Academia de Studii Economice

Facultatea de Cibernetica, Statistica si Informatica Economica

**Proiect SGBD**

**Gestiunea datelor in domeniul aviatic**

**Profesor coordonator Student**

[RADU Andreea -](https://online.ase.ro/user/view.php?id=4568&course=1) Oana Lovin Andrei

Grupa 1054, Seria D

**Cuprins:**

[Creati o baza de date dintr-un domeniu ales de voi care sa contina cel putin 6 tabele, fiecare tabela continand cel putin 10 inregistrari 3](#_Toc135595128)

[A. Schema conceptuala 9](#_Toc135595129)

[B. Interactiunea cu serverul Oracle prin intermediul comenzilor SQL 10](#_Toc135595130)

[C. Structuri alternative si repetitive. 11](#_Toc135595131)

[D. Colectii de date (index by table, nested table, varray) 16](#_Toc135595132)

[E. Tratarea exceptiilor 20](#_Toc135595133)

[F. Gestionarea cursorilor: impliciti si expliciti 28](#_Toc135595134)

[G. Functii, proceduri, includerea acestora în pachete. 31](#_Toc135595135)

[H. Declansatori la nivel de instructiune si de rand 46](#_Toc135595136)

# Creati o baza de date dintr-un domeniu ales de voi care sa contina cel putin 6 tabele, fiecare tabela continand cel putin 10 inregistrari

CREATE TABLE TARI\_2 (

Id\_tara VARCHAR2(5) CONSTRAINT PK\_tari PRIMARY KEY,

Nume\_tara VARCHAR2(200) NOT NULL);



CREATE TABLE AEROPORTURI (

Id\_aeroport NUMBER(3) CONSTRAINT PK\_aeroporturi PRIMARY KEY,

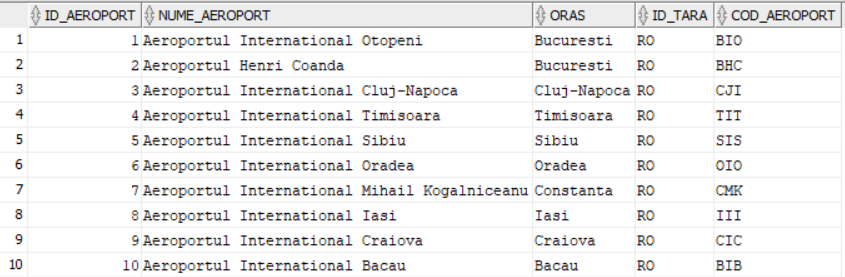
Nume\_aeroport VARCHAR2(200) NOT NULL,

Oras VARCHAR2(200) NOT NULL,

Id\_tara VARCHAR2(5) NOT NULL,

Cod\_aeroport VARCHAR2(5) NOT NULL,

CONSTRAINT FK\_tari FOREIGN KEY (Id\_tara) REFERENCES TARI\_2 (Id\_tara));



CREATE TABLE COMPANII\_AERIENE (

Id\_companie NUMBER(3) CONSTRAINT PK\_companii PRIMARY KEY,

Nume\_companie VARCHAR2(200) NOT NULL,

Nume\_tara VARCHAR2(200) NOT NULL,

An\_fondare NUMBER(4) NOT NULL);



CREATE TABLE CURSE (

Id\_cursa NUMBER(3) CONSTRAINT PK\_curse PRIMARY KEY,

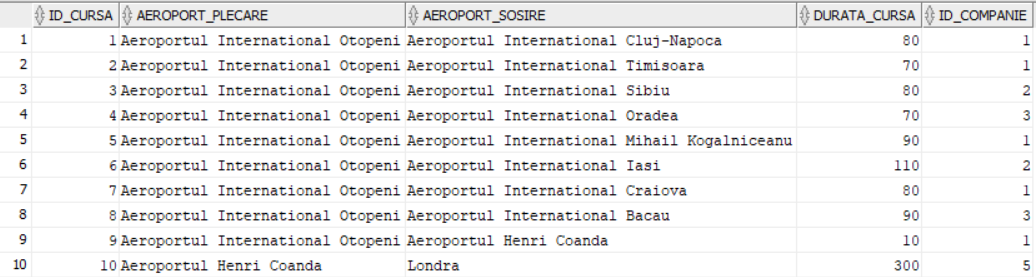
Aeroport\_plecare VARCHAR2(200) NOT NULL,

Aeroport\_sosire VARCHAR2(200) NOT NULL,

Durata\_cursa NUMBER(5) NOT NULL,

Id\_companie NUMBER(3) NOT NULL,

CONSTRAINT FK\_companii FOREIGN KEY (Id\_companie) REFERENCES COMPANII\_AERIENE (Id\_companie));



CREATE TABLE ZBORURI (

Id\_zbor NUMBER(3) CONSTRAINT PK\_zboruri PRIMARY KEY,

Nume\_zbor VARCHAR2(200) NOT NULL,

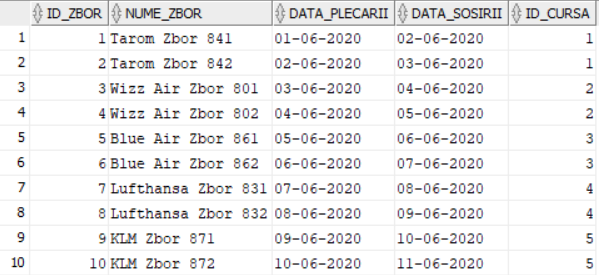
Data\_plecarii DATE NOT NULL,

Data\_sosirii DATE NOT NULL,

Id\_cursa NUMBER(3) NOT NULL,

CONSTRAINT FK\_curse FOREIGN KEY (Id\_cursa) REFERENCES CURSE (Id\_cursa)

);



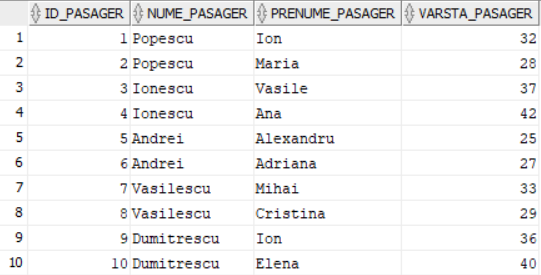
CREATE TABLE PASAGERI (

Id\_pasager NUMBER(3) CONSTRAINT PK\_pasageri PRIMARY KEY,

Nume\_pasager VARCHAR2(200) NOT NULL,

Prenume\_pasager VARCHAR2(200) NOT NULL,

Varsta\_pasager NUMBER(3) NOT NULL);



CREATE TABLE BILETE (

Id\_bilet NUMBER(3) CONSTRAINT PK\_bilete PRIMARY KEY,

Id\_zbor NUMBER(3) NOT NULL,

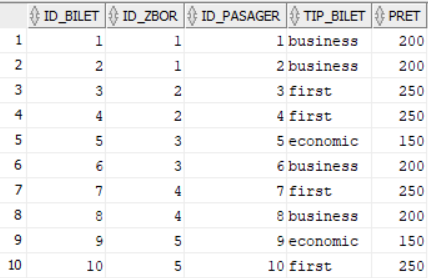
Id\_pasager NUMBER(3) NOT NULL,

Tip\_bilet VARCHAR2(200) NOT NULL,

Pret NUMBER(4) NOT NULL,

CONSTRAINT FK\_zboruri FOREIGN KEY (Id\_zbor) REFERENCES ZBORURI (Id\_zbor),

CONSTRAINT FK\_pasageri FOREIGN KEY (Id\_pasager) REFERENCES PASAGERI (Id\_pasager));



CREATE TABLE ECHIPAJ (

Id\_echipaj NUMBER(3),

Id\_zbor NUMBER(3) NOT NULL,

Id\_membru NUMBER(3) NOT NULL CONSTRAINT PK\_membru\_echipaj PRIMARY KEY,

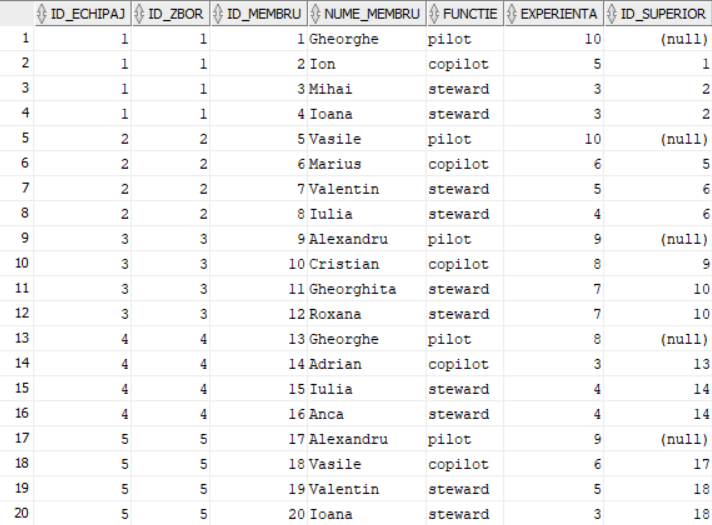
Nume\_membru VARCHAR2(200) NOT NULL,

Functie VARCHAR2(200) NOT NULL,

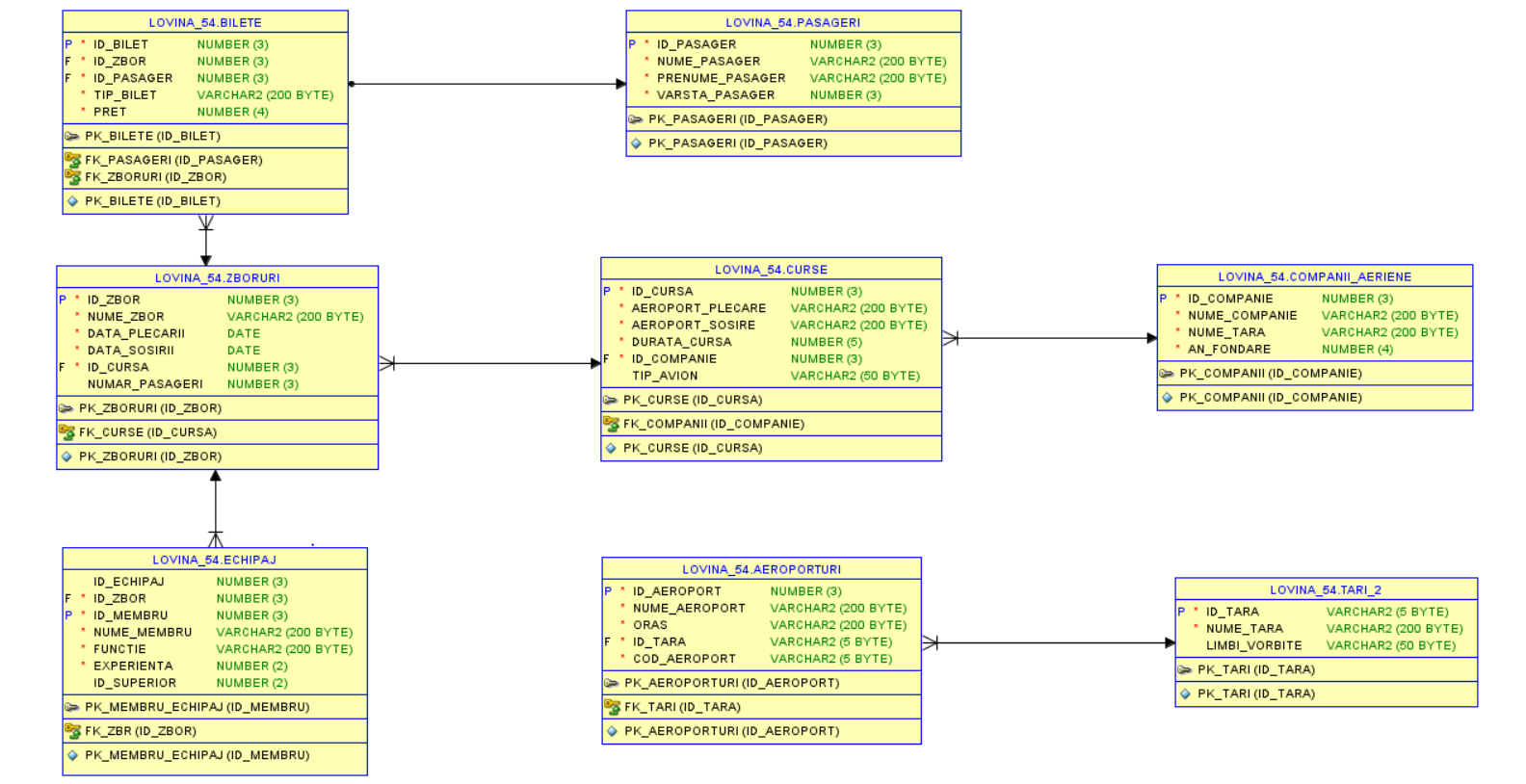
Experienta NUMBER(2) NOT NULL,

Id\_superior NUMBER(2),

CONSTRAINT FK\_zbr FOREIGN KEY (Id\_zbor) REFERENCES ZBORURI (Id\_zbor));



# A. Schema conceptuala



Schema este nemodificata.

# B. Interactiunea cu serverul Oracle prin intermediul comenzilor SQL

Creati un bloc PL/SQL care sterge biletul al carui cod este citit de la tastatura. In cazul in care acesta nu exista se va invoca o exceptie.

SET SERVEROUTPUT ON

DECLARE

v\_id\_bilet NUMBER(3) := &id;

BEGIN

EXECUTE IMMEDIATE 'DELETE FROM BILETE WHERE Id\_bilet = :id' USING v\_id\_bilet;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Biletul cu ID-ul introdus nu exista in tabela BILETE.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Biletul cu ID-ul ' || v\_id\_bilet || ' a fost sters din tabela BILETE.');

END IF;

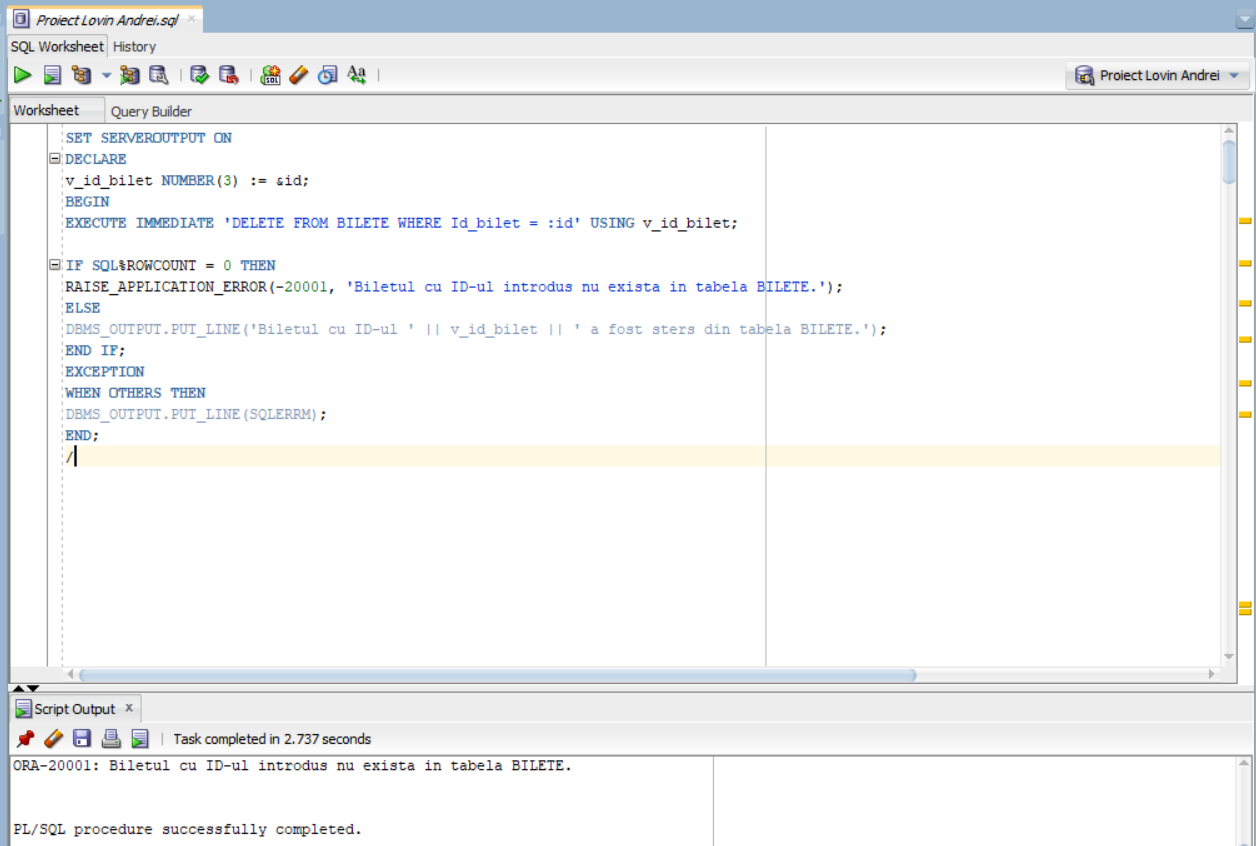
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/



# C. Structuri alternative si repetitive.

Creati un bloc PL/SQL care calculeaza numarul de curse care pleaca din Romania.

DECLARE

v\_id\_tara TARI\_2.Id\_tara%TYPE := 'RO';

v\_numar\_curse INTEGER := 0;

BEGIN

IF v\_id\_tara IS NOT NULL THEN

SELECT COUNT(\*)

INTO v\_numar\_curse

FROM CURSE c

INNER JOIN AEROPORTURI ap ON c.Aeroport\_plecare = ap.Nume\_aeroport

INNER JOIN TARI\_2 t ON ap.Id\_tara = t.Id\_tara

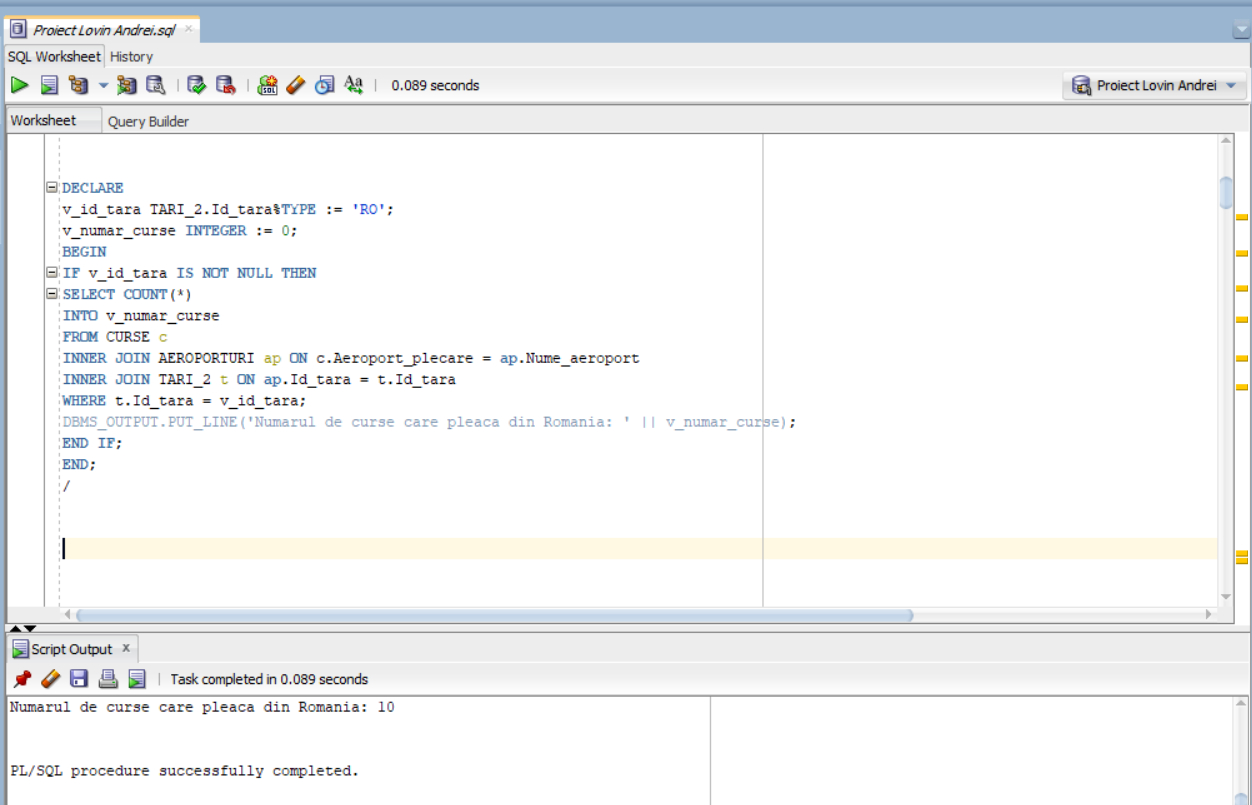
WHERE t.Id\_tara = v\_id\_tara;

DBMS\_OUTPUT.PUT\_LINE('Numarul de curse care pleaca din Romania: ' || v\_numar\_curse);

END IF;

END;

/



Creati un bloc PL/SQL care afiseaza toate aeroporturile dintr-o tara din baza de date.

DECLARE

v\_tara VARCHAR2(5) := 'RO';

v\_aeroport VARCHAR2(200);

CURSOR c\_aeroporturi IS

SELECT Nume\_aeroport

FROM AEROPORTURI a

JOIN TARI\_2 t ON a.Id\_tara = t.Id\_tara

WHERE t.Id\_tara = v\_tara;

BEGIN

OPEN c\_aeroporturi;

LOOP

FETCH c\_aeroporturi INTO v\_aeroport;

EXIT WHEN c\_aeroporturi%NOTFOUND;

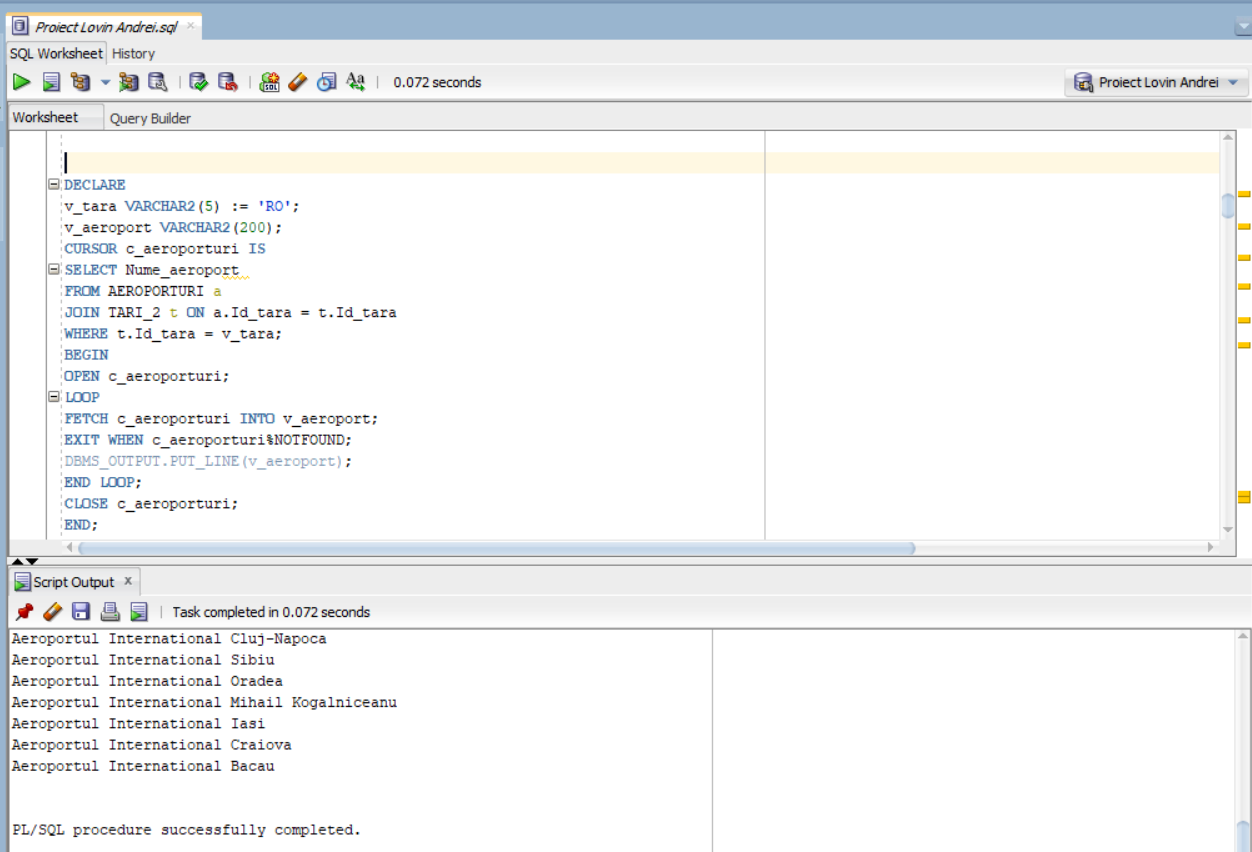
DBMS\_OUTPUT.PUT\_LINE(v\_aeroport);

END LOOP;

CLOSE c\_aeroporturi;

END;

/



Creati un bloc PL/SQL care afiseaza toate biletele achizitionate de un anumit pasager.

DECLARE

v\_pasager VARCHAR2(200) := 'Popescu';

v\_zbor VARCHAR2(200);

v\_pret NUMBER(4);

CURSOR c\_bilete IS

SELECT z.Nume\_zbor, b.Pret

FROM BILETE b

JOIN PASAGERI p ON b.Id\_pasager = p.Id\_pasager

JOIN ZBORURI z ON b.Id\_zbor = z.Id\_zbor

WHERE p.Nume\_pasager = v\_pasager;

BEGIN

OPEN c\_bilete;

LOOP

FETCH c\_bilete INTO v\_zbor, v\_pret;

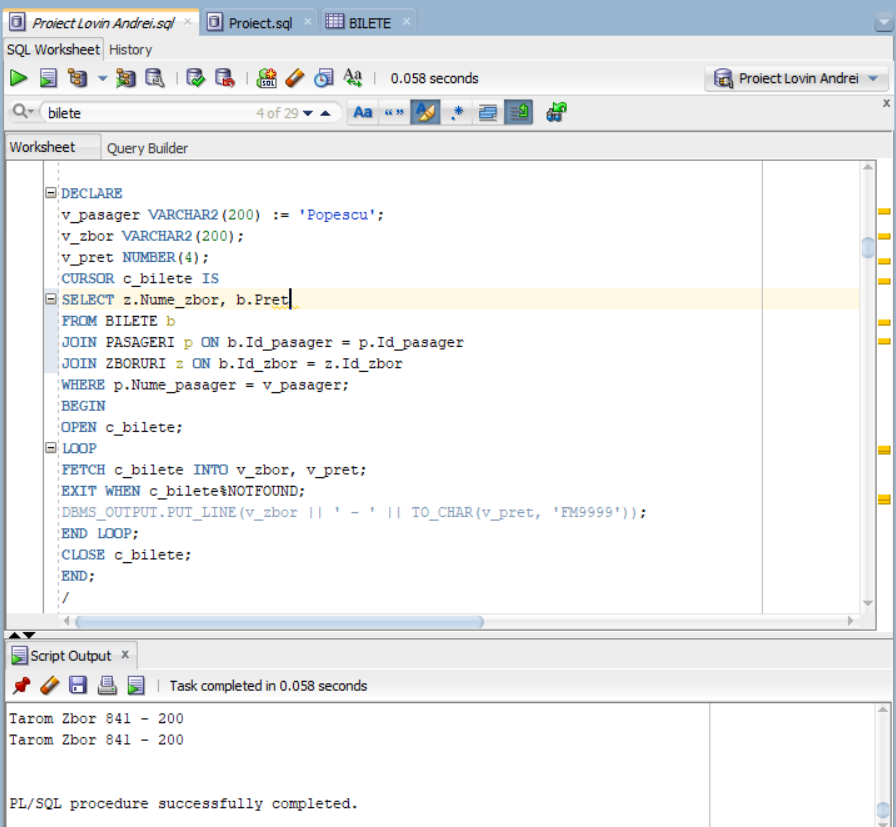
EXIT WHEN c\_bilete%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_zbor || ' - ' || TO\_CHAR(v\_pret, 'FM9999'));

END LOOP;

CLOSE c\_bilete;

END;



# D. Colectii de date (index by table, nested table, varray)

Creati un bloc PL/SQL care selectează toate înregistrările din tabelul PASAGERI și le stochează într-un tabel "pasageri\_tab". Apoi, se parcurge fiecare înregistrare din tabel și se afișează id-ul pasagerului, numele și prenumele acestuia.

DECLARE

TYPE pasageri\_tab IS TABLE OF PASAGERI%ROWTYPE INDEX BY PLS\_INTEGER;

t\_pasageri pasageri\_tab;

BEGIN

SELECT \*

BULK COLLECT INTO t\_pasageri

FROM PASAGERI;

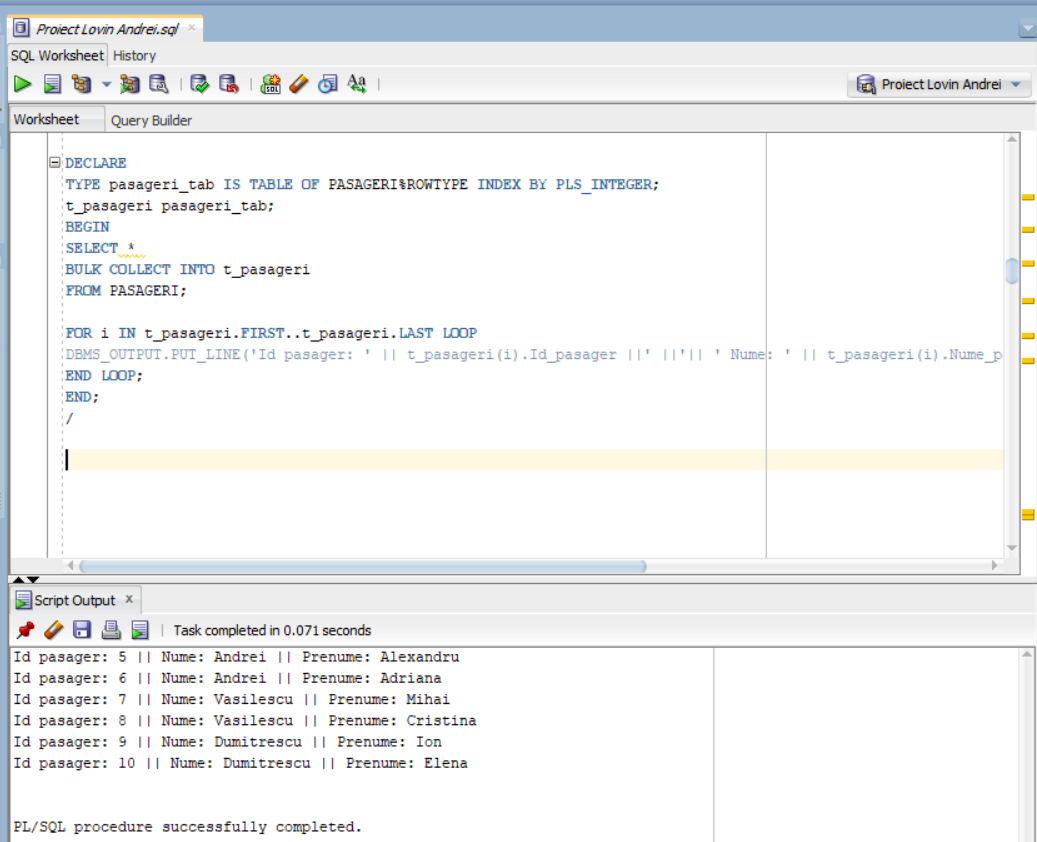
FOR i IN t\_pasageri.FIRST..t\_pasageri.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE('Id pasager: ' || t\_pasageri(i).Id\_pasager ||' ||'|| ' Nume: ' || t\_pasageri(i).Nume\_pasager ||' ||'|| ' Prenume: ' || t\_pasageri(i).Prenume\_pasager);

END LOOP;

END;

/



Creati un bloc PL/SQL care extrage identificatorii de tara si numele corespunzator din tabelul TARI\_2 si le afiseaza, utilizand nested table.

DECLARE

TYPE IdListType IS TABLE OF TARI\_2.Id\_tara%TYPE;

IdList IdListType := IdListType();

NumeTara TARI\_2.Nume\_tara%TYPE;

BEGIN

SELECT Id\_tara

BULK COLLECT INTO IdList

FROM TARI\_2;

FOR i IN 1..IdList.COUNT LOOP

SELECT Nume\_tara

INTO NumeTara

FROM TARI\_2

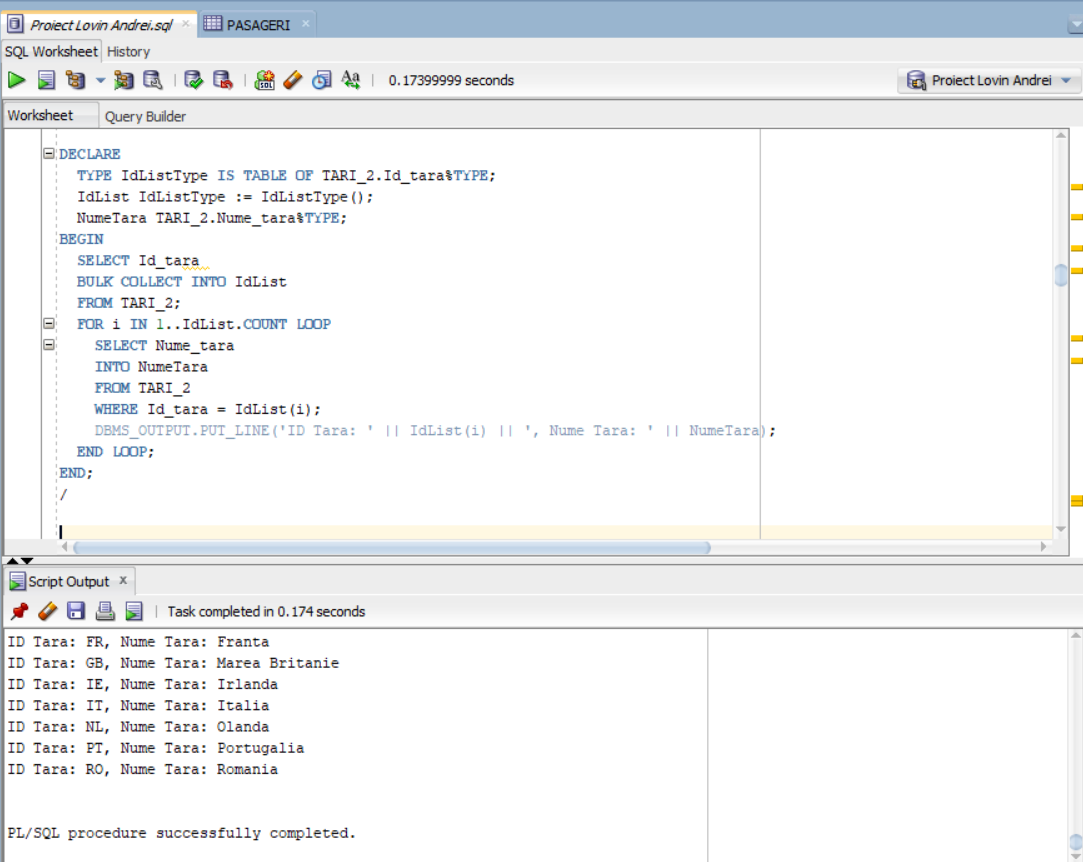
WHERE Id\_tara = IdList(i);

DBMS\_OUTPUT.PUT\_LINE('ID Tara: ' || IdList(i) || ', Nume Tara: ' || NumeTara);

END LOOP;

END;

/



Creati un bloc PL/SQL care insereaza înregistrari în tabelul PASAGERI folosind un varray.

DECLARE

TYPE PasagerListaType IS VARRAY(10) OF PASAGERI.Id\_pasager%TYPE;

PasagerLista PasagerListaType := PasagerListaType(201, 202, 203);

BEGIN

FOR i IN 1..PasagerLista.COUNT LOOP

INSERT INTO PASAGERI (Id\_pasager, Nume\_pasager, Prenume\_pasager, Varsta\_pasager)

VALUES (PasagerLista(i), 'Nume Pasager ' || PasagerLista(i), 'Prenume Pasager ' || PasagerLista(i), 30);

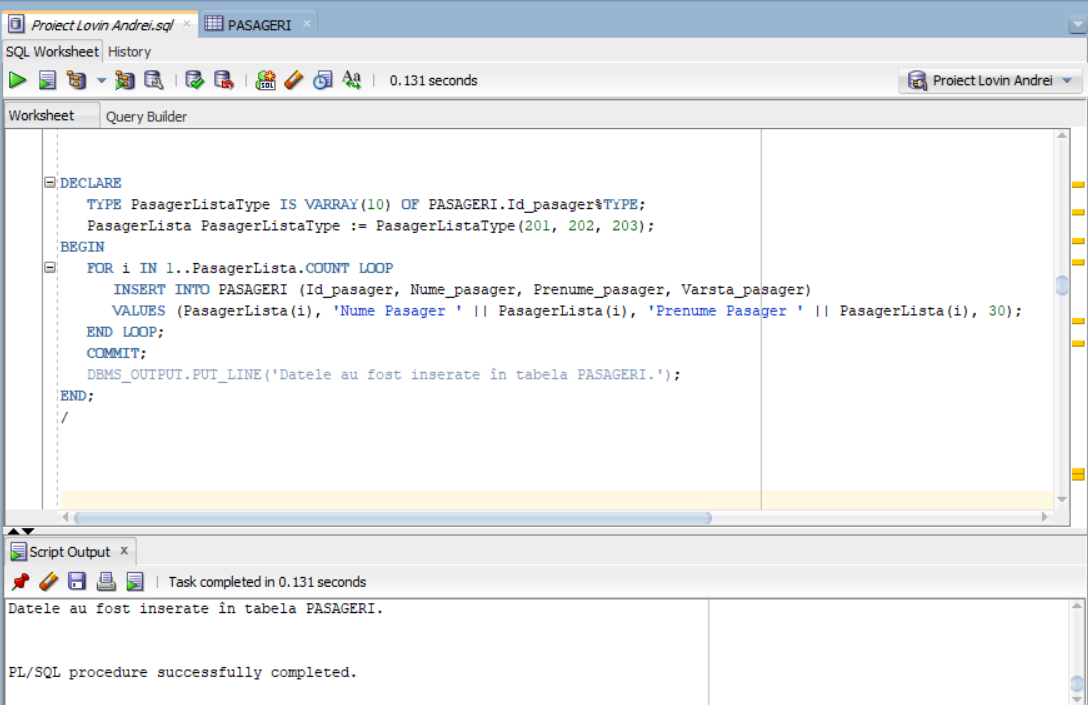
END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Datele au fost inserate în tabela PASAGERI.');

END;

/



# E. Tratarea exceptiilor

Creati un bloc PL/SQL care insereaza înregistrari în tabelul TARI\_2. Sa se trateze excepțiile ce pot apărea.

BEGIN

INSERT INTO TARI\_2 (Id\_tara, Nume\_tara) VALUES ('RO', 'Romania');

INSERT INTO TARI\_2 (Id\_tara, Nume\_tara) VALUES ('US', 'United States');

INSERT INTO TARI\_2 (Id\_tara, Nume\_tara) VALUES ('FR', 'France');

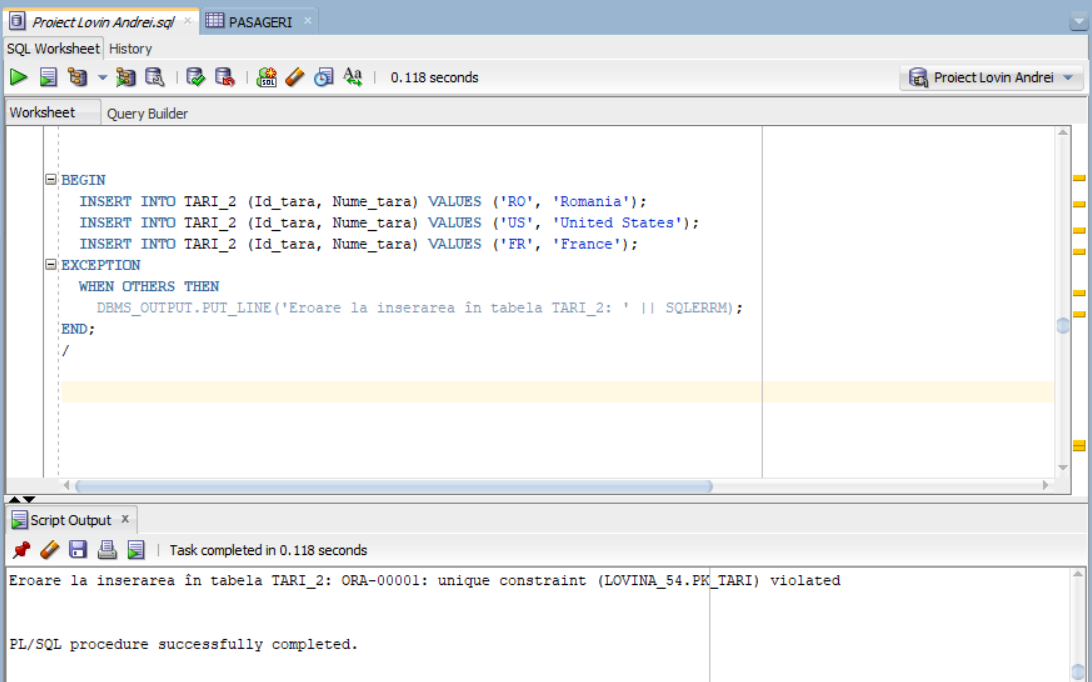
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare la inserarea în tabela TARI\_2: ' || SQLERRM);

END;

/



Creati un bloc PL/SQL care efectueaza o interogare SELECT în tabelul CURSE și gestioneaza posibilele erori care pot aparea în timpul executiei interogarii.

DECLARE

v\_Nume\_zbor CURSE.aeroport\_plecare%TYPE;

BEGIN

SELECT aeroport\_plecare INTO v\_Nume\_zbor

FROM CURSE

WHERE Id\_cursa = 100;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

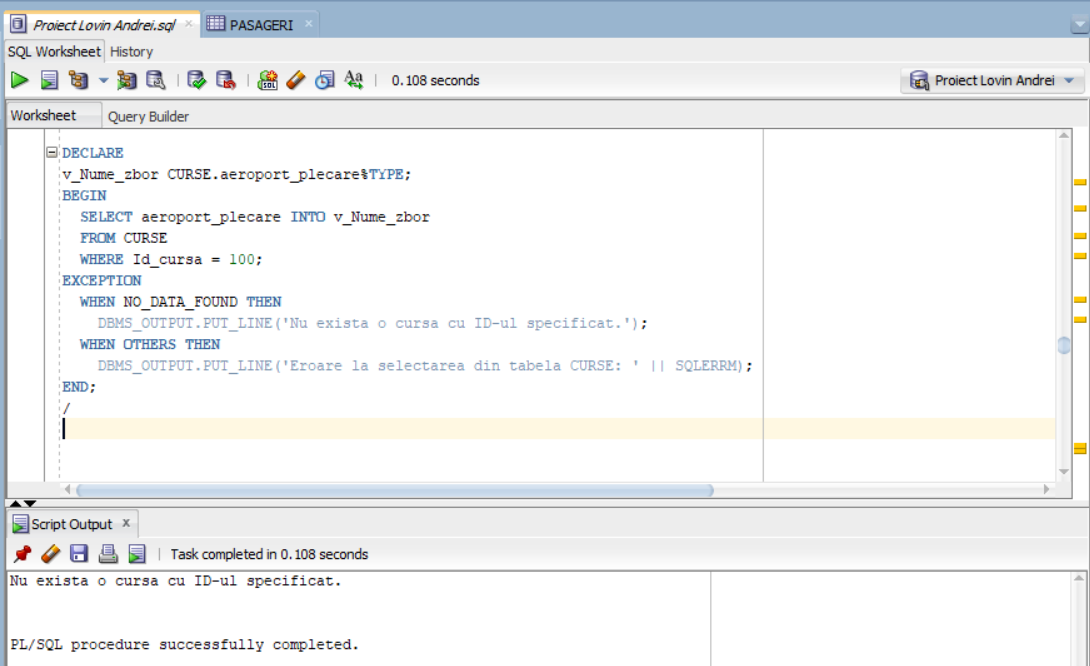
DBMS\_OUTPUT.PUT\_LINE('Nu exista o cursa cu ID-ul specificat.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare la selectarea din tabela CURSE: ' || SQLERRM);

END;

/



Creati un bloc PL/SQL care insereaza inregistrari in mai multe tabele. Sa se trateze exceptiile ce pot aparea.

BEGIN

INSERT INTO TARI\_2 (Id\_tara, Nume\_tara)

VALUES ('RO', 'Romania');

INSERT INTO AEROPORTURI (Id\_aeroport, Nume\_aeroport, Oras, Id\_tara, Cod\_aeroport)

VALUES (1, 'Aeroportul Otopeni', 'Bucuresti', 'RO', 'OTP');

INSERT INTO CURSE (Id\_cursa, Aeroport\_plecare, Aeroport\_sosire, Durata\_cursa, Id\_companie)

VALUES (1, 'OTP', 'CDG', 120, 1);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Eroare: Inserarea exista deja în una dintre tabele.');

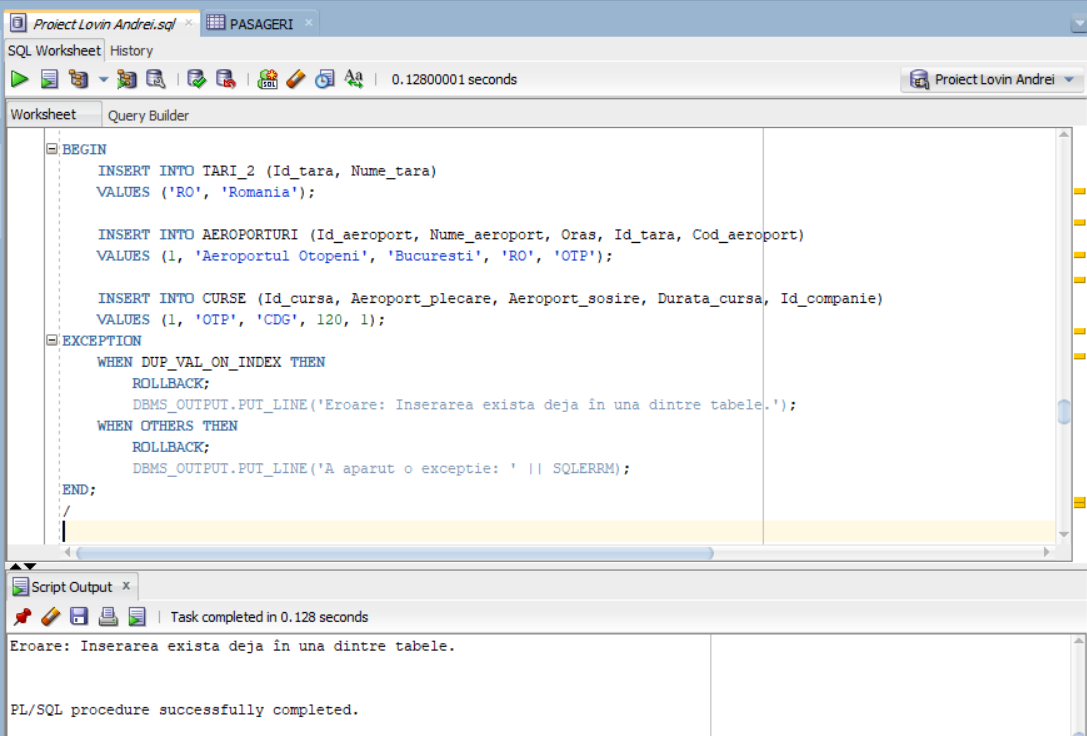
WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('A aparut o exceptie: ' || SQLERRM);

END;

/



Creati un bloc PL/SQL care actualizeaza ID-ul tarii unui aeroport din tabelul AEROPORTURI. Sa se trateze exceptiile ce pot aparea.

DECLARE

v\_id\_aeroport AEROPORTURI.Id\_aeroport%TYPE := 1;

v\_id\_tara AEROPORTURI.Id\_tara%TYPE := 'DE';

BEGIN

UPDATE AEROPORTURI

SET Id\_tara = v\_id\_tara

WHERE Id\_aeroport = v\_id\_aeroport;

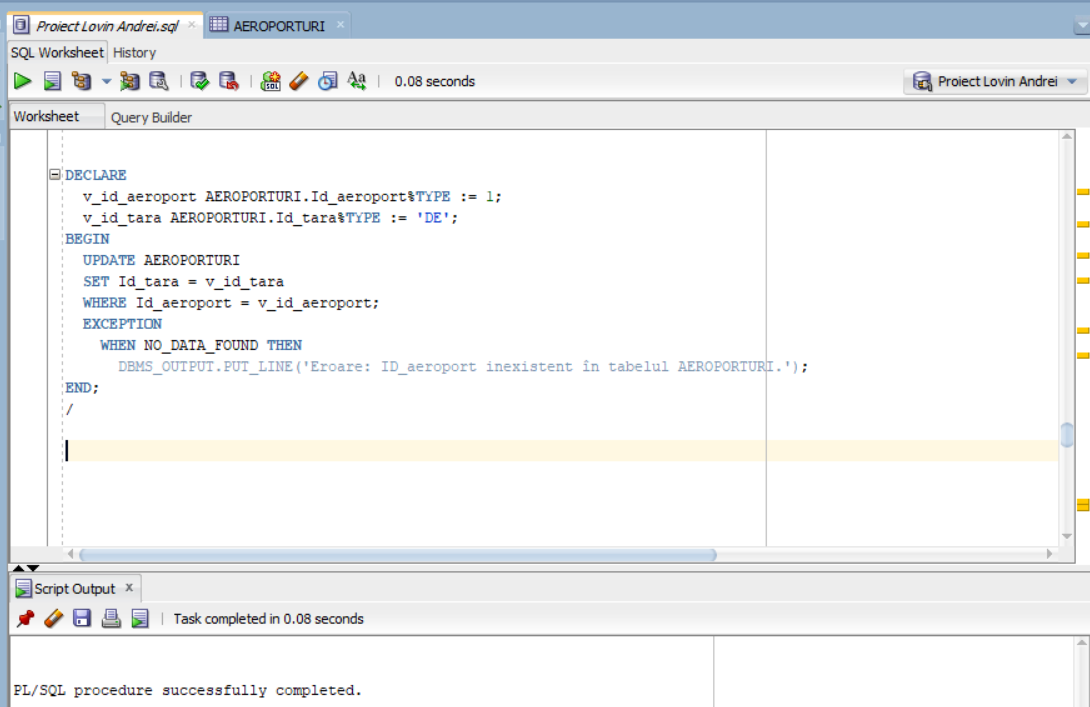
EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ID\_aeroport inexistent în tabelul AEROPORTURI.');

END;

/



Creati un bloc PL/SQL care cauta un membru al unui echipaj dupa ID din tabelul ECHIPAJ si afiseaza numele acestuia. Sa se trateze exceptiile ce pot aparea.

DECLARE

v\_nume\_membru ECHIPAJ.Nume\_membru%TYPE;

BEGIN

SELECT Nume\_membru INTO v\_nume\_membru

FROM ECHIPAJ

WHERE Id\_membru = 111;

DBMS\_OUTPUT.PUT\_LINE('Membru echipaj gasit: ' || v\_nume\_membru);

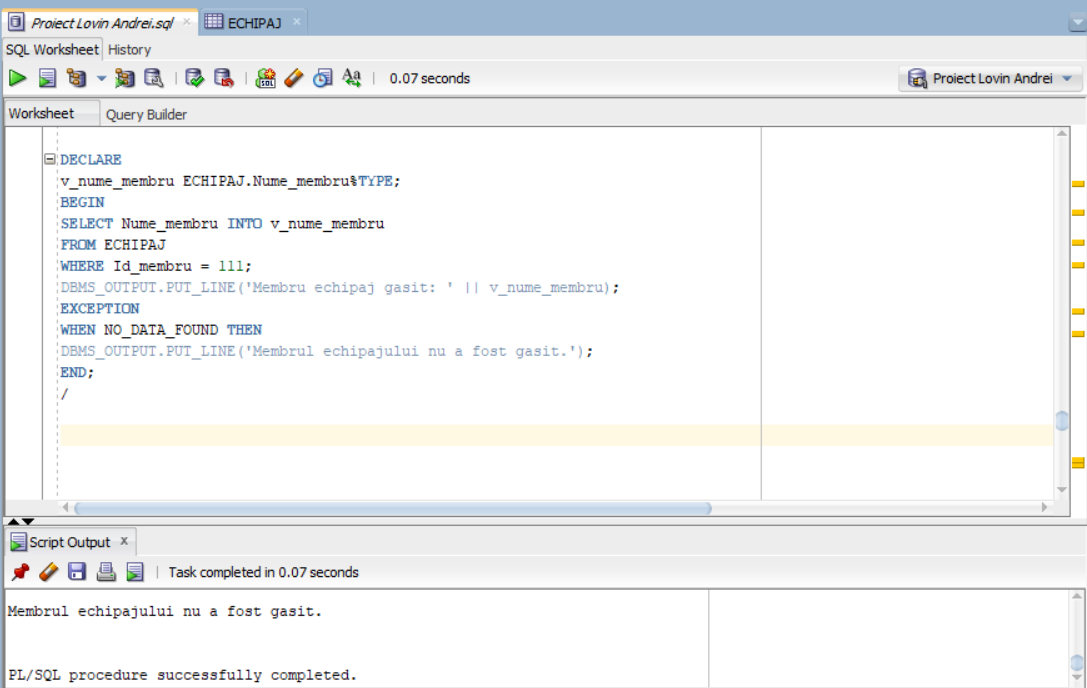
EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Membrul echipajului nu a fost gasit.');

END;

/



# F. Gestionarea cursorilor: impliciti si expliciti

Creati un bloc PL/SQL care afiseaza toate tarile din tabelul TARI\_2 si afiseaza informatii despre aeroporturile din tara respectiva.

DECLARE

CURSOR c\_tari IS

SELECT Id\_tara, Nume\_tara

FROM TARI\_2;

CURSOR c\_aeroporturi(p\_Id\_tara VARCHAR2) IS

SELECT Id\_aeroport, Nume\_aeroport, Oras, Cod\_aeroport

FROM AEROPORTURI

WHERE Id\_tara = p\_Id\_tara;

BEGIN

FOR r IN c\_tari LOOP

DBMS\_OUTPUT.PUT\_LINE('Tara: ' || r.Nume\_tara);

DBMS\_OUTPUT.PUT\_LINE('Aeroporturi:');

FOR rec\_aeroport IN c\_aeroporturi(r.Id\_tara) LOOP

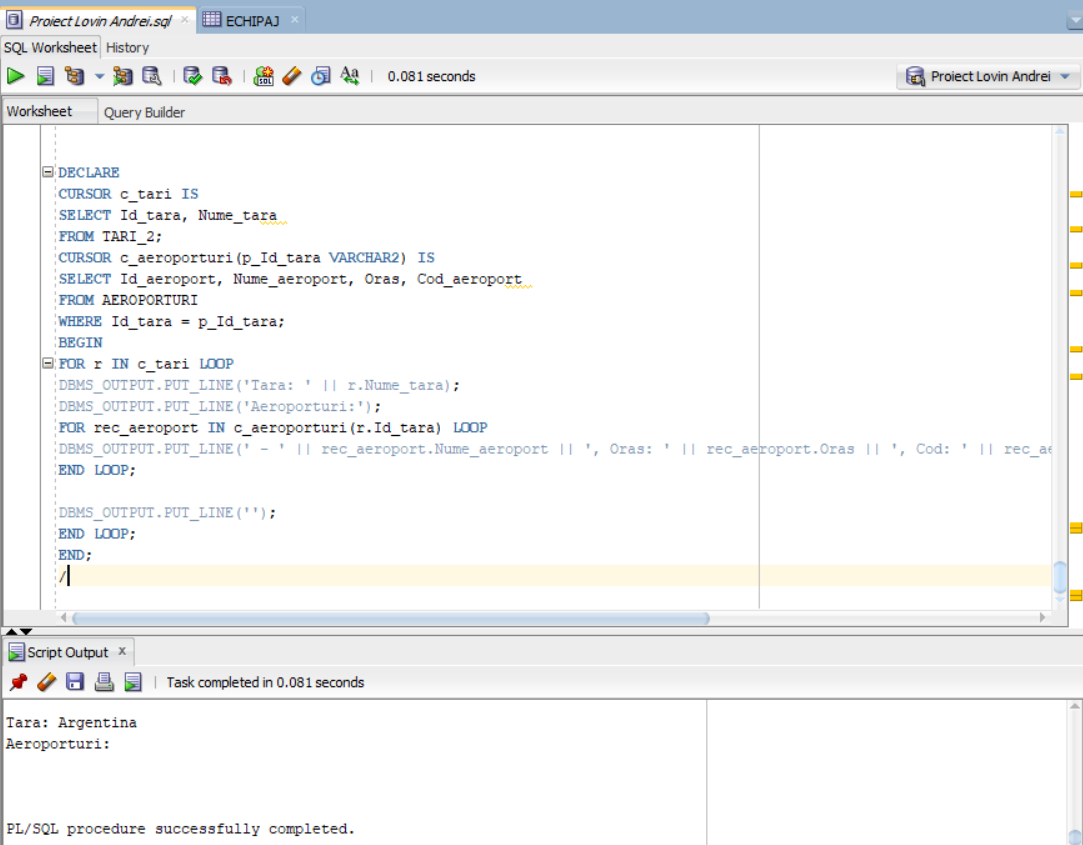
DBMS\_OUTPUT.PUT\_LINE(' - ' || rec\_aeroport.Nume\_aeroport || ', Oras: ' || rec\_aeroport.Oras || ', Cod: ' || rec\_aeroport.Cod\_aeroport);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

END;



Creati un bloc PL/SQL care modifica una sau mai multe inregistrari din tabelului TARI\_2 si afiseaza numarul de modificari care au fost realizate.

DECLARE

v\_count NUMBER;

BEGIN

UPDATE TARI\_2 SET Nume\_tara = 'Bulgaria' WHERE Id\_tara = 'RO';

v\_count := SQL%ROWCOUNT;

IF v\_count > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Au fost actualizate ' || v\_count || ' înregistrari.');

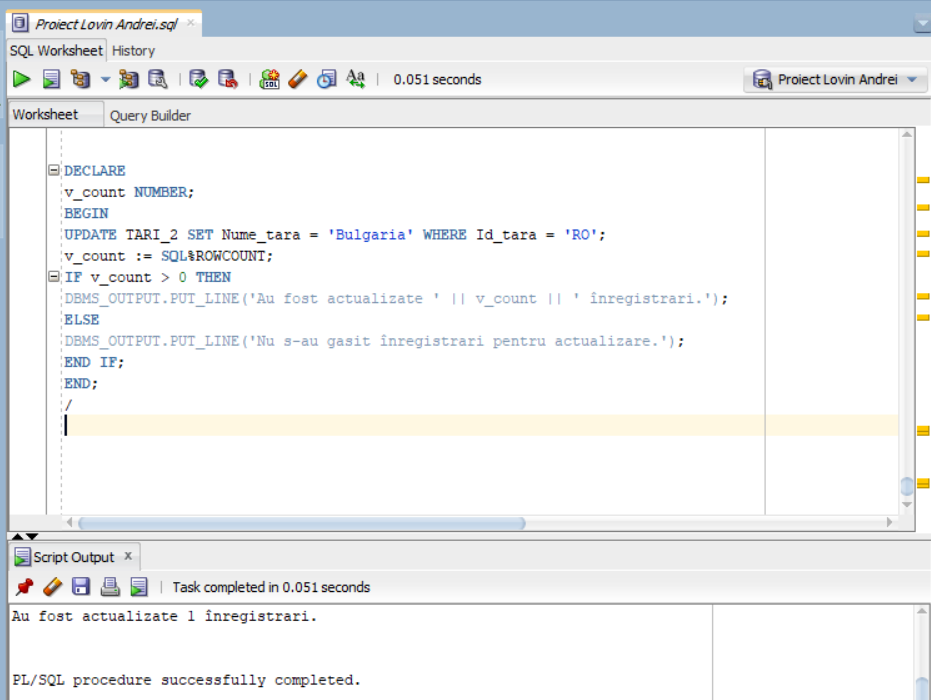
ELSE

DBMS\_OUTPUT.PUT\_LINE('Nu s-au gasit înregistrari pentru actualizare.');

END IF;

END;

/



# G. Functii, proceduri, includerea acestora în pachete.

Creati un bloc PL/SQL care calculeaza si returneaza durata totala a tuturor curselor de zbor pentru o anumita companie aeriana.

CREATE OR REPLACE FUNCTION calcul\_durata\_totala(p\_id\_companie IN COMPANII\_AERIENE.Id\_companie%TYPE)

RETURN NUMBER IS

v\_durata\_totala NUMBER(10) := 0;

BEGIN

SELECT SUM(c.Durata\_cursa) INTO v\_durata\_totala

FROM COMPANII\_AERIENE ca

JOIN CURSE c ON ca.Id\_companie = c.Id\_companie

WHERE ca.Id\_companie = p\_id\_companie;

RETURN v\_durata\_totala;

END;

/

DECLARE

v\_id\_companie COMPANII\_AERIENE.Id\_companie%TYPE := 1;

v\_durata\_totala NUMBER(10);

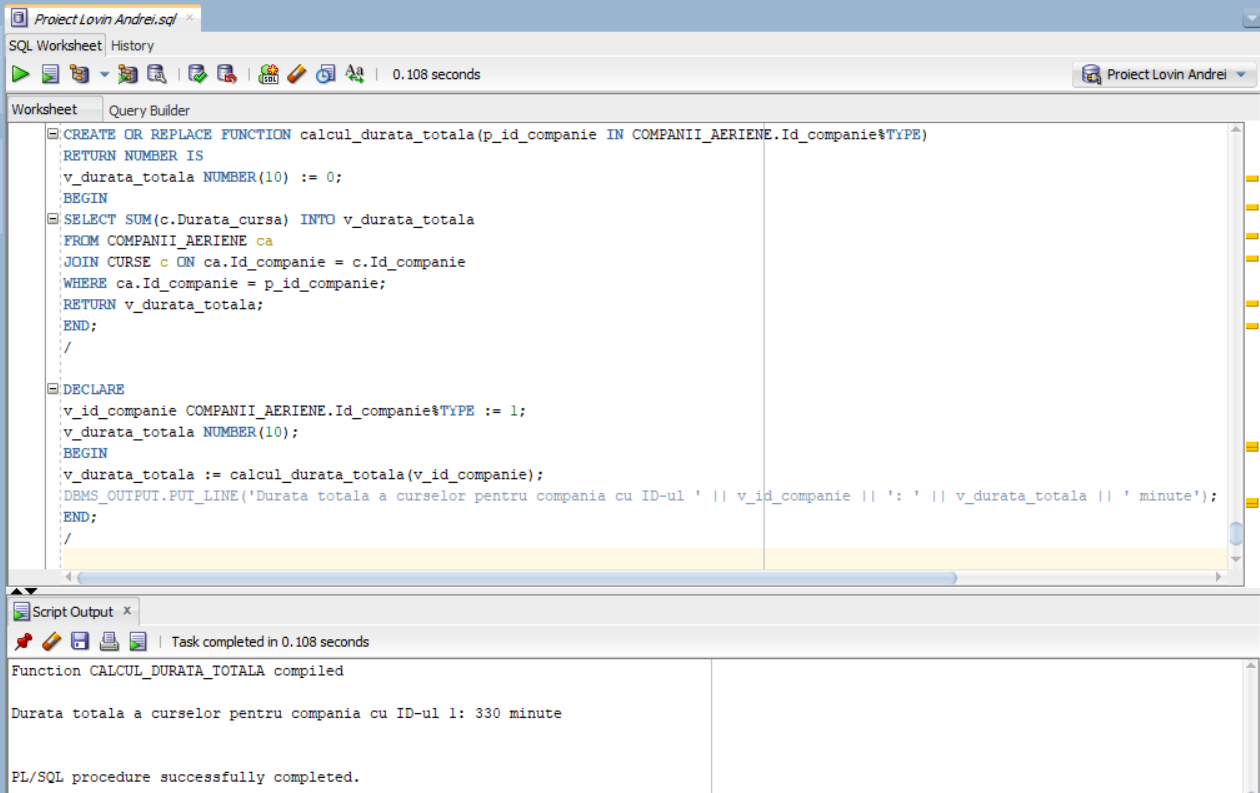
BEGIN

v\_durata\_totala := calcul\_durata\_totala(v\_id\_companie);

DBMS\_OUTPUT.PUT\_LINE('Durata totala a curselor pentru compania cu ID-ul ' || v\_id\_companie || ': ' || v\_durata\_totala || ' minute');

END;

/



Creati un bloc PL/SQL care calculeaza si returneaza numarul total de bilete achizitionate de un pasager.

CREATE OR REPLACE FUNCTION numar\_bilete(p\_id\_pasager IN PASAGERI.Id\_pasager%TYPE)

RETURN NUMBER IS

v\_numar\_bilete NUMBER(10) := 0;

BEGIN

SELECT COUNT(\*) INTO v\_numar\_bilete

FROM BILETE

WHERE Id\_pasager = p\_id\_pasager;

RETURN v\_numar\_bilete;

END;

/

DECLARE

v\_id\_pasager PASAGERI.Id\_pasager%TYPE := 2;

v\_numar\_bilete NUMBER(10);

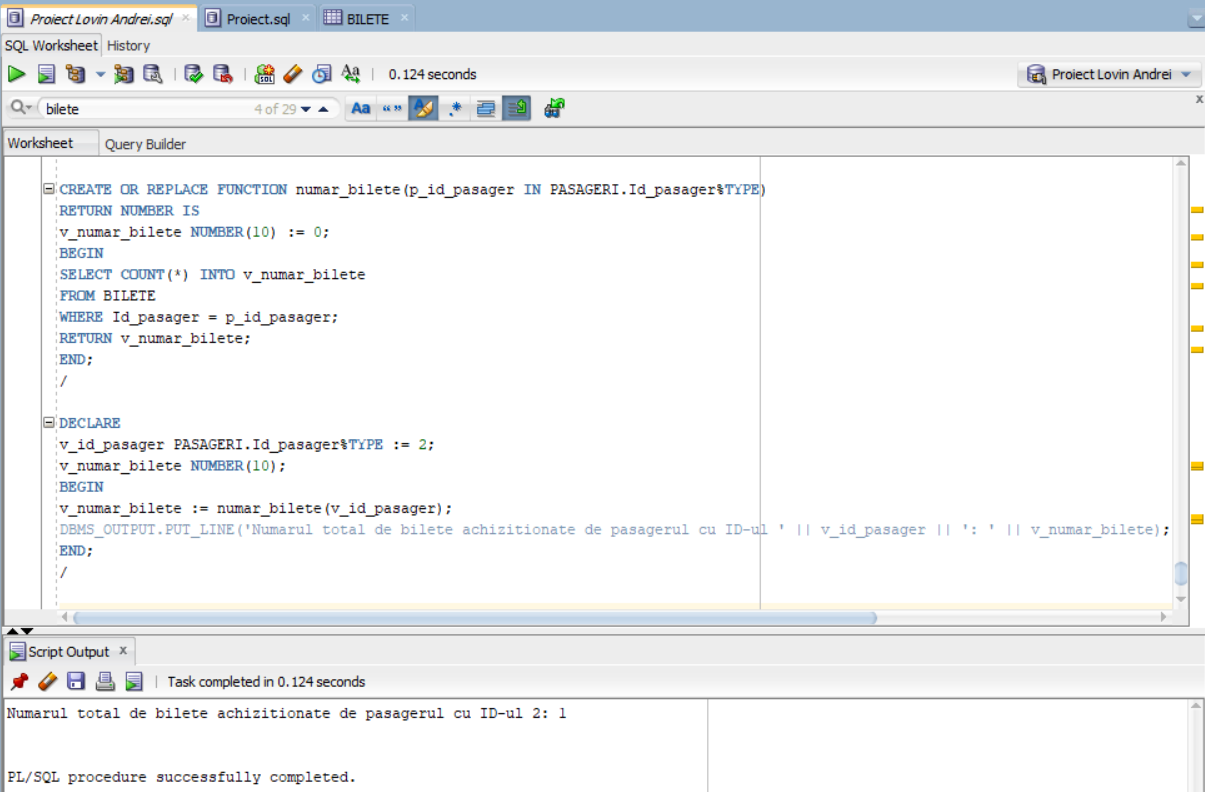
BEGIN

v\_numar\_bilete := numar\_bilete(v\_id\_pasager);

DBMS\_OUTPUT.PUT\_LINE('Numarul total de bilete achizitionate de pasagerul cu ID-ul ' || v\_id\_pasager || ': ' || v\_numar\_bilete);

END;

/



Creati un bloc PL/SQL care actualizeaza experienta unui membru al unui echipajului si returneaza numarul total de membrii ai tuturor echipajelor cu o experienta mai mare decat cea specificata.

CREATE OR REPLACE FUNCTION actualizare\_experienta(p\_id\_membru IN ECHIPAJ.Id\_membru%TYPE, p\_experienta\_noua IN ECHIPAJ.Experienta%TYPE)

RETURN NUMBER IS

v\_numar\_membri\_experienta\_mare NUMBER(10) := 0;

BEGIN

UPDATE ECHIPAJ

SET Experienta = p\_experienta\_noua

WHERE Id\_membru = p\_id\_membru;

SELECT COUNT(\*) INTO v\_numar\_membri\_experienta\_mare

FROM ECHIPAJ

WHERE Experienta > p\_experienta\_noua;

RETURN v\_numar\_membri\_experienta\_mare;

END;

/

DECLARE

v\_id\_membru ECHIPAJ.Id\_membru%TYPE := 1;

v\_experienta\_noua ECHIPAJ.Experienta%TYPE := 7;

v\_numar\_membri\_experienta\_mare NUMBER(10);

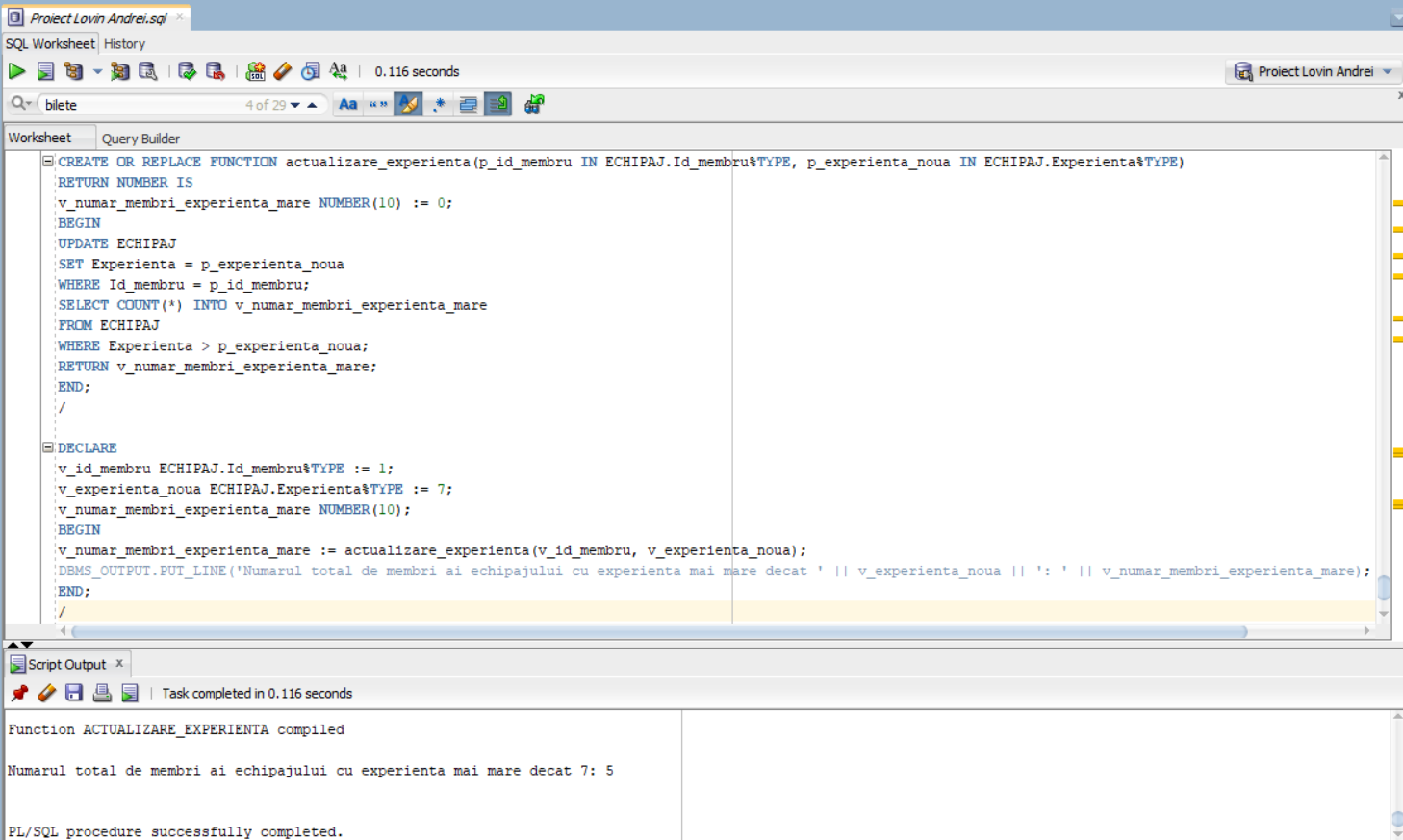
BEGIN

v\_numar\_membri\_experienta\_mare :=actualizare\_experienta(v\_id\_membru, v\_experienta\_noua);

DBMS\_OUTPUT.PUT\_LINE('Numarul total de membri ai echipajului cu experienta mai mare decat ' || v\_experienta\_noua || ': ' || v\_numar\_membri\_experienta\_mare);

END;

/



Creati un bloc PL/SQL care insereaza un nou pasager in tabelul PASAGERI si afiseaza un mesaj.

CREATE OR REPLACE PROCEDURE inserare\_pasager(

p\_id\_pasager IN PASAGERI.Id\_pasager%TYPE,

p\_nume\_pasager IN PASAGERI.Nume\_pasager%TYPE,

p\_prenume\_pasager IN PASAGERI.Prenume\_pasager%TYPE,

p\_varsta\_pasager IN PASAGERI.Varsta\_pasager%TYPE

) IS

BEGIN

INSERT INTO PASAGERI(Id\_pasager, Nume\_pasager, Prenume\_pasager, Varsta\_pasager)

VALUES (p\_id\_pasager, p\_nume\_pasager, p\_prenume\_pasager, p\_varsta\_pasager);

DBMS\_OUTPUT.PUT\_LINE('Pasagerul cu ID-ul ' || p\_id\_pasager || ' a fost inserat cu succes.');

END;

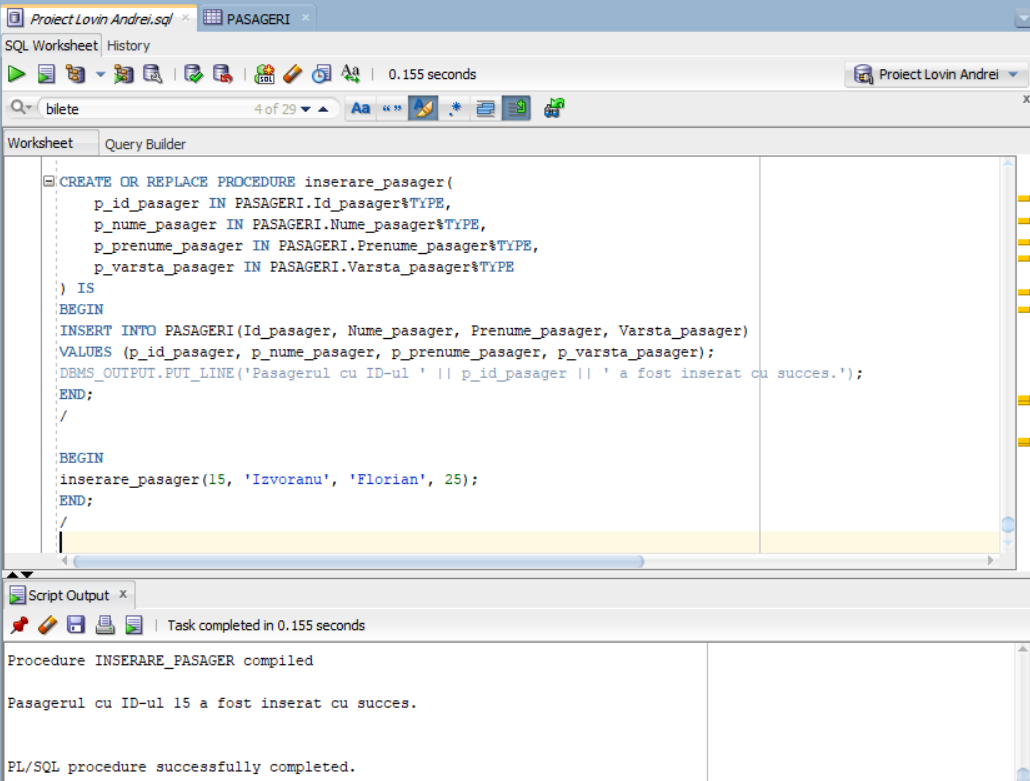
/

BEGIN

inserare\_pasager(15, 'Izvoranu', 'Florian', 25);

END;

/



Creati un bloc PL/SQL care actualizeaza durata unei curse de zbor din tabelul CURSE pe baza ID – ului cursei si afiseaza un mesaj.

CREATE OR REPLACE PROCEDURE actualizare\_durata\_cursa(

p\_id\_cursa IN CURSE.Id\_cursa%TYPE,

p\_durata\_cursa IN CURSE.Durata\_cursa%TYPE

) IS

BEGIN

UPDATE CURSE

SET Durata\_cursa = p\_durata\_cursa

WHERE Id\_cursa = p\_id\_cursa;

DBMS\_OUTPUT.PUT\_LINE('Durata cursei cu ID-ul ' || p\_id\_cursa || ' a fost actualizată cu succes.');

END;

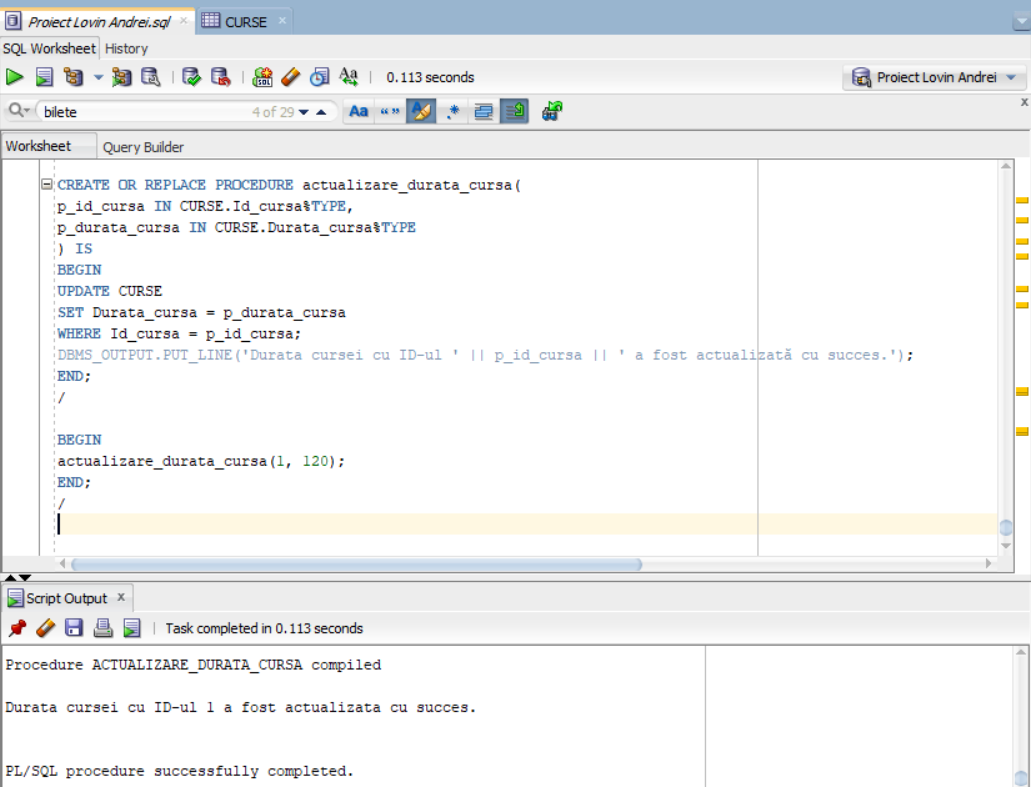
/

BEGIN

actualizare\_durata\_cursa(1, 120);

END;

/



Creati un bloc PL/SQL care sterge un membru al unui echipaj din tabelul ECHIPAJ pe baza ID – ului si afiseaza un mesaj.

CREATE OR REPLACE PROCEDURE stergere\_membru\_echipaj(

p\_id\_membru IN ECHIPAJ.Id\_membru%TYPE

) IS

BEGIN

DELETE FROM ECHIPAJ

WHERE Id\_membru = p\_id\_membru;

DBMS\_OUTPUT.PUT\_LINE('Membrul echipajului cu ID-ul ' || p\_id\_membru || ' a fost sters cu succes.');

END;

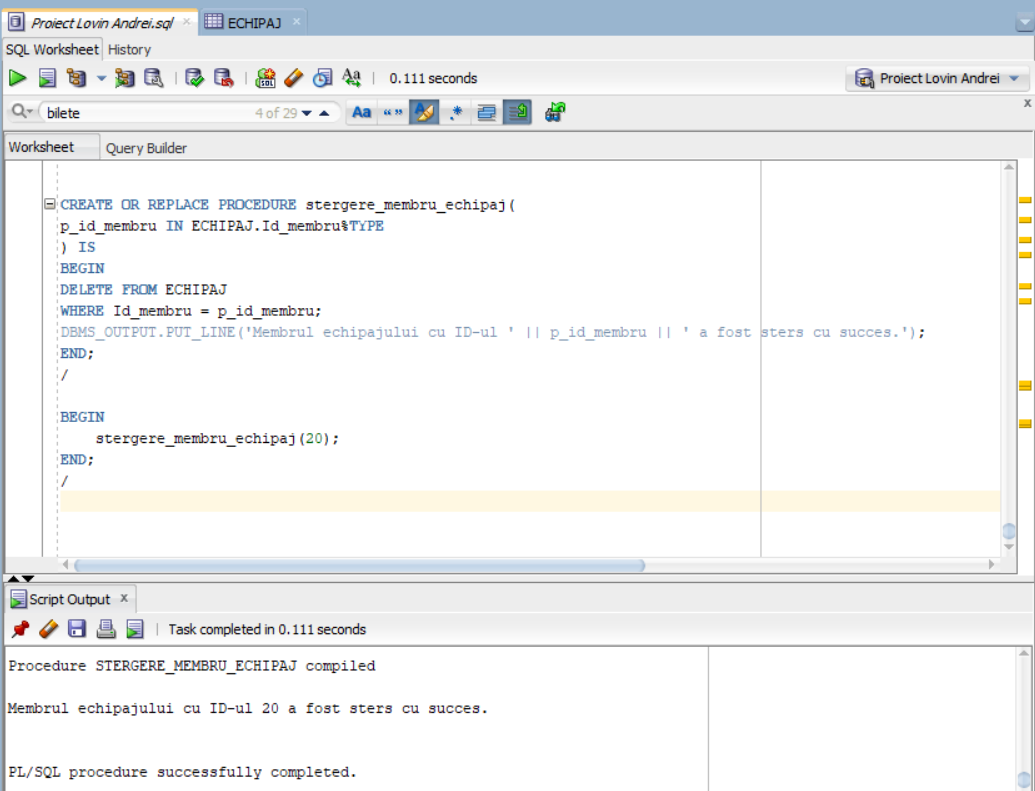
/

BEGIN

stergere\_membru\_echipaj(20);

END;

/



Creati un bloc PL/SQL care calculeaza durata unui zbor pe baza ID ului, insereaza un nou membru intr-un echipaj si determina numarul de bilete vandute pentru un anumit zbor, utilizand functii si proceduri ce vor fi incluse intr-un pachet.

CREATE OR REPLACE PACKAGE avioane\_pkg IS

FUNCTION calcul\_durata\_zbor(p\_id\_zbor IN ZBORURI.Id\_zbor%TYPE) RETURN NUMBER;

PROCEDURE inregistrare\_membru\_echipaj(

p\_id\_echipaj IN ECHIPAJ.Id\_echipaj%Type,

p\_id\_zbor IN ECHIPAJ.Id\_zbor%TYPE,

p\_id\_membru IN ECHIPAJ.Id\_membru%TYPE,

p\_nume\_membru IN ECHIPAJ.Nume\_membru%TYPE,

p\_functie IN ECHIPAJ.Functie%TYPE,

p\_experienta IN ECHIPAJ.Experienta%TYPE,

p\_id\_superior IN ECHIPAJ.Id\_superior%TYPE

);

FUNCTION numar\_bilete\_vandute(p\_id\_zbor IN ZBORURI.Id\_zbor%TYPE) RETURN NUMBER;

END avioane\_pkg;

/

CREATE OR REPLACE PACKAGE BODY avioane\_pkg IS

FUNCTION calcul\_durata\_zbor(p\_id\_zbor IN ZBORURI.Id\_zbor%TYPE) RETURN NUMBER IS

v\_durata NUMBER;

BEGIN

SELECT Durata\_cursa INTO v\_durata

FROM CURSE

WHERE Id\_cursa = (SELECT Id\_cursa FROM ZBORURI WHERE Id\_zbor = p\_id\_zbor);

RETURN v\_durata;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN NULL;

END calcul\_durata\_zbor;

PROCEDURE inregistrare\_membru\_echipaj(

p\_id\_echipaj IN ECHIPAJ.Id\_echipaj%Type,

p\_id\_zbor IN ECHIPAJ.Id\_zbor%TYPE,

p\_id\_membru IN ECHIPAJ.Id\_membru%TYPE,

p\_nume\_membru IN ECHIPAJ.Nume\_membru%TYPE,

p\_functie IN ECHIPAJ.Functie%TYPE,

p\_experienta IN ECHIPAJ.Experienta%TYPE,

p\_id\_superior IN ECHIPAJ.Id\_superior%TYPE

) IS

BEGIN

INSERT INTO ECHIPAJ(Id\_echipaj, Id\_zbor, Id\_membru, Nume\_membru, Functie, Experienta, Id\_superior)

VALUES (p\_id\_echipaj, p\_id\_zbor, p\_id\_membru, p\_nume\_membru, p\_functie, p\_experienta, p\_id\_superior);

END inregistrare\_membru\_echipaj;

FUNCTION numar\_bilete\_vandute(p\_id\_zbor IN ZBORURI.Id\_zbor%TYPE) RETURN NUMBER IS

v\_numar\_bilete NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_numar\_bilete

FROM BILETE

WHERE Id\_zbor = p\_id\_zbor;

RETURN v\_numar\_bilete;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN NULL;

END numar\_bilete\_vandute;

END avioane\_pkg;

/

BEGIN

DECLARE

v\_id\_zbor ZBORURI.Id\_zbor%TYPE := 7;

v\_durata\_zbor NUMBER;

BEGIN

v\_durata\_zbor := avioane\_pkg.calcul\_durata\_zbor(v\_id\_zbor);

DBMS\_OUTPUT.PUT\_LINE('Durata zborului cu ID ' || v\_id\_zbor || ' este: ' || v\_durata\_zbor || ' minute.');

END;

BEGIN

avioane\_pkg.inregistrare\_membru\_echipaj(6, 6, 21, 'Mircea', 'pilot', 15, null);

DBMS\_OUTPUT.PUT\_LINE('Membrul echipajului a fost înregistrat cu succes.');

END;

DECLARE

v\_id\_zbor ZBORURI.Id\_zbor%TYPE := 3;

v\_numar\_bilete NUMBER;

BEGIN

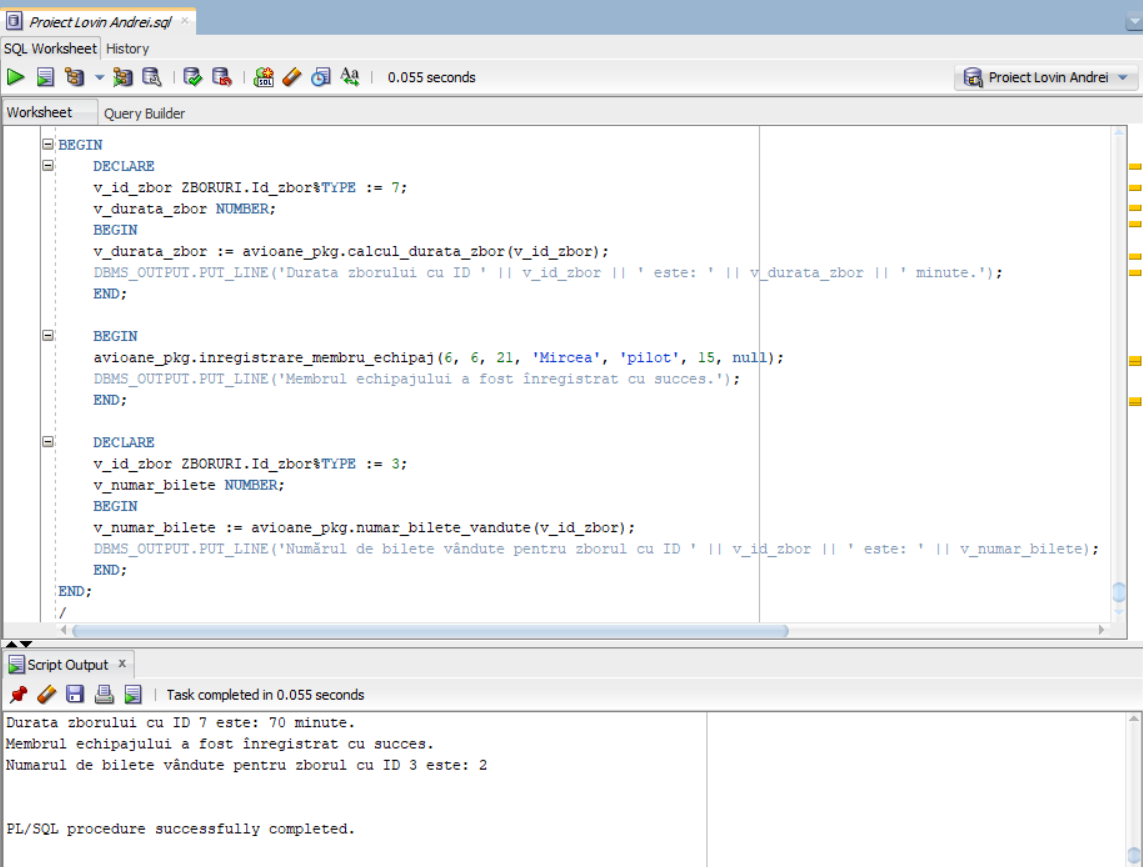
v\_numar\_bilete := avioane\_pkg.numar\_bilete\_vandute(v\_id\_zbor);

DBMS\_OUTPUT.PUT\_LINE('Numarul de bilete vandute pentru zborul cu ID ' || v\_id\_zbor || ' este: ' || v\_numar\_bilete);

END;

END;

/



# H. Declansatori la nivel de instructiune si de rand

Creati un bloc PL/SQL care activeaza un declansator inaitea inserarii unei noi intregistrari in tabelul PASAGERI.

CREATE OR REPLACE TRIGGER trg\_before\_insert\_pasageri

BEFORE INSERT ON PASAGERI

FOR EACH ROW

DECLARE

v\_full\_name VARCHAR2(200);

BEGIN

v\_full\_name := :NEW.Nume\_pasager || ' ' || :NEW.Prenume\_pasager;

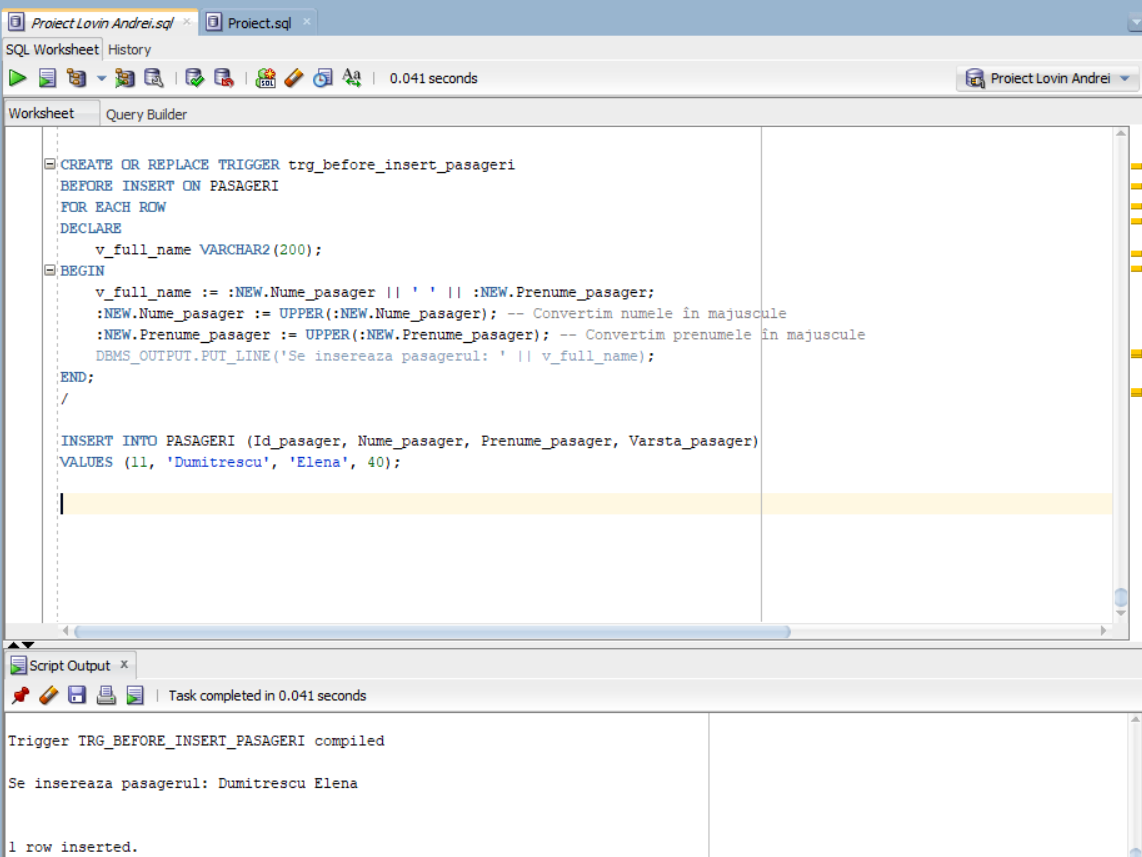
:NEW.Nume\_pasager := UPPER(:NEW.Nume\_pasager); -- Convertim numele în majuscule

:NEW.Prenume\_pasager := UPPER(:NEW.Prenume\_pasager); -- Convertim prenumele în majuscule

DBMS\_OUTPUT.PUT\_LINE('Se insereaza pasagerul: ' || v\_full\_name);

END;

/



Creati un bloc PL/SQL care activeaza un declansator dupa stergerea unei inregistrai din tabelul CURSE.

CREATE OR REPLACE TRIGGER trg\_delete\_cursa

AFTER DELETE ON CURSE

FOR EACH ROW

DECLARE

v\_message VARCHAR2(100);

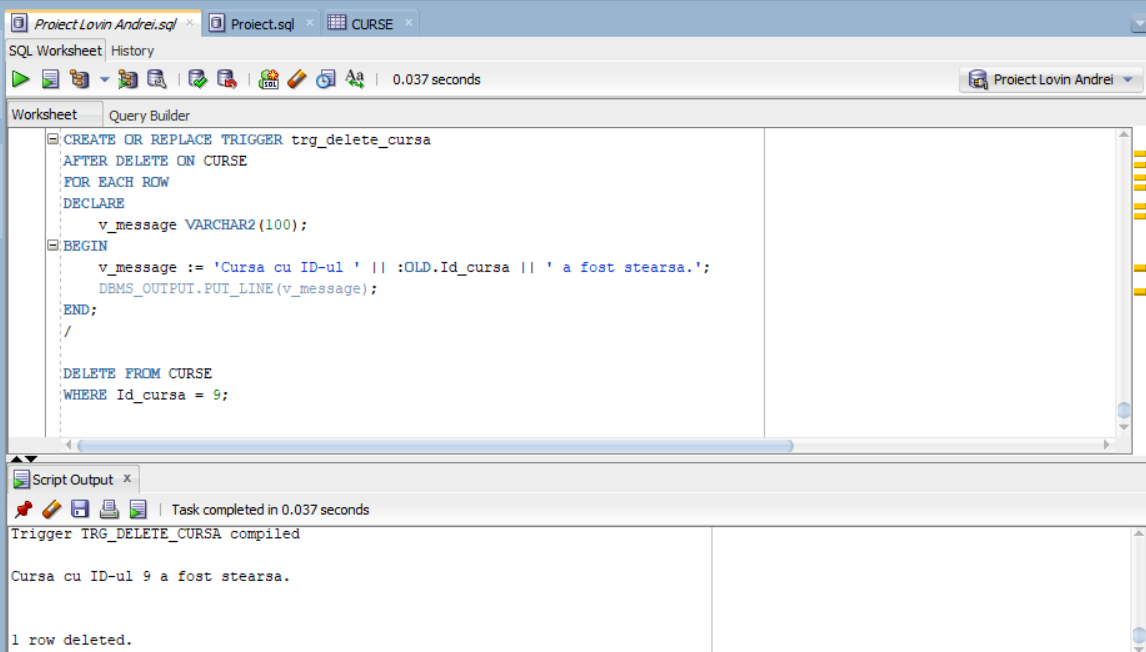
BEGIN

v\_message := 'Cursa cu ID-ul ' || :OLD.Id\_cursa || ' a fost stearsa.';

DBMS\_OUTPUT.PUT\_LINE(v\_message);

END;

/



Creati un bloc PL/SQL care activeaza un declansator ce nu te mai poata lasa sa inserezi sau sa modifici inregistrari din tabelul ZBORURI dupa ora 21:30.

CREATE OR REPLACE TRIGGER trg\_check\_zboruri\_time

BEFORE INSERT OR UPDATE ON ZBORURI

FOR EACH ROW

DECLARE

v\_ora\_curenta NUMBER := TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24'));

v\_minut\_curent NUMBER := TO\_NUMBER(TO\_CHAR(SYSDATE, 'MI'));

BEGIN

IF v\_ora\_curenta >= 22 OR (v\_ora\_curenta = 21 AND v\_minut\_curent > 30) THEN

IF INSERTING THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Nu se pot introduce zboruri după ora 21:30.');

ELSIF UPDATING THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Nu se pot modifica zborurile după ora 21:30.');

END IF;

END IF;

END;

/



Creati un bloc PL/SQL care activeaza un declansator ce nu te mai poata lasa sa stergi inregistrari din tabelul PASAGERI inafara intervalului 8:00 – 23:00.

CREATE OR REPLACE TRIGGER trg\_delete\_pasageri

BEFORE DELETE ON PASAGERI

FOR EACH ROW

DECLARE

v\_current\_hour NUMBER := TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24'));

BEGIN

IF v\_current\_hour >= 23 OR v\_current\_hour < 8 THEN

IF DELETING THEN

DBMS\_OUTPUT.PUT\_LINE('Nu se pot sterge pasagerii în afara intervalului orar 08:00 - 23:00.');

END IF;

END IF;

END;

/

