CREATE TABLE turist

(

id\_turist NUMBER(5) PRIMARY KEY,

nume VARCHAR2(64),

prenume VARCHAR2(64),

data\_nasterii DATE,

oras VARCHAR2(64)

);

CREATE TABLE agentie

(

id\_agentie NUMBER(5) PRIMARY KEY,

denumire VARCHAR2(64),

adresa VARCHAR2(64)

);

CREATE TABLE excursie

(

id\_excursie NUMBER(5) PRIMARY KEY,

denumire VARCHAR2(64),

pret NUMBER(10,2),

data\_inceput DATE,

data\_sfarsit DATE,

numar\_locuri NUMBER(3),

cod\_agentie NUMBER(5),

CONSTRAINT fk\_cod FOREIGN KEY (cod\_agentie) REFERENCES agentie(id\_agentie)

);

CREATE TABLE achizitioneaza

(

cod\_turist NUMBER(5) ,

cod\_excursie NUMBER(5),

data\_achizitie DATE,

PRIMARY KEY (cod\_turist, cod\_excursie),

FOREIGN KEY (cod\_turist) REFERENCES turist(id\_turist),

FOREIGN KEY (cod\_excursie) REFERENCES excursie(id\_excursie)

);

/

-- exercitiu 2

CREATE type tzile

AS

object

(

id\_exc NUMBER(5),

numar\_zile NUMBER(4)

)

; -- cate zile are fiecare exursie

CREATE type lista\_tzile

AS

TABLE OF tzile;

ALTER TABLE turist ADD lista lista\_tzile nested TABLE lista store

AS

turist\_lista;

/

DECLARE

v lista\_tzile;

BEGIN

FOR i IN

(SELECT \* FROM turist

)

LOOP

v := lista\_tzile

(

)

;

FOR j IN

(SELECT \* FROM achizitioneaza WHERE cod\_turist=i.id\_turist

)

LOOP

v.extend

(

)

;

v(v.last()):=tzile(j.cod\_excursie,0);

SELECT data\_sfarsit-data\_inceput

INTO v(v.last()).numar\_zile

FROM excursie

WHERE id\_excursie=j.cod\_excursie;

END LOOP;

UPDATE turist

SET lista =v

WHERE id\_turist=i.id\_turist;

END LOOP;

END;

/

SET serveroutput ON;

DECLARE

v NUMBER(4);

BEGIN

FOR i IN

(SELECT \* FROM turist

)

LOOP

v:=0;

FOR j IN 1..i.lista.count()

LOOP

v:=v+i.lista(j).numar\_zile;

END LOOP;

dbms\_output.put\_line(i.nume||' '||i.lista.count()||' '||v);

END LOOP;

END;

--end exercitiu 2 :)))))

/

--start exercitiu 3

CREATE

PROCEDURE plm\_nume(

orasel VARCHAR2)

IS

variabila NUMBER(4);

BEGIN

FOR i IN

(SELECT \* FROM turist WHERE orasel=oras

)

LOOP

dbms\_output.put\_line(i.nume);

FOR j IN

(SELECT e.denumire d

FROM achizitioneaza a

JOIN excursie e

ON (a.cod\_excursie=e.id\_excursie)

WHERE cod\_turist =i.id\_turist

)

LOOP

dbms\_output.put\_line(j.d);

END LOOP;

SELECT MIN(data\_inceput-sysdate)

INTO variabila

FROM achizitioneaza a

JOIN excursie e

ON (a.cod\_excursie=e.id\_excursie)

WHERE cod\_turist =i.id\_turist

AND sysdate <data\_inceput;

dbms\_output.put\_line(variabila);

END LOOP;

END;

/

--exercitiu 4

CREATE OR REPLACE

FUNCTION nume\_fct(

nume\_agentie VARCHAR2 )

RETURN NUMBER

IS

cea\_mai\_cea NUMBER(4);

BEGIN

SELECT id\_excursie

INTO cea\_mai\_cea

FROM

(SELECT id\_excursie

FROM excursie e

JOIN agentie a

ON e.cod\_agentie=a.id\_agentie

WHERE a.denumire=nume\_agentie

ORDER BY pret DESC

)

WHERE rownum<=1;

SELECT floor(months\_between(sysdate,data\_n) /12)

INTO cea\_mai\_cea

FROM

(SELECT data\_nasterii data\_n

FROM turist t

JOIN achizitioneaza a

ON t.id\_turist =a.cod\_turist

WHERE a.cod\_excursie = cea\_mai\_cea

ORDER BY 1 DESC

)

WHERE rownum <=1;

RETURN cea\_mai\_cea;

EXCEPTION

WHEN no\_data\_found THEN

raise\_application\_error (-22000,'N-avem') ;

END;

/

--exercitiu 5

CREATE OR REPLACE

PACKAGE pk

IS

type texc

IS

TABLE OF NUMBER INDEX BY pls\_integer;

exc texc;

END;

CREATE TRIGGER tr\_ins before

INSERT ON achizitioneaza BEGIN FOR i IN

(SELECT id\_excursie,

COUNT(cod\_turist) turisti

FROM achizitioneaza a

RIGHT JOIN excursie e

ON a.cod\_excursie=e.id\_excursie

GROUP BY id\_excursie

)

LOOP pk.exc

(

i.id\_excursie

)

:=i.turisti;

END LOOP;

END;

CREATE OR REPLACE TRIGGER tr\_bi before

INSERT ON achizitioneaza FOR EACH row DECLARE ceva NUMBER;

BEGIN

SELECT numar\_locuri

INTO ceva

FROM excursie

WHERE id\_excursie =:new.cod\_excursie;

IF pk.exc(:new.cod\_excursie)=ceva THEN

raise\_application\_error(-22213, 'locuri ocupate la '||:new.cod\_excursie);

END IF;

pk.exc(:new.cod\_excursie):=pk.exc(:new.cod\_excursie)+1;

END;by ady