Tema PLSQL 3 si 4

E6 PLSQL 3

E6. Adaptați cerința exercițiului 10 pentru diagrama proiectului prezentată la materia Baze de Date din anul I. Rezolvați subpunctul (a) al acestui exercițiu în PL/SQL, folosind baza de date proprie

Proiectul de anul trecut a constat in modelarea unei baze de date pentru program de streaming tip Spotify.

Exercitiul 10 (adaptat) : Pentru fiecare dintre playlist-urile 1, 2, 5, 7 (3, 4 nu au melodii inca adaugate), obtineti numele precum si lista numelor melodiilor care fac parte din acestea. Rezolvati folosind cele 3 tipuri de cursoare studiate.

1. Folosind cursor simplu

**Cod SQL:**

declare --cursor simplu

--cursor care va selecta id ul si titlul playlisturilor cerute

cursor playlist\_cursor is

select p.playlist\_id, p.titlu

from playlist p

where p.playlist\_id in (1,2,5,7);

--variabile folosite pt info ca titlu si id

v\_playlist\_id playlist.playlist\_id%type;

v\_titlu\_playlist playlist.titlu%type;

--cursor pt a lua melodiile pt fiecare playlist

cursor melodie\_cursor(p\_playlist\_id playlist.playlist\_id%type) is

select m.titlu

from melodie m

join melodie\_in\_playlist\_utilizator mp on m.melodie\_id = mp.melodie\_id

where mp.playlist\_id = p\_playlist\_id;

--variabila pt a lua titlul melodiilor

v\_titlu\_melodie melodie.titlu%type;

begin

--deschidem cursorul pt playlisturi

open playlist\_cursor;

--trecem prin fiecare playlist din el

loop

--punem info din cursor in variabile

fetch playlist\_cursor into v\_playlist\_id, v\_titlu\_playlist;

--iesim cand nu mai avem la ce da fetch

exit when playlist\_cursor%notfound;

dbms\_output.put\_line('Playlist-ul ' || v\_titlu\_playlist || ':');

--deschidem cursorul cu melodiile din acele playlisturi

open melodie\_cursor(v\_playlist\_id);

--trecem prin toate, facem fetch si ne oprim cand nu mai avem la ce sa facem

loop

fetch melodie\_cursor into v\_titlu\_melodie;

exit when melodie\_cursor%notfound;

dbms\_output.put\_line(' -> ' || v\_titlu\_melodie);

end loop;

--inchidem cursor melodie

close melodie\_cursor;

end loop;

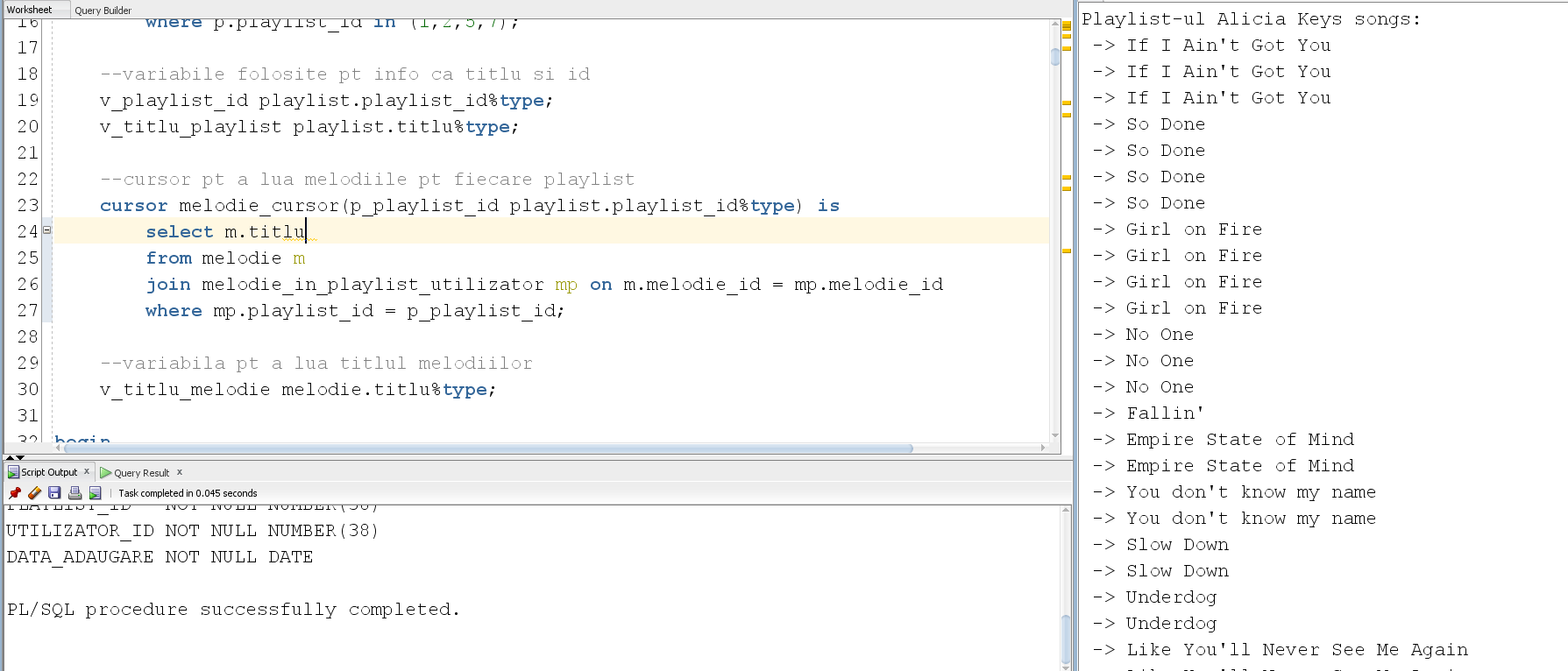
--inchdi cursor playlist

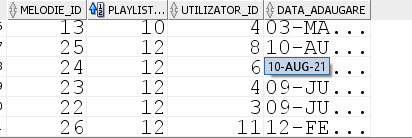
close playlist\_cursor;

end;

/

**Print screen-uri:**



Pentru verificare, deoarece aceste playlist-uri au un volum mare de melodii adaugate, si apar in duplicat deoarece aplicatia a fost gandita pentru playlist-uri publice in care utilizatorii pot adauga melodii. Vom apela procedura pentru playlist ul cu id = 12 deoarece contine doar 4 melodii. 



1. Folosind ciclu cursor

**Cod SQL:**

declare

--cursor care va selecta id ul si titlul playlisturilor cerute

cursor playlist\_cursor is

select p.playlist\_id, p.titlu

from playlist p

where p.playlist\_id in (1,2,5,7);

--variabile folosite pt info ca titlu si id

v\_playlist\_id playlist.playlist\_id%type;

v\_titlu\_playlist playlist.titlu%type;

--cursor pt a lua melodiile pt fiecare playlist

cursor melodie\_cursor(p\_playlist\_id playlist.playlist\_id%type) is

select m.titlu

from melodie m

join melodie\_in\_playlist\_utilizator mp on m.melodie\_id = mp.melodie\_id

where mp.playlist\_id = p\_playlist\_id;

--variabila pt a lua titlul melodiilor

v\_titlu\_melodie melodie.titlu%type;

begin

--facem for pt a lua fiecare playlist din cursor

for playlist\_rec in playlist\_cursor loop

v\_playlist\_id := playlist\_rec.playlist\_id;

v\_titlu\_playlist := playlist\_rec.titlu;

dbms\_output.put\_line('Playlist-ul ' || v\_titlu\_playlist || ':');

--acum iteram cu for prin melodiile din playlist ul la care am ajuns

for melodie\_rec in melodie\_cursor(v\_playlist\_id) loop

--atribuim valoarea din cursor variabilei

v\_titlu\_melodie := melodie\_rec.titlu;

dbms\_output.put\_line(' -> ' || v\_titlu\_melodie);

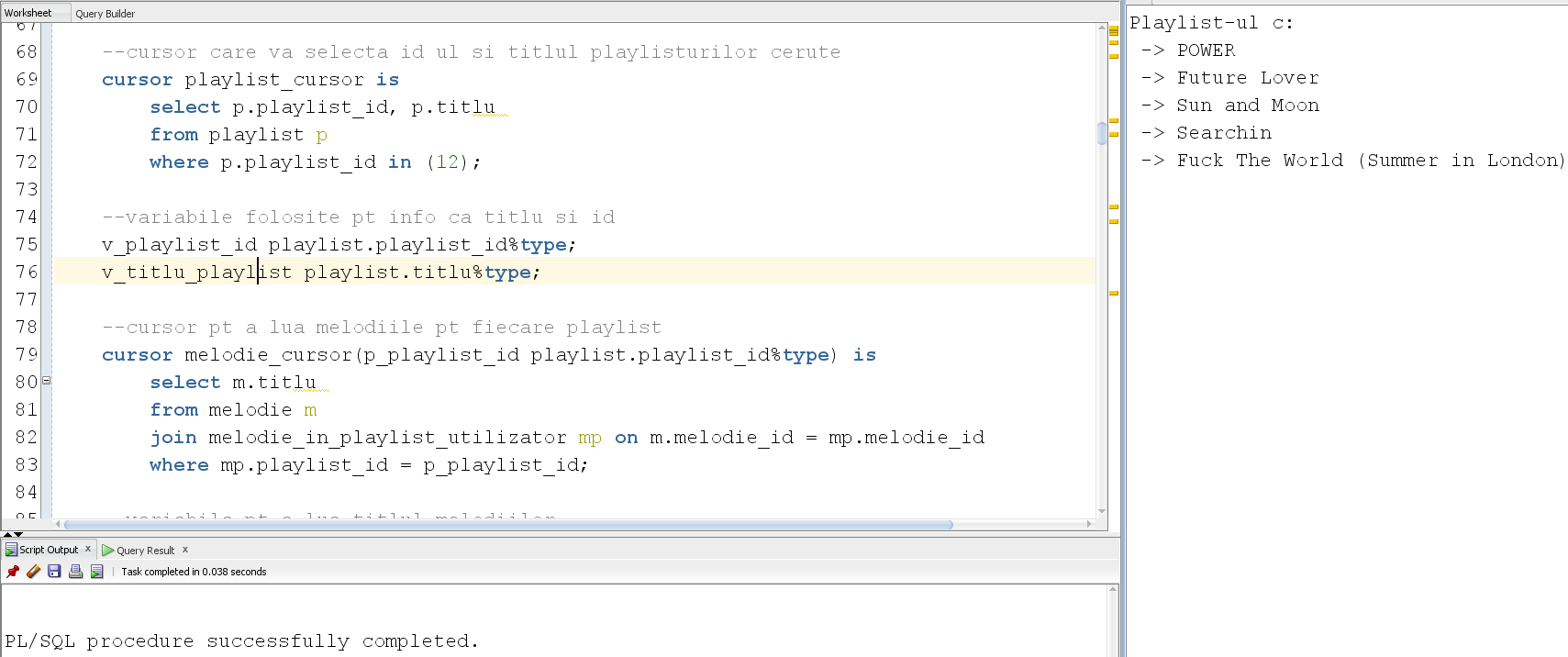
end loop;

end loop;

end;

/

**Print screen-uri: (vom rula procedura cu acelasi exemplu cu playlist-ul 12)**

****

1. **Folosind ciclu cursor cu subcerere** (folosind tot apelul cu playlist-ul cu id 12)

**Cod SQL:**

declare

--variabile folosite pt info ca titlu si id

v\_playlist\_id playlist.playlist\_id%type;

v\_titlu\_playlist playlist.titlu%type;

--variabila pt a lua titlul melodiilor

v\_titlu\_melodie melodie.titlu%type;

begin

for playlist\_rec in (

select p.playlist\_id, p.titlu

from playlist p

where p.playlist\_id in (12)

) loop

v\_playlist\_id := playlist\_rec.playlist\_id;

v\_titlu\_playlist := playlist\_rec.titlu;

dbms\_output.put\_line('Playlist-ul ' || v\_titlu\_playlist || ':');

for melodie\_rec in(

select m.titlu

from melodie m

join melodie\_in\_playlist\_utilizator mp on m.melodie\_id = mp.melodie\_id

where mp.playlist\_id = playlist\_rec.playlist\_id)

loop

v\_titlu\_melodie := melodie\_rec.titlu;

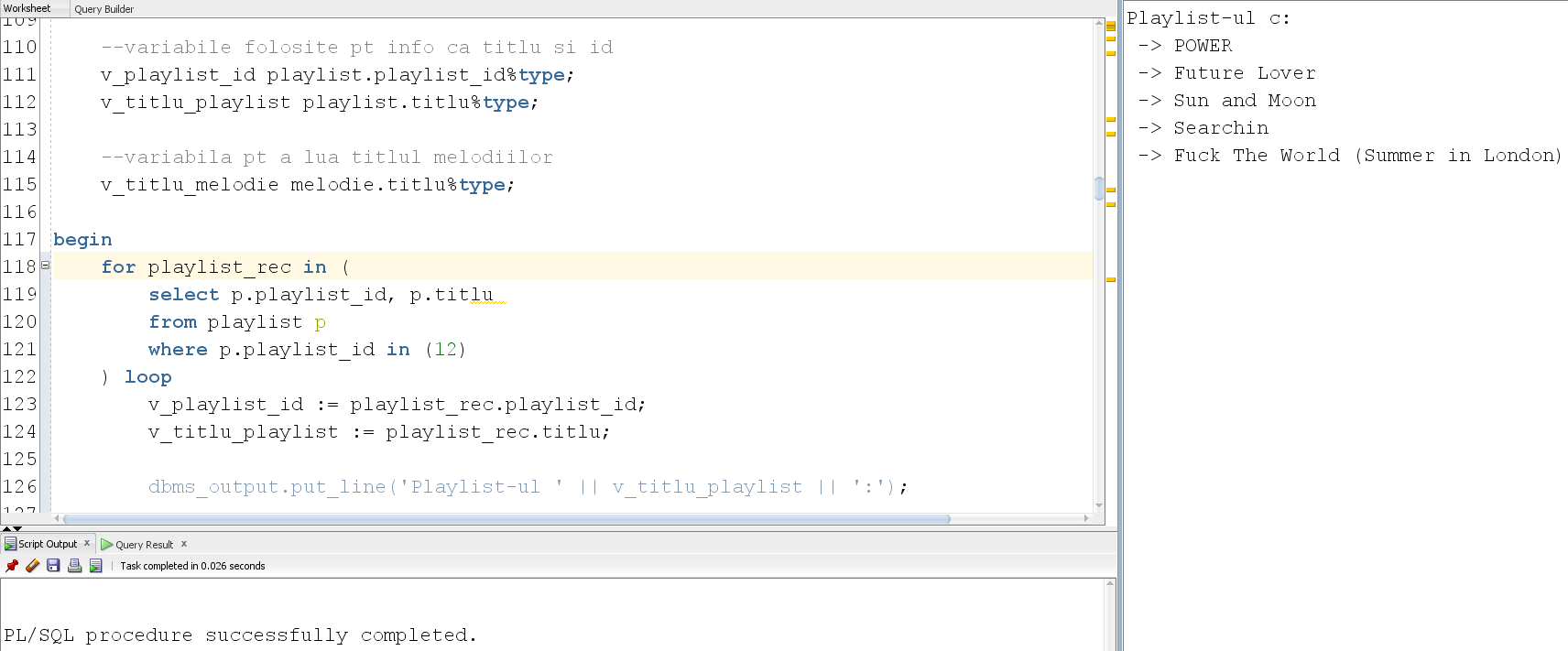
dbms\_output.put\_line(' -> ' || v\_titlu\_melodie);

end loop;

end loop;

end;

/

**Print screen-uri: **

**Laborator 4 PL/SQL**

E7. Adaptați cerințele exercițiilor 2 și 4 (folosind ca bază cerințele exercițiilor 1, respectiv 3) pentru diagrama proiectului prezentată la materia Baze de Date din anul I. Rezolvați aceste două exerciții în PL/SQL, folosind baza de date proprie.

Exercitiul 2 adaptat: Modificati functia definita la exercitiul 2 (folosind o functie stocata definiti un subprogram prin care sa obtineti numele unui artist a carei varsta este specificata. Tratati toate exceptiile ce pot fi generate), respectiv procedura definita la exercitiul 4(folosind o procedura stocata obtineti numele unui artist a carei varsta este specificata. Tratati toate exceptiile ce pot fi generate) astfel incat sa determine inserarea in tabelul info\_sma a informatiilor corespunzatoare fiecarui caz determinat de valoarea data ca parametru:

* Exista un singur artist cu varsta specificata
* Exista mai multi artisti cu varsta specificata
* Nu exista artisti cu varsta specificata

**Cod SQL pentru functia modificata:**

create or replace function f2\_sma (

v\_varsta artist.varsta%type default null)

return varchar2 is

v\_nume artist.nume%type;

begin

select nume into v\_nume

from artist

where varsta = v\_varsta;

--inserare in tabelul info\_sma pt fiecare caz

if v\_varsta is not null then --daca ajunge aici inseamna ca e un sg artist

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

values (user, sysdate, 'f2\_sma', 1, null);

else

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

values (user, sysdate, 'f2\_sma', 0 , 'Varsta specificata este NULL');

end if;

return v\_nume;

exception

when no\_data\_found then

--inserare pt atunci cand niciun artist nu are varsta asta

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

values(user, sysdate, 'f2\_sma', 0 , 'Nu exista artisti cu aceasta varsta');

raise\_application\_error(-20000, 'Nu exista artisti cu aceasta varsta');

when too\_many\_rows then

--inserare pt atunci cand mai multi artisti au varsta data

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

values(user, sysdate, 'f2\_sma', 0 ,'Exista mai multi artisti cu aceasta varsta');

raise\_application\_error(-20001, 'Exista mai multi artisti cu aceasta varsta');

when others then

--inserare pt cazul altor erori

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

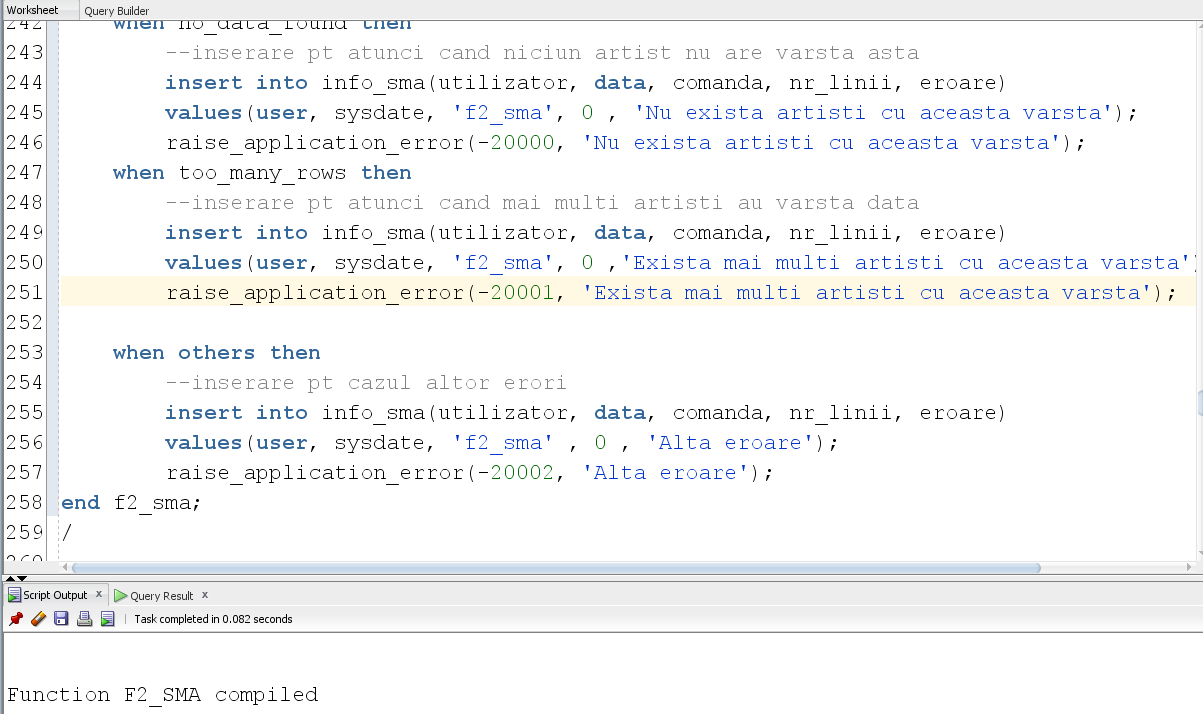
values(user, sysdate, 'f2\_sma' , 0 , 'Alta eroare');

raise\_application\_error(-20002, 'Alta eroare');

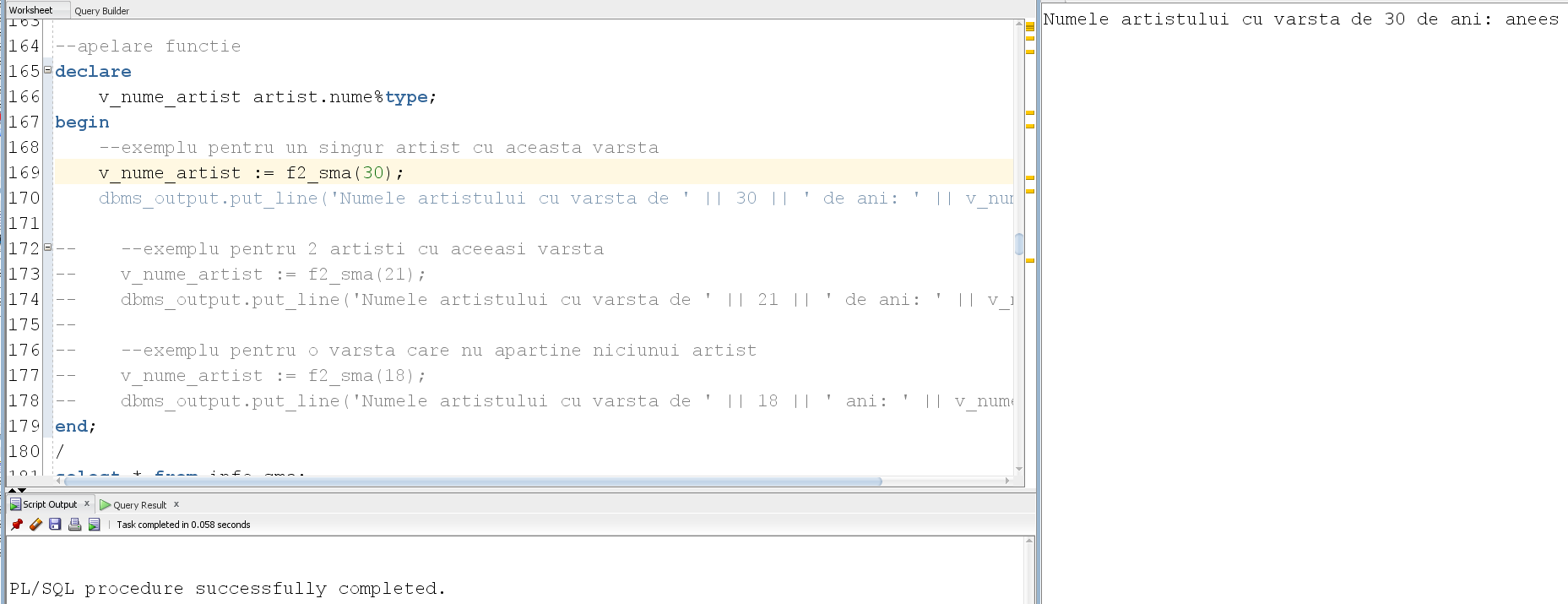
end f2\_sma;

/

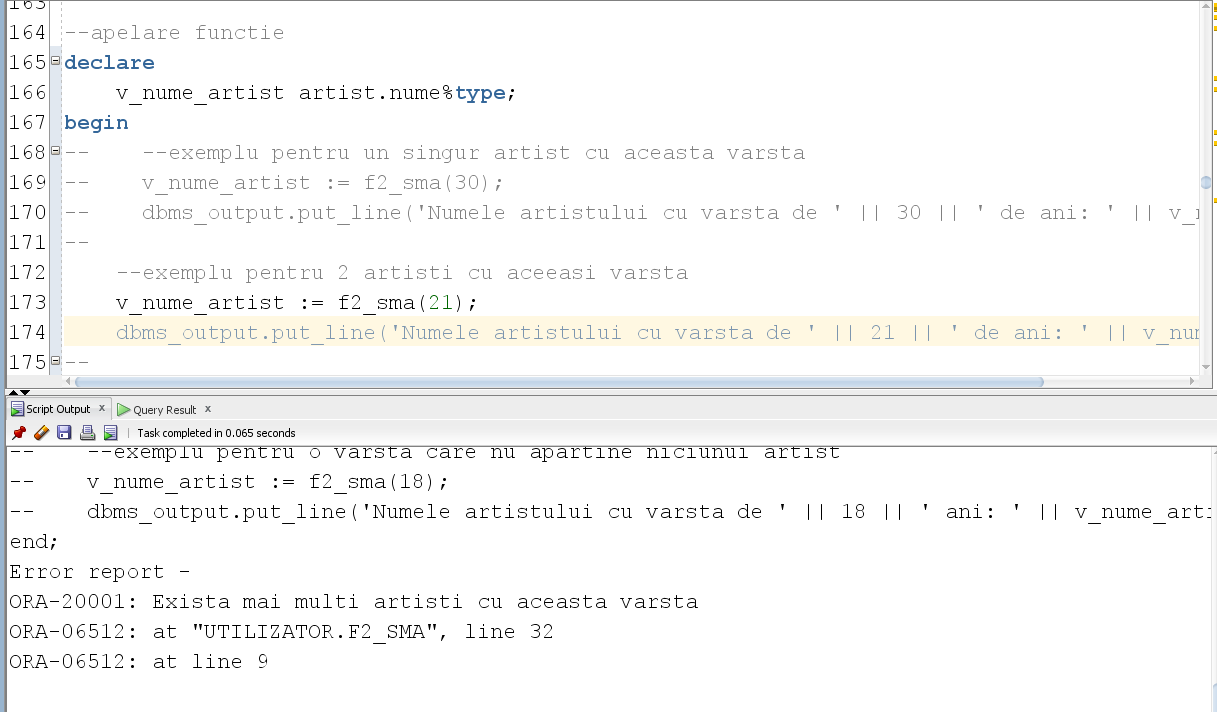
**Print screen-uri:**

****

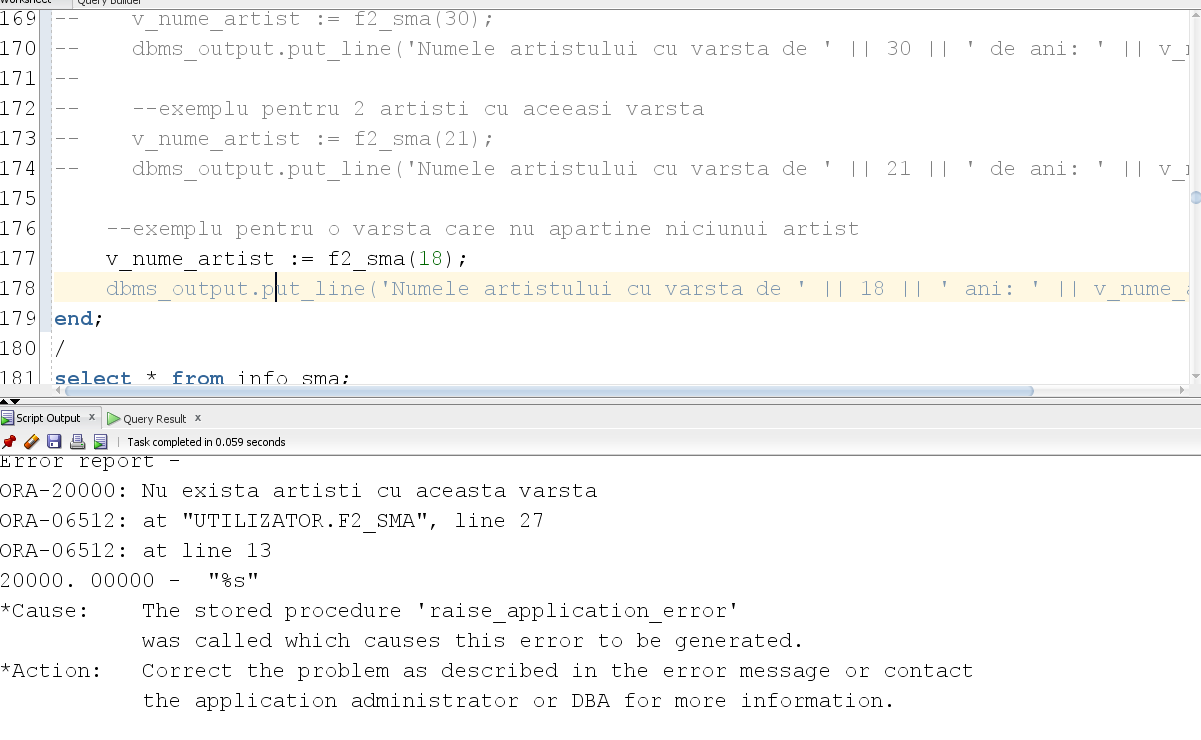
**Apelare functie pentru o valoare care determina un singur artist**

****

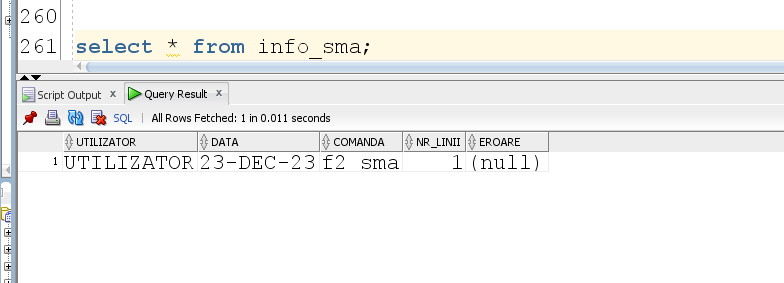
**Apelare functie pentru o valoare care determina mai multi artisti:**

****

**Apelare functie pentru o valoare care nu determina niciun artist:**

****

**Tabel info\_sma dupa apelarea procedurii:**

****

**Procedura modificata:**

**Cod SQL:**

create or replace procedure p4\_sma(

v\_varsta artist.varsta%type default null

) is

v\_nume artist.nume%type;

begin

select nume into v\_nume

from artist

where varsta= v\_varsta;

--afisare nume artist

dbms\_output.put\_line('Numele artistului cu varsta ' || v\_varsta || ' este: ' || v\_nume);

--inserare pt fiecare caz

if v\_varsta is not null then --inserare cand s a gasit un artist cu varsta data

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

values(user, sysdate, 'p4\_sma' , 1, null);

else --procedura a fost apelata cu varsta null

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

values(user, sysdate, 'p4\_sma', 0, 'Varsta specificata este null.');

end if;

exception

when no\_data\_found then

--inserare pt cand nu exista niciun artist

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

values(user, sysdate, 'p4\_sma', 0 ,'Nu exista artisti cu aceasta varsta');

raise\_application\_error(-20000,'Nu exista artisti cu aceasta varsta');

when too\_many\_rows then

--inserare pt atunci cand mai multi artisti au varsta data

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

values(user, sysdate, 'p4\_sma', 0 ,'Exista mai multi artisti cu aceasta varsta');

raise\_application\_error(-20001, 'Exista mai multi artisti cu aceasta varsta');

when others then

--inserare pt cazul altor erori

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

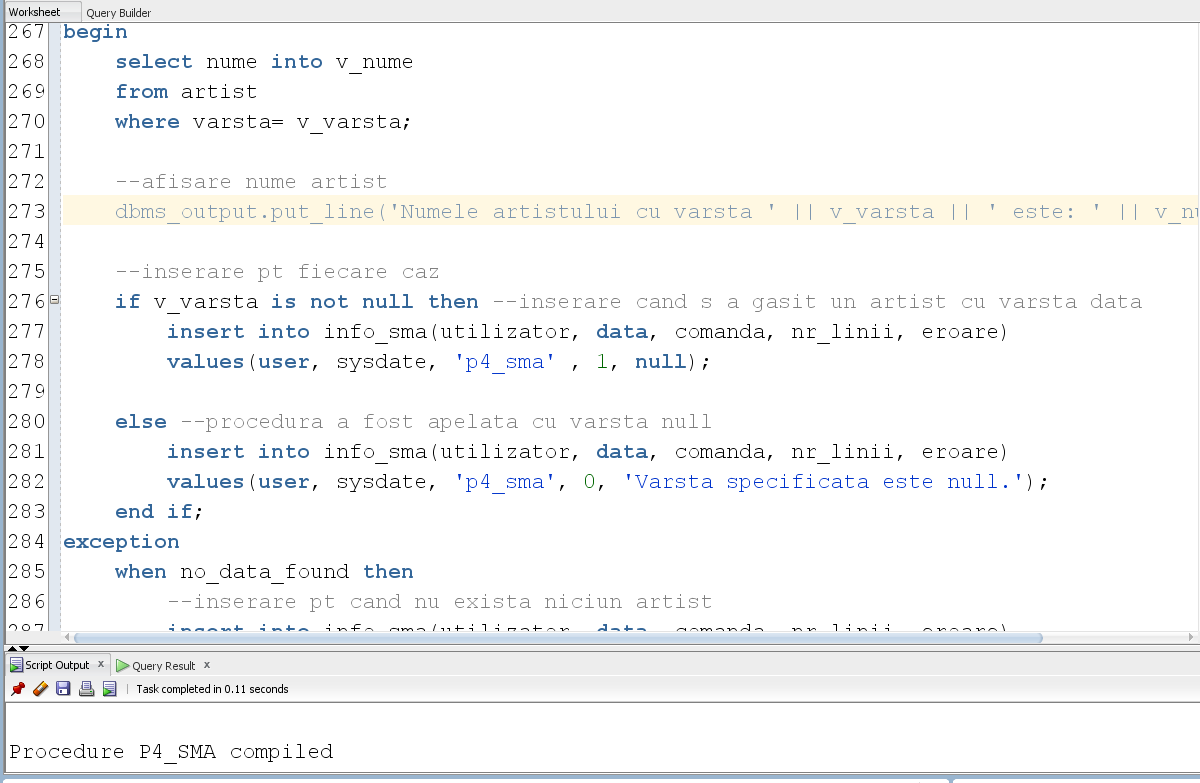
values(user, sysdate, 'p4\_sma' , 0 , 'Alta eroare');

raise\_application\_error(-20002, 'Alta eroare');

end p4\_sma;

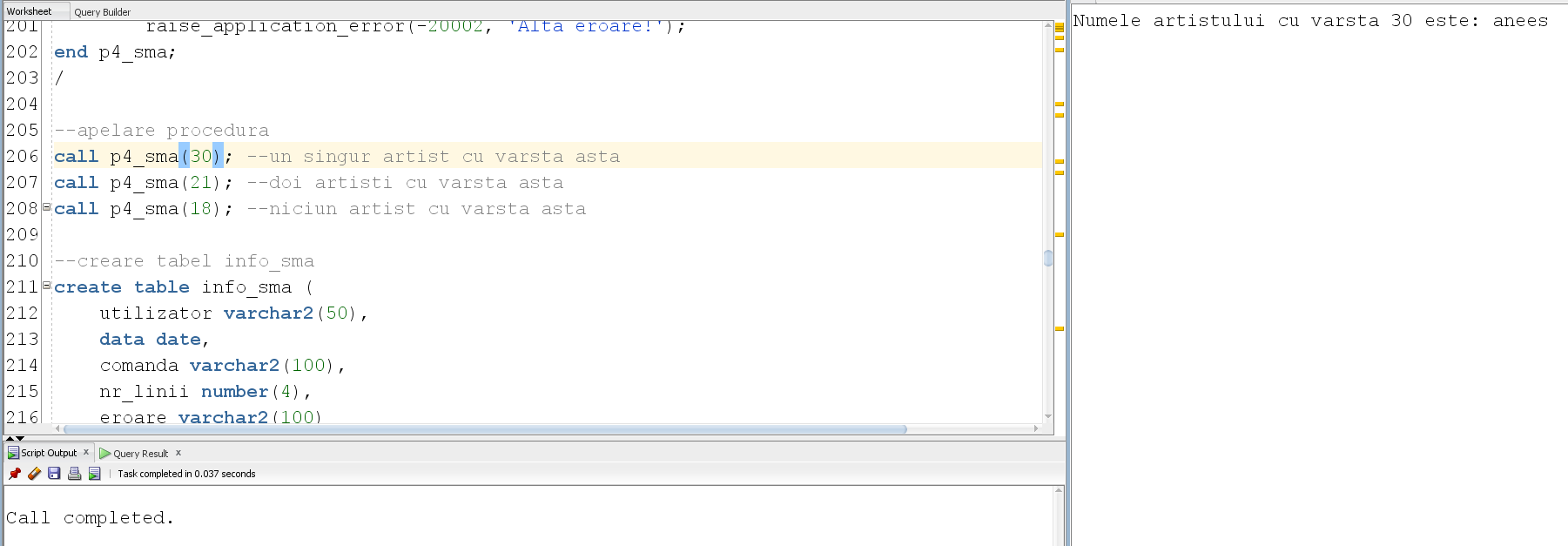
/

**Print screen-uri:**

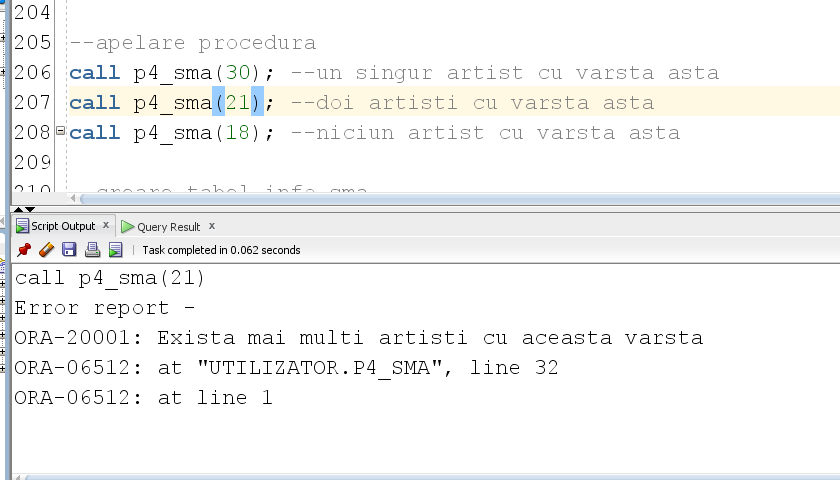
****

**Apelare procedura:**

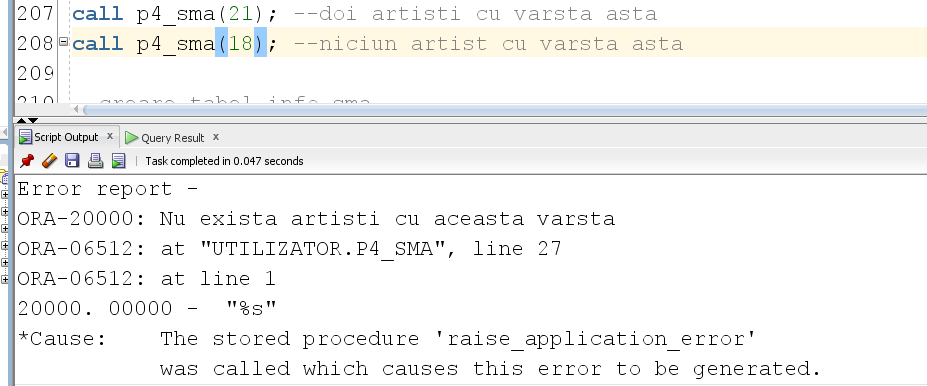
**-pentru un singur artist rezultat**

****

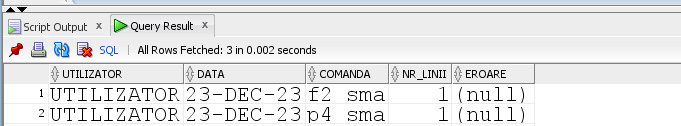
**Apel din care rezulta mai multi artisti cu varsta respectiva:**

****

**Apel din care nu rezulta niciun artist cu acea varsta:**

****

**Tabelul info\_sma dupa apelurile procedurii:**

****

**Cerinta exercitiul 4 adaptata:** Definiti o procedura stocata care mareste cu 10% pretul fiecarui album lansat de un artist al carui cod este dat ca parametru. Tratati cazul in care nu exista niciun artist cu codul dat. Inserati in tabelul info\_sma informatiile corespunzatoare fiecarui caz determinat de valoarea data pentru parametru.

**Cod SQL:**

create or replace procedure marire\_pret(

p\_id\_artist artist.artist\_id%type)

is

v\_procent number := 1.10; --marire cu 10%

v\_artist\_exista number;

nr\_linii number;

begin

--verificam daca exista artistul cu codul dat

select count(\*)

into v\_artist\_exista

from artist

where artist\_id = p\_id\_artist;

if v\_artist\_exista = 0 then

--inseram in info\_sma pentru cazul asta

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

values(user, sysdate, 'marire\_pret' ,0 , 'Nu exista artist cu id-ul specificat.');

raise\_application\_error(-20000, 'Nu exista artist cu id-ul specificat.');

else

--marim preturile albumelor asociate acestui artist identificat

update album

set pret = pret \* v\_procent

where artist\_id = p\_id\_artist;

--obtinem cate linii s au actualizat

nr\_linii := SQL%ROWCOUNT;

--inseram in info\_Sma

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

values(user, sysdate, 'marire\_pret' , nr\_linii, null);

end if;

exception --cum id u este cheie primara, nu poate exista decat eroare ca nu exista, tratata mai sus

--deci aici ramane doar eroarea others

when others then

--inseram in info\_sma ca a aparut o eroare

insert into info\_sma(utilizator, data, comanda, nr\_linii, eroare)

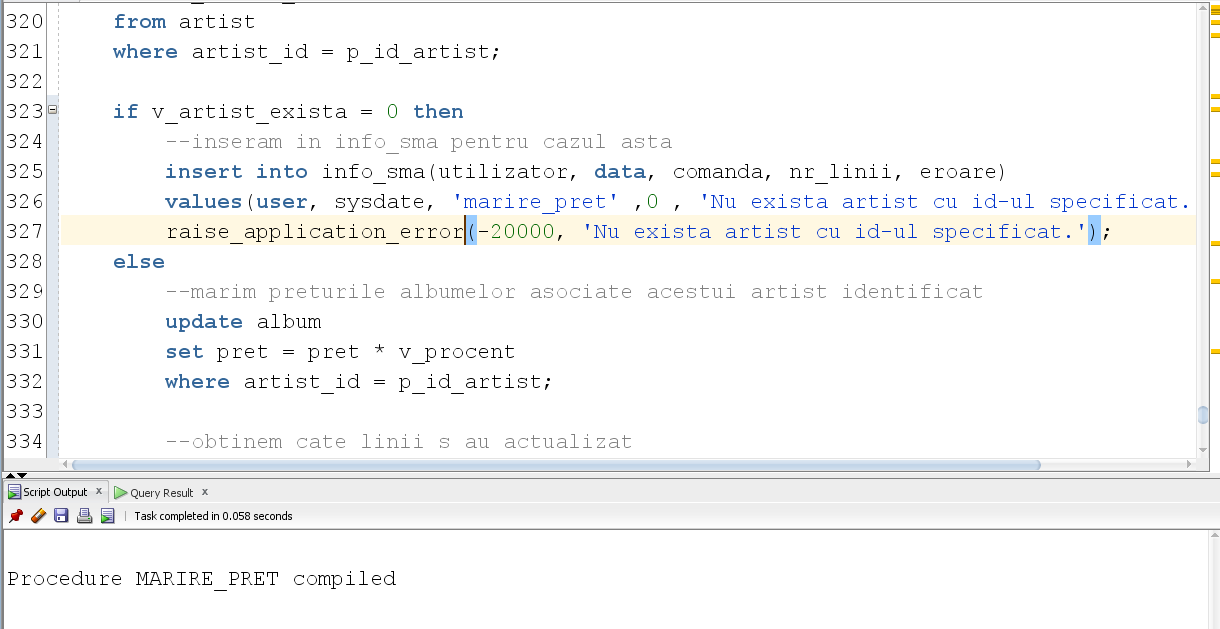
values(user, sysdate, 'marire\_pret' ,0 , 'Nu exista artist cu id-ul specificat.');

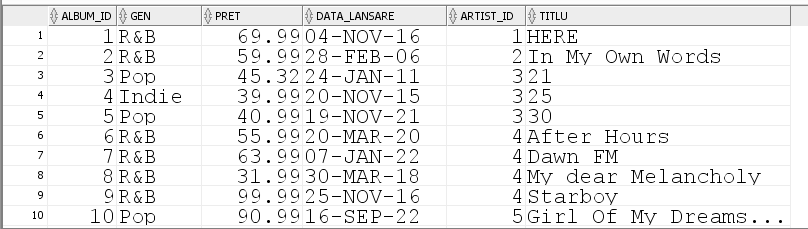
raise\_application\_error(-20001, 'A aparut o alta eroare');

end marire\_pret;

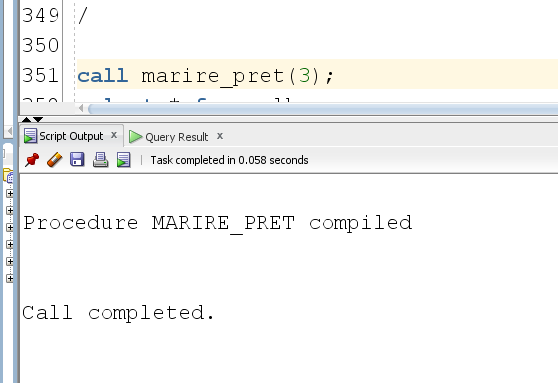
/

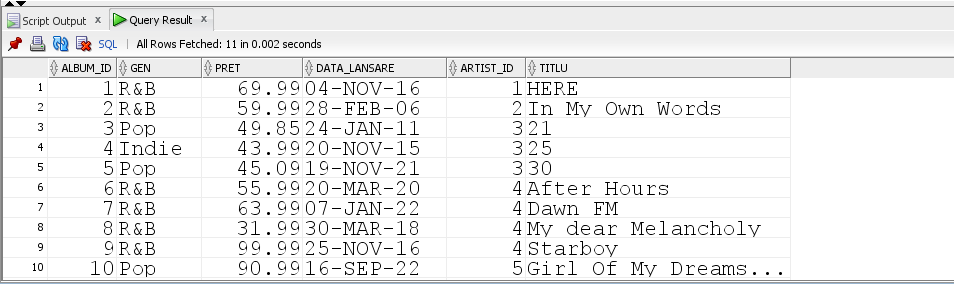
**Print screen-uri:**



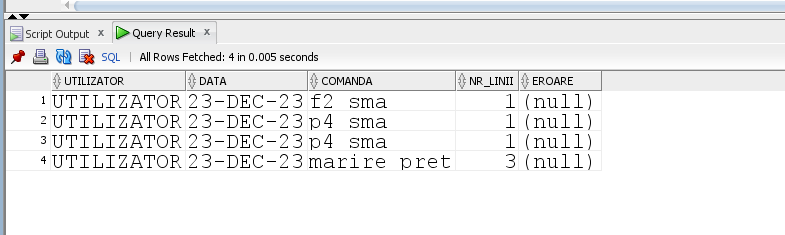
Preturi albume inainte de apelul procedurii: 

Apelare procedura pentru artistul cu id 3



Preturi albume dupa apelul procedurii: 

Tabelul info\_sma dupa apelul procedurii:



Apelare procedura cu id inexistent:

