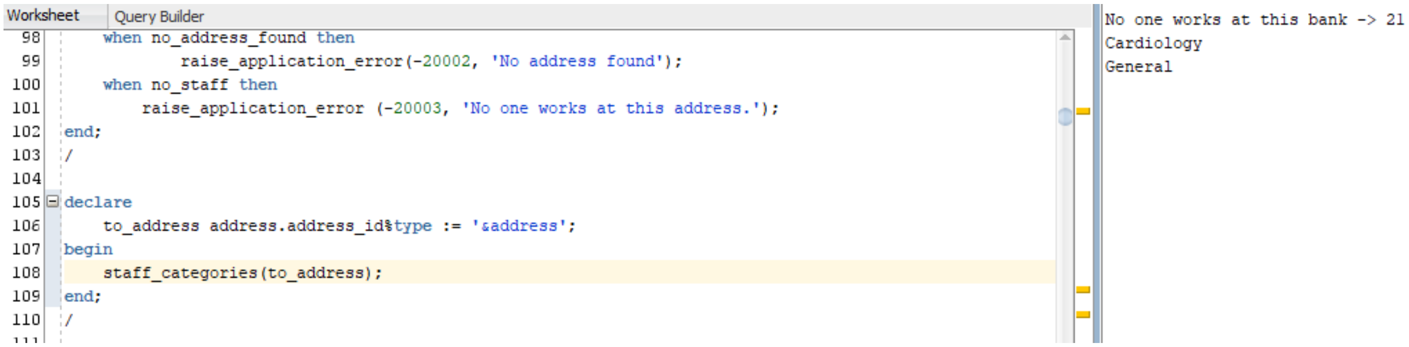
Cu adresa 5



```
        raise_application_error (-20003, 'No one works at this address.');
end;
/

declare
    to_address address.address_id%type := '&address';
begin
    staff_categories(to_address);
end;
/

select * from bank_staff;
rollback;
```

```
Cardiology
OBGYN
General
```

Cu adresa 1



```
    when no_address_found then
            raise_application_error(-20002, 'No address found');
    when no_staff then
        raise_application_error (-20003, 'No one works at this address.');
end;
/

declare
    to_address address.address_id%type := '&address';
begin
    staff_categories(to_address);
end;
/
```

```
No one works at this bank -> 21
Cardiology
General
```

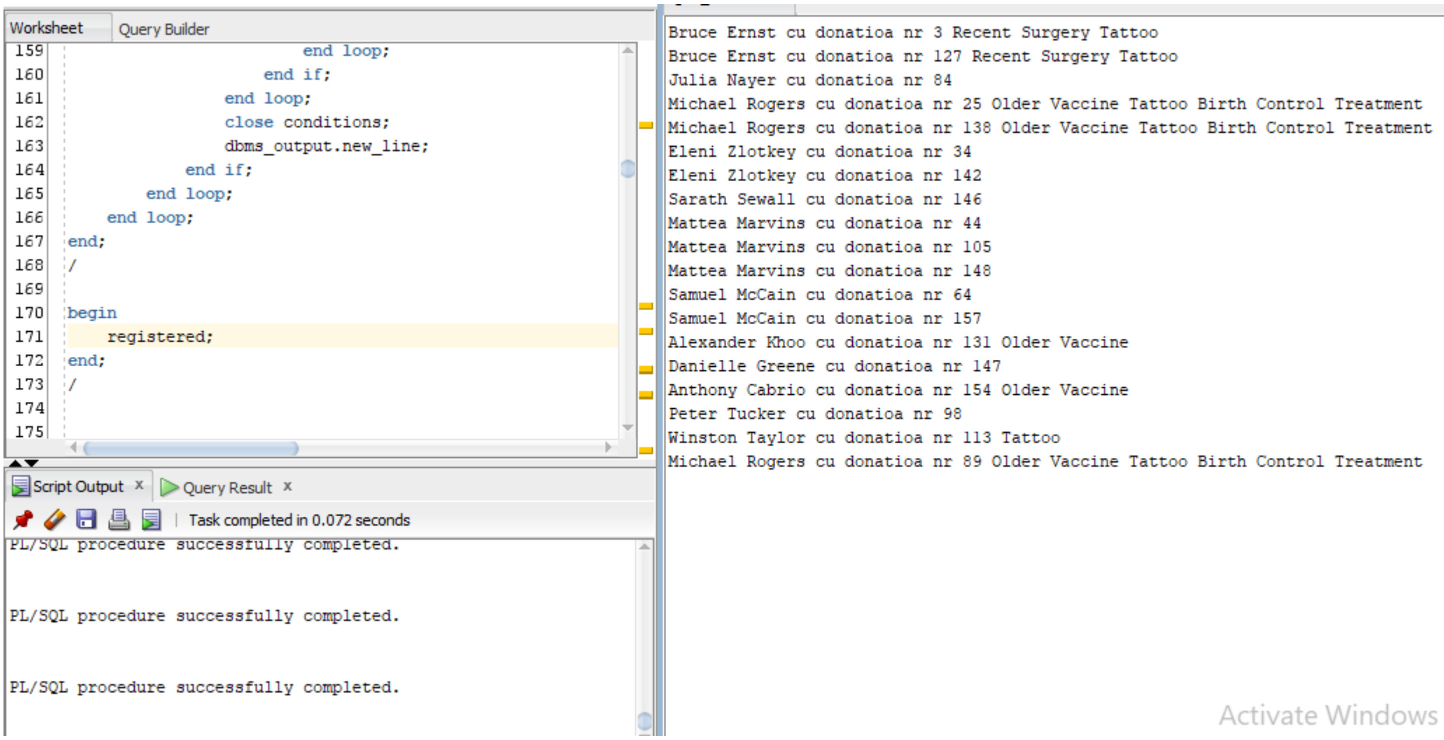
7) Subprogram care utilizeaza tipuri de cursori studiati

Sa se afiseze numele, prenumele, donatia si conditiile medicale ale donatorilor care au facut donatii la bancile la care sunt inregistrati.

Pentru a rezolva cerinta, am folosit mai multe tipuri de cursoare: pentru parcurgerea inregistrarilor dintre banci si donatori – ciclu cursor, pentru parcurgerea donatiilor – cursor cu subcereri, pentru parcurgerea conditiilor fiecarui donator – expresie cursor.

```
create or replace procedure registered
is
  TYPE refcursor IS REF CURSOR;
  cursor conditions is
    select donor_id, d.condition_id,
      cursor (select condition_name
          from medical_condition
          where condition_id = d.condition_id)
    from donor_condition d;

  cursor donor_reg is
    select bank_id, donor_id
    from registration;

  v_cursor refcursor;
  v_donor_id donor.donor_id%type;
  v_condition_id MEDICAL_CONDITION.CONDITION_ID%type;
  v_condition_name MEDICAL_CONDITION.CONDITION_NAME%type;
  f_name donor.first_name%type;
  l_name donor.last_name%type;
begin
  for i in donor_reg
  loop
    exit when donor_reg%notfound;
    for j in (select bank_id, donor_id, donation_id from donation)
    loop
```

```
            if i.bank_id = j.bank_id and i.donor_id = j.donor_id then
                select first_name, last_name
                into f_name, l_name
                from donor
                where donor_id = i.donor_id;

                dbms_output.put(f_name || ' ' || l_name || ' cu donatioa nr ' || j.donation_id);
                open conditions;
                loop
                    fetch conditions into v_donor_id, v_condition_id, v_cursor;
                    exit when conditions%notfound;
                    if v_donor_id = i.donor_id then
                        loop
                            fetch v_cursor into v_condition_name;
                            exit when v_cursor%notfound;
                            dbms_output.put(' ' || v_condition_name);
                        end loop;
                    end if;
                end loop;
                close conditions;
                dbms_output.new_line;
            end if;
        end loop;
    end loop;
end;
/
```



```
                        end loop;
                    end if;
                end loop;
                close conditions;
                dbms_output.new_line;
            end if;
        end loop;
    end loop;
end;
/

begin
    registered;
end;
/
```

Task completed in 0.072 seconds

```
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

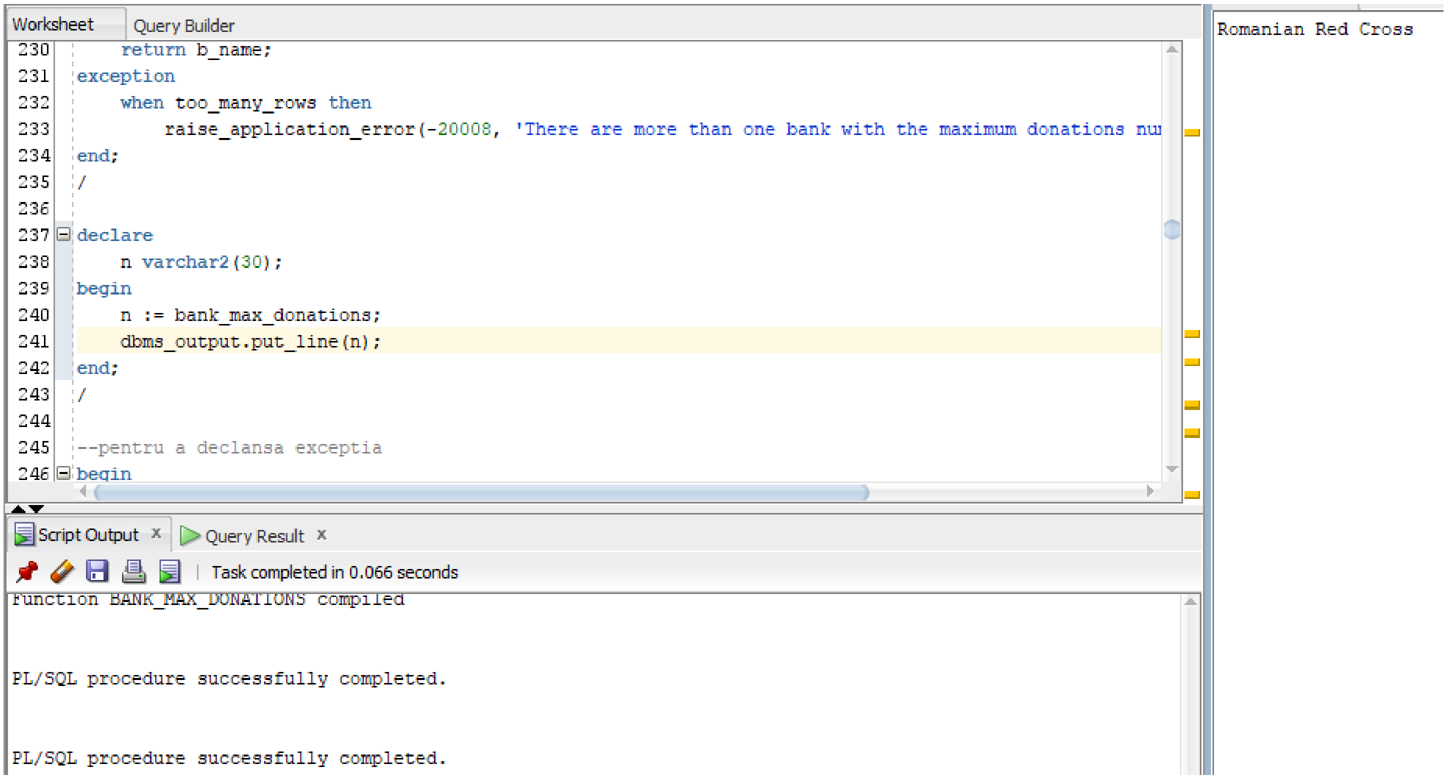
```
Bruce Ernst cu donatioa nr 3 Recent Surgery Tattoo
Bruce Ernst cu donatioa nr 127 Recent Surgery Tattoo
Julia Nayer cu donatioa nr 84
Michael Rogers cu donatioa nr 25 Older Vaccine Tattoo Birth Control Treatment
Michael Rogers cu donatioa nr 138 Older Vaccine Tattoo Birth Control Treatment
Eleni Zlotkey cu donatioa nr 34
Eleni Zlotkey cu donatioa nr 142
Sarath Sewall cu donatioa nr 146
Mattea Marvins cu donatioa nr 44
Mattea Marvins cu donatioa nr 105
Mattea Marvins cu donatioa nr 148
Samuel McCain cu donatioa nr 64
Samuel McCain cu donatioa nr 157
Alexander Khoo cu donatioa nr 131 Older Vaccine
Danielle Greene cu donatioa nr 147
Anthony Cabrio cu donatioa nr 154 Older Vaccine
Peter Tucker cu donatioa nr 98
Winston Taylor cu donatioa nr 113 Tattoo
Michael Rogers cu donatioa nr 89 Older Vaccine Tattoo Birth Control Treatment
```

8) Functie care utilizeaza cel putin 3 tabele

Sa se creeze o functie care returneaza numele bancii la care au fost efectuate cele mai multe donatii de sange de tip 0 negativ.

Pentru a rezolva cerinta, am folosit tabelele **BOOD_TYPES** (pentru id ul sangelui cerut), **DONATION** (pentru a numara donatiile si obtine id ul bancii) si **BLOOD_BANK** (pentru numele bancii).

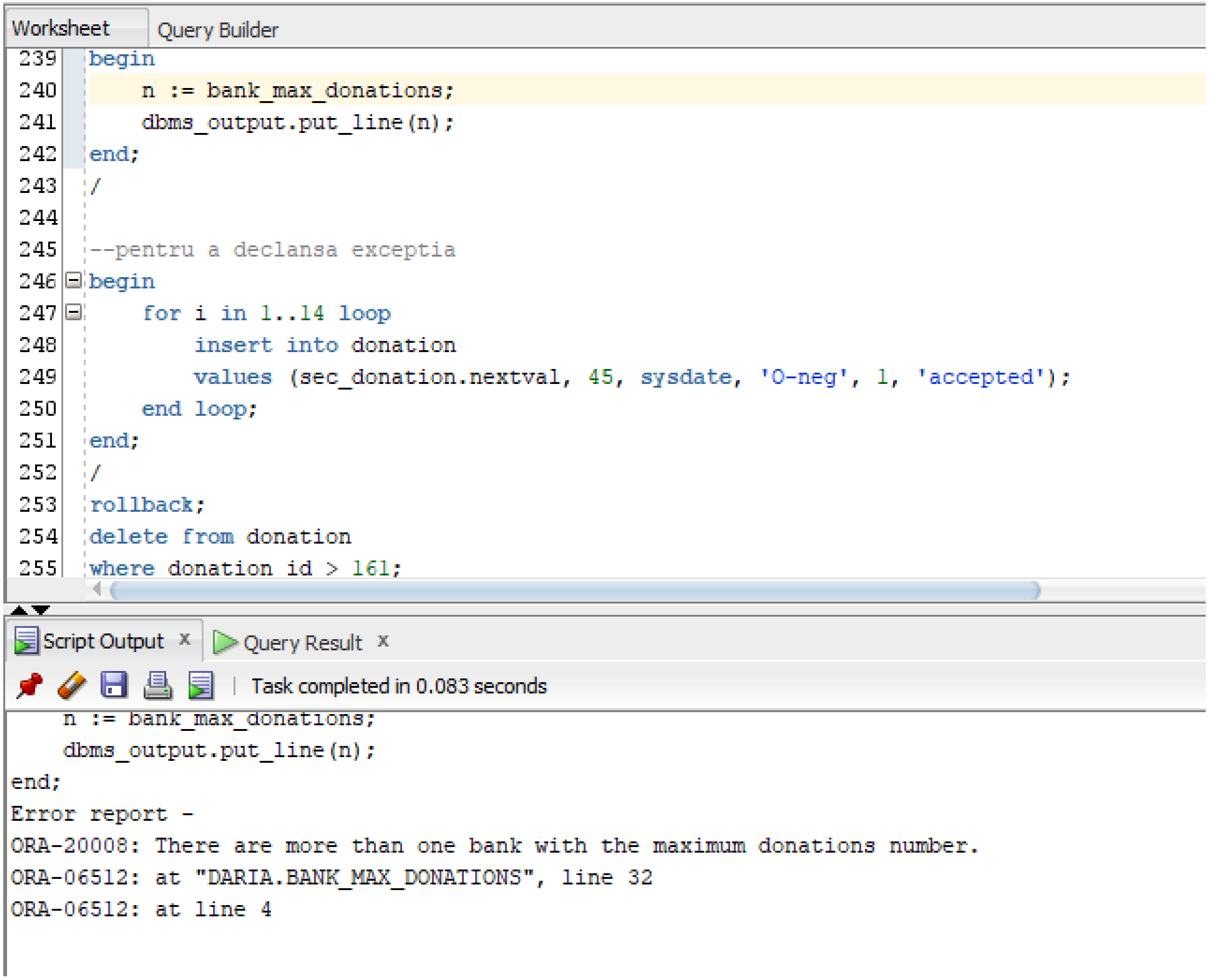
```
create or replace function bank_max_donations
return varchar2
is
    bank blood_bank.bank_id%type;
    blood blood_types.blood_id%type;
    b_name varchar2(30);
begin

    select blood_id
    into blood
    from blood_types
    where Rh = 'negative' and blood_id like 'O%';

    select bank_id
    into bank
    from donation
    where blood_type = blood
    group by bank_id
    having count(*) = ( select max(count(*))
                        from donation
                        where blood_type = blood
                        group by bank_id);

    select bank_name
    into b_name
    from blood_bank
    where bank_id = bank;


    return b_name;
exception
    when too_many_rows then
        raise_application_error(-20008, 'There are more than one bank with the maximum donations
number.');
end;
/
```



```
    return b_name;
exception
    when too_many_rows then
        raise_application_error(-20008, 'There are more than one bank with the maximum donations nu
end;
/

declare
    n varchar2(30);
begin
    n := bank_max_donations;
    dbms_output.put_line(n);
end;
/

--pentru a declansa exceptia
begin
```

```
Function BANK_MAX_DONATIONS compiled


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

```
Romanian Red Cross
```



```
begin
    n := bank_max_donations;
    dbms_output.put_line(n);
end;
/

--pentru a declansa exceptia
begin
    for i in 1..14 loop
        insert into donation
        values (sec_donation.nextval, 45, sysdate, '0-neg', 1, 'accepted');
    end loop;
end;
/
rollback;
delete from donation
where donation_id > 161;
```

Task completed in 0.083 seconds

```
    n := bank_max_donations;
    dbms_output.put_line(n);
end;
Error report -
ORA-20008: There are more than one bank with the maximum donations number.
ORA-06512: at "DARIA.BANK_MAX_DONATIONS", line 32
ORA-06512: at line 4
```

9) Procedura care utilizeaza cel putin 5 tabele

Sa se creeze o procedura care afiseaza donatiile de un tip dat ca parametru care s-au facut la bancile la care sunt inregistrati donatori cu o conditie data ca parametru si banca la care s-a facut donatia.

Pentru a rezolva cerinta, am folosit tabelele **BLOOD_TYPES, DONATION, DONOR, MEDICAL_CONDITION**, *DONOR_CONDITION*, *REGISTRATION*, **BLOOD_BANK**.

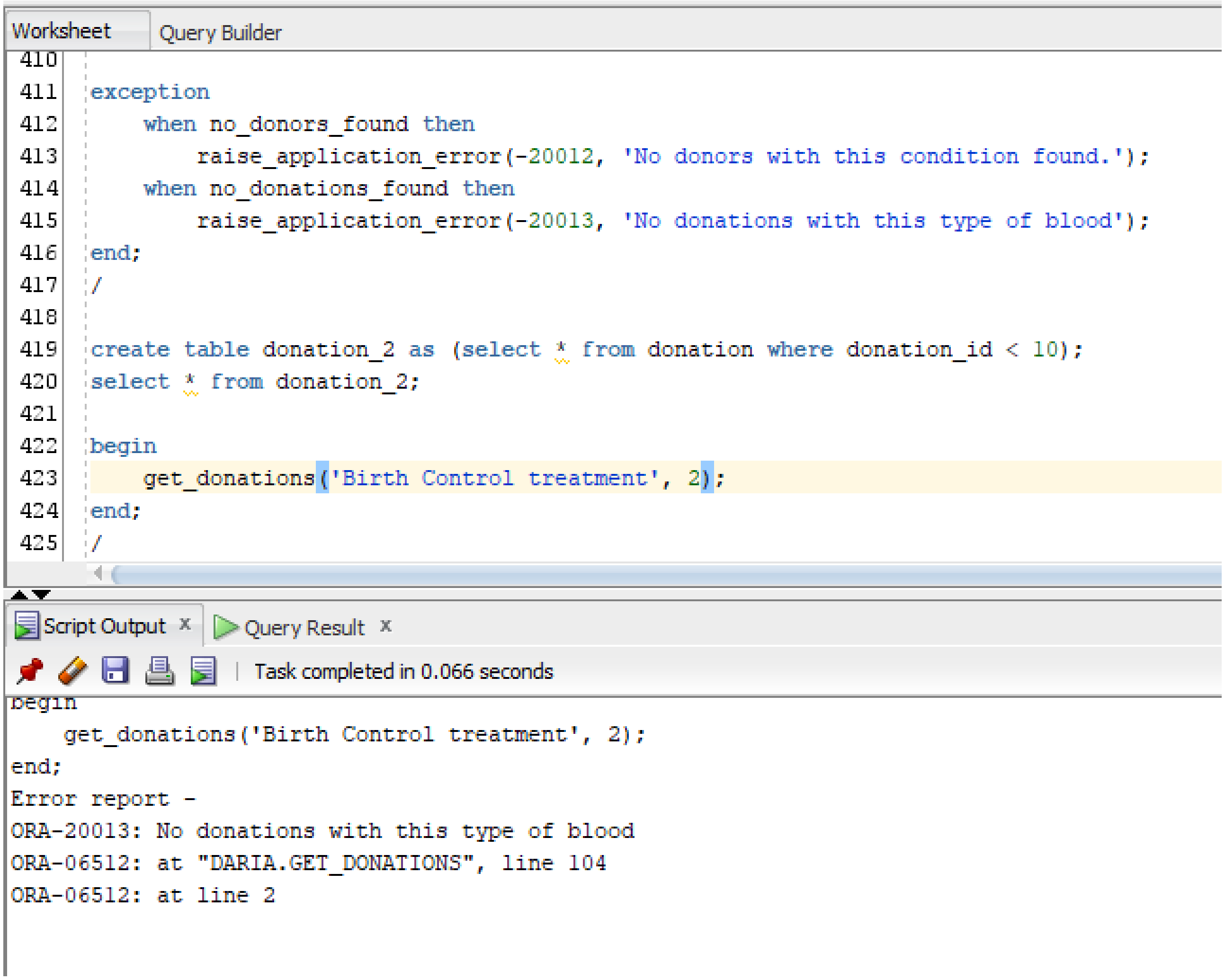
Pentru a evidentia exceptiile, am creat un tabel donation_2 cu mai putine linii care sa se incadreze in cerintele tratate de exceptii.

```
create or replace procedure get_donations
    (condition medical_condition.condition_name%type,
     blood blood_types.blood_group%type)
is
    blood_code char_list := get_blood_id(blood);
    condition_code medical_condition.condition_id%type := get_condition_id(condition);
    donations number_list := number_list();
    donors number_list := number_list();
    banks number_list := number_list();
    nr number;
    mt number;
    ind number := 0;
    b_name blood_bank.bank_name%type;

    no_donations_found exception;
    no_donors_found exception;
    no_banks_found exception;
begin

    dbms_output.put_line('Donations of blood type ' || blood || ' made at banks where people with ' || condition || ' are registred');
    dbms_output.put_line('----------------------------');

    for i in (select * from donation_2)
    loop
    for j in blood_code.first..blood_code.last
    loop
        if i.blood_type = blood_code(j) then
            donations.extend;
            donations(donations.last) := i.donation_id;
        end if;
    end loop;
end loop;

if donations.count() = 0 then
    raise no_donations_found;
end if;

select donor_id
bulk collect into donors
from donor
where donor_id in (select donor_id from donor_condition where condition_id = condition_code);

if donors.count() = 0 then
    raise no_donors_found;
end if;

for i in donors.first..donors.last
loop
    for j in (select * from registration where donor_id = donors(i))
    loop
        banks.extend();
        banks(banks.last) := j.bank_id;
    end loop;
end loop;
```

```
    for i in donations.first..donations.last
    loop
        select bank_id
        into mt
        from donation_2
        where donation_id = donations(i);

        for j in banks.first..banks.last
        loop
            if banks(j) = mt then
                select bank_name
                into b_name
                from blood_bank
                where bank_id = banks(j);

                dbms_output.put_line('Donation ' || donations(i) || ' from bank ' || b_name);
                ind := ind + 1;
                exit;
            end if;
        end loop;
    end loop;

    if ind = 0 then
        dbms_output.put_line('No donations.');
    end if;

exception
    when no_donors_found then
        raise_application_error(-20012, 'No donors with this condition found.');
    when no_donations_found then
        raise_application_error(-20013, 'No donations with this type of blood');
```

end;

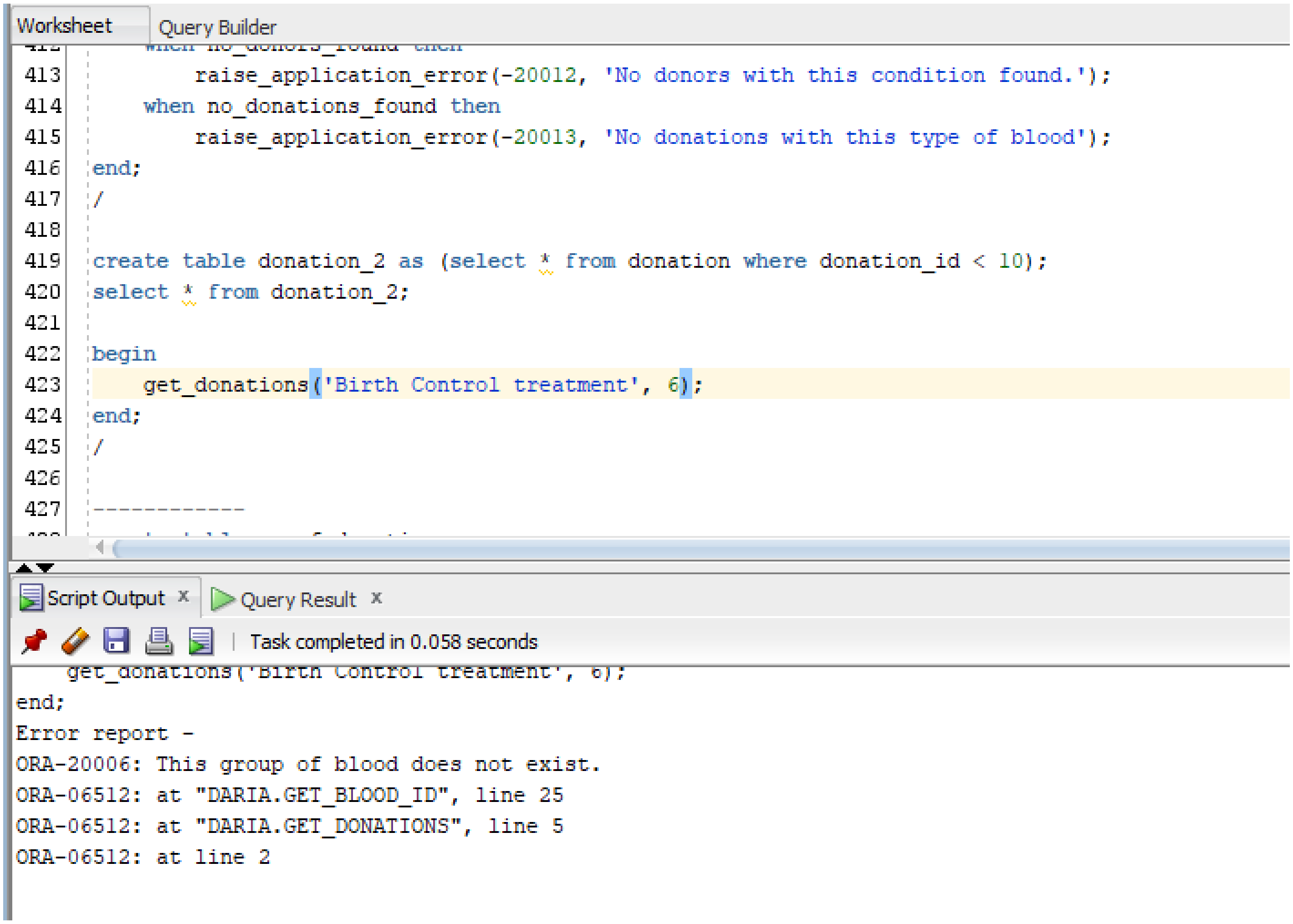
/



```

exception
    when no_donors_found then
        raise_application_error(-20012, 'No donors with this condition found.');
    when no_donations_found then
        raise_application_error(-20013, 'No donations with this type of blood');
end;
/

create table donation_2 as (select * from donation where donation_id < 10);
select * from donation_2;

begin
    get_donations('Birth Control treatment', 2);
end;
/
```

Task completed in 0.066 seconds

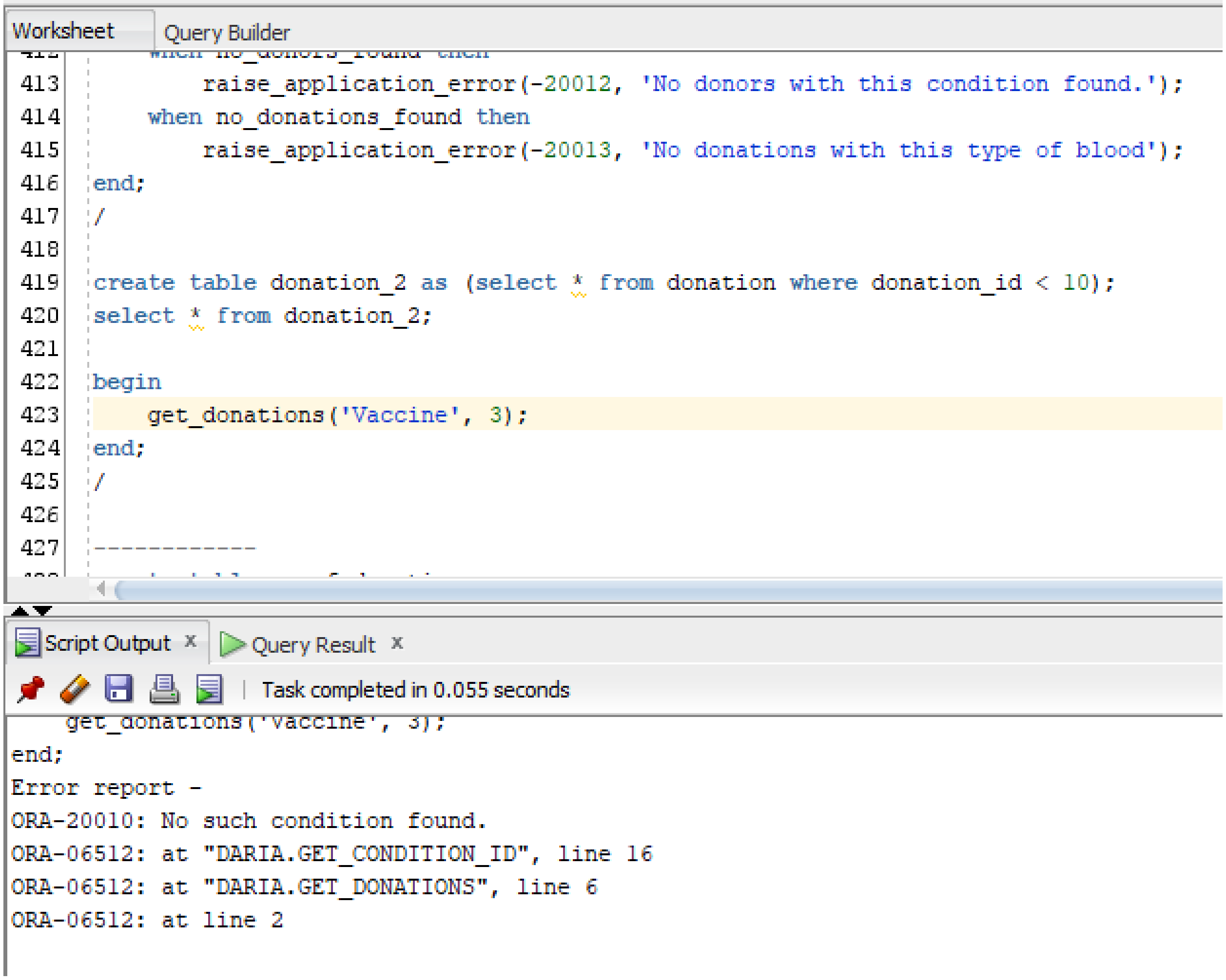
```
begin
    get_donations('Birth Control treatment', 2);
end;
Error report -
ORA-20013: No donations with this type of blood
ORA-06512: at "DARIA.GET_DONATIONS", line 104
ORA-06512: at line 2
```



```
    when no_donors_found then
        raise_application_error(-20012, 'No donors with this condition found.');
    when no_donations_found then
        raise_application_error(-20013, 'No donations with this type of blood');
end;
/

create table donation_2 as (select * from donation where donation_id < 10);
select * from donation_2;

begin
    get_donations('Birth Control treatment', 6);
end;
/

------------
```

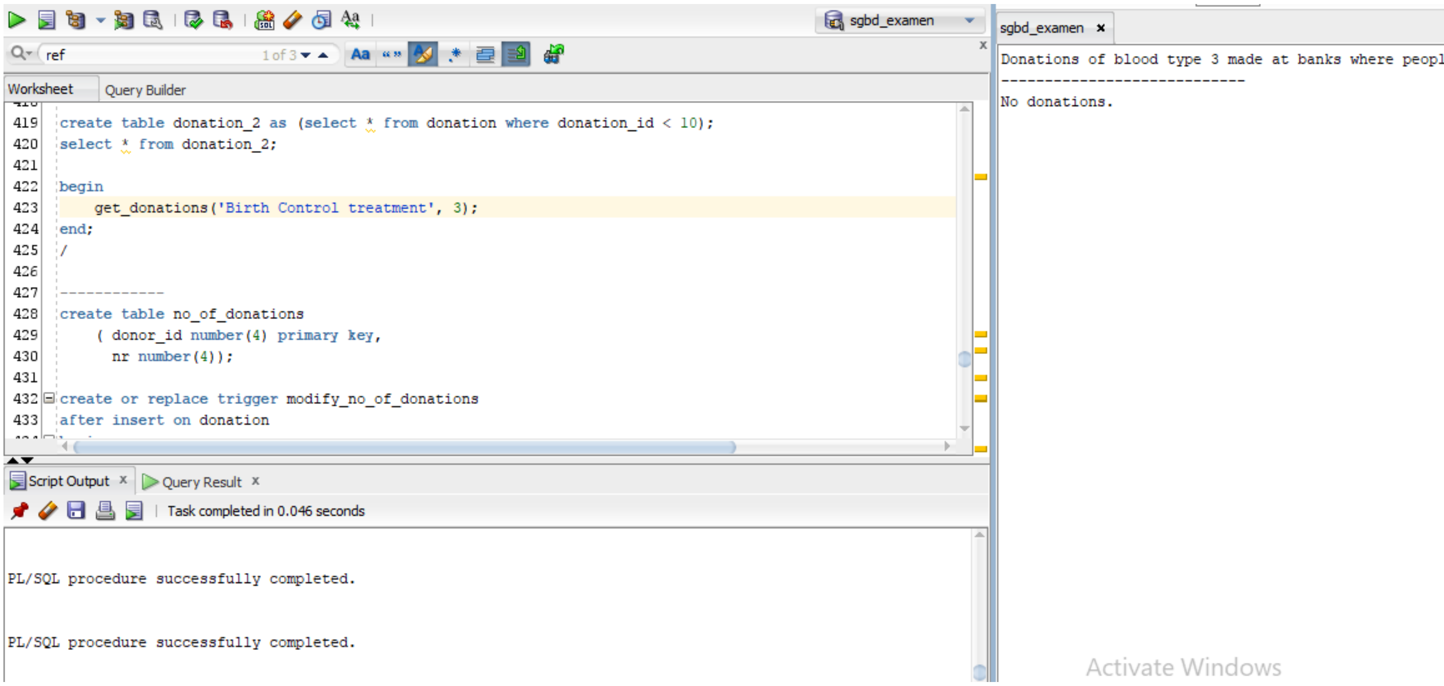
Task completed in 0.058 seconds

```
    get_donations('Birth Control treatment', 6);
end;
Error report -
ORA-20006: This group of blood does not exist.
ORA-06512: at "DARIA.GET_BLOOD_ID", line 25
ORA-06512: at "DARIA.GET_DONATIONS", line 5
ORA-06512: at line 2
```



```
    when no_donors_found then
        raise_application_error(-20012, 'No donors with this condition found.');
    when no_donations_found then
        raise_application_error(-20013, 'No donations with this type of blood');
end;
/

create table donation_2 as (select * from donation where donation_id < 10);
select * from donation_2;

begin
    get_donations('Vaccine', 3);
end;
/

------------
```

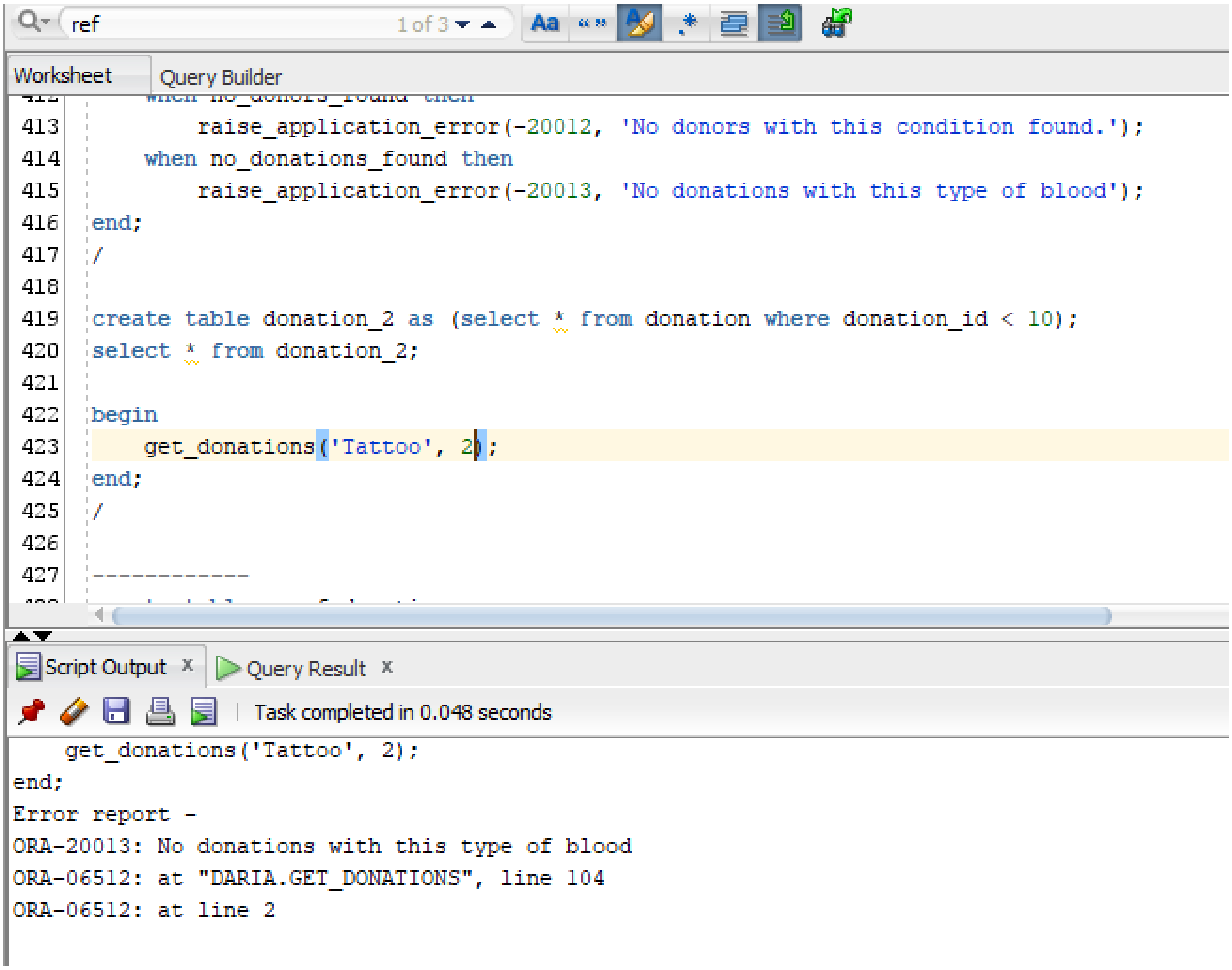
```
    get_donations('Vaccine', 3);
end;
Error report -
ORA-20010: No such condition found.
ORA-06512: at "DARIA.GET_CONDITION_ID", line 16
ORA-06512: at "DARIA.GET_DONATIONS", line 6
ORA-06512: at line 2
```



```
create table donation_2 as (select * from donation where donation_id < 10);
select * from donation_2;

begin
    get_donations('Birth Control treatment', 3);
end;
/

------------
create table no_of_donations
    ( donor_id number(4) primary key,
      nr number(4));

create or replace trigger modify_no_of_donations
after insert on donation
```

```
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

```
Donations of blood type 3 made at banks where peopl
-----------------------------
No donations.
```



```
    when no_donors_found then
        raise_application_error(-20012, 'No donors with this condition found.');
    when no_donations_found then
        raise_application_error(-20013, 'No donations with this type of blood');
end;
/

create table donation_2 as (select * from donation where donation_id < 10);
select * from donation_2;

begin
    get_donations('Tattoo', 2);
end;
/

------------
```

Task completed in 0.048 seconds

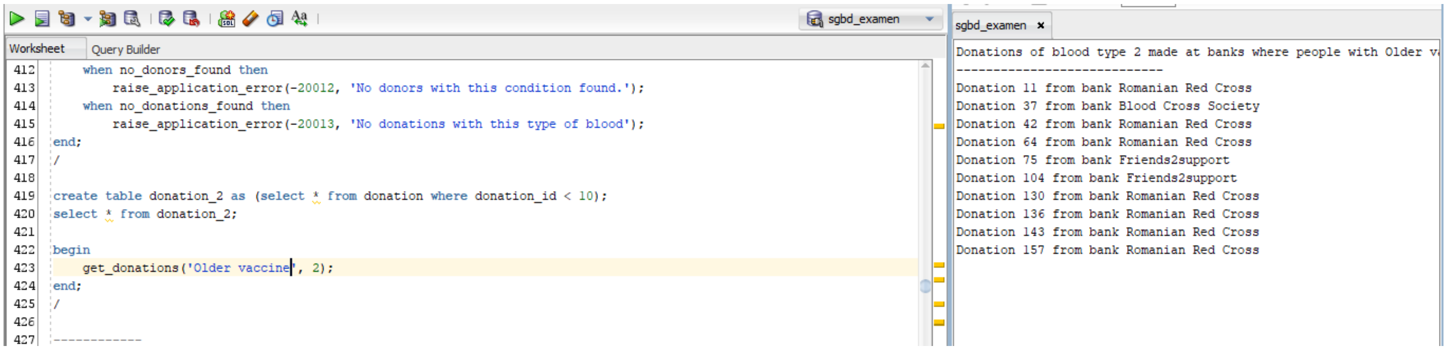
```
    get_donations('Tattoo', 2);
end;
Error report -
ORA-20013: No donations with this type of blood
ORA-06512: at "DARIA.GET_DONATIONS", line 104
ORA-06512: at line 2
```

Varianta finala a procedurii este:

```
create or replace procedure get_donations
    (condition medical_condition.condition_name%type,
     blood blood_types.blood_group%type)
is
    blood_code char_list := get_blood_id(blood);
    condition_code medical_condition.condition_id%type := get_condition_id(condition);
    donations number_list := number_list();
    donors number_list := number_list();
    banks number_list := number_list();
    nr number;
    mt number;
    ind number := 0;
    b_name blood_bank.bank_name%type;
    no_donations_found exception;
    no_donors_found exception;
    no_banks_found exception;
begin

    dbms_output.put_line('Donations of blood type ' || blood || ' made at banks where people with ' || condition || ' are registred');
    dbms_output.put_line('----------------------------');

    for i in (select * from donation)
    loop
        for j in blood_code.first..blood_code.last
        loop
            if i.blood_type = blood_code(j) then
                donations.extend;
                donations(donations.last) := i.donation_id;
            end if;
        end loop;
    end loop;

    if donations.count() = 0 then
        raise no_donations_found;
    end if;

    select donor_id
    bulk collect into donors
    from donor
    where donor_id in (select donor_id from donor_condition where condition_id = condition_code);

    if donors.count() = 0 then
        raise no_donors_found;
```

```
end if;

for i in donors.first..donors.last
loop
   for j in (select * from registration where donor_id = donors(i))
   loop
      banks.extend();
      banks(banks.last) := j.bank_id;
   end loop;
end loop;

for i in donations.first..donations.last
loop
   select bank_id
   into mt
   from donation
   where donation_id = donations(i);

   for j in banks.first..banks.last
   loop
      if banks(j) = mt then
         select bank_name
         into b_name
         from blood_bank
         where bank_id = banks(j);

         dbms_output.put_line('Donation ' || donations(i) || ' from bank ' || b_name);
         ind := ind + 1;
         exit;
      end if;
   end loop;
```

```
    end loop;


    if ind = 0 then
        dbms_output.put_line('No donations.');
    end if;


exception
    when no_donors_found then
        raise_application_error(-20012, 'No donors with this condition found.');
    when no_donations_found then
        raise_application_error(-20013, 'No donations with this type of blood');
end;
/
```



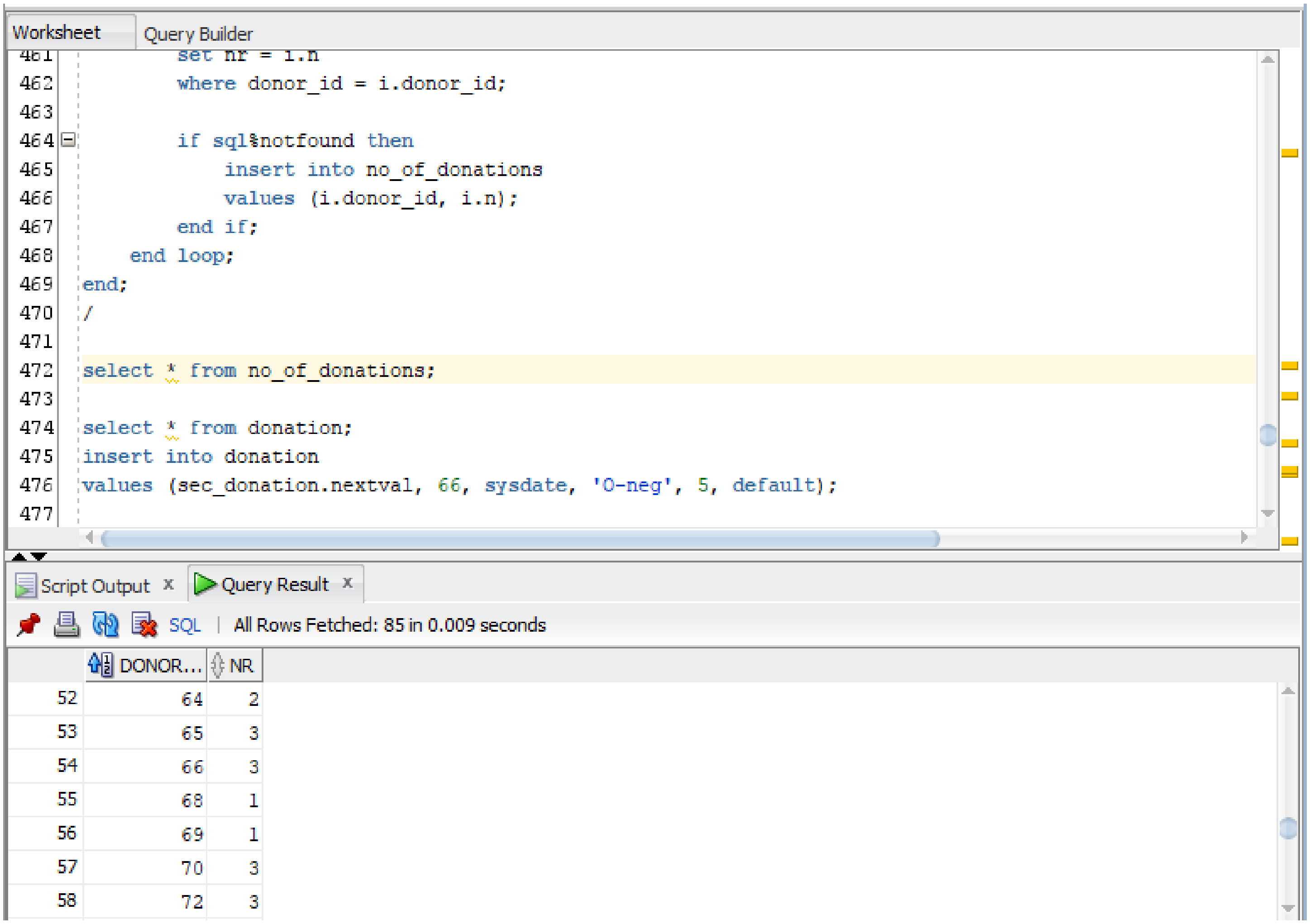
```
    when no_donors_found then
        raise_application_error(-20012, 'No donors with this condition found.');
    when no_donations_found then
        raise_application_error(-20013, 'No donations with this type of blood');
end;
/

create table donation_2 as (select * from donation where donation_id < 10);
select * from donation_2;

begin
    get_donations('Older vaccine', 2);
end;
/

------------
```

```
Donations of blood type 2 made at banks where people with Older v
----------------------------
Donation 11 from bank Romanian Red Cross
Donation 37 from bank Blood Cross Society
Donation 42 from bank Romanian Red Cross
Donation 64 from bank Romanian Red Cross
Donation 75 from bank Friends2support
Donation 104 from bank Friends2support
Donation 130 from bank Romanian Red Cross
Donation 136 from bank Romanian Red Cross
Donation 143 from bank Romanian Red Cross
Donation 157 from bank Romanian Red Cross
```

10) Trigger de tip LMD la nivel de comanda

Sa se creeze un tabel care sa contina cate donatii a facut fiecare donator.Sa se creeze un trigger la nivel de comanda care actualizeaza datele din tabelul creat.

```
create or replace trigger modify_no_of_donations

after insert on donation

begin

  for i in (select donor_id, count(*) as n

         from donation

         group by donor_id)

   loop

      update no_of_donations

      set nr = i.n

      where donor_id = i.donor_id;


      if sql%notfound then

         insert into no_of_donations

         values (i.donor_id, i.n);

      end if;

   end loop;

end;

/
```

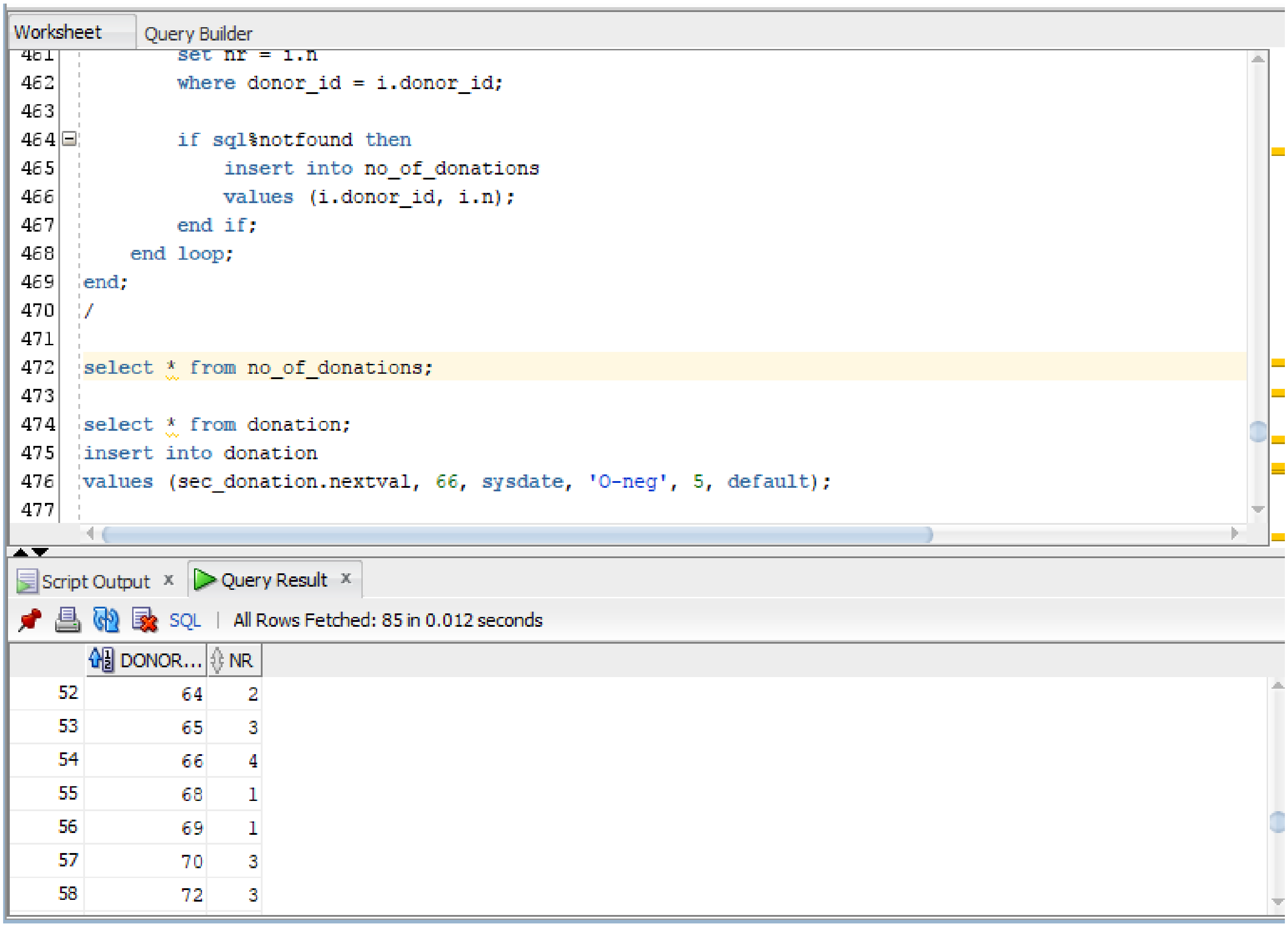


```
        set nr = i.n
        where donor_id = i.donor_id;

        if sql%notfound then
            insert into no_of_donations
            values (i.donor_id, i.n);
        end if;
    end loop;
end;
/

select * from no_of_donations;

select * from donation;
insert into donation
values (sec_donation.nextval, 66, sysdate, 'O-neg', 5, default);

```

All Rows Fetched: 85 in 0.009 seconds

	DONOR...	NR
52	64	2
53	65	3
54	66	3
55	68	1
56	69	1
57	70	3
58	72	3

Am mai rulat inca o data insert.



```
        set nr = i.n
        where donor_id = i.donor_id;

        if sql%notfound then
            insert into no_of_donations
            values (i.donor_id, i.n);
        end if;
    end loop;
end;
/

select * from no_of_donations;

select * from donation;
insert into donation
values (sec_donation.nextval, 66, sysdate, 'O-neg', 5, default);

```

All Rows Fetched: 85 in 0.012 seconds

	DONOR...	NR
52	64	2
53	65	3
54	66	4
55	68	1
56	69	1
57	70	3
58	72	3

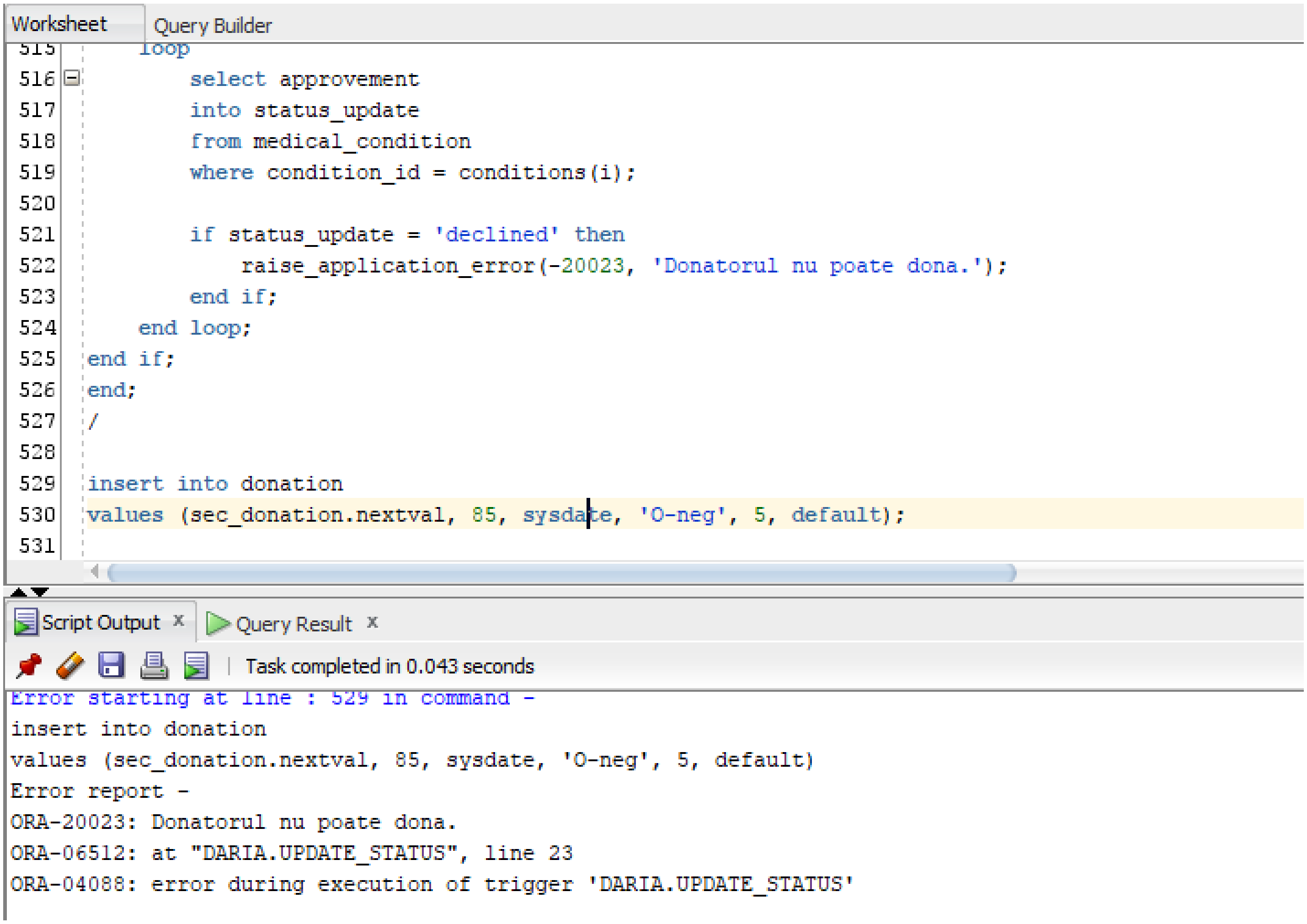
Donatiile nu se pot sterge (triggerul no_delete); exista doar varinata in care se poate schimba statusul in ‘declined’. Am dat disable la trigger pentru a sterge donatiile pur demonstrative pentru exceptii.

11) Trigger de tip LMD la nivel de linie

Sa se creeze un trigger care nu mai lasa donatorii care au conditiile nepotrivite pentru a dona sa mai doneze in continuare.

```
create or replace trigger update_status
before insert on donation
for each row
declare
    status_update varchar2(30);
    donor_code number;
    conditions number_list := number_list();
    new_donation number;
begin
    donor_code := :new.donor_id;

    select condition_id
    bulk collect into conditions
    from donor_condition
    where donor_id = donor_code;

    if conditions.count() <> 0 then
    for i in conditions.first..conditions.last
    loop
        select approvement
        into status_update
        from medical_condition
        where condition_id = conditions(i);

        if status_update = 'declined' then
            raise_application_error(-20023, 'Donatorul nu poate dona.');
        end if;
    end loop;
end if;
```

```
end;

/
```



```
515         loop
516             select approvement
517             into status_update
518             from medical_condition
519             where condition_id = conditions(i);
520
521             if status_update = 'declined' then
522                 raise_application_error(-20023, 'Donatorul nu poate dona.');
523             end if;
524         end loop;
525     end if;
526     end;
527     /
528
529     insert into donation
530     values (sec_donation.nextval, 85, sysdate, 'O-neg', 5, default);
531
```

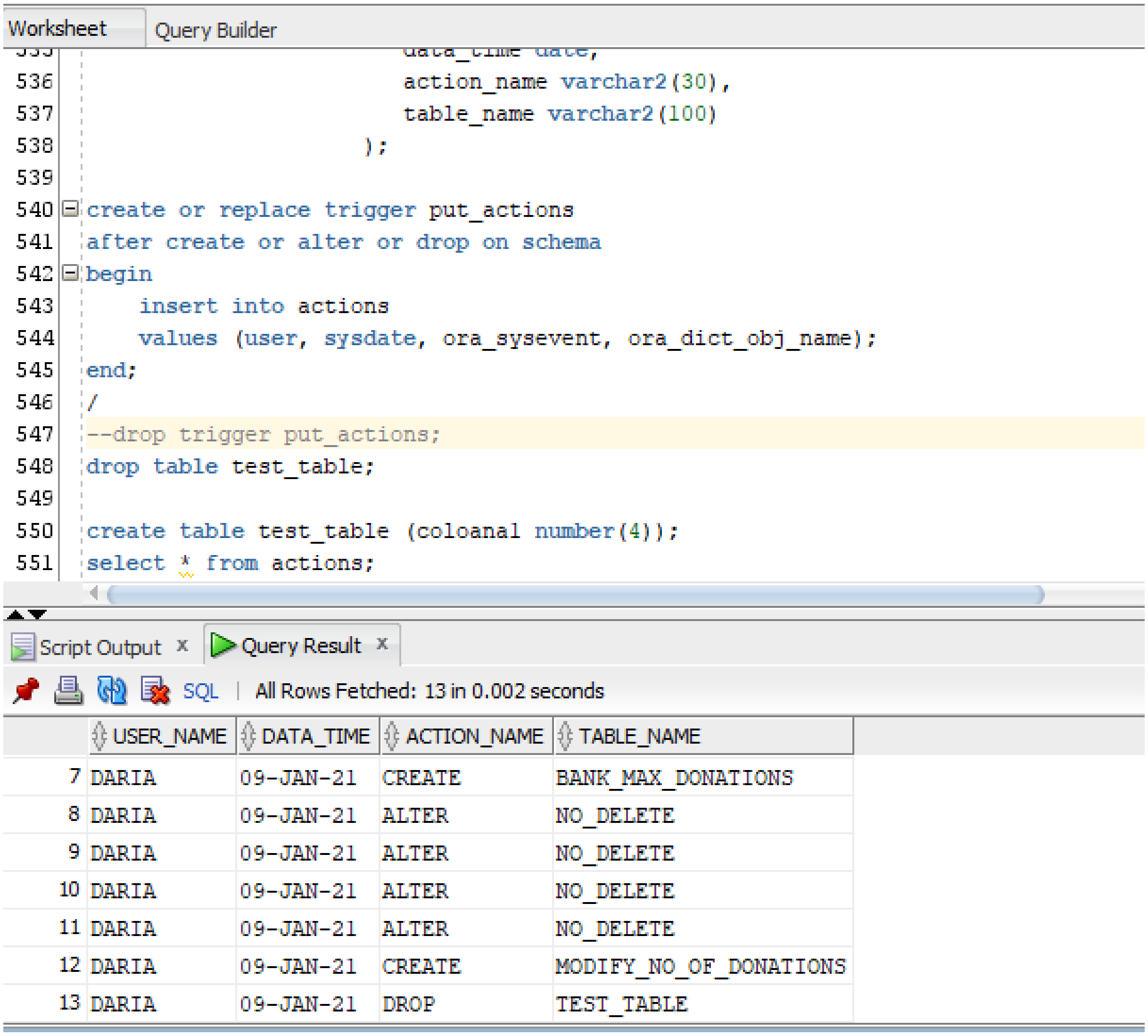
Task completed in 0.043 seconds

```
Error starting at line : 529 in command -
insert into donation
values (sec_donation.nextval, 85, sysdate, 'O-neg', 5, default)
Error report -
ORA-20023: Donatorul nu poate dona.
ORA-06512: at "DARIA.UPDATE_STATUS", line 23
ORA-04088: error during execution of trigger 'DARIA.UPDATE_STATUS'
```

12) Trigger de tip LDD

Sa se creeze un trigger care introduce in tabelul actions utilizatorul, data la care a fost facuta comanda LDD, numele comenzii si obiectul aspra caruia s-a apelat comanda.

```
create or replace trigger put_actions

after create or alter or drop on schema

begin

   insert into actions

   values (user, sysdate, ora_sysevent, ora_dict_obj_name);

end;

/
```



```
                        data_time date,
                        action_name varchar2(30),
                        table_name varchar2(100)
                     );

create or replace trigger put_actions
after create or alter or drop on schema
begin
    insert into actions
    values (user, sysdate, ora_sysevent, ora_dict_obj_name);
end;
/
--drop trigger put_actions;
drop table test_table;

create table test_table (coloanal number(4));
select * from actions;
```

All Rows Fetched: 13 in 0.002 seconds

	USER_NAME	DATA_TIME	ACTION_NAME	TABLE_NAME
7	DARIA	09-JAN-21	CREATE	BANK_MAX_DONATIONS
8	DARIA	09-JAN-21	ALTER	NO_DELETE
9	DARIA	09-JAN-21	ALTER	NO_DELETE
10	DARIA	09-JAN-21	ALTER	NO_DELETE
11	DARIA	09-JAN-21	ALTER	NO_DELETE
12	DARIA	09-JAN-21	CREATE	MODIFY_NO_OF_DONATIONS
13	DARIA	09-JAN-21	DROP	TEST_TABLE

Pentru cerinte am folosit si urmatoarele functii ajutatoare:

```
create or replace function get_staff (l_name staff.last_name%type)
    return number
is
    to_id staff.staff_id%type;
begin
    select staff_id into to_id
    from staff
    where last_name = initcap(l_name);

    return to_id;
exception
    when no_data_found then
        dbms_output.put_line('No staff with this name found.');
    when too_many_rows then
        dbms_output.put_line('More than one staff with this name.');
        for i in (  select staff_id, first_name, last_name
                    from staff
                    where last_name = initcap(l_name) )
        loop
            dbms_output.put_line (i.first_name || ' ' || i.last_name);
        end loop;
end;
/


create or replace function get_blood_id
    ( blood blood_types.blood_group%type)
return char_list
is
    bloods char_list;
    nr number;
```

```
begin
    select count(*)
    into nr
    from blood_types
    where blood_group = blood;

    if nr = 0 then
        raise no_data_found;
    end if;

    select blood_id
    bulk collect into bloods
    from blood_types
    where blood_group = blood;

    return bloods;
exception
    when no_data_found then
        raise_application_error(-20006, 'This group of blood does not exist.');
end;
/

create or replace function get_condition_id
    (condition medical_condition.condition_name%type)
return medical_condition.condition_id%type
is
    condition_code medical_condition.condition_id%type;
begin
    select condition_id
    into condition_code
    from medical_condition
    where condition_name = initcap(condition);


    return condition_code;


exception
    when no_data_found then
        raise_application_error(-20010, 'No such condition found.');
    when too_many_rows then
        raise_application_error(-20011, 'More than one condition with this name');
end;
/
```

13) Pachet cu functiile create mai sus

```
create or replace package blood_bank_management is
 procedure staff_categories (
    to_address address.address_id%type
);
procedure registered;

function get_staff (l_name staff.last_name%type)
    return number;

function bank_max_donations
return varchar2;

function get_blood_id
    ( blood blood_types.blood_group%type)
return char_list;

function get_condition_id
    (condition medical_condition.condition_name%type)
return medical_condition.condition_id%type;

procedure get_donations
    (condition medical_condition.condition_name%type,
     blood blood_types.blood_group%type);

end blood_bank_management;
/

create or replace package body blood_bank_management is
    procedure staff_categories (
        to_address address.address_id%type
    ) as
        nr_address number(4);
        lista_emp number_list := number_list();
        lista_bank number_list := number_list();
        lista_spec char_list := char_list();
        spec STAFF_CATEGORY.SPECIALITY%type;
        ind number(4) := 0;
        is_in_list boolean;
        no_bank_found exception;
        no_address_found exception;
        no_staff exception;
    begin
        select count(*) into nr_address
        from address
        where address_id = to_address;

        if nr_address = 0 then
            raise no_address_found;
        end if;

        select bank_id
        bulk collect into lista_bank
        from blood_bank
        where address_id = to_address;

        if lista_bank.count() = 0 then
            raise no_bank_found;
        end if;

        for i in lista_bank.first..lista_bank.last
        loop
```

```
select staff_category_id
bulk collect into lista_emp
from staff
where bank_id = lista_bank(i);

insert into bank_staff
values (lista_bank(i), lista_emp);

if lista_emp.count() = 0 then
    dbms_output.put_line('No one works at this bank -> ' || lista_bank(i));
else
    for j in lista_emp.first..lista_emp.last
    loop
        select speciality into spec
        from staff_category
        where category_id = lista_emp(j);

        is_in_list := true;
        if lista_spec.count() <> 0 then
            for k in lista_spec.first..lista_spec.last
            loop
                --dbms_output.put_line(lista_spec(k));
                if lista_spec(k) = spec then
                    is_in_list := false;
                end if;
            end loop;
        end if;
            if is_in_list = true then
                                ind := ind + 1;

                lista_spec.extend();
```

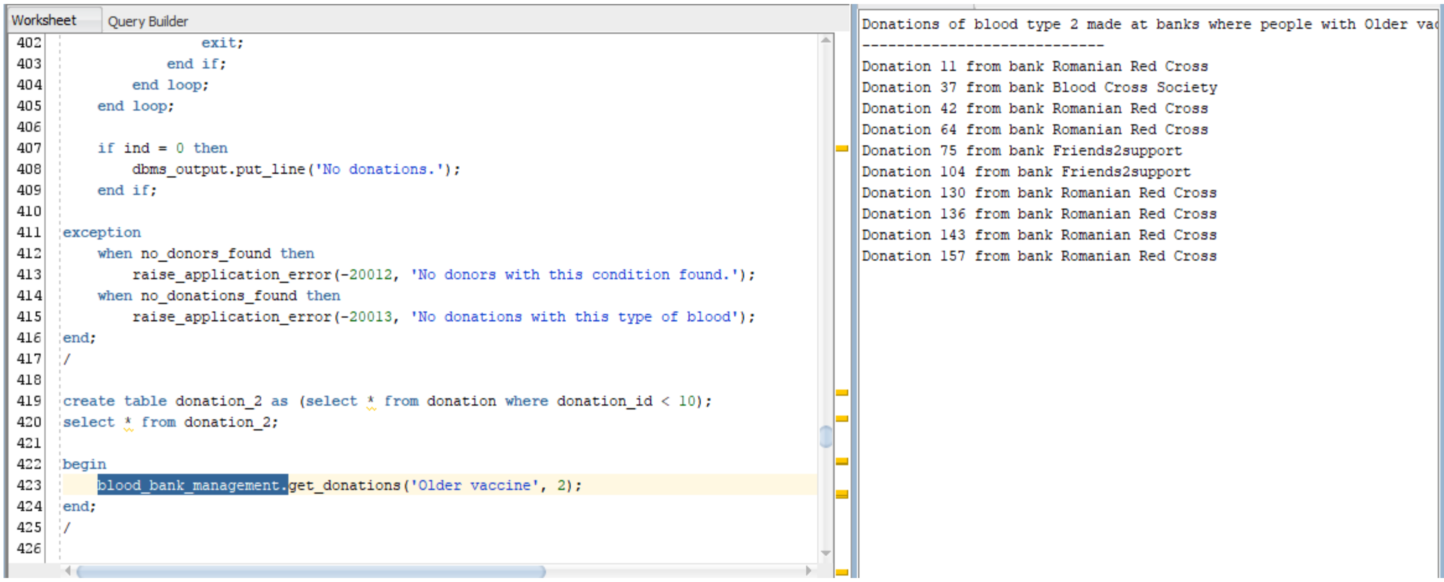
```
                            lista_spec(ind) := spec;
                        end if;
                end loop;
            end if;

        end loop;

        for i in lista_spec.first..lista_spec.last
        loop
            if lista_spec(i) <> 'None' then
                dbms_output.put_line(lista_spec(i));
            end if;
        end loop;
    exception
        when no_bank_found then
            raise_application_error (-20001, 'No bank found at this address.');
        when no_address_found then
                raise_application_error(-20002, 'No address found');
        when no_staff then
            raise_application_error (-20003, 'No one works at this address.');
    end;

    procedure registered
is
    TYPE refcursor IS REF CURSOR;
    cursor conditions is
        select donor_id, d.condition_id,
            cursor (select condition_name
                    from medical_condition
                    where condition_id = d.condition_id)
        from donor_condition d;
    cursor donor_reg is
        select bank_id, donor_id
        from registration;

    v_cursor refcursor;
    v_donor_id donor.donor_id%type;
    v_condition_id MEDICAL_CONDITION.CONDITION_ID%type;
    v_condition_name MEDICAL_CONDITION.CONDITION_NAME%type;
    f_name donor.first_name%type;
    l_name donor.last_name%type;
begin
    for i in donor_reg
    loop
        exit when donor_reg%notfound;
        for j in (select bank_id, donor_id, donation_id from donation)
        loop
            if i.bank_id = j.bank_id and i.donor_id = j.donor_id then
                select first_name, last_name
                into f_name, l_name
                from donor
                where donor_id = i.donor_id;

                dbms_output.put(f_name || ' ' || l_name || ' cu donatioa nr ' || j.donation_id);
                open conditions;
                loop
                    fetch conditions into v_donor_id, v_condition_id, v_cursor;
                    exit when conditions%notfound;
                    if v_donor_id = i.donor_id then
                        loop
                            fetch v_cursor into v_condition_name;
                            exit when v_cursor%notfound;
                            dbms_output.put(' ' || v_condition_name);
                        end loop;
                    end if;
                end loop;
                close conditions;
                dbms_output.new_line;
            end if;
        end loop;
    end loop;
end;

function get_staff (l_name staff.last_name%type)
    return number
is
    to_id staff.staff_id%type;
begin
    select staff_id into to_id
    from staff
    where last_name = initcap(l_name);

    return to_id;
exception
    when no_data_found then
        dbms_output.put_line('No staff with this name found.');
    when too_many_rows then
        dbms_output.put_line('More than one staff with this name.');
        for i in (  select staff_id, first_name, last_name
                    from staff
                    where last_name = initcap(l_name) )
        loop
```

```
            dbms_output.put_line (i.first_name || ' ' || i.last_name);
        end loop;
end;

function bank_max_donations
return varchar2
is
    bank blood_bank.bank_id%type;
    blood blood_types.blood_id%type;
    b_name varchar2(30);
begin

    select blood_id
    into blood
    from blood_types
    where Rh = 'negative' and blood_id like 'O%';

    select bank_id
    into bank
    from donation
    where blood_type = blood
    group by bank_id
    having count(*) = ( select max(count(*))
                        from donation
                        where blood_type = blood
                        group by bank_id);

    select bank_name
    into b_name
    from blood_bank
    where bank_id = bank;
    return b_name;
exception
    when too_many_rows then
        raise_application_error(-20008, 'There are more than one bank with the maximum donations number.');
end;

function get_blood_id
    ( blood blood_types.blood_group%type)
return char_list
is
    bloods char_list;
    nr number;
begin
    select count(*)
    into nr
    from blood_types
    where blood_group = blood;

    if nr = 0 then
        raise no_data_found;
    end if;

    select blood_id
    bulk collect into bloods
    from blood_types
    where blood_group = blood;

    return bloods;
exception
    when no_data_found then
```

```
        raise_application_error(-20006, 'This group of blood does not exist.');
end;

function get_condition_id
    (condition medical_condition.condition_name%type)
return medical_condition.condition_id%type
is
    condition_code medical_condition.condition_id%type;
begin
    select condition_id
    into condition_code
    from medical_condition
    where condition_name = initcap(condition);

    return condition_code;

exception
    when no_data_found then
        raise_application_error(-20010, 'No such condition found.');
    when too_many_rows then
        raise_application_error(-20011, 'More than one condition with this name');
end;

procedure get_donations
    (condition medical_condition.condition_name%type,
     blood blood_types.blood_group%type)
is
    blood_code char_list := get_blood_id(blood);
    condition_code medical_condition.condition_id%type := get_condition_id(condition);
    donations number_list := number_list();
    donors number_list := number_list();
    banks number_list := number_list();
    nr number;
    mt number;
    ind number := 0;
    b_name blood_bank.bank_name%type;

    no_donations_found exception;
    no_donors_found exception;
    no_banks_found exception;
begin

    dbms_output.put_line('Donations of blood type ' || blood || ' made at banks where people with ' || condition || '
are registred');
    dbms_output.put_line('----------------------------');

    for i in (select * from donation)
    loop
        for j in blood_code.first..blood_code.last
        loop
            if i.blood_type = blood_code(j) then
                donations.extend;
                donations(donations.last) := i.donation_id;
            end if;
        end loop;
    end loop;

    if donations.count() = 0 then
        raise no_donations_found;
    end if;

    select donor_id
    bulk collect into donors
```

```
from donor
where donor_id in (select donor_id from donor_condition where condition_id = condition_code);

if donors.count() = 0 then
    raise no_donors_found;
end if;

for i in donors.first..donors.last
loop
    for j in (select * from registration where donor_id = donors(i))
    loop
        banks.extend();
        banks(banks.last) := j.bank_id;
    end loop;
end loop;

for i in donations.first..donations.last
loop
    select bank_id
    into mt
    from donation
    where donation_id = donations(i);

    for j in banks.first..banks.last
    loop
        if banks(j) = mt then
            select bank_name
            into b_name
            from blood_bank
            where bank_id = banks(j);
```

```
                dbms_output.put_line('Donation ' || donations(i) || ' from bank ' || b_name);
                ind := ind + 1;
                exit;
            end if;
        end loop;
    end loop;

    if ind = 0 then
        dbms_output.put_line('No donations.');
    end if;

exception
    when no_donors_found then
        raise_application_error(-20012, 'No donors with this condition found.');
    when no_donations_found then
        raise_application_error(-20013, 'No donations with this type of blood');
end;

end blood_bank_management;
/
```



```
                exit;
            end if;
        end loop;
    end loop;

    if ind = 0 then
        dbms_output.put_line('No donations.');
    end if;

exception
    when no_donors_found then
        raise_application_error(-20012, 'No donors with this condition found.');
    when no_donations_found then
        raise_application_error(-20013, 'No donations with this type of blood');
end;
/

create table donation_2 as (select * from donation where donation_id < 10);
select * from donation_2;

begin
    blood_bank_management.get_donations('Older vaccine', 2);
end;
/
```

```
Donations of blood type 2 made at banks where people with Older vac
---------------------------
Donation 11 from bank Romanian Red Cross
Donation 37 from bank Blood Cross Society
Donation 42 from bank Romanian Red Cross
Donation 64 from bank Romanian Red Cross
Donation 75 from bank Friends2support
Donation 104 from bank Friends2support
Donation 130 from bank Romanian Red Cross
Donation 136 from bank Romanian Red Cross
Donation 143 from bank Romanian Red Cross
Donation 157 from bank Romanian Red Cross
```

14) Pachet cu tipuri de date complexe si obiecte necesa pentru actiuni integrate

In acest proiect am ales sa pun in evidenta overload pe proceduri intr-un pachet.

Pachetul salary_management contine o functie care seteaza toate salariile default, o functie care mareste salariile staff-ului care lucreaza la o banca anume si este de o anumita categorie (doctor, asistenta medicala sau resident) trimise ca parametru si o functie care mareste salariile staff-ului care lucreaza intr-o specialitaze anume si au un minim de experienta, trimise ca parametru.

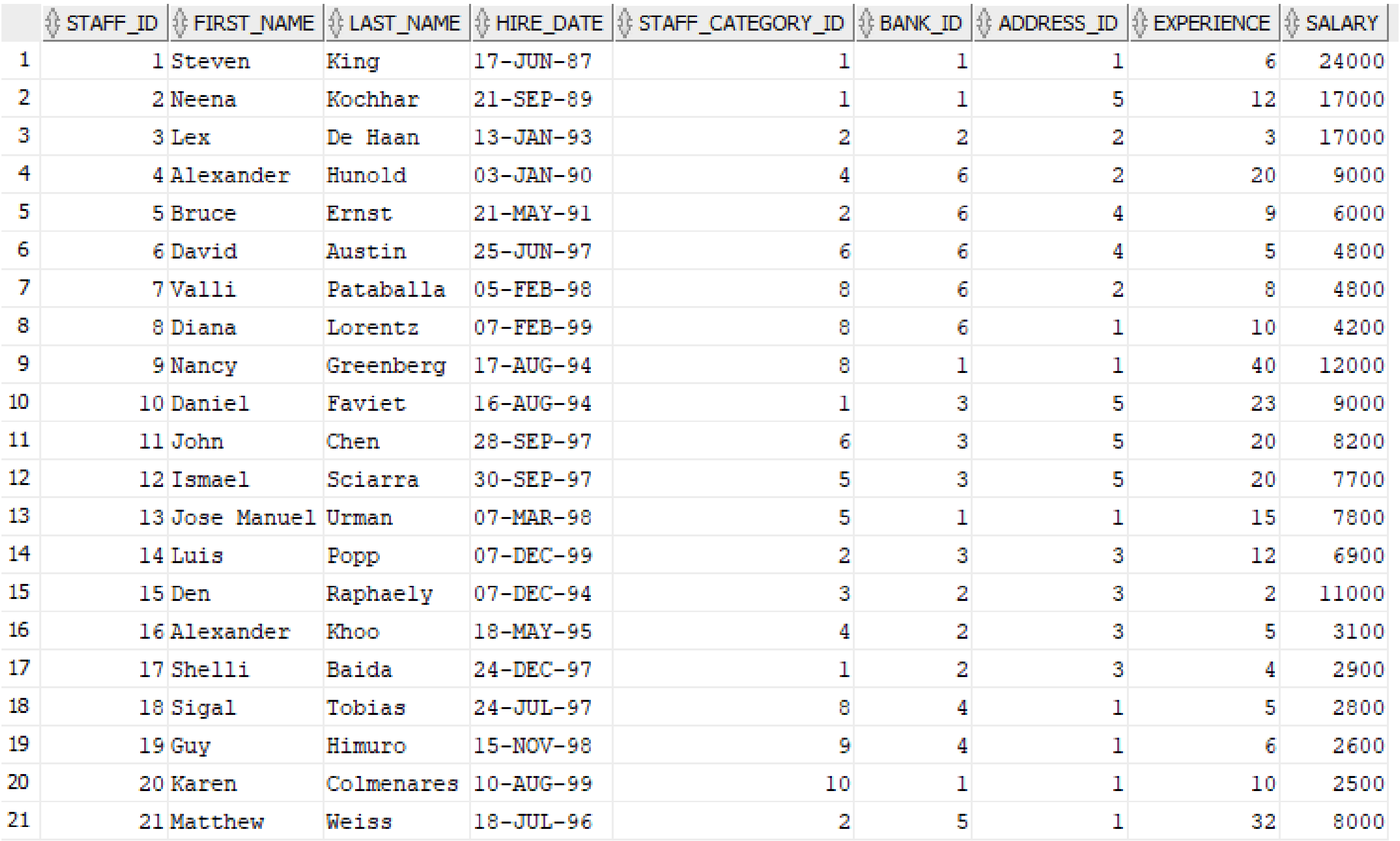
```
create or replace package salary_management

is



    procedure set_default;



    procedure upgrade_salary( bank blood_bank.bank_name%type,

                             categ_name staff_category.category_name%type);



    procedure upgrade_salary(spec staff_category.speciality%type,

                            exp staff.experience%type);



    function get_bank_id(bank blood_bank.bank_name%type)

        return blood_bank.bank_id%type;



    function get_category_id(categ staff_category.category_name%type)

        return number_list;



    function get_category_id_from_spec (categ staff_category.speciality%type)

        return number_list;



end salary_management;

/



create or replace package body salary_management

is
```

```
procedure set_default is
    id_categ staff_category.category_id%type;
    spec staff_category.category_name%type;
begin
    for i in (select * from staff)
    loop
        select staff_category_id
        into id_categ
        from staff
        where staff_id = i.staff_id;

        select category_name
        into spec
        from staff_category
        where category_id = id_categ;

        if spec = 'Doctor' then
            update staff
            set salary = 10000
            where staff_id = i.staff_id;
        elsif spec = 'Nurse' then
            update staff
            set salary = 5000
            where staff_id = i.staff_id;
        elsif spec = 'Resident' then
            update staff
            set salary = 4000
            where staff_id = i.staff_id;
        elsif spec = 'None' then
            update staff
            set salary = 2000
```

```
          where staff_id = i.staff_id;
        end if;
    end loop;
end;



function get_bank_id(bank blood_bank.bank_name%type)
        return blood_bank.bank_id%type
is
    b_id blood_bank.bank_id%type;
begin
    select bank_id
    into b_id
    from blood_bank
    where bank_name = initcap(bank);

    return b_id;
exception
    when no_data_found then
        raise_application_error (-20015, 'No bank found.');
    when too_many_rows then
        raise_application_error(-20016, 'Too many banks with this name.');
end;

function get_category_id(categ staff_category.category_name%type)
        return number_list
is
    categ_id number_list;
begin
    select category_id
    bulk collect into categ_id
    from staff_category
    where category_name = initcap(categ);

    if categ_id.count() = 0 then
        raise no_data_found;
    end if;
        return categ_id;
exception
    when no_data_found then
        raise_application_error (-20015, 'No category found.');
end;

procedure upgrade_salary( bank blood_bank.bank_name%type,
                           categ_name staff_category.category_name%type)
is
    b_id blood_bank.bank_id%type := get_bank_id(bank);
    categ_list number_list := get_category_id(categ_name);
    categ staff.staff_category_id%type;
    to_upgrade boolean;
begin
for i in (select * from staff)
    loop
        select staff_category_id
        into categ
        from staff
        where staff_id = i.staff_id;
        to_upgrade := false;
        for j in categ_list.first..categ_list.last
        loop
            if categ_list(j) = categ then
                to_upgrade := true;
            end if;
        end loop;
        if to_upgrade = true then
            update staff
            set salary = salary * 1.1
            where staff_id = i.staff_id and bank_id = b_id;
        end if;
    end loop;
end;

function get_category_id_from_spec (categ staff_category.speciality%type)
        return number_list
is
    categ_list number_list;
begin
    select category_id
    bulk collect into categ_list
    from staff_category
    where speciality = categ;

    if categ_list.count() = 0 then
        raise no_data_found;
    end if;
    return categ_list;
exception
    when no_data_found then
        raise_application_error(-20016, 'No such speciality.');
end;

procedure upgrade_salary(spec staff_category.speciality%type,
                        exp staff.experience%type)
```

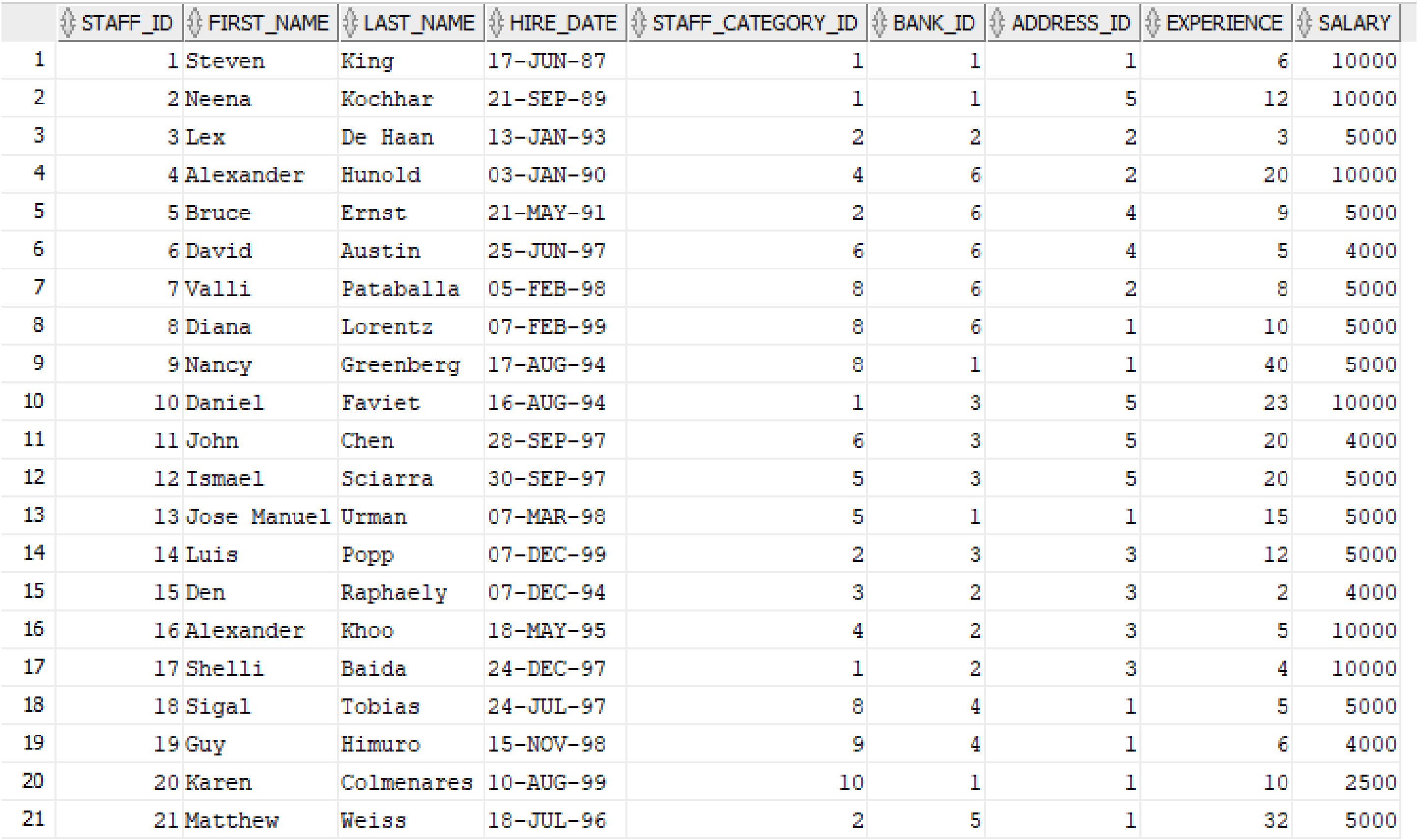
```
is
    categ staff.staff_category_id%type;
    to_upgrade boolean;
    categ_list number_list := get_category_id_from_spec(spec);
     no_staff_in_categ exception;
begin
    for i in (select * from staff)
    loop
        select staff_category_id
        into categ
        from staff
        where staff_id = i.staff_id;

        if categ_list.count() > 0 then
            to_upgrade := false;
            for j in categ_list.first..categ_list.last
            loop
                if categ_list(j) = categ then
                    to_upgrade := true;
                end if;
            end loop;
            if to_upgrade = true then
                update staff
                set salary = salary * 1.3
                where staff_id = i.staff_id and i.experience > exp;
            end if;
        end if;
    end loop;
end;
end salary_management;
/
```

Tabel initial Staff



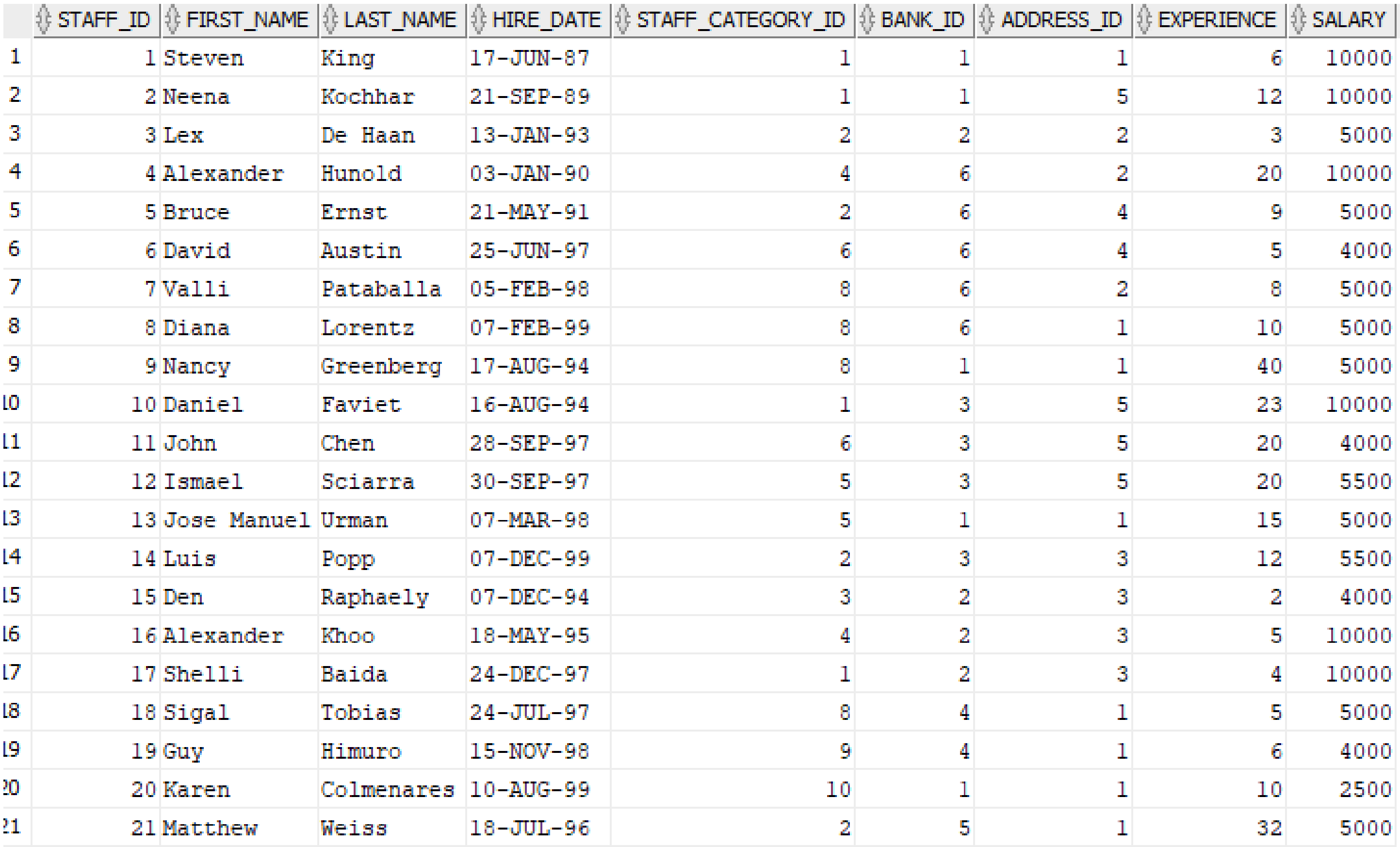
	STAFF_ID	FIRST_NAME	LAST_NAME	HIRE_DATE	STAFF_CATEGORY_ID	BANK_ID	ADDRESS_ID	EXPERIENCE	SALARY
1	1	Steven	King	17-JUN-87	1	1	1	6	24000
2	2	Neena	Kochhar	21-SEP-89	1	1	5	12	17000
3	3	Lex	De Haan	13-JAN-93	2	2	2	3	17000
4	4	Alexander	Hunold	03-JAN-90	4	6	2	20	9000
5	5	Bruce	Ernst	21-MAY-91	2	6	4	9	6000
6	6	David	Austin	25-JUN-97	6	6	4	5	4800
7	7	Valli	Pataballa	05-FEB-98	8	6	2	8	4800
8	8	Diana	Lorentz	07-FEB-99	8	6	1	10	4200
9	9	Nancy	Greenberg	17-AUG-94	8	1	1	40	12000
10	10	Daniel	Faviet	16-AUG-94	1	3	5	23	9000
11	11	John	Chen	28-SEP-97	6	3	5	20	8200
12	12	Ismael	Sciarra	30-SEP-97	5	3	5	20	7700
13	13	Jose Manuel	Urman	07-MAR-98	5	1	1	15	7800
14	14	Luis	Popp	07-DEC-99	2	3	3	12	6900
15	15	Den	Raphaely	07-DEC-94	3	2	3	2	11000
16	16	Alexander	Khoo	18-MAY-95	4	2	3	5	3100
17	17	Shelli	Baida	24-DEC-97	1	2	3	4	2900
18	18	Sigal	Tobias	24-JUL-97	8	4	1	5	2800
19	19	Guy	Himuro	15-NOV-98	9	4	1	6	2600
20	20	Karen	Colmenares	10-AUG-99	10	1	1	10	2500
21	21	Matthew	Weiss	18-JUL-96	2	5	1	32	8000

Dupa apelarea functiei salary_default (functie care seteaza salariile tuturor).



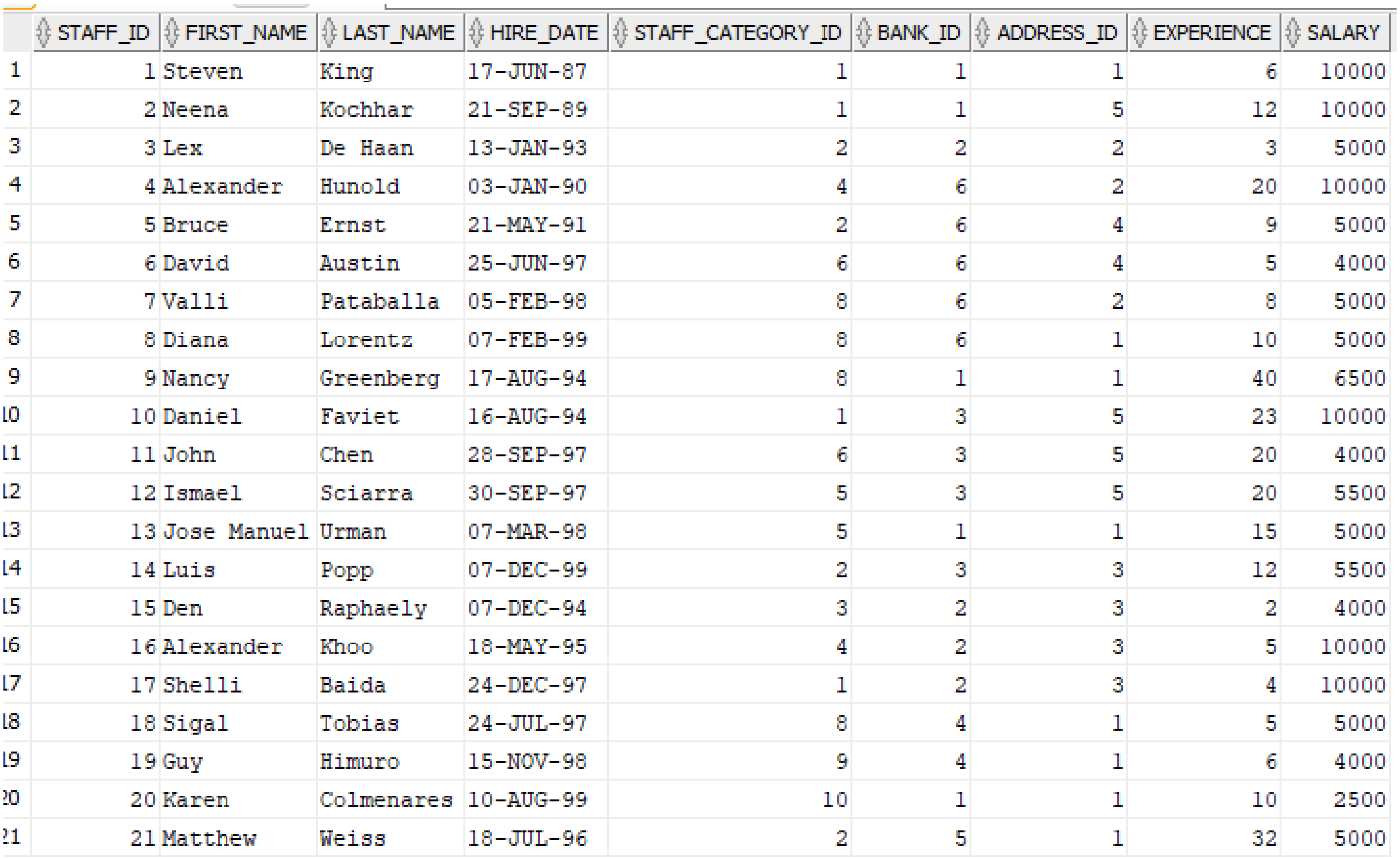
	STAFF_ID	FIRST_NAME	LAST_NAME	HIRE_DATE	STAFF_CATEGORY_ID	BANK_ID	ADDRESS_ID	EXPERIENCE	SALARY
1	1	Steven	King	17-JUN-87	1	1	1	6	10000
2	2	Neena	Kochhar	21-SEP-89	1	1	5	12	10000
3	3	Lex	De Haan	13-JAN-93	2	2	2	3	5000
4	4	Alexander	Hunold	03-JAN-90	4	6	2	20	10000
5	5	Bruce	Ernst	21-MAY-91	2	6	4	9	5000
6	6	David	Austin	25-JUN-97	6	6	4	5	4000
7	7	Valli	Pataballa	05-FEB-98	8	6	2	8	5000
8	8	Diana	Lorentz	07-FEB-99	8	6	1	10	5000
9	9	Nancy	Greenberg	17-AUG-94	8	1	1	40	5000
10	10	Daniel	Faviet	16-AUG-94	1	3	5	23	10000
11	11	John	Chen	28-SEP-97	6	3	5	20	4000
12	12	Ismael	Sciarra	30-SEP-97	5	3	5	20	5000
13	13	Jose Manuel	Urman	07-MAR-98	5	1	1	15	5000
14	14	Luis	Popp	07-DEC-99	2	3	3	12	5000
15	15	Den	Raphaely	07-DEC-94	3	2	3	2	4000
16	16	Alexander	Khoo	18-MAY-95	4	2	3	5	10000
17	17	Shelli	Baida	24-DEC-97	1	2	3	4	10000
18	18	Sigal	Tobias	24-JUL-97	8	4	1	5	5000
19	19	Guy	Himuro	15-NOV-98	9	4	1	6	4000
20	20	Karen	Colmenares	10-AUG-99	10	1	1	10	2500
21	21	Matthew	Weiss	18-JUL-96	2	5	1	32	5000

Dupa prima procedura de marire.



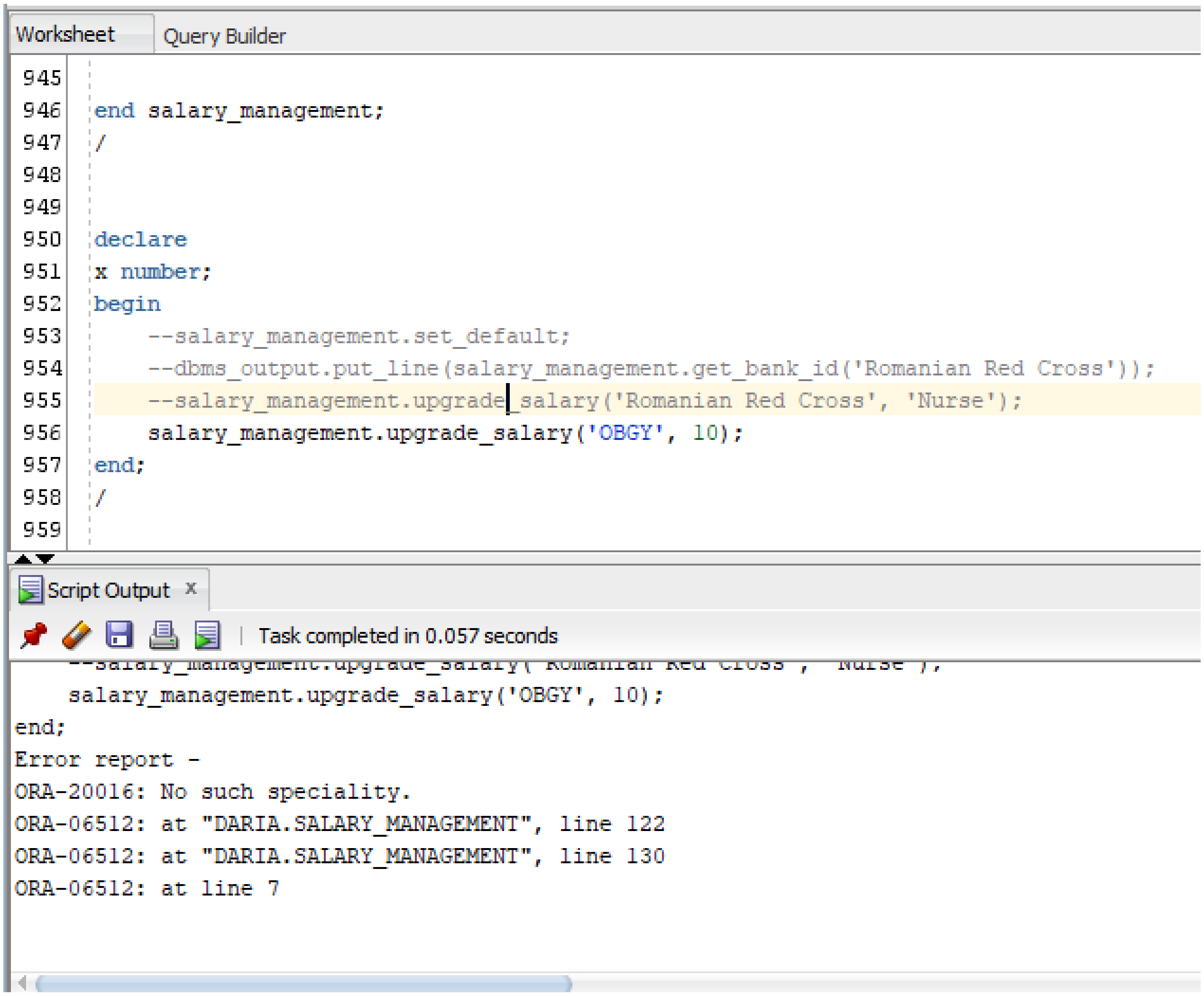
	STAFF_ID	FIRST_NAME	LAST_NAME	HIRE_DATE	STAFF_CATEGORY_ID	BANK_ID	ADDRESS_ID	EXPERIENCE	SALARY
1	1	Steven	King	17-JUN-87	1	1	1	6	10000
2	2	Neena	Kochhar	21-SEP-89	1	1	5	12	10000
3	3	Lex	De Haan	13-JAN-93	2	2	2	3	5000
4	4	Alexander	Hunold	03-JAN-90	4	6	2	20	10000
5	5	Bruce	Ernst	21-MAY-91	2	6	4	9	5000
6	6	David	Austin	25-JUN-97	6	6	4	5	4000
7	7	Valli	Pataballa	05-FEB-98	8	6	2	8	5000
8	8	Diana	Lorentz	07-FEB-99	8	6	1	10	5000
9	9	Nancy	Greenberg	17-AUG-94	8	1	1	40	5000
10	10	Daniel	Faviet	16-AUG-94	1	3	5	23	10000
11	11	John	Chen	28-SEP-97	6	3	5	20	4000
12	12	Ismael	Sciarra	30-SEP-97	5	3	5	20	5500
13	13	Jose Manuel	Urman	07-MAR-98	5	1	1	15	5000
14	14	Luis	Popp	07-DEC-99	2	3	3	12	5500
15	15	Den	Raphaely	07-DEC-94	3	2	3	2	4000
16	16	Alexander	Khoo	18-MAY-95	4	2	3	5	10000
17	17	Shelli	Baida	24-DEC-97	1	2	3	4	10000
18	18	Sigal	Tobias	24-JUL-97	8	4	1	5	5000
19	19	Guy	Himuro	15-NOV-98	9	4	1	6	4000
20	20	Karen	Colmenares	10-AUG-99	10	1	1	10	2500
21	21	Matthew	Weiss	18-JUL-96	2	5	1	32	5000

Dupa a doua procedura de marire



	STAFF_ID	FIRST_NAME	LAST_NAME	HIRE_DATE	STAFF_CATEGORY_ID	BANK_ID	ADDRESS_ID	EXPERIENCE	SALARY
1	1	Steven	King	17-JUN-87	1	1	1	6	10000
2	2	Neena	Kochhar	21-SEP-89	1	1	5	12	10000
3	3	Lex	De Haan	13-JAN-93	2	2	2	3	5000
4	4	Alexander	Hunold	03-JAN-90	4	6	2	20	10000
5	5	Bruce	Ernst	21-MAY-91	2	6	4	9	5000
6	6	David	Austin	25-JUN-97	6	6	4	5	4000
7	7	Valli	Pataballa	05-FEB-98	8	6	2	8	5000
8	8	Diana	Lorentz	07-FEB-99	8	6	1	10	5000
9	9	Nancy	Greenberg	17-AUG-94	8	1	1	40	6500
10	10	Daniel	Faviet	16-AUG-94	1	3	5	23	10000
11	11	John	Chen	28-SEP-97	6	3	5	20	4000
12	12	Ismael	Sciarra	30-SEP-97	5	3	5	20	5500
13	13	Jose Manuel	Urman	07-MAR-98	5	1	1	15	5000
14	14	Luis	Popp	07-DEC-99	2	3	3	12	5500
15	15	Den	Raphaely	07-DEC-94	3	2	3	2	4000
16	16	Alexander	Khoo	18-MAY-95	4	2	3	5	10000
17	17	Shelli	Baida	24-DEC-97	1	2	3	4	10000
18	18	Sigal	Tobias	24-JUL-97	8	4	1	5	5000
19	19	Guy	Himuro	15-NOV-98	9	4	1	6	4000
20	20	Karen	Colmenares	10-AUG-99	10	1	1	10	2500
21	21	Matthew	Weiss	18-JUL-96	2	5	1	32	5000

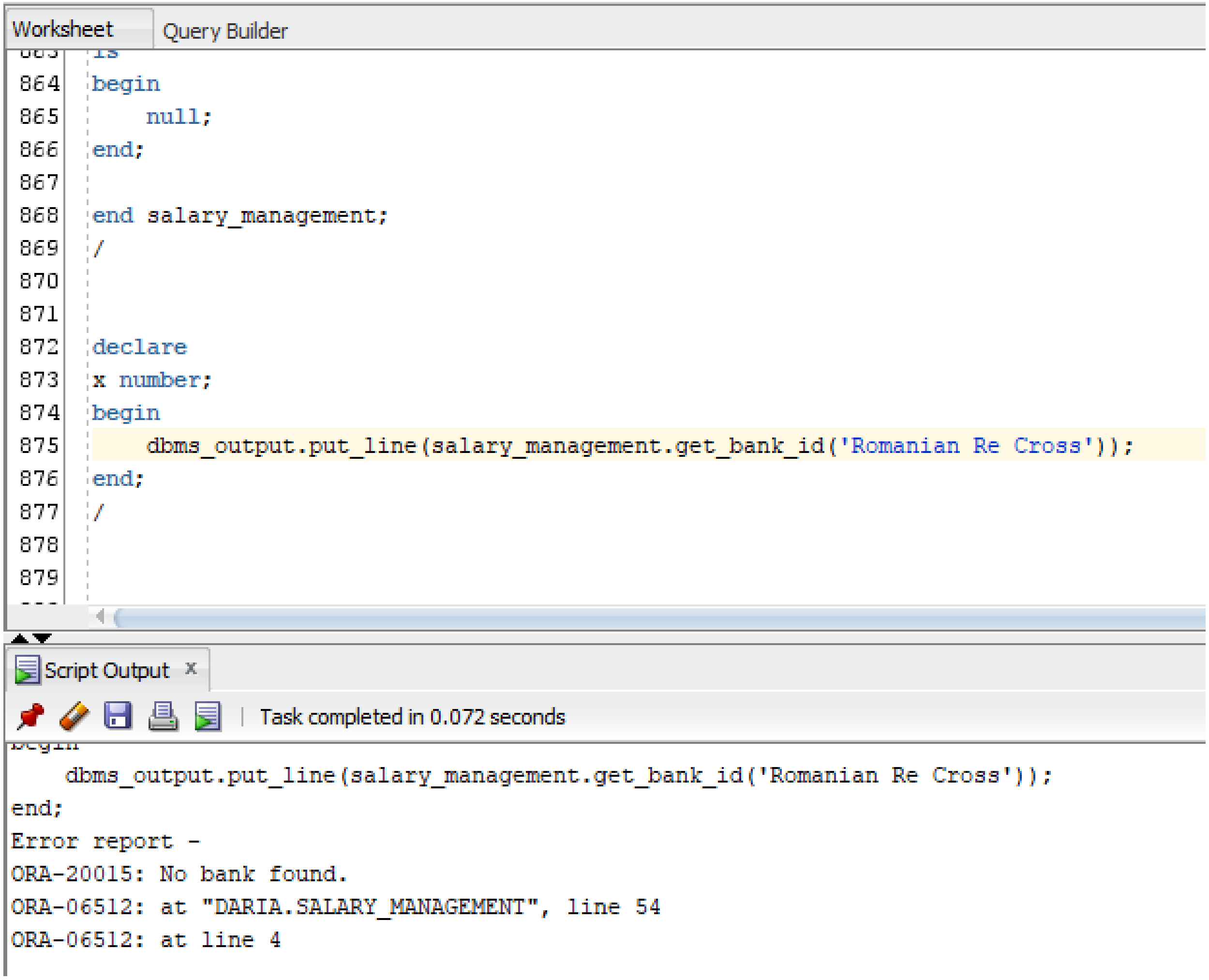
Am rulat pentru exceptii:



```

end salary_management;
/


declare
x number;
begin
    --salary_management.set_default;
    --dbms_output.put_line(salary_management.get_bank_id('Romanian Red Cross'));
    --salary_management.upgrade_salary('Romanian Red Cross', 'Nurse');
    salary_management.upgrade_salary('OBGY', 10);
end;
/

```

Task completed in 0.057 seconds

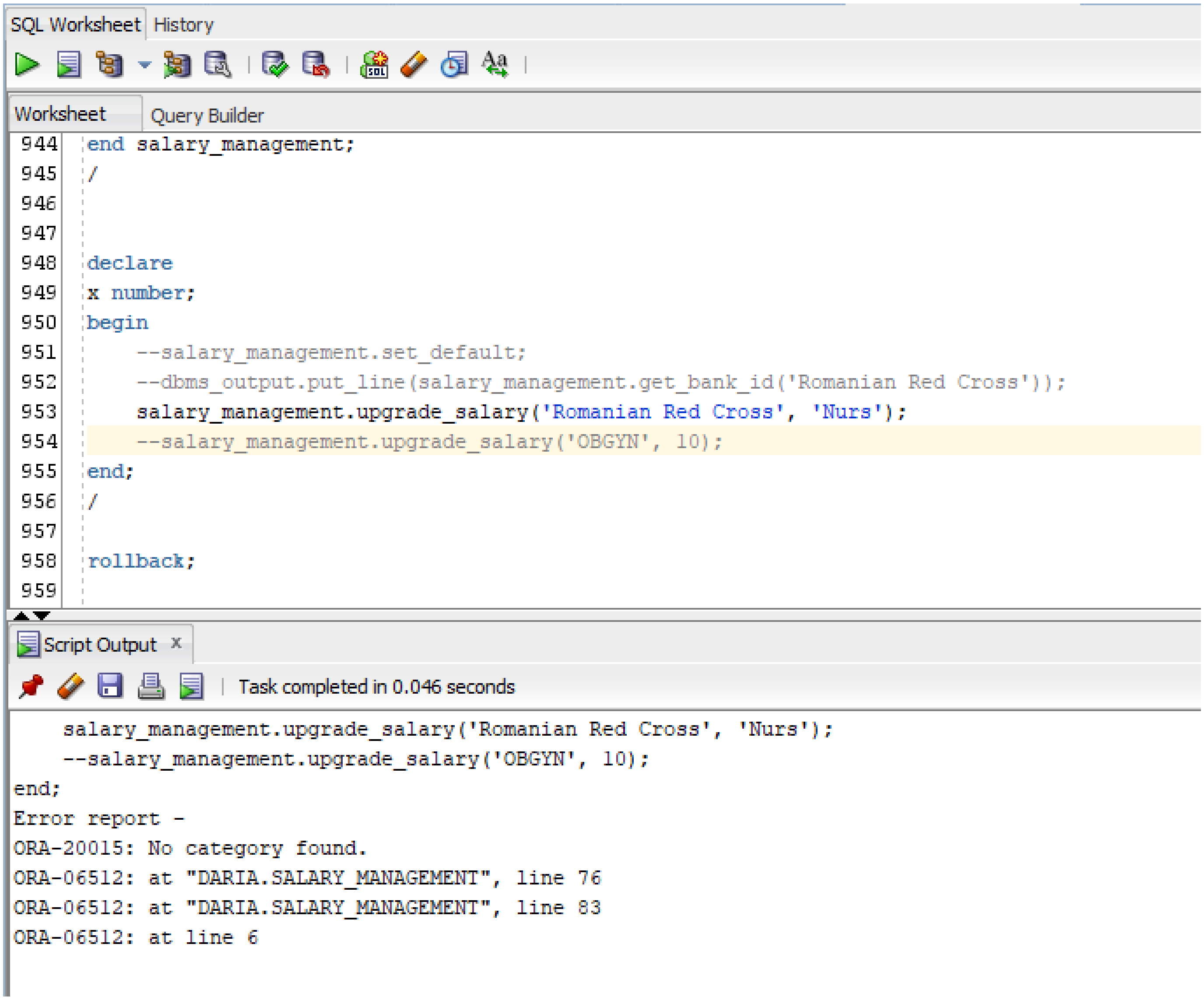
```
    --salary_management.upgrade_salary('Romanian Red Cross', 'Nurse');
    salary_management.upgrade_salary('OBGY', 10);
end;
Error report -
ORA-20016: No such speciality.
ORA-06512: at "DARIA.SALARY_MANAGEMENT", line 122
ORA-06512: at "DARIA.SALARY_MANAGEMENT", line 130
ORA-06512: at line 7
```



```
is
begin
    null;
end;

end salary_management;
/


declare
x number;
begin
    dbms_output.put_line(salary_management.get_bank_id('Romanian Re Cross'));
end;
/
```

Task completed in 0.072 seconds

```
begin
    dbms_output.put_line(salary_management.get_bank_id('Romanian Re Cross'));
end;
Error report -
ORA-20015: No bank found.
ORA-06512: at "DARIA.SALARY_MANAGEMENT", line 54
ORA-06512: at line 4
```



```
end salary_management;
/


declare
x number;
begin
    --salary_management.set_default;
    --dbms_output.put_line(salary_management.get_bank_id('Romanian Red Cross'));
    salary_management.upgrade_salary('Romanian Red Cross', 'Nurs');
    --salary_management.upgrade_salary('OBGYN', 10);
end;
/

rollback;

```

Task completed in 0.046 seconds

```
    salary_management.upgrade_salary('Romanian Red Cross', 'Nurs');
    --salary_management.upgrade_salary('OBGYN', 10);
end;
Error report -
ORA-20015: No category found.
ORA-06512: at "DARIA.SALARY_MANAGEMENT", line 76
ORA-06512: at "DARIA.SALARY_MANAGEMENT", line 83
ORA-06512: at line 6
```