

Tema proiectului: Sala de sport

Descrierea temei

Proiectul reprezintă baza de date a unei săli de sport.

Există mai multe sedii ale sălii de sport, fiecare sediu având denumire, capacitate și data de deschidere proprie. Fiecare sediu se află la o adresă care conține oraș, stradă și număr. Fiecare sediu are unul sau mai mulți angajați.

Pentru fiecare angajat știm numele, prenumele, telefonul, emailul, salariul, bonusul, sediul în care acesta lucrează, funcția acestuia și managerul care îi este superior. Fiecare angajat poate (daca acesta este antrenor) să susțină una sau mai multe ore de antrenament (ședințe) împreună cu un client.

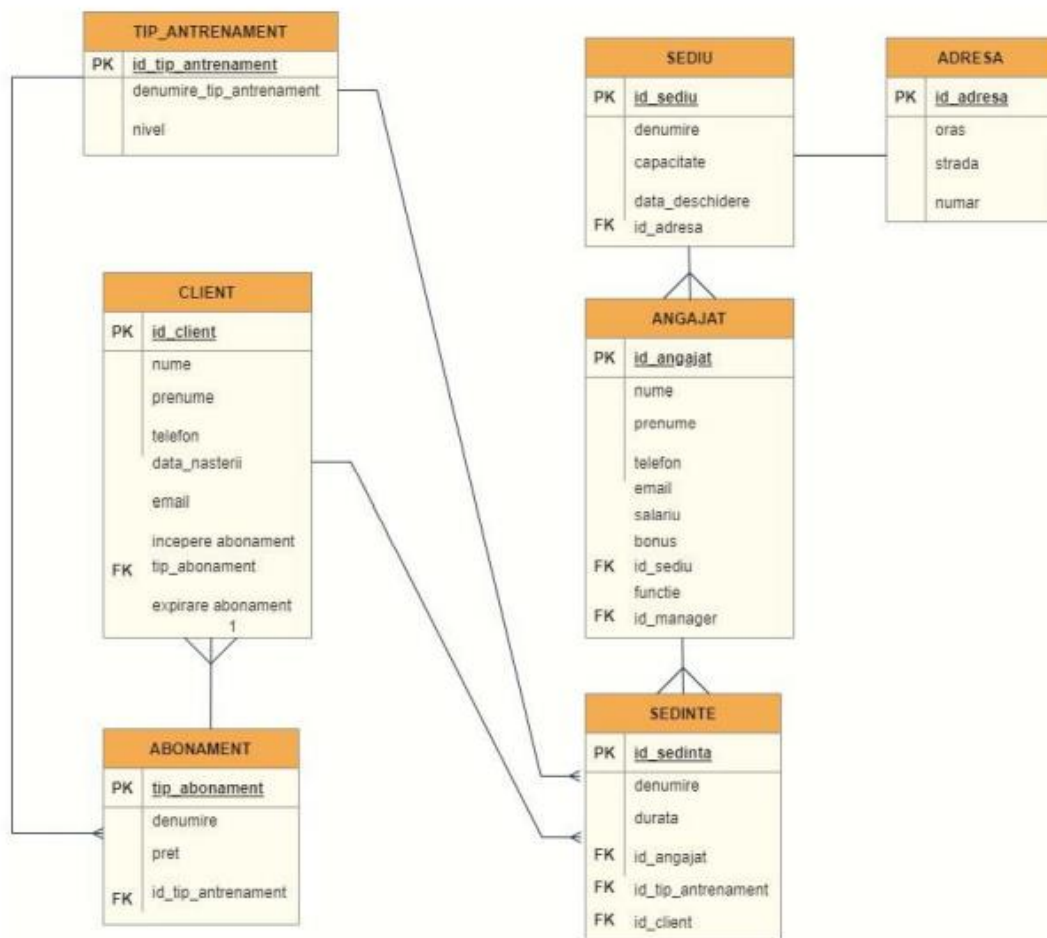
Fiecare ședință are un tip de antrenament (pentru fiecare tip de antrenament se cunosc denumirea care este alcătuită din inițiala sportului practicat și numărul ședinței respectivului client și nivelul - un număr de la 1 la 5, reprezentând dificultatea.

Fiecare client participă la una sau mai multe sedințe. Despre fiecare client se cunosc următoarele: numele, prenumele, telefonul, data nașterii, emailul, data de începere și data de expirare a abonamentului și tipul de abonament. Fiecare client are un abonament (iar un abonament poate fi avut de unul sau mai mulți clienți).

Fiecare abonament are un tip de antrenament aferent (fiecare tip de antrenament face parte dintr-un abonament).

Schema conceptuală pentru modelarea temei alese

Din schema va trebui să rezulte tipul legăturilor dintre entități după modelul schemei de la seminar. Se poate realiza folosind orice instrument sau site (e.g., www.draw.io)



C. Interacțiunea cu serverul Oracle prin intermediul comenzilor SQL

Să se creeze o noua tabelă numită `ad_abonamente_premium`, apoi să se includă în aceasta abonamentul cu id-ul 21 și să se afișeze tabela.

```
begin
```

```
--execute immediate 'DROP table ad_abonamente_premium';
```

```
execute immediate 'CREATE table ad_abonamente_premium AS SELECT * FROM  
ad_abonamente where 1=2';
```

```
end;
```

```
declare
```

```
v_tip ad_abonamente.tip_abonament%type;
```

```
v_den ad_abonamente.denumire%type;
```

```
v_pret ad_abonamente.pret%type;
```

```
v_id_antr ad_abonamente.id_tip_antrenament%type;
```

```
begin
```

```
select tip_abonament, denumire, pret, id_tip_antrenament into v_tip, v_den, v_pret, v_id_antr  
from ad_abonamente where tip_abonament = 21;
```

```
INSERT INTO ad_abonamente_premium (tip_abonament, denumire, pret, id_tip_antrenament)  
VALUES
```

```
(v_tip, v_den, v_pret, v_id_antr);
```

```
DBMS_OUTPUT.PUT_LINE ('S-a adaugat în tabela ad_abonamente_premium abonamentul cu  
id: '||v_tip||' denumire
```

```
'||v_den||' pret '||v_pret || ' și tip de antrenament ' || v_id_antr);
```

```
end;
```

```
/
```

```
select * from ad_abonamente_premium;
```

```

begin
--execute immediate 'DROP table ad_abonamente_premium';
execute immediate 'CREATE table ad_abonamente_premium AS SELECT * FROM ad_abonamente where l=2';
end;

declare
v_tip ad_abonamente.tip_abonament%type;
v_den ad_abonamente.denumire%type;
v_pret ad_abonamente.pret%type;
v_id_antr ad_abonamente.id_tip_antrenament%type;
begin
select tip_abonament, denumire, pret, id_tip_antrenament into v_tip, v_den, v_pret, v_id_antr
from ad_abonamente where tip_abonament = 21;

INSERT INTO ad_abonamente_premium (tip_abonament, denumire, pret, id_tip_antrenament) VALUES
(v_tip, v_den, v_pret, v_id_antr);
DBMS_OUTPUT.PUT_LINE ('S-a adaugat in tabela ad_abonamente_premium abonamentul cu id: '||v_tip||' denumire
'||v_den||' pret '||v_pret || ' si tip de antrenament ' || v_id_antr);
end;
/
select * from ad_abonamente_premium;

```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.008 seconds

| TIP_ABONAMENT | DENUMIRE | PRET | ID_TIP_ANTRENAMENT |
|---------------|----------|------|--------------------|
| 1 | 21 C | 150 | 1 |

E. Tratarea excepțiilor (minim 2 implicite, 2 explicite)

1. Sa se afișeze numele angajatului al cărui salariu este egal cu 3500 de lei. în cazul în care există mai mulți angajați cu acest salariu, tratați excepția și afișați un mesaj.

DECLARE

nume angajati.nume%type;

BEGIN

select nume into nume from ad_angajati where salariu=3500;

DBMS_OUTPUT.PUT_LINE('Angajatul cu salariul egal cu 3500 este: '||nume);

EXCEPTION

WHEN TOO_MANY_ROWS THEN

DBMS_OUTPUT.PUT_LINE('Există mai multi angajati care au salariul egal cu 3500, prin urmare, se recomanda utilizarea unui cursor și afisarea tuturor angajatilor cu salariul 3500');
END;

```

DECLARE
    nume angajati.nume%type;
BEGIN
    select nume into nume from ad_angajati where salariu=3500;
    DBMS_OUTPUT.PUT_LINE('Angajatul cu salariul egal cu 3500 este: '||nume);
EXCEPTION
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai multi angajati care au salariul egal cu 3500, prin urmare, se recomanda utilizarea unui cursor si afisarea tuturor angajatilor cu salariul 3500');
END;
  
```

Script Output x

Task completed in 0.047 seconds

Exista mai multi angajati care au salariul egal cu 3500, prin urmare, se recomanda utilizarea unui cursor si afisarea tuturor angajatilor cu salariul 3500

PL/SQL procedure successfully completed.

2. Să se ștergă sediile ale căror vechime depășește 3 ani sau a căror capacitate este mai mică de 25 de persoane. Dacă nu este posibil pentru că există angajați care lucrează în acest sediu, tratați excepția și afișați o eroare.

DECLARE

cod NUMBER;

mesaj VARCHAR2(255);

exceptie_stergere EXCEPTION;

PRAGMA EXCEPTION_INIT(exceptie_stergere, -2292);

BEGIN

DELETE FROM ad_sedii

where extract(year from data_deschidere)< extract(year from sysdate)-3 or capacitate<25 ;

EXCEPTION

WHEN exceptie_stergere THEN

dbms_output.put_line('Nu puteti sterge sediul deoarece există angajati care sunt asignati acestuia');

cod:=SQLCODE;

mesaj:=SQLERRM;

INSERT INTO AD_ERORI VALUES(USER, SYSDATE, cod, mesaj);

END;

```

DECLARE
cod NUMBER;
mesaj VARCHAR2(255);
exceptie_stergere EXCEPTION;
PRAGMA EXCEPTION_INIT(exceptie_stergere, -2292);
BEGIN
DELETE FROM ad_sedii
where extract(year from data_deschidere)< extract(year from sysdate)-3 or capacitate<25 ;
EXCEPTION
WHEN exceptie_stergere THEN
dbms_output.put_line('Nu puteti sterge sediul deoarece exista angajati care sunt asignati acestuia');
cod:=SQLCODE;
mesaj:=SQLERRM;
INSERT INTO AD_ERORI VALUES(USER, SYSDATE, cod, mesaj);
END;

select * from ad_erori;

```

Script Output x Query Result x

SQL | All Rows Fetched: 5 in 0.011 seconds

| UTILIZATOR | DATA | COD_EROARE | MESAJ_EROARE |
|--------------|-----------|--|--------------|
| DUMITRIUA_51 | 30-APR-22 | -2292 ORA-02292: integrity constraint (DUMITRIUA_51.FK_SEDIU) violated - child record found | |
| DUMITRIUA_51 | 30-APR-22 | -2292 ORA-02292: integrity constraint (DUMITRIUA_51.FK_SEDIU) violated - child record found | |
| DUMITRIUA_51 | 30-APR-22 | -2292 ORA-02292: integrity constraint (DUMITRIUA_51.PROD_COM_ID_PRODUS_FK) violated - child record found | |
| DUMITRIUA_51 | 30-APR-22 | -2292 ORA-02292: integrity constraint (DUMITRIUA_51.FK_SEDIU) violated - child record found | |
| DUMITRIUA_51 | 28-MAY-22 | -2292 ORA-02292: integrity constraint (DUMITRIUA_51.FK_SEDIU) violated - child record found | |

3. Scrieți un program care crește bonusul angajatului cu numele Zegreanu cu două puncte procentuale. În cazul în care acesta nu există, tratați excepția și afișați un mesaj de eroare.

DECLARE

invalid_angajat EXCEPTION;

BEGIN

UPDATE ad_angajati

SET bonus=0.2

WHERE lower(nume)='zegreanu';

IF SQL%NOTFOUND THEN

RAISE invalid_angajat;

END IF;

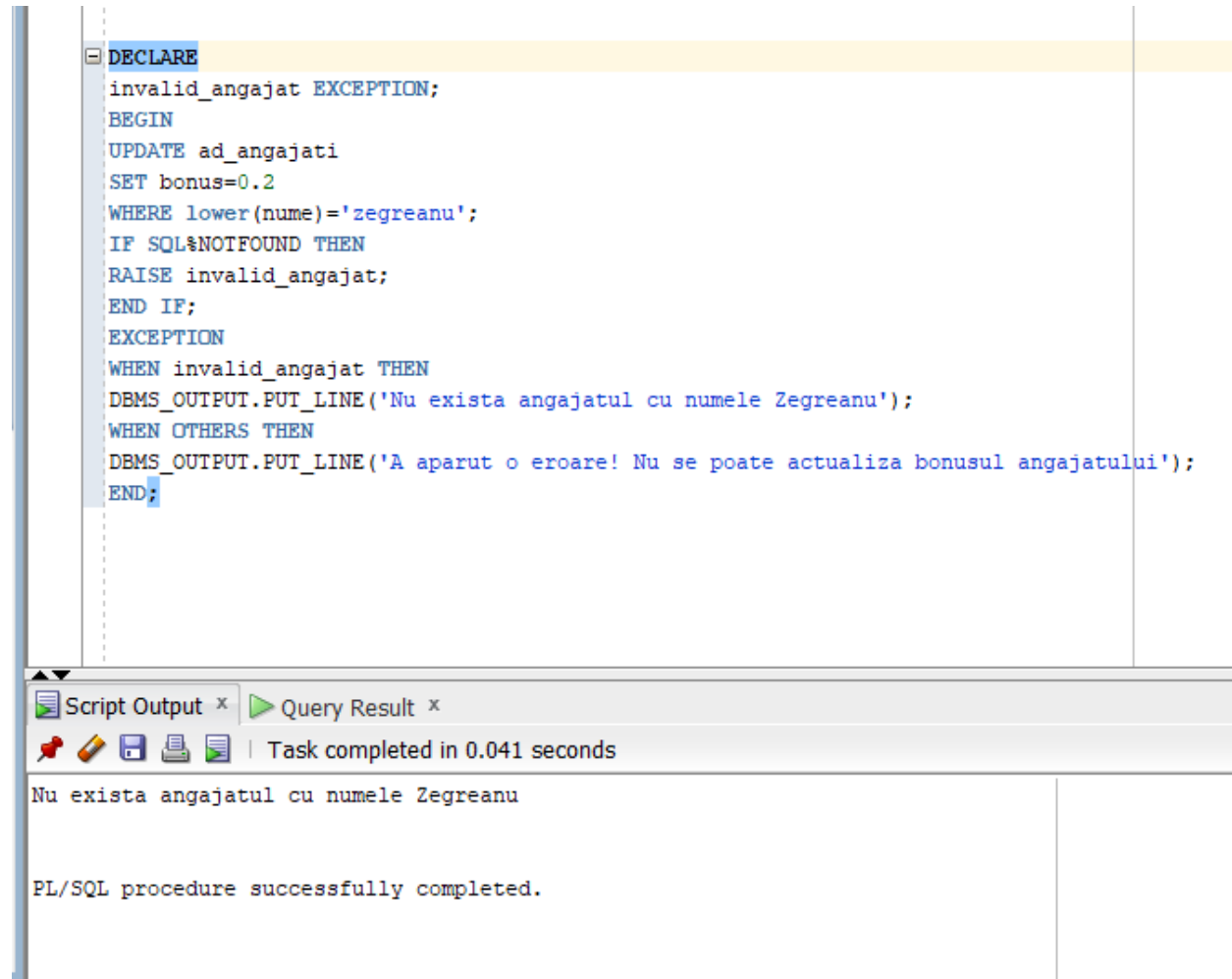
EXCEPTION

WHEN invalid_angajat THEN

DBMS_OUTPUT.PUT_LINE('Nu exista angajatul cu numele Zegreanu');

WHEN OTHERS THEN

```
DBMS_OUTPUT.PUT_LINE('A aparut o eroare! Nu se poate actualiza bonusul angajatului');  
END;
```



The screenshot displays a SQL IDE interface. The main editor window shows a PL/SQL procedure named `invalid_angajat`. The procedure declares an exception, attempts to update the `bonus` of employees with the name 'Zegreanu', and handles the `SQL%NOTFOUND` exception by raising `invalid_angajat`. It also includes a general exception handler for other errors. The procedure is executed, and the results are shown in the 'Script Output' and 'Query Result' tabs. The 'Script Output' tab shows the message 'Nu exista angajatul cu numele Zegreanu' and 'PL/SQL procedure successfully completed.' The 'Query Result' tab shows the message 'Task completed in 0.041 seconds'.

```
DECLARE  
invalid_angajat EXCEPTION;  
BEGIN  
UPDATE ad_angajati  
SET bonus=0.2  
WHERE lower(nume)='zegreanu';  
IF SQL%NOTFOUND THEN  
RAISE invalid_angajat;  
END IF;  
EXCEPTION  
WHEN invalid_angajat THEN  
DBMS_OUTPUT.PUT_LINE('Nu exista angajatul cu numele Zegreanu');  
WHEN OTHERS THEN  
DBMS_OUTPUT.PUT_LINE('A aparut o eroare! Nu se poate actualiza bonusul angajatului');  
END;
```

Script Output x Query Result x
Task completed in 0.041 seconds
Nu exista angajatul cu numele Zegreanu
PL/SQL procedure successfully completed.

4. Să se mareasca salariul angajatilor care au salariul mai mic de 2500 de lei, astfel încât acesta să devina 2500 de lei. Daca nu se produce nicio modificare, să se afișeze un mesaj, folosindu-se o excepție definită de utilizator.

```
DECLARE
```

```
exceptie_salariu EXCEPTION;
```

```
BEGIN
```

```
UPDATE ad_angajati SET salariu=2500 WHERE salariu<2500;
```

```
IF SQL%NOTFOUND THEN
```

```
    RAISE exceptie_salariu;
```

```
END IF;
```

```
EXCEPTION
```

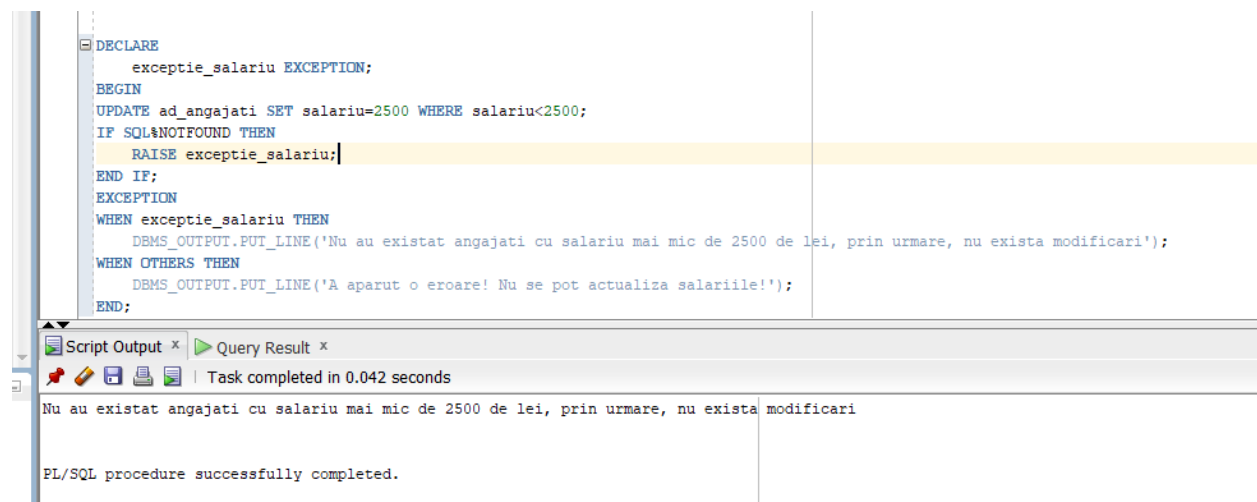
```
WHEN exceptie_salariu THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Nu au existat angajati cu salariu mai mic de 2500 de lei, prin  
    urmare, nu exista modificari');
```

```
WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.PUT_LINE('A aparut o eroare! Nu se pot actualiza salariile!');
```

```
END;
```



```
DECLARE
    exceptie_salariu EXCEPTION;
BEGIN
    UPDATE ad_angajati SET salariu=2500 WHERE salariu<2500;
    IF SQL%NOTFOUND THEN
        RAISE exceptie_salariu;
    END IF;
EXCEPTION
    WHEN exceptie_salariu THEN
        DBMS_OUTPUT.PUT_LINE('Nu au existat angajati cu salariu mai mic de 2500 de lei, prin urmare, nu exista modificari');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('A aparut o eroare! Nu se pot actualiza salariile!');
END;
```

Script Output x Query Result x

Task completed in 0.042 seconds

Nu au existat angajati cu salariu mai mic de 2500 de lei, prin urmare, nu exista modificari

PL/SQL procedure successfully completed.

F. Gestionarea cursorilor: minim 3 impliciți și 3 expliciți (cu și fără parametri)

1. Scrieți un program care afișează toți clientii care au abonamentul cu id-ul 21, folosind un cursor explicit.

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
cursor client_cursor is select id_client, nume, prenume, telefon, email, incepere_abonament from  
ad_clienti where
```

```
tip_abonament=21;
```



```

client_id ad_clienti.id_client%type;
client_nume ad_clienti.nume%type;
client_prenume ad_clienti.prenume%type;
client_telefon ad_clienti.telefon%type;
client_email ad_clienti.email%type;
client_incepre_abonament ad_clienti.incepere_abonament%type;

BEGIN

dbms_output.put_line('Sa se afiseze lista clientilor care au abonamentul cu id-ul 21');

open client_cursor;

loop

fetch client_cursor into client_id, client_nume, client_prenume,client_telefon,client_email,
client_incepre_abonament ;

exit when client_cursor%notfound;

dbms_output.put_line('Nume: '||client_nume||' Prenume: '||client_prenume || ' Telefon: ' ||
client_telefon || ' Email: ' || client_email || ' are abonamentul inceput în data de ' ||
client_incepre_abonament);

end loop;

close client_cursor;

end;

```

The screenshot displays the Oracle SQL Developer environment. The top pane shows a PL/SQL script in the Query Builder, which defines a cursor, opens it, and loops through the results to print client details and their subscription start dates. The bottom pane shows the 'Script Output' window, which reports that the task completed successfully in 0.099 seconds. Below this, the output of the script is shown, indicating that the procedure was completed successfully and displaying the list of clients for subscription ID 21.

```

DECLARE
cursor client_cursor is select id_client, nume, prenume, telefon, email, incepere_abonament from ad_clienti where
tip_abonament=21;
client_id ad_clienti.id_client%type;
client_nume ad_clienti.nume%type;
client_prenume ad_clienti.prenume%type;
client_telefon ad_clienti.telefon%type;
client_email ad_clienti.email%type;
client_incepre_abonament ad_clienti.incepere_abonament%type;
BEGIN
dbms_output.put_line('Sa se afiseze lista clientilor care au abonamentul cu id-ul 21');
open client_cursor;
loop
fetch client_cursor into client_id, client_nume, client_prenume,client_telefon,client_email, client_incepre_abonament ;
exit when client_cursor%notfound;
dbms_output.put_line('Nume: '||client_nume||' Prenume: '||client_prenume || ' Telefon: ' || client_telefon || ' Email: ' || client_email || ' are abonamentul inceput in data de ' || cli
end loop;
close client_cursor;
end;

```

Task completed in 0.099 seconds

Nu exista angajatul cu numele Zegreanu

PL/SQL procedure successfully completed.

Sa se afiseze lista clientilor care au abonamentul cu id-ul 21

| | | | | |
|-------------|-----------------|-----------------------|-----------------------------|--|
| Nume: Diana | Prenume: Riscov | Telefon: +40733823899 | Email: dianars@gmail.com | are abonamentul inceput in data de 10-NOV-21 |
| Nume: Ioana | Prenume: Echim | Telefon: +40733844657 | Email: echimioana@gmail.com | are abonamentul inceput in data de 08-APR-21 |

PL/SQL procedure successfully completed.

2. Scrieți un program care sa afișeze primii doi antrenori cu cele mai multe ședințe de antrenament susținute, folosind un cursor explicit.

```
DECLARE
```

```
CURSOR cursor_angajat IS
```

```
select a.id_angajat, a.numa, a.prenume, a.salariu, count(s.id_angajat) NRSedinte
```

```
from ad_angajati a, ad_sedinte s
```

```
where a.id_angajat=s.id_angajat
```

```
group by a.id_angajat, a.numa, a.prenume, a.salariu
```

```
order by count(s.id_angajat) desc;
```

```
rec_angajat cursor_angajat%rowtype;
```

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE('Primii doi angajati cu cele mai multe sedinte: ');
```

```
IF NOT cursor_angajat%ISOPEN THEN
```

```
    OPEN cursor_angajat;
```

```
END IF;
```

```
LOOP
```

```
    FETCH cursor_angajat INTO rec_angajat;
```

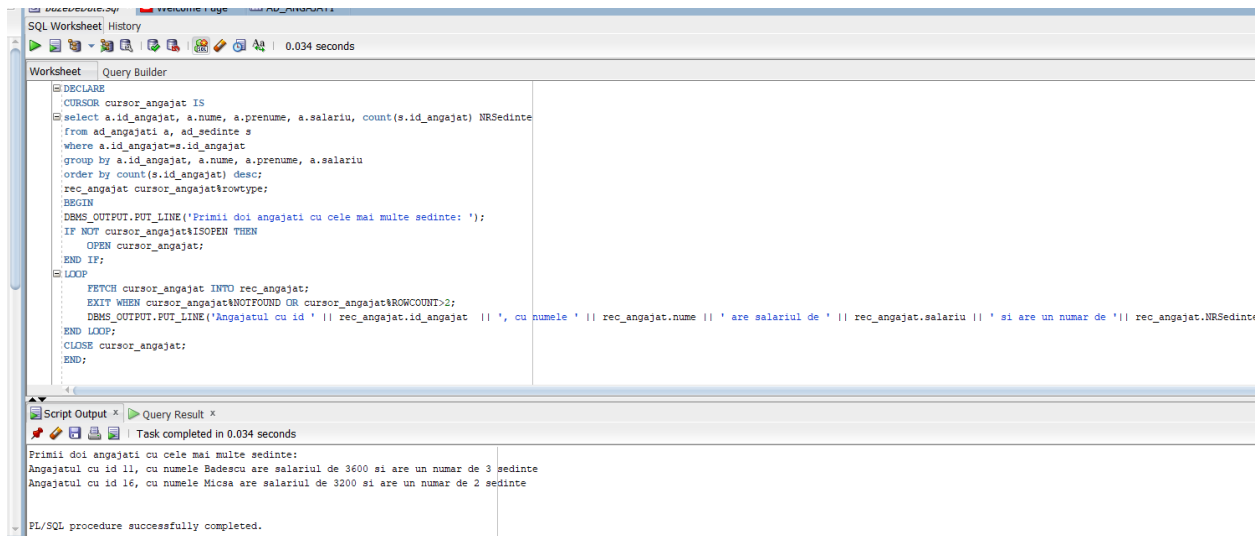
```
    EXIT WHEN cursor_angajat%NOTFOUND OR cursor_angajat%ROWCOUNT>2;
```

```
    DBMS_OUTPUT.PUT_LINE('Angajatul cu id ' || rec_angajat.id_angajat || ', cu numele ' ||  
rec_angajat.numa || ' are salariul de ' || rec_angajat.salariu || ' și are un numar de ' ||  
rec_angajat.NRSedinte || ' sedinte');
```

```
END LOOP;
```

```
CLOSE cursor_angajat;
```

```
END;
```



1. Scrieți un program care să afișeze primii patru angajati care au cel mai mare salariu, folosind un cursor explicit.

DECLARE

cursor c_salariati is

select * from ad_angajati

order by salariu desc;

rec_angajati c_salariati%rowtype;

BEGIN

DBMS_OUTPUT.PUT_LINE('Primii patru angajati în ordine în funcție de salariu:');

IF NOT c_salariati%ISOPEN THEN

OPEN c_salariati;

END IF;

LOOP

FETCH c_salariati INTO rec_angajati;

EXIT WHEN c_salariati%NOTFOUND OR c_salariati%ROWCOUNT>4;

DBMS_OUTPUT.PUT_LINE('Nume angajat: ' || rec_angajati.numa || ' Prenume: ' || rec_angajati.prenume || ' cu salariul ' || rec_angajati.salariu);

END LOOP;

CLOSE c_salariati;

END;

The screenshot shows a SQL IDE with a query window containing a PL/SQL procedure. The procedure declares a cursor c_salariati, fetches data from ad_angajati, and prints the first four employees in descending order of salary. The results pane shows the output of the procedure, listing the names and salaries of the first four employees.

```

DECLARE
cursor c_salariati is
select * from ad_angajati
order by salariu desc;
rec_angajati c_salariati%rowtype;
BEGIN
DBMS_OUTPUT.PUT_LINE('Primii patru angajati in ordine in functie de salariu:');
IF NOT c_salariati%ISOPEN THEN
OPEN c_salariati;
END IF;
LOOP
FETCH c_salariati INTO rec_angajati;
EXIT WHEN c_salariati%NOTFOUND OR c_salariati%ROWCOUNT>4;
DBMS_OUTPUT.PUT_LINE('Nume angajat: '||rec_angajati.num|| ' Prenume: '||rec_angajati.prenume ||' cu salariul '||rec_angajati.salariu);
END LOOP;
CLOSE c_salariati;
END;

```

Script Output x Query Result x

Task completed in 0.045 seconds

```

Primii patru angajati in ordine in functie de salariu:
Nume angajat: Dumitriu Prenume: Ana Maria cu salariul 4200
Nume angajat: Median Prenume: Sergiu cu salariul 4100
Nume angajat: Mircea Prenume: Andrei cu salariul 4000
Nume angajat: Macra Prenume: George cu salariul 3900

```

PL/SQL procedure successfully completed.

2. Pentru angajații sediului nou, cu id-ul egal cu 10, să se calculeze care este remunerația (salariu și bonus) și care este prima pe care o vor obține în prima luna de lucru (remuneratie + 1.500), folosindu-se un cursor implicit. Datele referitoare la salariu se vor pastra într-o nouă tabelă numită ad_salarii.

DROP TABLE AD_SALARII;

CREATE TABLE AD_SALARII AS

SELECT id_angajat,nume, prenume, SUM(salariu+bonus) as valoare

FROM ad_angajati

WHERE id_sediu= 10

GROUP BY id_angajat, nume, prenume

Order by valoare desc;

ALTER TABLE AD_SALARII

ADD(prima NUMBER(10));

DECLARE

CURSOR c_salarii IS

SELECT id_angajat,nume, prenume, valoare, prima

FROM AD_SALARII

FOR UPDATE OF prima NOWAIT;

BEGIN

FOR rec_salarii IN c_salarii LOOP

UPDATE AD_SALARII

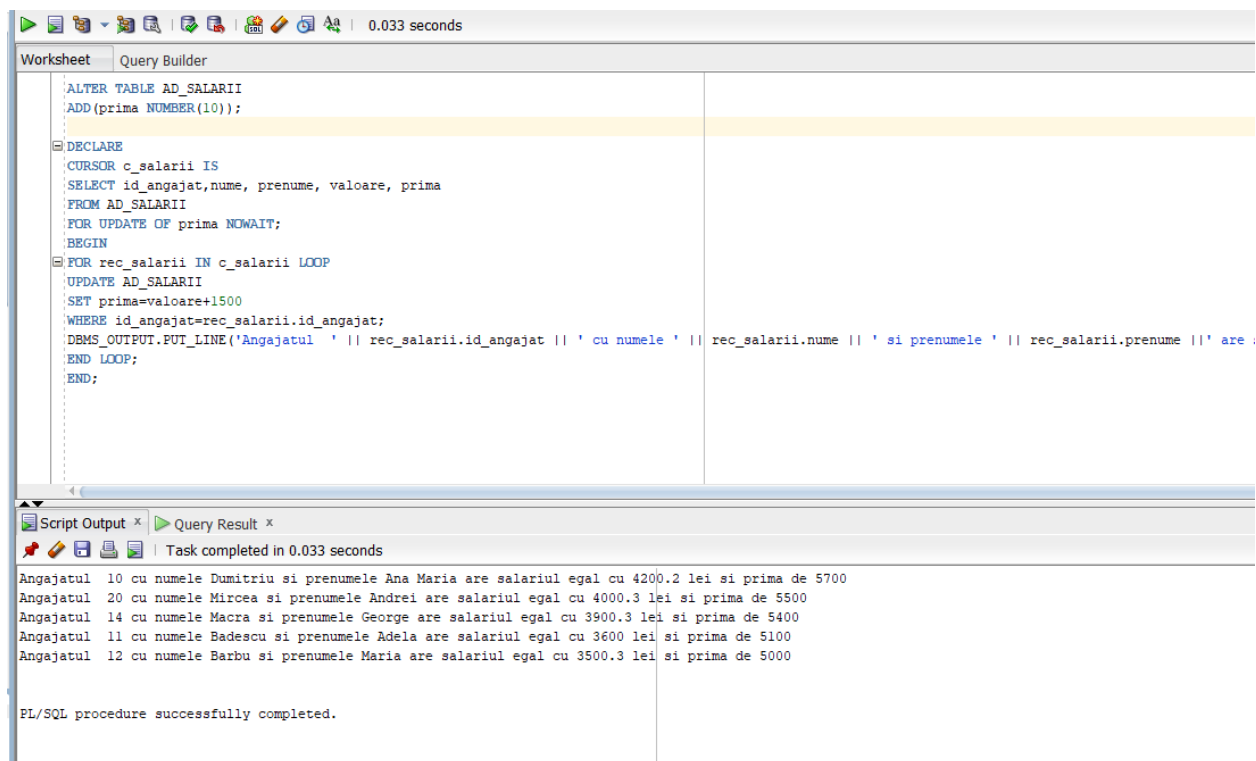
SET prima=valoare+1500

WHERE id_angajat=rec_salarii.id_angajat;

DBMS_OUTPUT.PUT_LINE('Angajatul ' || rec_salarii.id_angajat || ' cu numele ' ||
rec_salarii.nume || 'si prenumele ' || rec_salarii.prenume || ' are salariul egal cu
'||rec_salarii.valoare||' lei și prima de '||rec_salarii.prima);

END LOOP;

END;



The screenshot shows a SQL IDE interface. The top toolbar indicates a duration of 0.033 seconds. The main window is divided into two panes: 'Worksheet' and 'Query Builder'. The 'Worksheet' pane contains the following PL/SQL code:

```
ALTER TABLE AD_SALARII
ADD(prima NUMBER(10));

DECLARE
CURSOR c_salarii IS
SELECT id_angajat,nume, prenume, valoare, prima
FROM AD_SALARII
FOR UPDATE OF prima NOWAIT;
BEGIN
FOR rec_salarii IN c_salarii LOOP
UPDATE AD_SALARII
SET prima=valoare+1500
WHERE id_angajat=rec_salarii.id_angajat;
DBMS_OUTPUT.PUT_LINE('Angajatul ' || rec_salarii.id_angajat || ' cu numele ' || rec_salarii.nume || ' si prenumele ' || rec_salarii.prenume || ' are
END LOOP;
END;
```

The bottom pane is split into 'Script Output' and 'Query Result'. The 'Script Output' pane shows the following output:

```
Task completed in 0.033 seconds

Angajatul 10 cu numele Dumitriu si prenumele Ana Maria are salariul egal cu 4200.2 lei si prima de 5700
Angajatul 20 cu numele Mircea si prenumele Andrei are salariul egal cu 4000.3 lei si prima de 5500
Angajatul 14 cu numele Macra si prenumele George are salariul egal cu 3900.3 lei si prima de 5400
Angajatul 11 cu numele Badescu si prenumele Adela are salariul egal cu 3600 lei si prima de 5100
Angajatul 12 cu numele Barbu si prenumele Maria are salariul egal cu 3500.3 lei si prima de 5000

PL/SQL procedure successfully completed.
```

3. Folosind un cursor implicit, să se afișeze pentru fiecare sediu care este id-ul, numărul de angajați și capacitatea acestuia.

DECLARE

cursor c_sediu is

select s.id_sediu,s.capacitate, s.denumire, count(a.id_sediu) as NrAngajati

from ad_sedii s, ad_angajati a

where s.id_sediu = a.id_sediu

group by s.id_sediu, s.capacitate, s.denumire;

BEGIN

DBMS_OUTPUT.PUT_LINE('Dep și număr angajați:');

FOR rec_sediu în c_sediu loop

IF rec_sediu.capacitate >=20 and rec_sediu.NrAngajati >=rec_sediu.capacitate/10 then

DBMS_OUTPUT.PUT_LINE('Sediul '||rec_sediu.denumire || ' cu id-ul ' || rec_sediu.id_sediu
|| ' are un număr de '||rec_sediu.NrAngajati || ' angajați și capacitate de ' || rec_sediu.capacitate || '
clienți');

end if;

end loop;

END;

```

DECLARE
    cursor c_sediu is
    select s.id_sediu, s.capacitate, s.denumire, count(a.id_sediu) as NrAngajati
    from ad_sedii s, ad_angajati a
    where s.id_sediu = a.id_sediu
    group by s.id_sediu, s.capacitate, s.denumire;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Departamentul si numarul de angajati:');
    FOR rec_sediu in c_sediu loop
        IF rec_sediu.capacitate >=20 and rec_sediu.NrAngajati >=rec_sediu.capacitate/10 then
            DBMS_OUTPUT.PUT_LINE('Sediul '||rec_sediu.denumire || ' cu id-ul ' || rec_sediu.id_sediu || ' are un numar de '||rec_sediu.NrAngajati || ' angajati si
        end if;
    end loop;
END;

```

Script Output x Query Result x

Task completed in 0.035 seconds

```

Departamentul si numarul de angajati:
Sediul BestFitness cu id-ul 10 are un numar de 5 angajati si capacitate de 50 clienti
Sediul ZumbaGym cu id-ul 12 are un numar de 2 angajati si capacitate de 20 clienti
Sediul FitGym cu id-ul 11 are un numar de 3 angajati si capacitate de 30 clienti

```

PL/SQL procedure successfully completed.

4. Să se afișeze abonamentele și pentru fiecare abonament numele clientilor care dețin un astfel de abonament, utilizându-se doi cursori impliciti.

DECLARE

cursor c_abonament is

select tip_abonament, denumire

from ad_abonamente;

cursor c_clienti(p_id_abonament NUMBER) is

select c.id_client, c.prenume, c.num, c.email

from ad_clienti c

where c.tip_abonament = p_id_abonament

order by c.tip_abonament;

BEGIN

DBMS_OUTPUT.PUT_LINE('Tipul de abonament pentru fiecare client:');

FOR rec_abonament in c_abonament loop

```
DBMS_OUTPUT.PUT_LINE('Pentru abonamentul ' || rec_abonament.denumire || ' există  
urmatorii clienti: ');
```

```
for rec_clienti in c_clienti(rec_abonament.tip_abonament) loop
```

```
DBMS_OUTPUT.PUT_LINE('Nume: ' || rec_clienti.numa || ' Prenume: ' ||  
rec_clienti.prenume);
```

```
end loop;
```

```
DBMS_OUTPUT.PUT_LINE('-----');
```

```
end loop;
```

```
END;
```

The screenshot displays the Oracle SQL Developer interface. The top pane, titled 'Query Builder', contains a PL/SQL script. The script declares two cursors: 'c_abonament' which selects 'tip_abonament' and 'denumire' from 'ad_abonamente', and 'c_clienti' which is a function cursor that selects 'c.id_client', 'c.prenume', 'c.numa', and 'c.email' from 'ad_clienti' where 'c.tip_abonament' equals a parameter 'p_id_abonament', ordered by 'c.tip_abonament'. The script then begins a loop over 'c_abonament'. For each record, it prints the subscription type and then enters a loop over 'c_clienti' to print the names of clients associated with that subscription type. After each inner loop, it prints a separator line. The script ends with 'END;'.

The bottom pane, titled 'Script Output', shows the execution results. It confirms the subscription types and lists the clients for each:

- Tipul de abonament pentru fiecare client:
- Pentru abonamentul C există urmatorii clienti:
- Nume: Diana Prenume: Riscov
- Nume: Ioana Prenume: Echim
-
- Pentru abonamentul Z există urmatorii clienti:
- Nume: Catinca Prenume: Rebic
-
- Pentru abonamentul F există urmatorii clienti:
-
- Pentru abonamentul A există urmatorii clienti:
-
- Pentru abonamentul G există urmatorii clienti:
-
- Pentru abonamentul Y există urmatorii clienti:
- Nume: Stefan Prenume: Efimie
-

G. Funcții, proceduri, includerea acestora în pachete (minim 3 funcții, 3 proceduri și 2 pachete)

PROCEDURI

1. Scrieți o procedură care primește în lista de parametrii id-ul unui abonament și un procent și care crește pretul abonamentului primit ca parametru cu procentul primit ca parametru.

tip_abonament = p_tip_abonament:

create or replace procedure modificare_pret_abonament

(p_tip_abonament în ad_abonamente.tip_abonament%type, procent în number) is

v_pret ad_abonamente.pret%type;

begin

select pret into v_pret from ad_abonamente where tip_abonament=p_tip_abonament;

dbms_output.put_line('Abonamentul are pretul de '||v_pret);

update ad_abonamente

set pret = pret+ procent/100*pret

where tip_abonament = p_tip_abonament;

select pret into v_pret from ad_abonamente where tip_abonament=p_tip_abonament;

dbms_output.put_line('Abonamentul are pretul de '||v_pret);

end;

--apel

execute modificare_pret_abonament(21, 25);

The screenshot displays the Oracle SQL Developer environment. The top window, titled 'BazeDeDate.sql', shows a PL/SQL procedure named 'modificare_pret_abonament'. The procedure takes two parameters: 'p_tip_abonament' (a string) and 'procent' (a number). It first selects the current price ('pret') for the specified tip from the 'ad_abonamente' table. It then calculates a new price by adding a percentage increase to the current price. Finally, it updates the 'pret' column in the 'ad_abonamente' table with the new value. The procedure is executed with '22' as the tip and '5' as the percentage.

```
tip_abonament = p_tip_abonament;  
create or replace procedure modificare_pret_abonament  
(p_tip_abonament in ad_abonamente.tip_abonament%type, procent in number) is  
v_pret ad_abonamente.pret%type;  
begin  
select pret into v_pret from ad_abonamente where tip_abonament=p_tip_abonament;  
dbms_output.put_line('Abonamentul are pretul de '||v_pret);  
update ad_abonamente  
set pret = pret+ procent/100*pret  
where tip_abonament = p_tip_abonament;  
select pret into v_pret from ad_abonamente where tip_abonament=p_tip_abonament;  
dbms_output.put_line('Abonamentul are pretul de '||v_pret);  
end;  
--apel  
execute modificare_pret_abonament(22, 5);
```

The bottom window, titled 'Script Output', shows the execution results. It indicates that the procedure was compiled successfully and that the price for tip 22 was updated from 74 to 78. The execution completed in 0.046 seconds.

```
Error starting at line : 291 in command -  
tip_abonament = p_tip_abonament:  
Error report -  
Unknown Command  
  
SP2-0044: For a list of known commands enter HELP  
and to leave enter EXIT.  
  
Procedure MODIFICARE_PRET_ABONAMENT compiled  
  
Abonamentul are pretul de 74  
Abonamentul are pretul de 78  
  
PL/SQL procedure successfully completed.
```

2. Scrieți o procedură care afișează numele clientului și abonamentul pe care acesta îl are în funcție de id-ul clientului primit ca parametru

```
create or replace procedure afisare_date_client
```

```
(p_id în ad_clienti.id_client%type, p_nume out ad_clienti.nume%type, p_denumire out ad_abonamente.denumire%type)
```

```
is
```

```
begin
```

```
select nume, denumire into p_nume, p_denumire
```

```
from ad_clienti c, ad_abonamente a
```

```
where c.tip_abonament = a.tip_abonament;
```

```
end;
```

```
--APEL, NU MERGE
```

```
declare v_nume ad_clienti.nume%type;
```

```
v_denumire ad_abonamente.denumire%type;
```

```
begin
```

```
afisare_date_client(102, v_nume, v_denumire);
```

```
dbms_output.put_line('Clientul cu numele '||v_nume||' are abonamentul'||v_denumire);
```

```
end;
```

3. Sa se creeze o procedură prin care sa se modifice bonusul primit de către un angajat al cărui id este primit ca parametru de intrare cu o valoare primită ca parametru. Să se afișeze noua valoare a bonusului și valoarea salariului înainte și după modificare.

```
id_angajat=p_id_angajat:
```

```
CREATE OR REPLACE PROCEDURE modificare_bonus
```

```
(p_id_angajat în ad_angajati.id_angajat%type, punct în number) IS
```

```
v_bonus ad_angajati.bonus%type;
```

```
v_salariu ad_angajati.salariu%type;
```

```
BEGIN
```

```
Select bonus, salariu into v_bonus, v_salariu from ad_angajati where id_angajat=p_id_angajat;
```

```
dbms_output.put_line('Angajatul are bonus de 0'||v_bonus || ' și salariul de ' ||
v_bonus*v_salariu);
```

```
Update ad_angajati
```

```
Set bonus=bonus+punct
```

```
Where id_angajat=p_id_angajat;
```

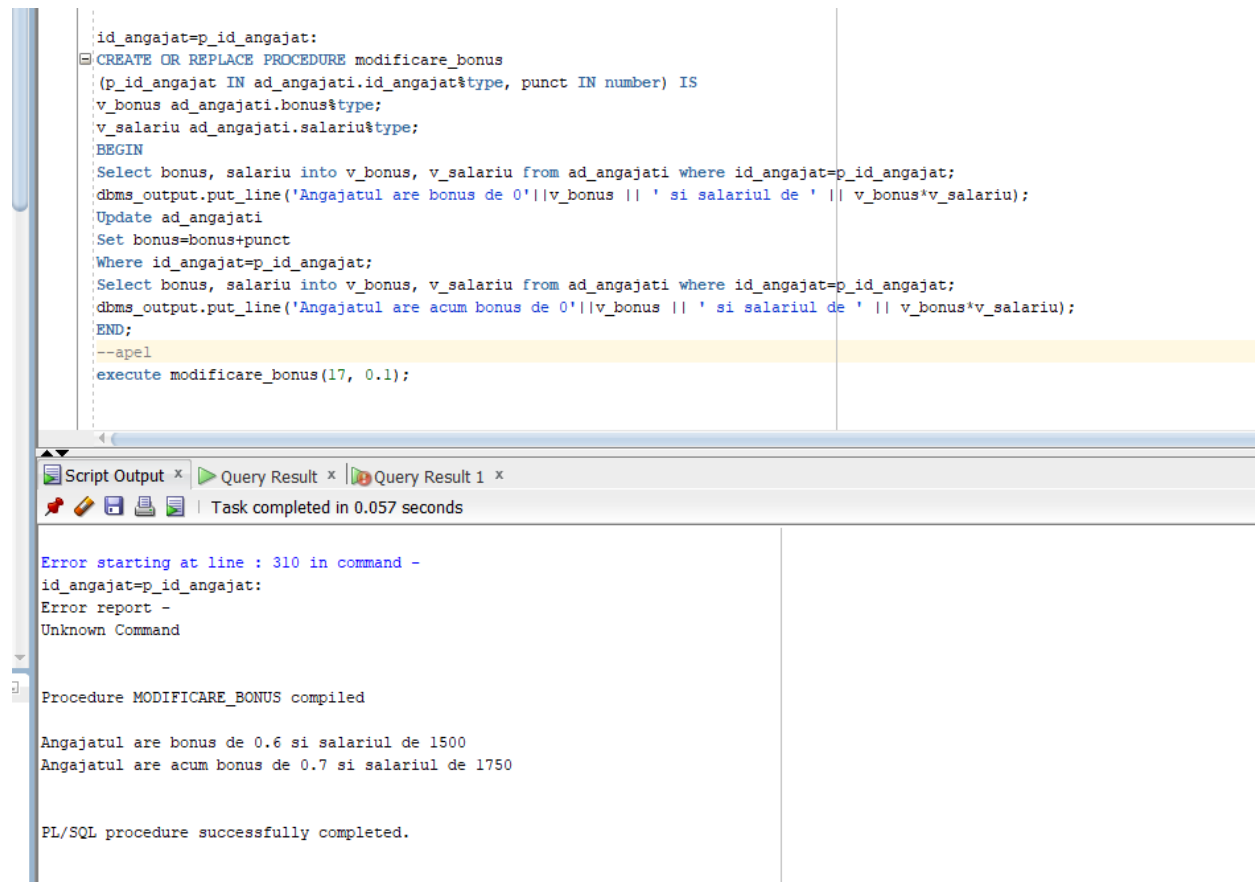
```
Select bonus, salariu into v_bonus, v_salariu from ad_angajati where id_angajat=p_id_angajat;
```

```
dbms_output.put_line('Angajatul are acum bonus de 0'||v_bonus || ' și salariul de ' ||
v_bonus*v_salariu);
```

```
END;
```

```
--apel
```

```
execute modificare_bonus(17, 0.1);
```



```

id_angajat=p_id_angajat:
CREATE OR REPLACE PROCEDURE modificare_bonus
(p_id_angajat IN ad_angajati.id_angajat%type, punct IN number) IS
v_bonus ad_angajati.bonus%type;
v_salariu ad_angajati.salariu%type;
BEGIN
Select bonus, salariu into v_bonus, v_salariu from ad_angajati where id_angajat=p_id_angajat;
dbms_output.put_line('Angajatul are bonus de 0'||v_bonus || ' si salariul de ' || v_bonus*v_salariu);
Update ad_angajati
Set bonus=bonus+punct
Where id_angajat=p_id_angajat;
Select bonus, salariu into v_bonus, v_salariu from ad_angajati where id_angajat=p_id_angajat;
dbms_output.put_line('Angajatul are acum bonus de 0'||v_bonus || ' si salariul de ' || v_bonus*v_salariu);
END;
--apel
execute modificare_bonus(17, 0.1);

```

Script Output x Query Result x Query Result 1 x

Task completed in 0.057 seconds

```

Error starting at line : 310 in command -
id_angajat=p_id_angajat:
Error report -
Unknown Command

Procedure MODIFICARE_BONUS compiled

Angajatul are bonus de 0.6 si salariul de 1500
Angajatul are acum bonus de 0.7 si salariul de 1750

PL/SQL procedure successfully completed.

```

FUNCTII

4. Să se scrie o funcție care returnează valoarea salariului net. Aceasta primește ca parametrii valoarea salariului brut și valoarea procentuala a impozitului.

create or replace function salarii_nete(salariu în number, impozit în number) return number is

begin

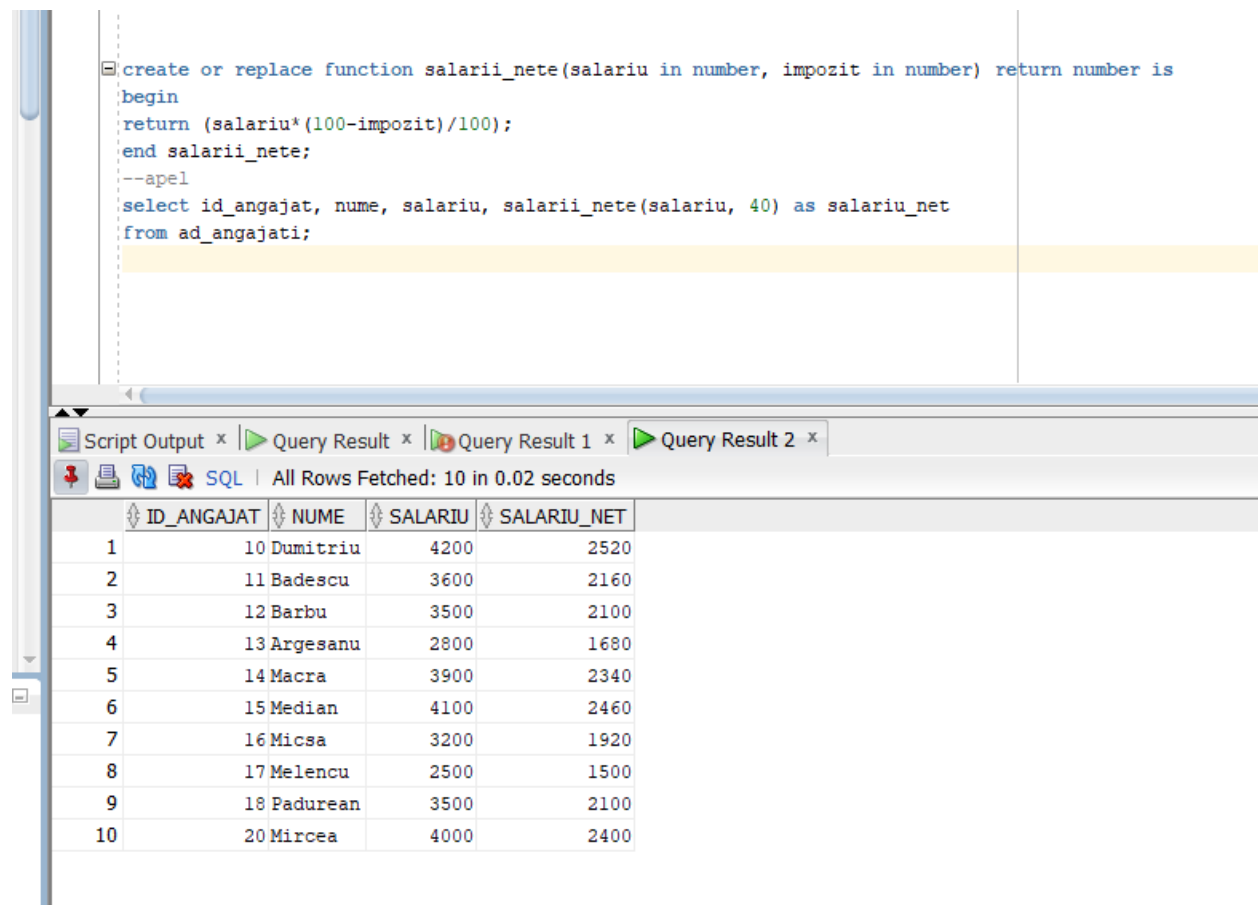
return (salariu*(100-impozit)/100);

end salarii_nete;

--apel

select id_angajat, nume, salariu, salarii_nete(salariu, 40) as salariu_net

from ad_angajati;



The screenshot displays a SQL IDE interface. The top pane shows the following SQL code:

```
create or replace function salarii_nete(salariu in number, impozit in number) return number is
begin
return (salariu*(100-impozit)/100);
end salarii_nete;
--apel
select id_angajat, nume, salariu, salarii_nete(salariu, 40) as salariu_net
from ad_angajati;
```

The bottom pane shows the 'Query Result' tab with the following data:

| ID_ANGAJAT | NUME | SALARIU | SALARIU_NET |
|------------|-------------|---------|-------------|
| 1 | 10 Dumitriu | 4200 | 2520 |
| 2 | 11 Badescu | 3600 | 2160 |
| 3 | 12 Barbu | 3500 | 2100 |
| 4 | 13 Argesanu | 2800 | 1680 |
| 5 | 14 Macra | 3900 | 2340 |
| 6 | 15 Median | 4100 | 2460 |
| 7 | 16 Micsa | 3200 | 1920 |
| 8 | 17 Melencu | 2500 | 1500 |
| 9 | 18 Padurean | 3500 | 2100 |
| 10 | 20 Mircea | 4000 | 2400 |

5. Să se creeze o funcție care returnează nivelul de dificultate pentru un tip de antrenament în felul urmator: 1-2 usor, 3-4 mediu, 5 dificil.

create or replace function nivel_antrenament(id_tip number) return varchar2

```
is
v_nivel ad_tip_antrenamente.nivel%type;
begin
select nivel into v_nivel from ad_tip_antrenamente where id_tip_antrenament=id_tip;
if v_nivel=1 or v_nivel=2 then return 'Nivel usor - necesita o incalzire usoara, nu necesita
ajutor/supraveghere atenta';
elsif v_nivel=3 or v_nivel=4 then return 'Nivel mediu - necesita incalzire și supraveghere din
partea antrenorului';
else return 'Nivel ridicat - necesita incalzire corespunzatoare și ajutor din partea antrenorului';
end if;
exception
when no_data_found then return 'Nu există antrenamentul cu acest id';
end;
--apel
declare
nivel varchar2(250);
begin
nivel:=nivel_antrenament(1);
dbms_output.put_line('Antrenamentul are ' || nivel);
end;
```

The screenshot shows the Oracle SQL Developer interface. The top pane is the 'Query Builder' tab, displaying a PL/SQL function named `nivel_antrenament`. The function takes `id_tip` as a parameter and returns a `varchar2`. It uses a `select` statement to retrieve the level from the `ad_tip_antrenamente` table. The function logic is as follows:

```

create or replace function nivel_antrenament(id_tip number) return varchar2
is
v_nivel ad_tip_antrenamente.nivel%type;
begin
select nivel into v_nivel from ad_tip_antrenamente where id_tip_antrenament=id_tip;
if v_nivel=1 or v_nivel=2 then return 'Nivel usor - necesita o incalzire usoara, nu necesita ajutor/supraveghere atenta';
elsif v_nivel=3 or v_nivel=4 then return 'Nivel mediu - necesita incalzire si supraveghere din partea antrenorului';
else return 'Nivel ridicat - necesita incalzire corespunzatoare si ajutor din partea antrenorului';
end if;
exception
when no_data_found then return 'Nu exista antrenamentul cu acest id';
end;
--apel
declare
nivel varchar2(250);
begin
nivel:=nivel_antrenament(1);
dbms_output.put_line('Antrenamentul are ' || nivel);
end;

```

The bottom pane shows the 'Script Output' tab, which displays the following messages:

```

Function NIVEL_ANTRENAMENT compiled
Antrenamentul are Nivel mediu - necesita incalzire si supraveghere din partea antrenorului
PL/SQL procedure successfully completed.

```

6. Să se scrie o funcție care primește ca parametru de intrare id-ul unei ședințe și returnează true (1) pentru acele sedințe care durează mai mult decât media duratelor tuturor ședințelor și false (0) altfel.

create or replace function f_durata

(p_id_sedinte în ad_sedinte.durata%type) return number

is

v_durata_total ad_sedinte.durata%type;

v_durata ad_sedinte.durata%type;

begin

select sum(durata)/count(id_sedinta) into v_durata_total from ad_sedinte;

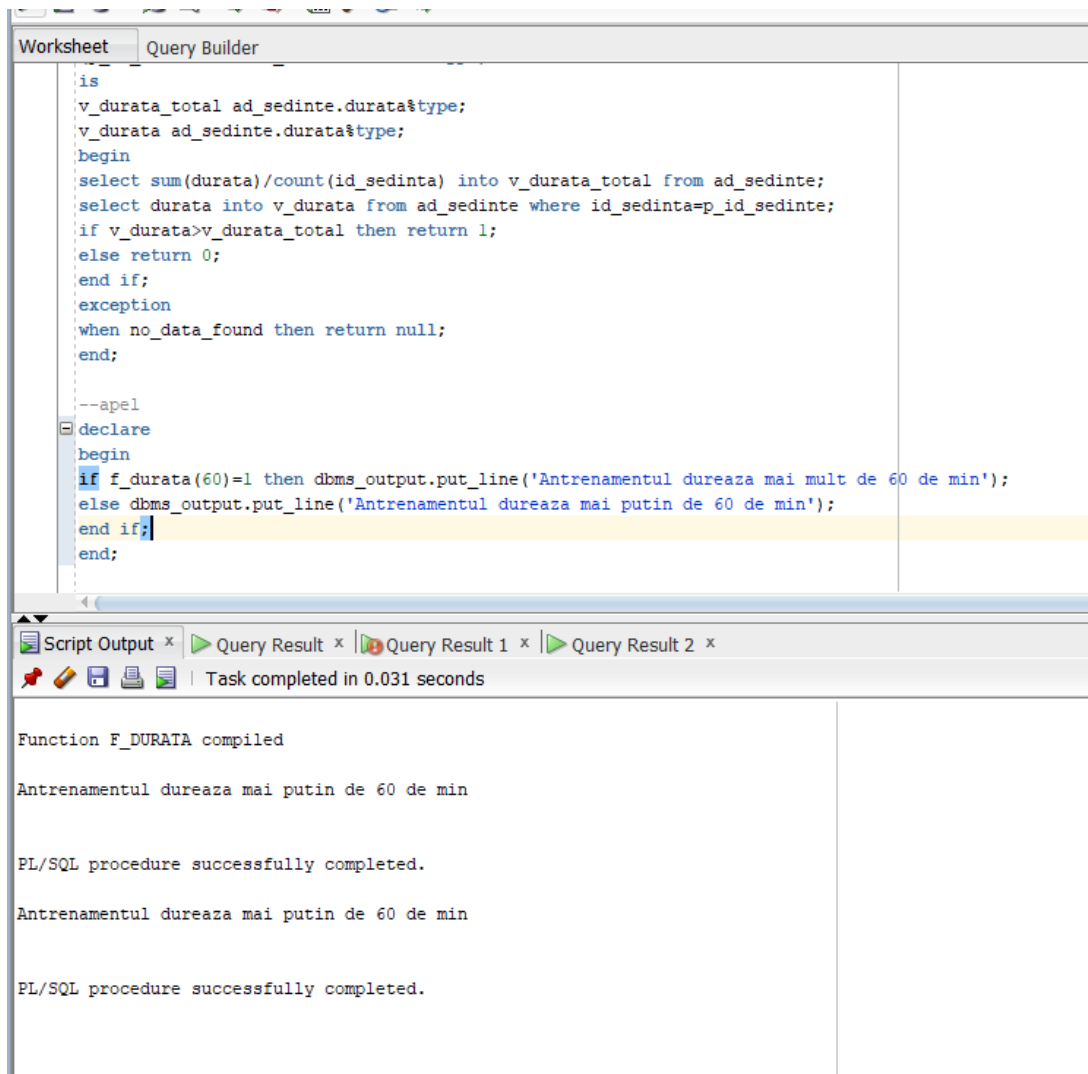
select durata into v_durata from ad_sedinte where id_sedinta=p_id_sedinte;

if v_durata>v_durata_total then return 1;

else return 0;

end if;

```
exception
when no_data_found then return null;
end;
--apel
declare
begin
if f_durata(70)=1 then dbms_output.put_line('Antrenamentul dureaza mai mult de 60 de min');
else dbms_output.put_line('Antrenamentul dureaza mai putin de 60 de min');
end if;
end;
```



The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Query Builder', contains a PL/SQL script. The script defines a function `f_durata` that calculates the average duration of training sessions and compares it to a specified value. It includes an exception handler for `no_data_found`. Below the function definition, there is a call to `f_durata(70)` and a conditional statement that prints the result using `dbms_output.put_line`. The bottom pane, titled 'Script Output', shows the execution results. It indicates that the function `F_DURATA` was compiled successfully. The output shows the message 'Antrenamentul dureaza mai putin de 60 de min' being printed twice, followed by 'PL/SQL procedure successfully completed.' twice. A status bar at the bottom of the output pane indicates 'Task completed in 0.031 seconds'.

```
is
v_durata_total ad_sedinte.durata%type;
v_durata ad_sedinte.durata%type;
begin
select sum(durata)/count(id_sedinta) into v_durata_total from ad_sedinte;
select durata into v_durata from ad_sedinte where id_sedinta=p_id_sedinte;
if v_durata>v_durata_total then return 1;
else return 0;
end if;
exception
when no_data_found then return null;
end;

--apel
declare
begin
if f_durata(70)=1 then dbms_output.put_line('Antrenamentul dureaza mai mult de 60 de min');
else dbms_output.put_line('Antrenamentul dureaza mai putin de 60 de min');
end if;
end;
```

Script Output x Query Result x Query Result 1 x Query Result 2 x

Task completed in 0.031 seconds

Function F_DURATA compiled

Antrenamentul dureaza mai putin de 60 de min

PL/SQL procedure successfully completed.

Antrenamentul dureaza mai putin de 60 de min

PL/SQL procedure successfully completed.

PACHETE

PACHET 1: CLIENT

Să se creeze un pachet care să conțină funcții și proceduri pentru prelucrarea datelor din tabela ad_clinti.

create or replace package actualizare_client IS

procedure adauga_client

```
(p_id_client ad_clinti.id_client%type,  
p_nume ad_clinti.nume%type,  
p_prename ad_clinti.prename%type,  
p_telefon ad_clinti.telefon%type,  
p_data_nasterii ad_clinti.data_nasterii%type,  
p_email ad_clinti.email%type,  
p_incepere_abonament ad_clinti.incepere_abonament%type,  
p_tip_abonament ad_clinti.tip_abonament%type,  
p_expirare_abonament ad_clinti.expirare_abonament%type);
```

procedure modifica_client

```
(p_id_client ad_clinti.id_client%type,  
p_nume ad_clinti.nume%type,  
p_prename ad_clinti.prename%type,  
p_telefon ad_clinti.telefon%type,  
p_data_nasterii ad_clinti.data_nasterii%type,  
p_email ad_clinti.email%type,  
p_incepere_abonament ad_clinti.incepere_abonament%type,  
p_tip_abonament ad_clinti.tip_abonament%type,  
p_expirare_abonament ad_clinti.expirare_abonament%type);
```

```
procedure modifica_client
```

```
(p_id_client ad_clienti.id_client%type,  
p_incepere_abonament ad_clienti.incepere_abonament%type,  
p_tip_abonament ad_clienti.tip_abonament%type,  
p_expirare_abonament ad_clienti.expirare_abonament%type);
```

```
procedure sterge_client
```

```
(p_id_client ad_clienti.id_client%type);
```

```
function există_client
```

```
(p_id_client ad_clienti.id_client%type) return boolean;
```

```
function abonament_valid
```

```
(p_id_client ad_clienti.id_client%type) return boolean;
```

```
exceptie exception;
```

```
END;
```

```
/
```

```
create or replace package body actualizare_client is
```

```
procedure adauga_client
```

```
(p_id_client ad_clienti.id_client%type,  
p_nume ad_clienti.nume%type,  
p_prenume ad_clienti.prenume%type,  
p_telefon ad_clienti.telefon%type,  
p_data_nasterii ad_clienti.data_nasterii%type,  
p_email ad_clienti.email%type,  
p_incepere_abonament ad_clienti.incepere_abonament%type,
```

```
p_tip_abonament ad_clienti.tip_abonament%type,  
p_expirare_abonament ad_clienti.expirare_abonament%type)  
is  
begin  
if există_client(p_id_client) then raise exceptie;  
else insert into ad_clienti values (p_id_client, p_nume, p_prenume, p_telefon, p_data_nasterii,  
    p_email, p_incepere_abonament, p_tip_abonament, p_expirare_abonament);  
end if;  
exception  
when exceptie then dbms_output.put_line('Există deja un client cu acest id');  
end; --20
```

```
procedure modifica_client
```

```
(p_id_client ad_clienti.id_client%type,  
p_nume ad_clienti.nume%type,  
p_prenume ad_clienti.prenume%type,  
p_telefon ad_clienti.telefon%type,  
p_data_nasterii ad_clienti.data_nasterii%type,  
p_email ad_clienti.email%type,  
p_incepere_abonament ad_clienti.incepere_abonament%type,  
p_tip_abonament ad_clienti.tip_abonament%type,  
p_expirare_abonament ad_clienti.expirare_abonament%type)  
is  
begin  
if există_client(p_id_client) then  
update ad_clienti set nume=p_nume, prenume =p_prenume, telefon=p_telefon,  
data_nasterii=p_data_nasterii, email=p_email, incepere_abonament = p_incepere_abonament,  
tip_abonament=p_tip_abonament, expirare_abonament=p_expirare_abonament
```

```
where id_client=p_id_client;
else raise exceptie;
end if;--40
exception
when exceptie then dbms_output.put_line('Nu există clientul cu acest id');
end;

procedure modifica_client
    (p_id_client ad_clienti.id_client%type,
    p_incepere_abonament ad_clienti.incepere_abonament%type,
    p_tip_abonament ad_clienti.tip_abonament%type,
    p_expirare_abonament ad_clienti.expirare_abonament%type)
is
begin --50
if există_client(p_id_client) then
update ad_clienti set incepere_abonament = p_incepere_abonament,
tip_abonament=p_tip_abonament, expirare_abonament=p_expirare_abonament
where id_client = p_id_client;
else raise exceptie;
end if;
exception
when exceptie then dbms_output.put_line('Nu există clientul cu acest id');
end;

procedure sterge_client
    (p_id_client ad_clienti.id_client%type)
is
begin
```

```
if există_client(p_id_client) then
delete from ad_clienti where id_client = p_id_client;
dbms_output.put_line('Clientul cu id '|| p_id_client || ' a fost sters');
else raise exceptie;
end if;
exception
when exceptie then dbms_output.put_line('Nu există clientul cu acest id');
end;
```

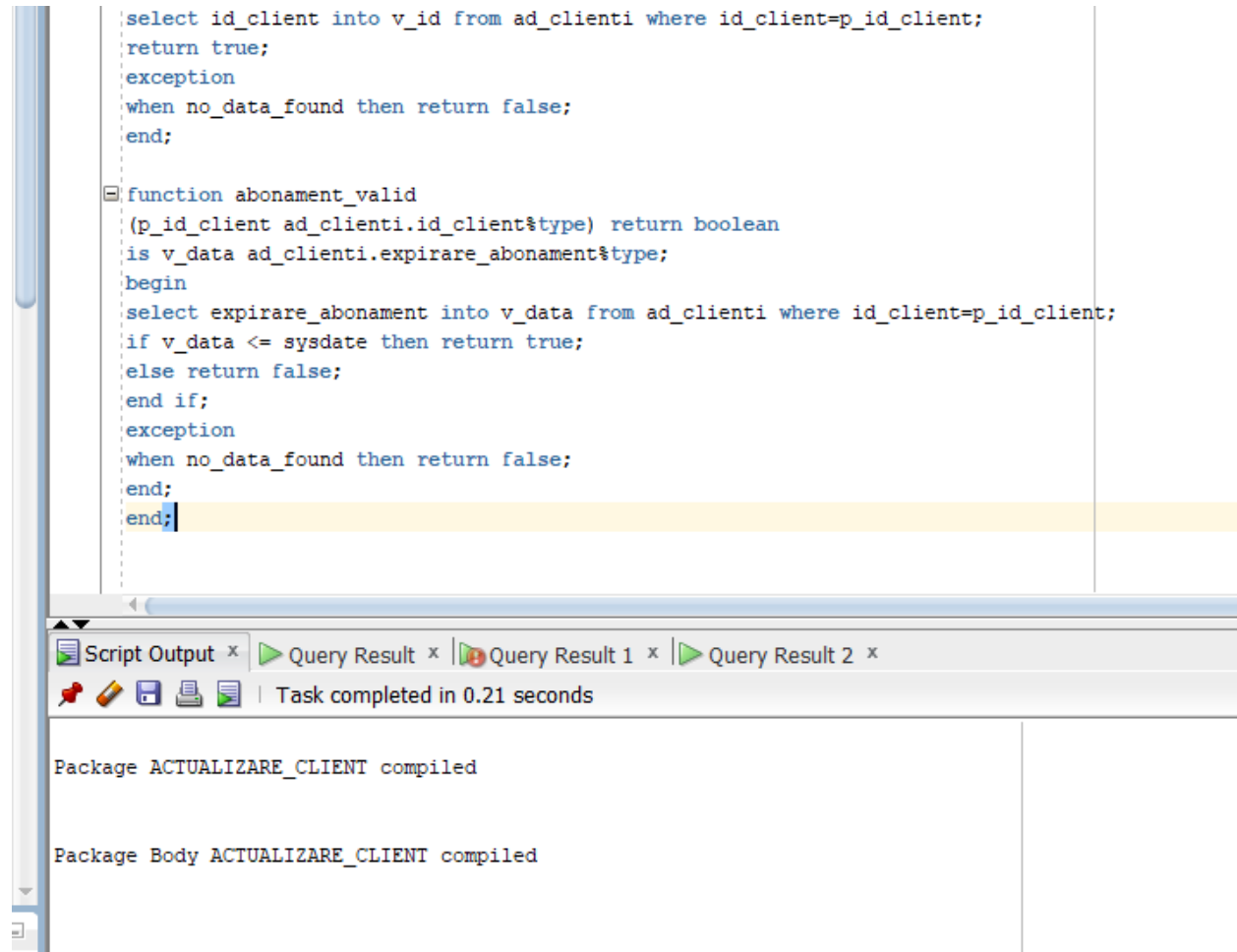
```
function există_client
(p_id_client ad_clienti.id_client%type) return boolean
is v_id number;
begin
select id_client into v_id from ad_clienti where id_client=p_id_client;
return true;
exception
when no_data_found then return false;
end;
```

```
function abonament_valid
(p_id_client ad_clienti.id_client%type) return boolean
is v_data ad_clienti.expirare_abonament%type;
begin
select expirare_abonament into v_data from ad_clienti where id_client=p_id_client;
if v_data <= sysdate then return true;
else return false;
end if;
exception
```

when no_data_found then return false;

end;

end;



The screenshot displays the Oracle SQL Developer environment. The main editor window contains a PL/SQL script with the following code:

```
select id_client into v_id from ad_clienti where id_client=p_id_client;
return true;
exception
when no_data_found then return false;
end;

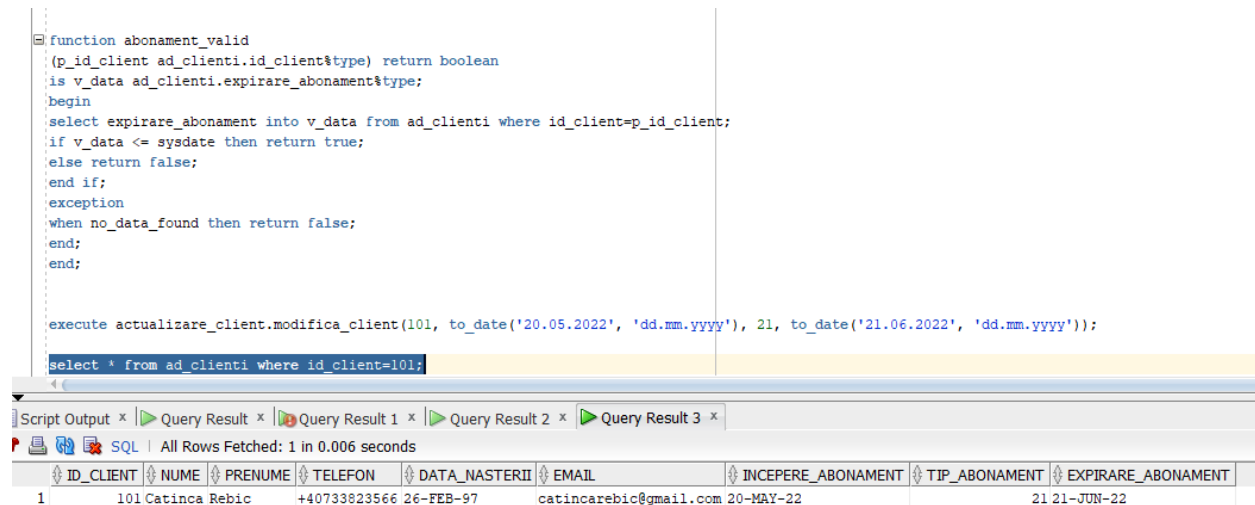
function abonament_valid
(p_id_client ad_clienti.id_client%type) return boolean
is v_data ad_clienti.expirare_abonament%type;
begin
select expirare_abonament into v_data from ad_clienti where id_client=p_id_client;
if v_data <= sysdate then return true;
else return false;
end if;
exception
when no_data_found then return false;
end;
end;
```

Below the editor, the 'Script Output' tab is active, showing the execution results:

Task completed in 0.21 seconds

Package ACTUALIZARE_CLIENT compiled

Package Body ACTUALIZARE_CLIENT compiled



PACHET 2: ABONAMENTE

Să se creeze un pachet care să conțină funcții și proceduri pentru prelucrarea datelor din tabela `ad_abonamente`.

create or replace package `actualizare_abonamente` IS

procedure `adauga_abonament`

```

(p_tip_abonament ad_abonamente.tip_abonament%type,
p_denumire ad_abonamente.denumire%type,
p_pret ad_abonamente.pret%type,
p_tip_antrenament ad_abonamente.id_tip_antrenament%type);

```

procedure `modifica_abonament`

```

(p_tip_abonament ad_abonamente.tip_abonament%type,
p_denumire ad_abonamente.denumire%type,
p_pret ad_abonamente.pret%type,
p_tip_antrenament ad_abonamente.id_tip_antrenament%type);

```

```
procedure modifica_abonament
```

```
(p_tip_abonament ad_abonamente.tip_abonament%type,  
p_pret ad_abonamente.pret%type);
```

```
procedure sterge_abonament
```

```
(p_tip_abonament ad_abonamente.tip_abonament%type);
```

```
function există_abonament
```

```
(p_tip_abonament ad_abonamente.tip_abonament%type) return boolean;
```

```
exceptie exception;
```

```
END;
```

```
/
```

```
create or replace package body actualizare_abonamente is
```

```
procedure adauga_abonament
```

```
(p_tip_abonament ad_abonamente.tip_abonament%type,  
p_denumire ad_abonamente.denumire%type,  
p_pret ad_abonamente.pret%type,  
p_tip_antrenament ad_abonamente.id_tip_antrenament%type)
```

```
is
```

```
begin
```

```
if există_abonament(p_tip_abonament) then raise exceptie;
```

```
else
```

```
insert into ad_abonamente values (p_tip_abonament, p_denumire, p_pret, p_tip_antrenament);
```

```
end if;
```

```
exception
```



```
when exceptie then dbms_output.put_line('Există deja abonamentul cu acest id');  
end;--15
```

```
procedure modifica_abonament
```

```
    (p_tip_abonament ad_abonamente.tip_abonament%type,  
     p_denumire ad_abonamente.denumire%type,  
     p_pret ad_abonamente.pret%type,  
     p_tip_antrenament ad_abonamente.id_tip_antrenament%type)
```

```
is
```

```
begin
```

```
if există_abonament(p_tip_abonament) then
```

```
update ad_abonamente set tip_abonament = p_tip_abonament, denumire=p_denumire,  
pret=p_pret, id_tip_antrenament = p_tip_antrenament
```

```
where id_tip_antrenament = p_tip_antrenament;
```

```
else raise exceptie;
```

```
end if;
```

```
exception--28
```

```
when exceptie then dbms_output.put_line('Nu există abonamentul cu acest id');
```

```
end;
```

```
procedure modifica_abonament
```

```
    (p_tip_abonament ad_abonamente.tip_abonament%type,  
     p_pret ad_abonamente.pret%type)
```

```
is
```

```
begin
```

```
if există_abonament(p_tip_abonament) then
```

```
update ad_abonamente set pret=p_pret
```

```
where tip_abonament = p_tip_abonament;
```

```
else raise exceptie;--40
```

```
end if;
exception
when exceptie then dbms_output.put_line('Nu există abonamentul cu acest id');
end;
```

```
procedure sterge_abonament
    (p_tip_abonament ad_abonamente.tip_abonament%type)
is
begin
if există_abonament(p_tip_abonament)then
delete from ad_abonamente where p_tip_abonament=tip_abonament;
else raise exceptie;
end if;
exception
when exceptie then dbms_output.put_line('Nu există abonamentul cu acest id');
end;
```

```
function există_abonament
    (p_tip_abonament ad_abonamente.tip_abonament%type) return boolean
is
v_tip number;
begin
select tip_abonament into v_tip from ad_abonamente where p_tip_abonament =tip_abonament;
return true;
exception
when no_data_found then return false;
end;
end;
```

Worksheet Query Builder

```

if exista_abonament(p_tip_abonament) then
delete from ad_abonamente where p_tip_abonament=tip_abonament;
else raise exception;
end if;
exception
when exception then dbms_output.put_line('Nu exista abonamentul cu acest id');
end;

function exista_abonament
(p_tip_abonament ad_abonamente.tip_abonament%type) return boolean
is
v_tip number;
begin
select tip_abonament into v_tip from ad_abonamente where p_tip_abonament =tip_abonament;
return true;
exception
when no_data_found then return false;
end;
end;

```

Script Output x Query Result x Query Result 1 x Query Result 2 x Query Result 3 x

Task completed in 0.109 seconds

Package ACTUALIZARE_ABONAMENTE compiled

Package Body ACTUALIZARE_ABONAMENTE compiled

```

end;

execute ACTUALIZARE_ABONAMENTE.modifica_abonament(21, 150);
select * from ad_abonamente where tip_abonament =21;

```

Script Output x Query Result x Query Result 1 x Query Result 2 x Query Result 3 x

SQL | All Rows Fetched: 1 in 0.012 seconds

| | TIP_ABONAMENT | DENUMIRE | PRET | ID_TIP_ANTRENAMENT |
|---|---------------|----------|------|--------------------|
| 1 | 21 C | | 150 | 1 |

H. Declanșatori (minim 3)

1. Să se creeze un trigger care să nu permită valori mai mici de 30 de lei sau mai mari de 300 de lei pentru prețul unui abonament.

```
create or replace trigger trigger_pret
```

```
before insert or update on ad_abonamente for each row
```

```
begin
```

```
if :new.pret < 30 then raise_application_error(-20101, 'Pretul pe care doriti sa il setati pentru  
acest abonament este mai mic de 30 de lei');
```

```
elsif :new.pret > 300 then raise_application_error(-20102, 'Pretul pe care doriti sa il setati pentru  
acest abonament este mai mare de 300 de lei');
```

```
end if;
```

```
end;
```

```
--folosire
```

```
update ad_abonamente set pret = 20 where tip_abonament=21;
```

```

create or replace trigger trigger_pret
before insert or update on ad_abonamente for each row
begin
  if :new.pret < 30 then raise_application_error(-20101, 'Pretul pe care doriti sa il setati pentru acest abonament este mai mic de 30 de lei');
  elsif :new.pret > 300 then raise_application_error(-20102, 'Pretul pe care doriti sa il setati pentru acest abonament este mai mare de 300 de lei');
  end if;
end;

--folosire
update ad_abonamente set pret = 20 where tip_abonament=21;

```

Script Output x | Query Result x | Query Result 1 x | Query Result 2 x | Query Result 3 x

Task completed in 0.07 seconds

Trigger TRIGGER_PRET compiled

Error starting at line : 119 in command -
update ad_abonamente set pret = 20 where tip_abonament=21

Error report -
ORA-20101: Pretul pe care doriti sa il setati pentru acest abonament este mai mic de 30 de lei
ORA-06512: at "DUMITRIUA_51.TRIGGER_PRET", line 2
ORA-04088: error during execution of trigger 'DUMITRIUA_51.TRIGGER_PRET'

2. Să se creeze o tabelă în care să se rețină operația (insert, update, delete) pentru tabela ad_sedinte, cine a efectuat-o și data efectuării care sunt asociate unui trigger care a fost declansat.

```
create table triggeri_folositi
```

```
(denumire_instructiune varchar2(15),
```

```
utilizator varchar2(25),
data date default sysdate);

create or replace trigger t
before insert or update or delete on ad_sedinte
declare
v_instructiune triggeri_folositi.denumire_instructiune%type;
begin
case
    when INSERTING then v_instructiune:='Inserare';
    when UPDATING then v_instructiune:='Update';
    else v_instructiune:='Stergere';
end case;
insert into triggeri_folositi values(v_instructiune, user, sysdate);
end;

--folosire
delete from ad_sedinte where id_sedinta=77;
select * from triggeri_folositi;
```

Worksheet | Query Builder

```

(denumire_instructiune varchar2(15),
utilizator varchar2(25),
data date default sysdate);

create or replace trigger t
before insert or update or delete on ad_sedinte
declare
v_instructiune triggeri_folositi.denumire_instructiune%type;
begin
case
when INSERTING then v_instructiune:='Inserare';
when UPDATING then v_instructiune:='Update';
else v_instructiune:='Stergere';
end case;
insert into triggeri_folositi values(v_instructiune, user, sysdate);
end;
--fol
delete from ad_sedinte where id_sedinta=77;
select * from triggeri_folositi;

```

Script Output x | Query Result x | Query Result 1 x | Query Result 2 x | Query Result 3 x

SQL | All Rows Fetched: 2 in 0.005 seconds

| | DENUMIRE_INSTRUCTIUNE | UTILIZATOR | DATA |
|---|-----------------------|--------------|-----------|
| 1 | Stergere | DUMITRIUA_51 | 28-MAY-22 |
| 2 | Stergere | DUMITRIUA_51 | 20-MAY-22 |

5. Sa se creeze un trigger pentru tabela ad_sedii care să nu permită introducerea unei denumiri pentru sediu care sa aibă mai puțin de 3 litere sau 3, să nu permită setarea unei capacități mai mari de 100 de oameni sau mai mici de 10 oameni și sa nu mermita setarea pentru data de deschidere mai veche de azi pentru un sediu.

create or replace trigger trigger_sediu

before insert or update on ad_sedii for each row

begin

if length(:new.denumire)<=3 then raise_application_error(-20106, 'Nu se poate seta o denumire care are trei sau mai puțin de trei litere');

```
end if;

if :new.capacitate < 10 then raise_application_error(-20103, 'Capacitatea pe care doriti sa o setati
pentru acest abonament este mai mic de 10 oameni');

elsif :new.capacitate > 100 then raise_application_error(-20104, 'Capacitatea pe care doriti sa o
setati pentru acest abonament este mai mare de 100 de oameni');

end if;

if :new.data_deschidere < sysdate then raise_application_error(-20105, 'Nu se poate seta o data de
deschidere inainte de ziua curenta');

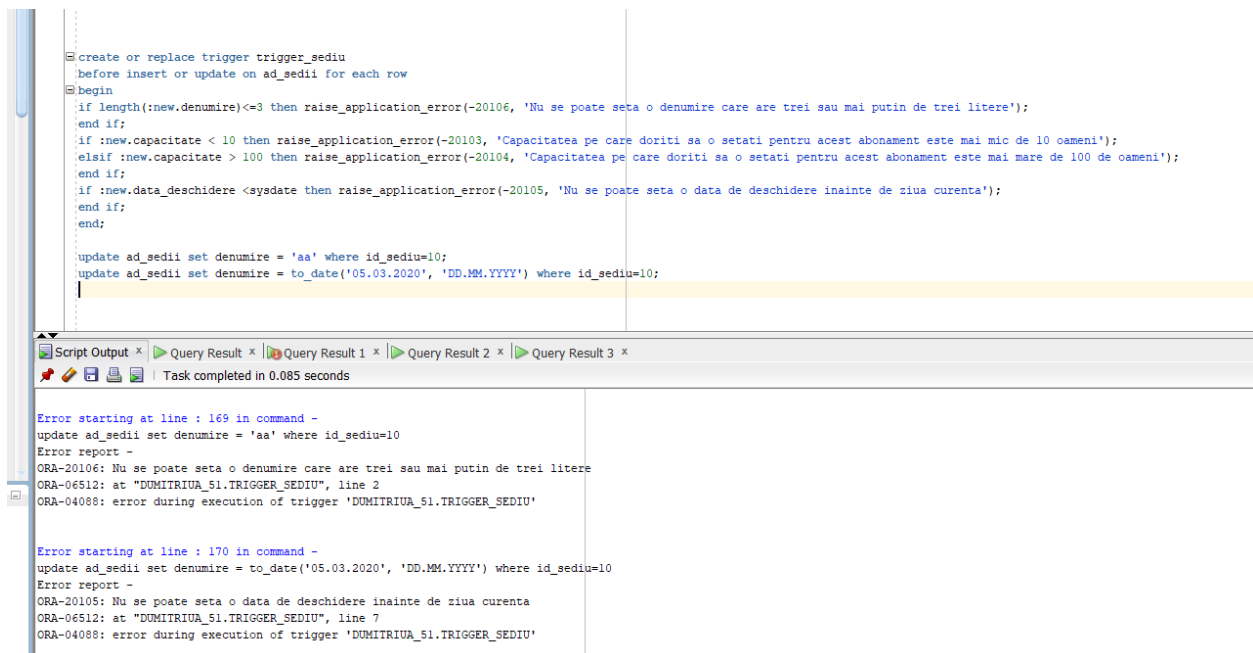
end if;

end;

--folosire

update ad_sedii set denumire = 'aa' where id_sediu=10;

update ad_sedii set denumire = to_date('05.03.2020', 'DD.MM.YYYY') where id_sediu=10;
```



```
create or replace trigger trigger_sediu
before insert or update on ad_sedii for each row
begin
if length(:new.denumire)<=3 then raise_application_error(-20106, 'Nu se poate seta o denumire care are trei sau mai putin de trei litere');
end if;
if :new.capacitate < 10 then raise_application_error(-20103, 'Capacitatea pe care doriti sa o setati pentru acest abonament este mai mic de 10 oameni');
elsif :new.capacitate > 100 then raise_application_error(-20104, 'Capacitatea pe care doriti sa o setati pentru acest abonament este mai mare de 100 de oameni');
end if;
if :new.data_deschidere < sysdate then raise_application_error(-20105, 'Nu se poate seta o data de deschidere inainte de ziua curenta');
end if;
end;

update ad_sedii set denumire = 'aa' where id_sediu=10;
update ad_sedii set denumire = to_date('05.03.2020', 'DD.MM.YYYY') where id_sediu=10;
```

Script Output x Query Result x Query Result 1 x Query Result 2 x Query Result 3 x

Task completed in 0.085 seconds

Error starting at line : 169 in command -
update ad_sedii set denumire = 'aa' where id_sediu=10
Error report -
ORA-20106: Nu se poate seta o denumire care are trei sau mai putin de trei litere
ORA-06512: at "DUMITRIUA_51.TRIGGER_SEDIU", line 2
ORA-04088: error during execution of trigger 'DUMITRIUA_51.TRIGGER_SEDIU'

Error starting at line : 170 in command -
update ad_sedii set denumire = to_date('05.03.2020', 'DD.MM.YYYY') where id_sediu=10
Error report -
ORA-20105: Nu se poate seta o data de deschidere inainte de ziua curenta
ORA-06512: at "DUMITRIUA_51.TRIGGER_SEDIU", line 7
ORA-04088: error during execution of trigger 'DUMITRIUA_51.TRIGGER_SEDIU'