

TEMA LABORATOR 10  
(Laborator PLSQL 5)

1. Definiți un pachet care să permită gestiunea angajaților companiei. Pachetul va conține:

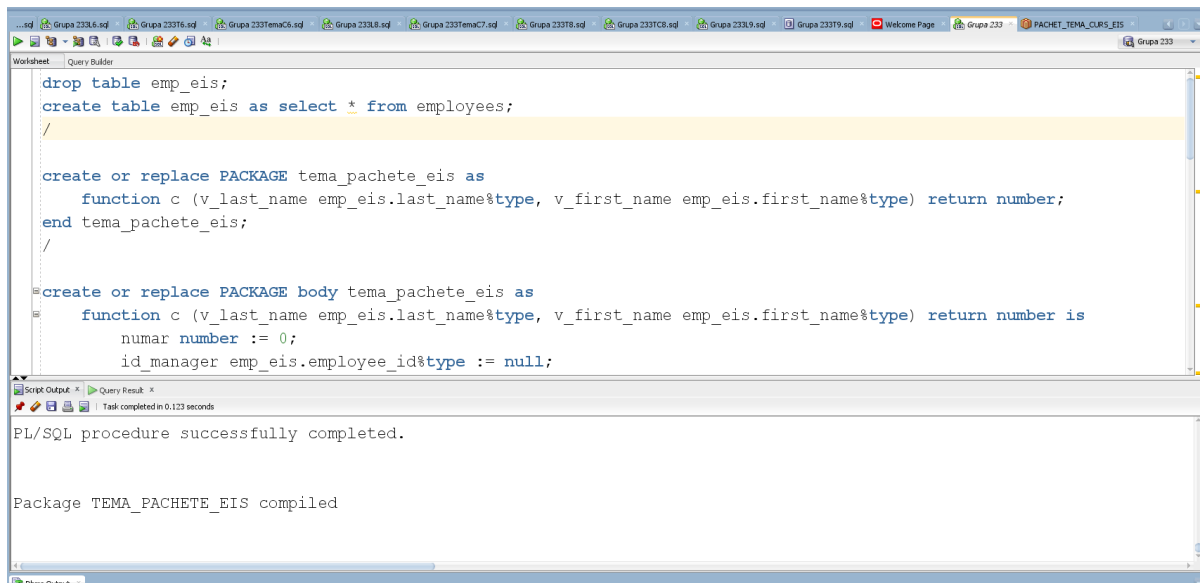
c. o funcție care întoarce numărul de subalterni direcți sau indirecti ai unui angajat al cărui nume și prenume sunt date ca parametrii;

Observație: Tratați toate excepțiile.

Am rezolvat doar pentru subalternii direcți.

```
drop table emp_eis;  
create table emp_eis as select * from employees;  
/
```

```
create or replace PACKAGE tema_pachete_eis as  
    function c (v_last_name emp_eis.last_name%type, v_first_name emp_eis.first_name%type) return  
    number;  
end tema_pachete_eis;  
/
```



```
create or replace PACKAGE body tema_pachete_eis as  
    function c (v_last_name emp_eis.last_name%type, v_first_name emp_eis.first_name%type) return  
    number is  
        numar number := 0;  
        id_manager emp_eis.employee_id%type := null;  
    begin  
        select employee_id into id_manager
```

```

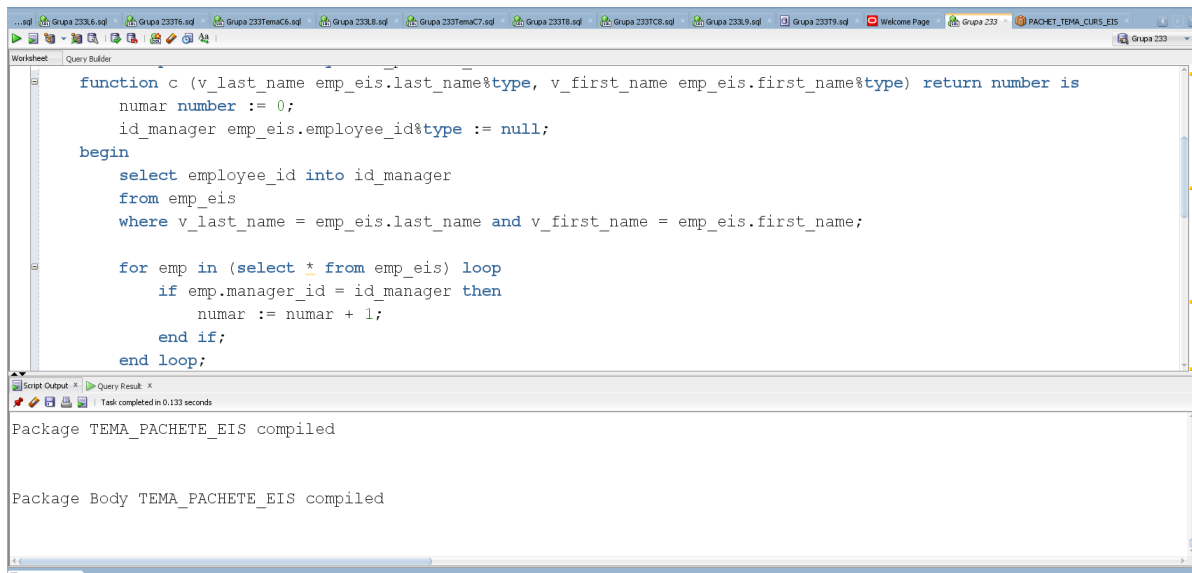
from emp_eis
where v_last_name = emp_eis.last_name and v_first_name = emp_eis.first_name;

for emp in (select * from emp_eis) loop
    if emp.manager_id = id_manager then
        numar := numar + 1;
    end if;
end loop;

return numar;

exception
when NO_DATA_FOUND then dbms_output.put_line('No data found!');
return 0;
when TOO_MANY_ROWS then dbms_output.put_line('Too many rows!');
return 0;
end c;
end tema_pachete_eis;
/

```



```

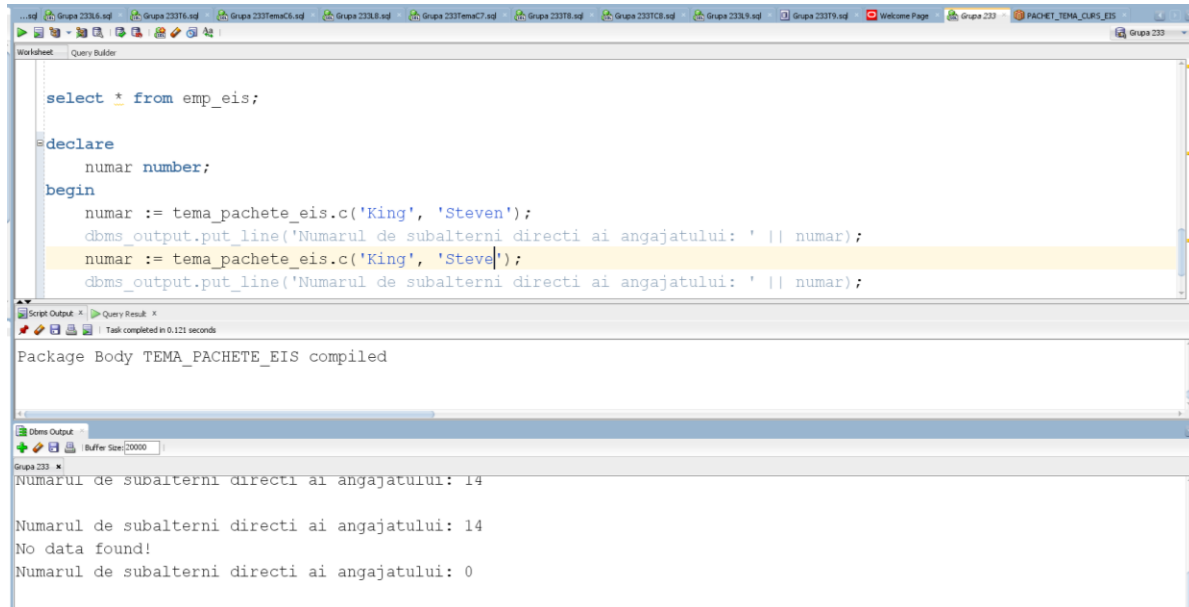
declare
    numar number;
begin
    numar := tema_pachete_eis.c('King', 'Steven');
    dbms_output.put_line('Numarul de subalterni directi ai angajatului: ' || numar);
    numar := tema_pachete_eis.c('King', 'Steve');
    dbms_output.put_line('Numarul de subalterni directi ai angajatului: ' || numar);
end;
/

```

Primul caz functioneaza, angajatul are subalterni.

La al doilea caz nu gaseste un angajat cu numele respectiv, deci o sa avem eroarea No data found si un numar de subalterni egal cu 0.

La al treilea caz nu exista exemplu in tabel (atunci cand avem doi angajati cu acelasi nume), dar exceptia de Too many rows functioneaza similar cu cea cu No data found, deci merge.



```
select * from emp_eis;

declare
    numar number;
begin
    numar := tema_pachete_eis.c('King', 'Steven');
    dbms_output.put_line('Numarul de subalterni directi ai angajatului: ' || numar);
    numar := tema_pachete_eis.c('King', 'Steve');
    dbms_output.put_line('Numarul de subalterni directi ai angajatului: ' || numar);
end;
```

Package Body TEMA\_PACHETE\_EIS compiled

Group 233

Numarul de subalterni directi ai angajatului: 14

Numarul de subalterni directi ai angajatului: 14

No data found!

Numarul de subalterni directi ai angajatului: 0

e. o procedură prin care se actualizează cu o valoare dată ca parametru salariul unui angajat al cărui nume este dat ca parametru:

- se va verifica dacă valoarea dată pentru salariu respectă limitele impuse pentru acel job;
- dacă sunt mai mulți angajați care au același nume, atunci se va afișa un mesaj corespunzător și de asemenea se va afișa lista acestora;
- dacă nu există angajați cu numele dat, atunci se va afișa un mesaj corespunzător;



```
drop table emp_eis;
create table emp_eis as select * from employees;
/

create or replace PACKAGE tema_pachete_eis as
    function c (v_last_name emp_eis.last_name%type, v_first_name emp_eis.first_name%type) return number;
    procedure e (v_last_name emp_eis.last_name%type, v_valoare emp_eis.salary%type);
end tema_pachete_eis;
/

create or replace PACKAGE body tema_pachete_eis as
    procedure e (v_last_name emp_eis.last_name%type, v_valoare emp_eis.salary%type) is

```

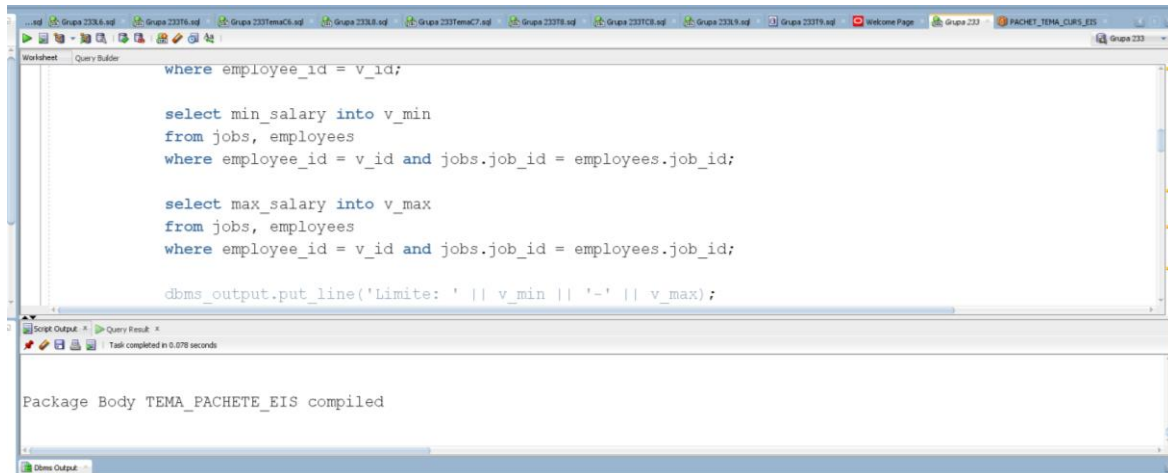
Package TEMA\_PACHETE\_EIS compiled

create or replace PACKAGE tema\_pachete\_eis as

```

function c (v_last_name emp_eis.last_name%type, v_first_name emp_eis.first_name%type) return
number;
procedure e (v_last_name emp_eis.last_name%type, v_valoare emp_eis.salary%type);
end tema_pachete_eis;
/

```



```

create or replace PACKAGE body tema_pachete_eis as
procedure e (v_last_name emp_eis.last_name%type, v_valoare emp_eis.salary%type) is
    v_id emp_eis.employee_id%type;
    v_salariu emp_eis.salary%type;
    v_min jobs.min_salary%type;
    v_max jobs.max_salary%type;
    v_salariu_nou emp_eis.salary%type;
begin
    select employee_id into v_id
    from emp_eis
    where v_last_name = last_name;

    select salary into v_salariu
    from emp_eis
    where employee_id = v_id;

    select min_salary into v_min
    from jobs, employees
    where employee_id = v_id and jobs.job_id = employees.job_id;

    select max_salary into v_max
    from jobs, employees
    where employee_id = v_id and jobs.job_id = employees.job_id;

    dbms_output.put_line('Limite: ' || v_min || '-' || v_max);

    v_salariu_nou := v_salariu + v_valoare;

```

```

        if (v_salariu_nou >= v_min and v_salariu_nou <= v_max) then
            update emp_eis
            set salary = v_salariu_nou
            where employee_id = v_id;
            dbms_output.put_line('Salariul ' || v_salariu_nou || ' dupa marire se incadreaza in limite: ' ||
v_min || '-' || v_max);
        else
            dbms_output.put_line('Salariul ' || v_salariu_nou || ' dupa marire nu se incadreaza in limite: '
|| v_min || '-' || v_max);
        end if;

    exception
        when TOO_MANY_ROWS then
            dbms_output.put_line('Avem mai multi angajati cu acest nume: ');
            for emp in (select * from emp_eis where last_name = v_last_name) loop
                dbms_output.put_line(emp.last_name || ' ' || emp.first_name);
            end loop;
        when NO_DATA_FOUND then
            dbms_output.put_line('Nu avem angajati cu acest nume!');
        end e;

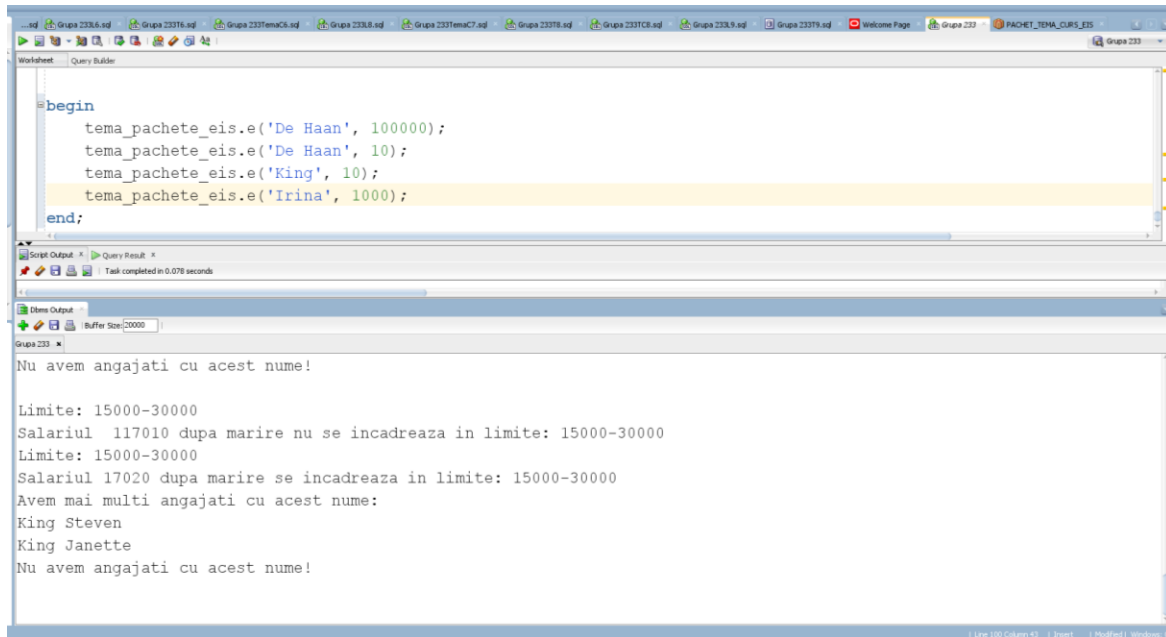
    function c (v_last_name emp_eis.last_name%type, v_first_name emp_eis.first_name%type) return
number is
        numar number := 0;
        id_manager emp_eis.employee_id%type := null;
    begin
        select employee_id into id_manager
        from emp_eis
        where v_last_name = emp_eis.last_name and v_first_name = emp_eis.first_name;

        for emp in (select * from emp_eis) loop
            if emp.manager_id = id_manager then
                numar := numar + 1;
            end if;
        end loop;

        return numar;

    exception
        when NO_DATA_FOUND then dbms_output.put_line('No data found!');
        return 0;
        when TOO_MANY_ROWS then dbms_output.put_line('Too many rows!');
        return 0;
    end c;
end tema_pachete_eis;
/

```



```

begin
    tema_pachete_eis.e('De Haan', 100000);
    tema_pachete_eis.e('De Haan', 10);
    tema_pachete_eis.e('King', 10);
    tema_pachete_eis.e('Irina', 1000);
end;
/

```

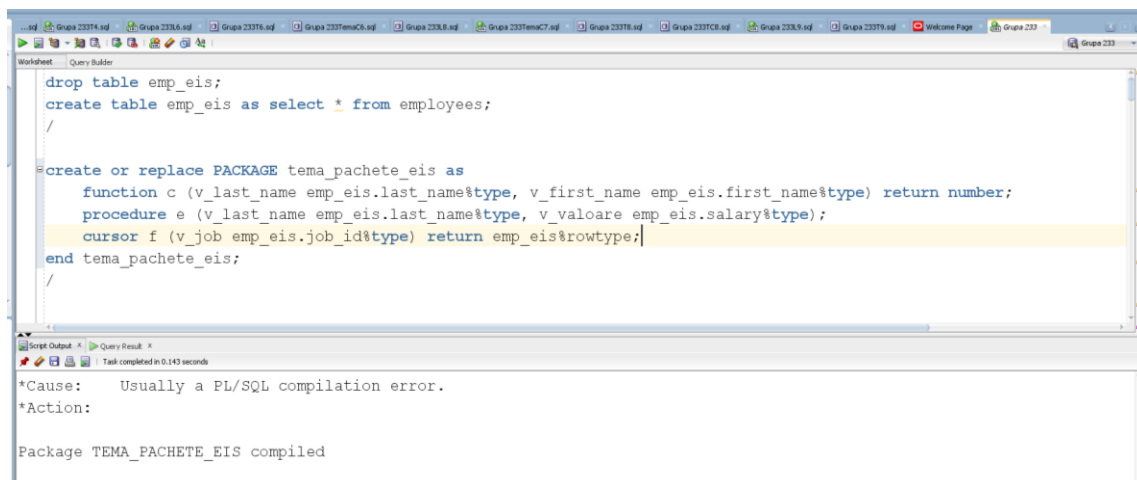
In primul caz, salariul nou depaseste, deci nu actualizam.

In al doilea caz, putem actualiza.

In al treilea caz avem mai multi angajati.

In al patrulea caz nu avem angajati cu acel nume.

f. un cursor care obține lista angajaților care lucrează pe un job al cărui cod este dat ca parametru;



```

create or replace PACKAGE tema_pachete_eis as
    function c (v_last_name emp_eis.last_name%type, v_first_name emp_eis.first_name%type) return
number;
    procedure e (v_last_name emp_eis.last_name%type, v_valoare emp_eis.salary%type);
    cursor f (v_job emp_eis.job_id%type) return emp_eis%rowtype;
end tema_pachete_eis;
/

```



```

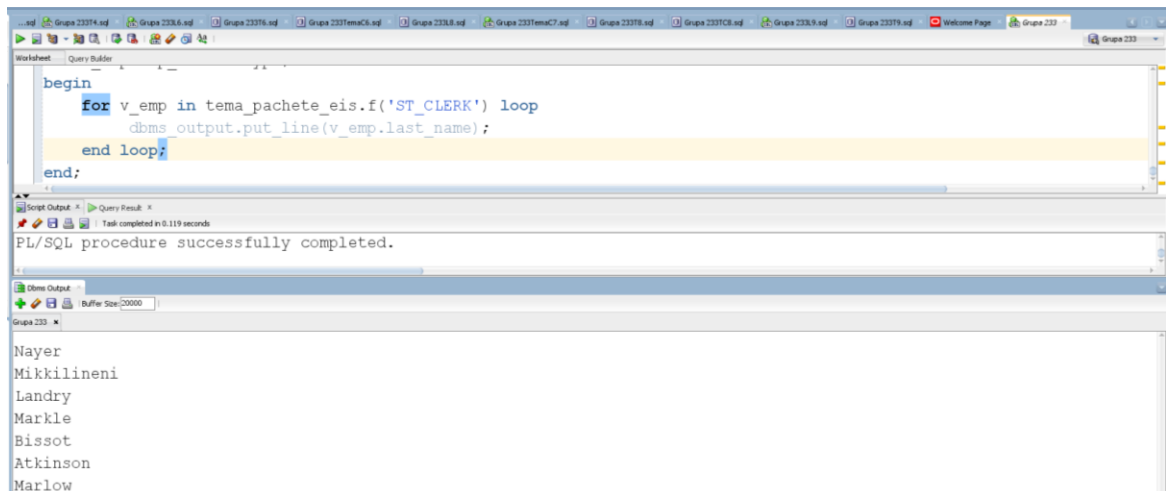
create or replace PACKAGE body tema_pachete_eis as

    cursor f (v_job emp_eis.job_id%type) return emp_eis%rowtype is
        select * from emp_eis where v_job = job_id;

-- ... restul functiilor si procedurilor de mai sus

end tema_pachete_eis;
/

```

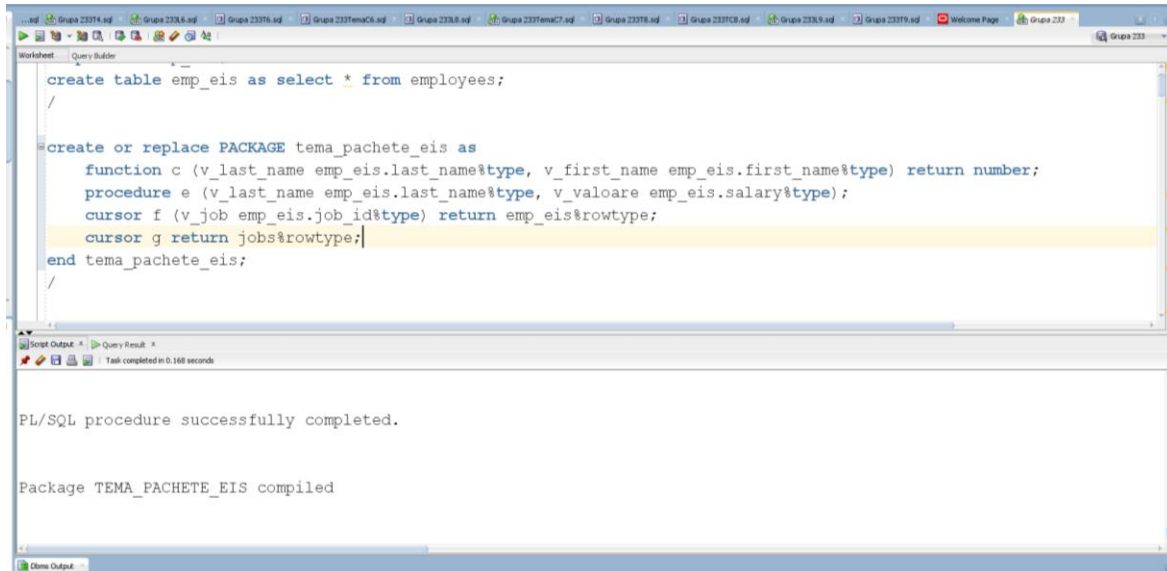


```

declare
    v_emp emp_eis%rowtype;
begin
    for v_emp in tema_pachete_eis.f('ST_CLERK') loop
        dbms_output.put_line(v_emp.last_name);
    end loop;
end;

g. un cursor care obține lista tuturor joburilor din companie;

```



```

create or replace PACKAGE tema_pachete_eis as
    function c (v_last_name emp_eis.last_name%type, v_first_name emp_eis.first_name%type) return
number;
    procedure e (v_last_name emp_eis.last_name%type, v_valoare emp_eis.salary%type);
    cursor f (v_job emp_eis.job_id%type) return emp_eis%rowtype;
    cursor g return jobs%rowtype;
end tema_pachete_eis;
/

```

```

create or replace PACKAGE body tema_pachete_eis as

```

```

    cursor f (v_job emp_eis.job_id%type) return emp_eis%rowtype is
        select * from emp_eis where v_job = job_id;

```

```

    cursor g return jobs%rowtype is
        select * from jobs;

```

```

-- ... restul functiilor si procedurilor de mai sus

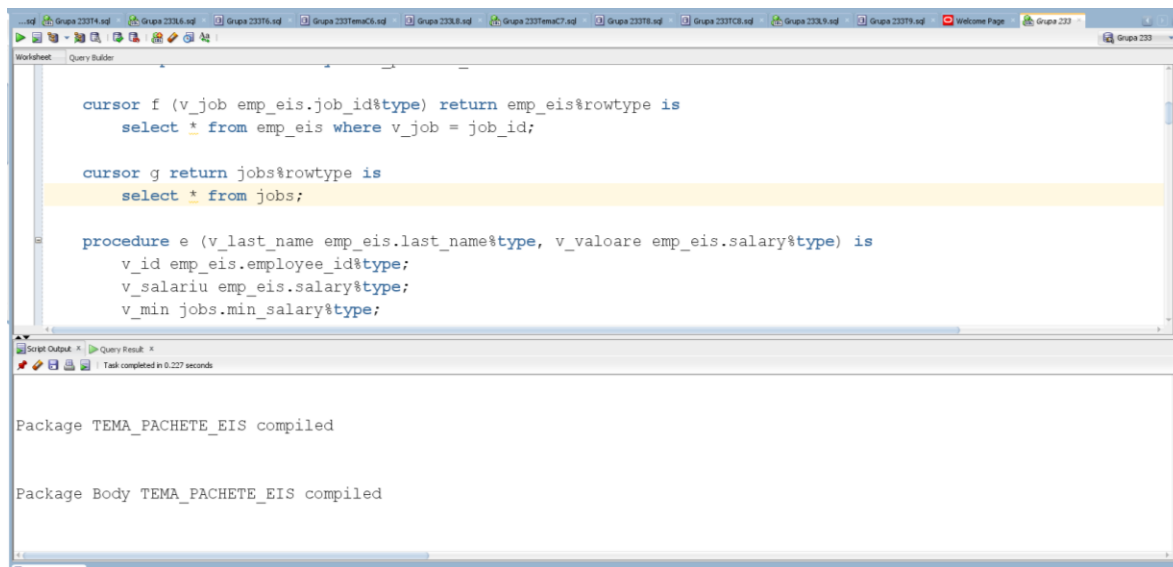
```

```

end tema_pachete_eis;
/

```





```

declare
    v_job jobs%rowtype;
begin
    for v_job in tema_pachete_eis.g loop
        dbms_output.put_line(v_job.job_title);
    end loop;
end;

```



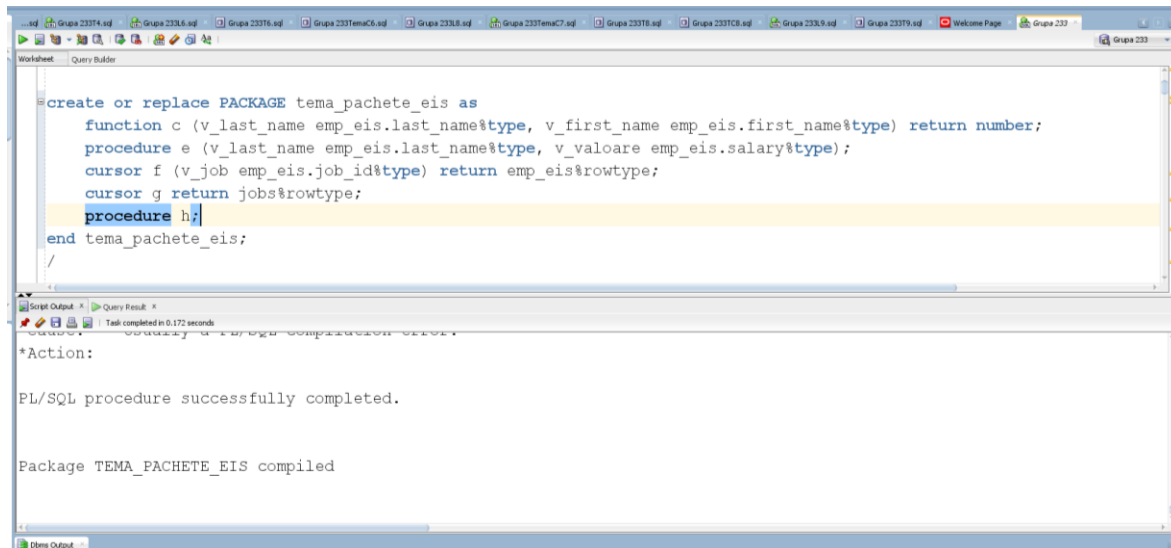
h. o procedură care utilizează cele două cursoare definite anterior și obține pentru fiecare job numele acestuia și lista angajaților care lucrează în prezent pe acel job; în plus, pentru fiecare angajat să se specifice dacă în trecut a mai avut sau nu jobul respectiv.

create or replace PACKAGE tema\_pachete\_eis as

```

function c (v_last_name emp_eis.last_name%type, v_first_name emp_eis.first_name%type) return
number;
procedure e (v_last_name emp_eis.last_name%type, v_valoare emp_eis.salary%type);
cursor f (v_job emp_eis.job_id%type) return emp_eis%rowtype;
cursor g return jobs%rowtype;
procedure h;
end tema_pachete_eis;
/

```



create or replace PACKAGE body tema\_pachete\_eis as

```

cursor f (v_job emp_eis.job_id%type) return emp_eis%rowtype is
select * from emp_eis where v_job = job_id;

```

```

cursor g return jobs%rowtype is
select * from jobs;

```

```

procedure h is
begin
  for v_job in g loop
    dbms_output.put_line('Job: ' || v_job.job_title);
    for v_emp in f(v_job.job_id) loop
      dbms_output.put_line(v_emp.last_name || ' ' || v_emp.first_name);
    end loop;
    dbms_output.put_line("");
  end loop;
end h;

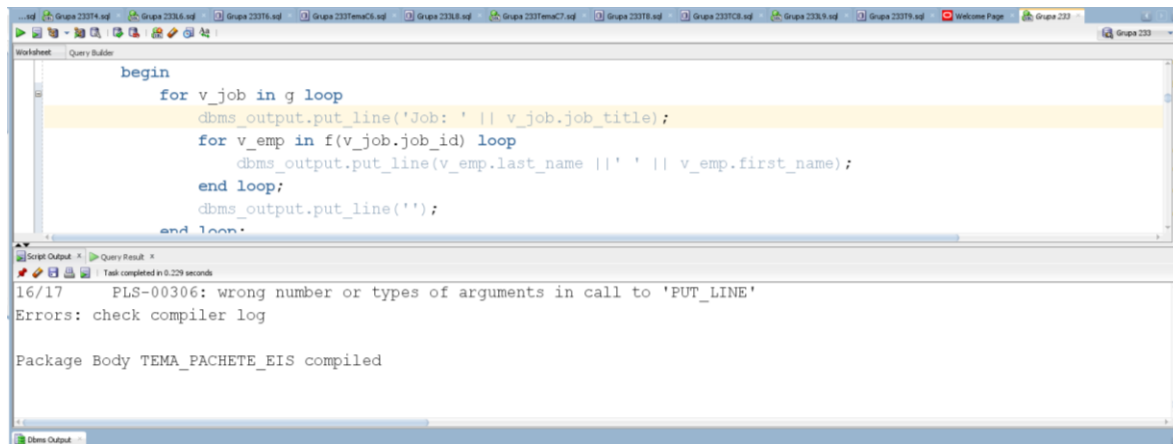
```

-- ... restul functiilor si procedurilor de mai sus

```

end tema_pachete_eis;
/

```



```

begin
  tema_pachete_eis.h;
end;

```

