RZOLVARE EXERCITIU ZODII

```
declare
cursor distributie_zodie is
  select nume_zodie, count(*) distributie from zodiac z join utilizatori u on
    to_date(to_char(u.data_nastere, 'DD-MM'), 'DD-MM') between to_date(z.data_inceput, 'DD-MM')
and to_date(z.data_sfarsit, 'DD-MM')
    group by nume_zodie;

cursor stud(zodie varchar2) is select nume, prenume, data_nastere from utilizatori u join zodiac z on
  to_date(to_char(u.data_nastere, 'DD-MM'), 'DD-MM') between to_date(z.data_inceput, 'DD-MM') and
to_date(z.data_sfarsit, 'DD-MM')
  where z.nume_zodie = zodie;

begin
  for v_zodie in distributie_zodie loop
    dbms_output.put_line(v_zodie.nume_zodie||' '||v_zodie.distributie);
    for rand in stud(v_zodie.nume_zodie) LOOP
      dbms_output.put_line(rand.nume||' '||rand.prenume||' '||rand.data_nastere );
      end loop;
  end loop;
end;
```

Exercitii cu functii si TRIGGERE

```
set serveroutput on;
drop table evaluare;
/
create table evaluare
(
```

```
id number not null primary key,
nume varchar2(30) not null,
curs varchar2(30) not null,
nota number not null,
data date
)
/

CREATE SEQUENCE contor
  MINVALUE 1
  MAXVALUE 100
  START WITH 1
  INCREMENT BY 1
  CACHE 30;
/
insert into evaluare values(contor.nextval,'Ionescu', 'SGBD', 7, '30.apr.2019' );
/
insert into evaluare values(contor.nextval,'Popescu', 'SGBD', 8, '30.apr.2019' );
/
insert into evaluare values(contor.nextval,'Angelescu', 'SGBD', 9, '15.apr.2019' );
/
insert into evaluare values(contor.nextval,'Mirel', 'SGBD', 7, '10.apr.2019' );
/
insert into evaluare values(contor.nextval,'Gigi', 'SGBD', 5, '02.06.2019' );
/
insert into evaluare values(contor.nextval,'Mitica', 'SGBD', 9, '07.06.2019' );
/
insert into evaluare values(contor.nextval,'Bibi', 'SGBD', 9, '07.05.2019' );
/
```

```
insert into evaluare values(contor.nextval,'Popescu', 'SGBD', 7, '09.05.2019' );
/
insert into evaluare values(contor.nextval,'Angelescu', 'SGBD', 10, sysdate);
/
insert into evaluare values(contor.nextval,'Popescu', 'SGBD', 6, sysdate );
/
insert into evaluare values(contor.nextval,'Bibi', 'SGBD', 7, sysdate );
/
select * from evaluare;



--exercitiu cu exceptii
--Pentru tabela Evaluare sa se creeze o functie care afiseaza pentru un student dat ultima nota obtinuta
CREATE OR REPLACE FUNCTION nota_recenta_student(  p_nume IN evaluare.nume%type)
  RETURN VARCHAR2
AS
  nota_recenta INTEGER;
  mesaj       VARCHAR2(32767);
  counter      INTEGER;
BEGIN
  SELECT nota
  INTO nota_recenta
  FROM
    (SELECT nota
    FROM evaluare
    WHERE nume like p_nume
    ORDER BY data DESC
    )
  WHERE rownum <= 1;
```

```
  mesaj       := 'Cea mai recenta nota a studentului  ' || p_nume || ' este ' || nota_recenta || '.';
  RETURN mesaj;
EXCEPTION
WHEN no_data_found THEN
  SELECT COUNT(*) INTO counter FROM evaluare WHERE nume like p_nume;
  IF counter = 0 THEN
   mesaj   := 'Studentul  ' || p_nume || ' nu exista in baza de date.';
            -- raise_application_error (-20001,'Studentul  ' || p_nume || ' nu exista in baza de date.');
            -- se poate arunca o exceptie ca cea de sus sau se poate returna un mesaj de eroare
   return mesaj;
  END IF;
END nota_recenta_student;
/
```

--testare functie. Functia se poate folosi direct intr-o comanda SELECT. Se va interoga tabela DUAL pentru a obtine o singura inregistrare.

```
select nota_recenta_student('Popescu') from dual; --student existent in baza de date
select nota_recenta_student('Marcel') from dual;  --student inexistent in baza de date
```

--Daca se interogheaza tabela Evaluare se vor parcurge toate inregistrarile din tabela si se obtine acelasi rezultat de atatea ori cate inregistrari avem in tabela

```
select nota_recenta_student('Popescu') from evaluare;
```

--se parcurg toate inregistrarile din tabela evaluare si se aplica functia dupa parametrul Popescu

--exercitiu cu triggere

```
CREATE OR REPLACE TRIGGER marire_nota
  before UPDATE OF nota ON evaluare
-- aici se executa numai cand modificam nota, intainte (BEFORE) de a o modifica(UPDATE) !
```

```
  FOR EACH ROW  --pentru fiecare rand in parte se declanseaza trigger-ul
  declare
  BEGIN
 dbms_output.put_line('Nume Student: ' || :OLD.nume||' are vechea nota '|| :OLD.nota||' modificam cu nota '||:NEW.nota);
  -- observati ca aveti acces si la alte campuri, nu numai la cele modificate
  -- totusi nu permitem sa facem update daca valoarea este mai mica (conform regulamentului universitatii):
  IF (:OLD.nota>=:NEW.nota) THEN
     :NEW.nota := :OLD.nota;
     dbms_output.put_line('Nu s-a facut modificarea notei');
  ELSE
     dbms_output.put_line('S-a facut modificarea notei');
  end if;
END;
/
drop trigger marire_nota;  -- se executa daca vrem sa stergem trigger-u
update evaluare set nota =8 where id in (1,2,3,4);  --testam declansarea trigger-ului
select valoare from note where id in(1,2,3,4);  --verificam modificarile in baza de date
rollback;


-- Exercitiul doi cu triggere
--select user from dual;
--drop table note_test;
create table note_test as select * from evaluare;


CREATE OR REPLACE TRIGGER del_note
  BEFORE delete ON note_test
  BEGIN
```

```
    RAISE_APPLICATION_ERROR (
      num => -20001,
      msg => 'can''t touch this');
  END;
/
--Triggerul arunca o exceptie prin raise_application_error cu un numar alocat si un mesaj de eroare
drop trigger del_note;  --se executa cand vrem sa stergem trigger-ul
select count(*) from note_test;
select * from note_test;
delete from note_test where id between 1 and 10;
rollback;
```