

Proiect SGBD

Nume Student: Atodiresei Irina

Grupa: 1053, Seria C

Conf. univ. dr. Alexandra Corbea

Descrierea bazei de date

Proiectul are ca scop analizarea activității unei biblioteci.

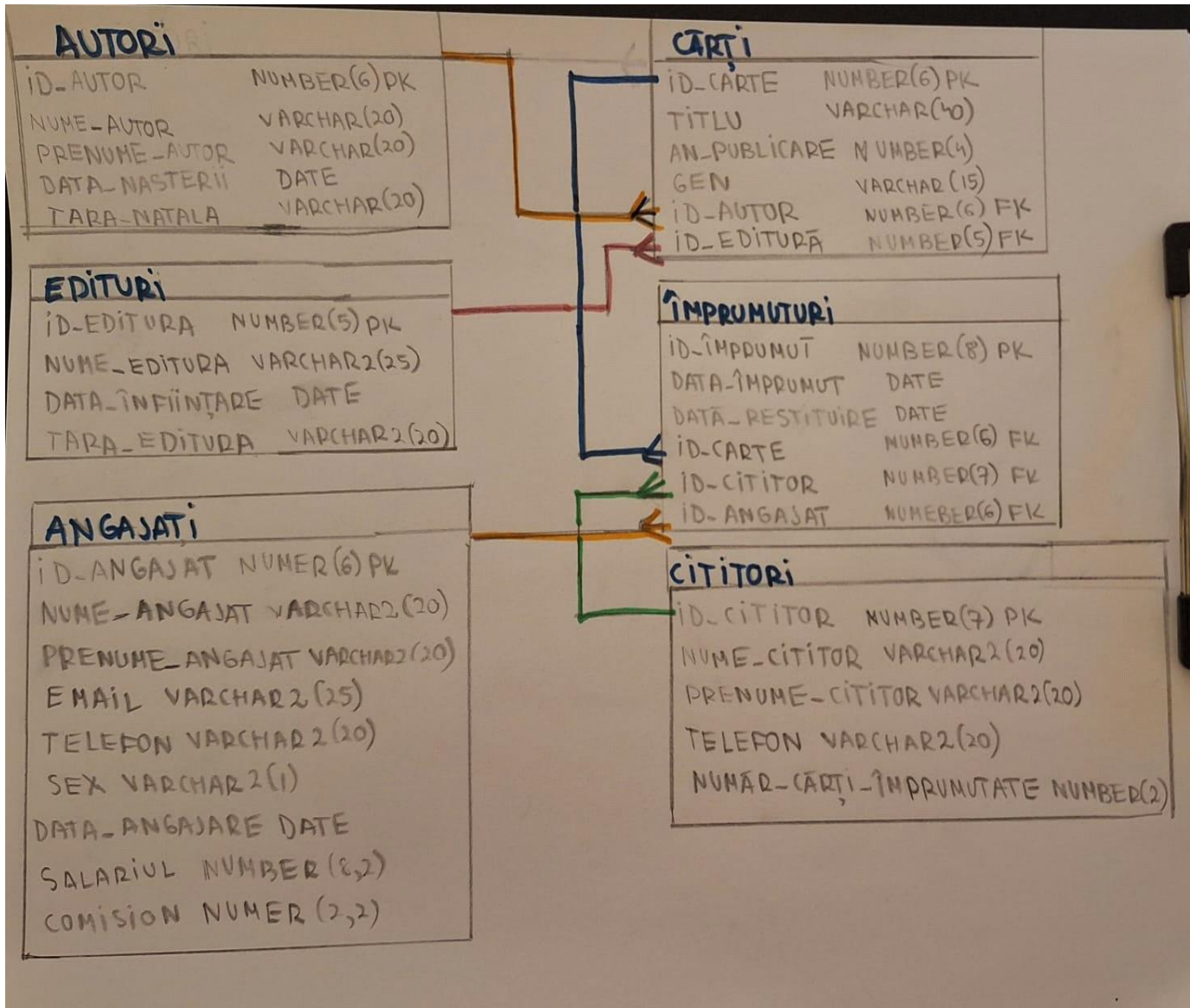
Tema aleasă pentru baza de date a unei biblioteci este concepută pentru a gestiona eficient informațiile legate de cărți, autori, edituri, cititori, angajați și împrumuturi în cadrul unei instituții bibliotecare. Fiecare tabelă din această bază de date contribuie la organizarea și accesarea ușoară a datelor.

- Tabela Cărți: Stochează detalii despre cărți, inclusiv titlul, autorul, editura, anul publicării și genul, oferind astfel informații esențiale despre fiecare carte din colecție. (fiecare carte având id-ul unic și cheile externe id_autor și id_editura)
- Tabela Scriitori: Conține informații despre autori, precum numele și prenumele acestora, data nasterii și țara natală, permitând gestionarea și căutarea simplă a autorilor. (fiecare scriitor având id-ul unic)
- Tabela Edituri: Păstrează detalii despre edituri, inclusiv numele, anul înființării și țara de proveniență, pentru a urmări cărțile publicate de diferite edituri. (fiecare editura având id-ul unic)
- Tabela Cititori: Menține informații despre cititori, precum nume, prenume, numărul de telefon, numărul de cărți împrumutate, permițând bibliotecii să țină evidența cititorilor săi. (fiecare cititor având id-ul unic)
- Tabela Angajați: Gestionează informații despre angajați, precum nume, prenume, email, telefon, sex, data angajării, salariul, comisionul și id-ul manager-ului căruia sunt subordonați, pentru a ține evidența personalului bibliotecii. (fiecare angajat având id-ul unic)
- Tabela Împrumuturi: Monitorizează tranzacțiile de împrumut, inclusiv cărțile împrumutate, cititorii implicați, angajații implicați, precum și datele de împrumut și restituire. (fiecare împrumut având id-ul unic și cheile externe id_carte, id_cititor, id_angajat)

Prin relațiile dintre aceste tabele, sistemul permite bibliotecii să gestioneze eficient colecția de cărți, să țină evidența autorilor și editurilor asociate, să monitorizeze cititorii și angajații și să urmărească tranzacțiile de împrumut.

2.Schema

(în SQL DEVELOPER tabelele au denumirea cu atodireseirina_ în față)



TEMA 1

-- 1. Afişarea numărului total de cărți disponibile (cursor implicit+structura de control if..then..else)

DECLARE

v_total_carti NUMBER;

BEGIN

SELECT COUNT(*) INTO v_total_carti FROM atodireseiriina_carti;

IF v_total_carti > 100 THEN

DBMS_OUTPUT.PUT_LINE('Biblioteca are peste 100 de carti.');

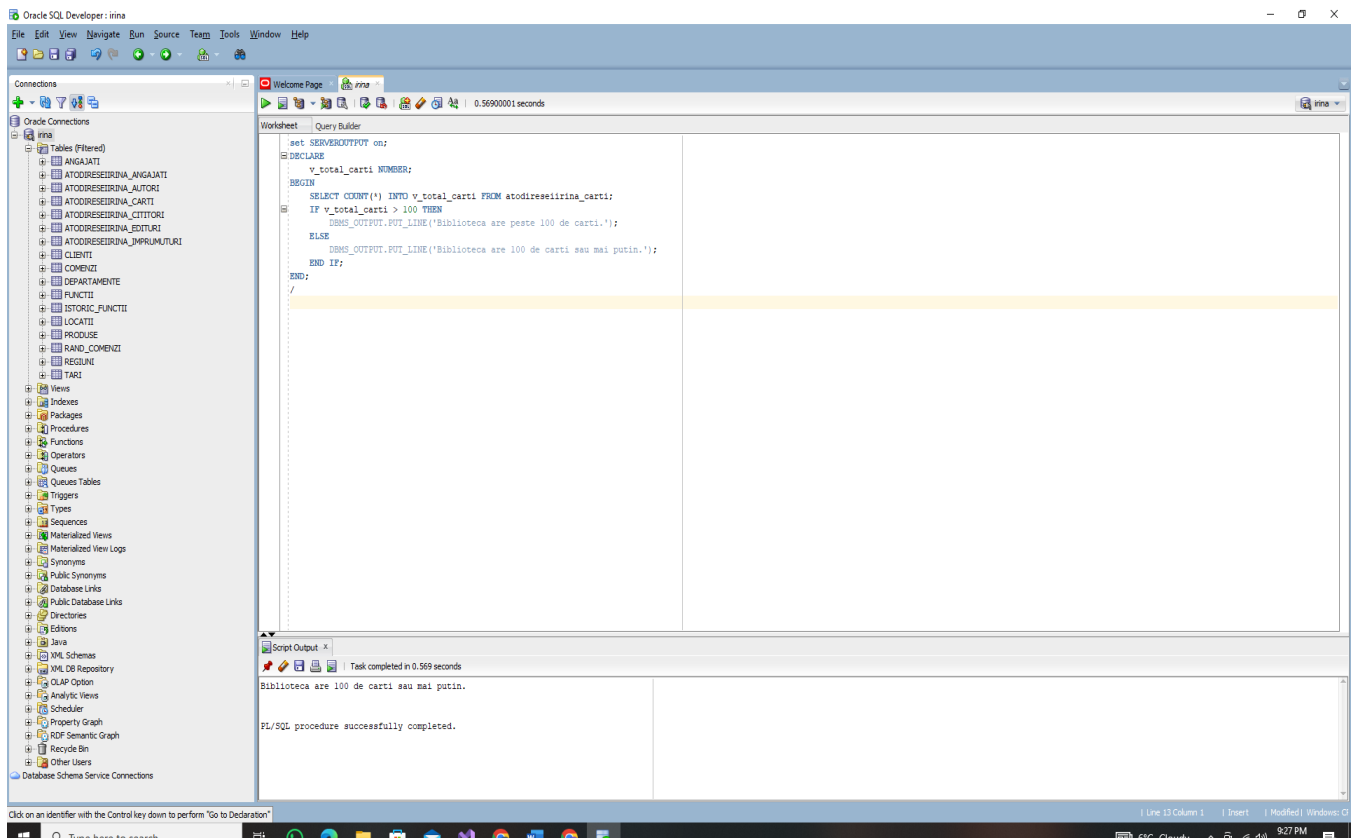
ELSE

DBMS_OUTPUT.PUT_LINE('Biblioteca are 100 de carti sau mai putin.');

END IF;

END;

/



-- 2. Creșterea salariului angajaților cu vechime mai mare de 5 ani (CASE+CORSOR EXPLICIT)

DECLARE

CORSOR c_angajati IS

SELECT id_angajat, salariu, data_angajare FROM atodireseiirina_angajati;

v_ani NUMBER;

BEGIN

FOR v_angajat IN c_angajati LOOP

v_ani := TRUNC(MONTHS_BETWEEN(SYSDATE, v_angajat.data_angajare) / 12);

CASE

WHEN v_ani > 5 THEN

UPDATE atodireseiirina_angajati

SET salariu = v_angajat.salariu * 1.10

WHERE id_angajat = v_angajat.id_angajat;

DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || ' are vechime de ' || v_ani || ' ani și
a primit +10%.');

WHEN v_ani <= 5 THEN

UPDATE atodireseiirina_angajati

SET salariu = v_angajat.salariu * 1.05

WHERE id_angajat = v_angajat.id_angajat;

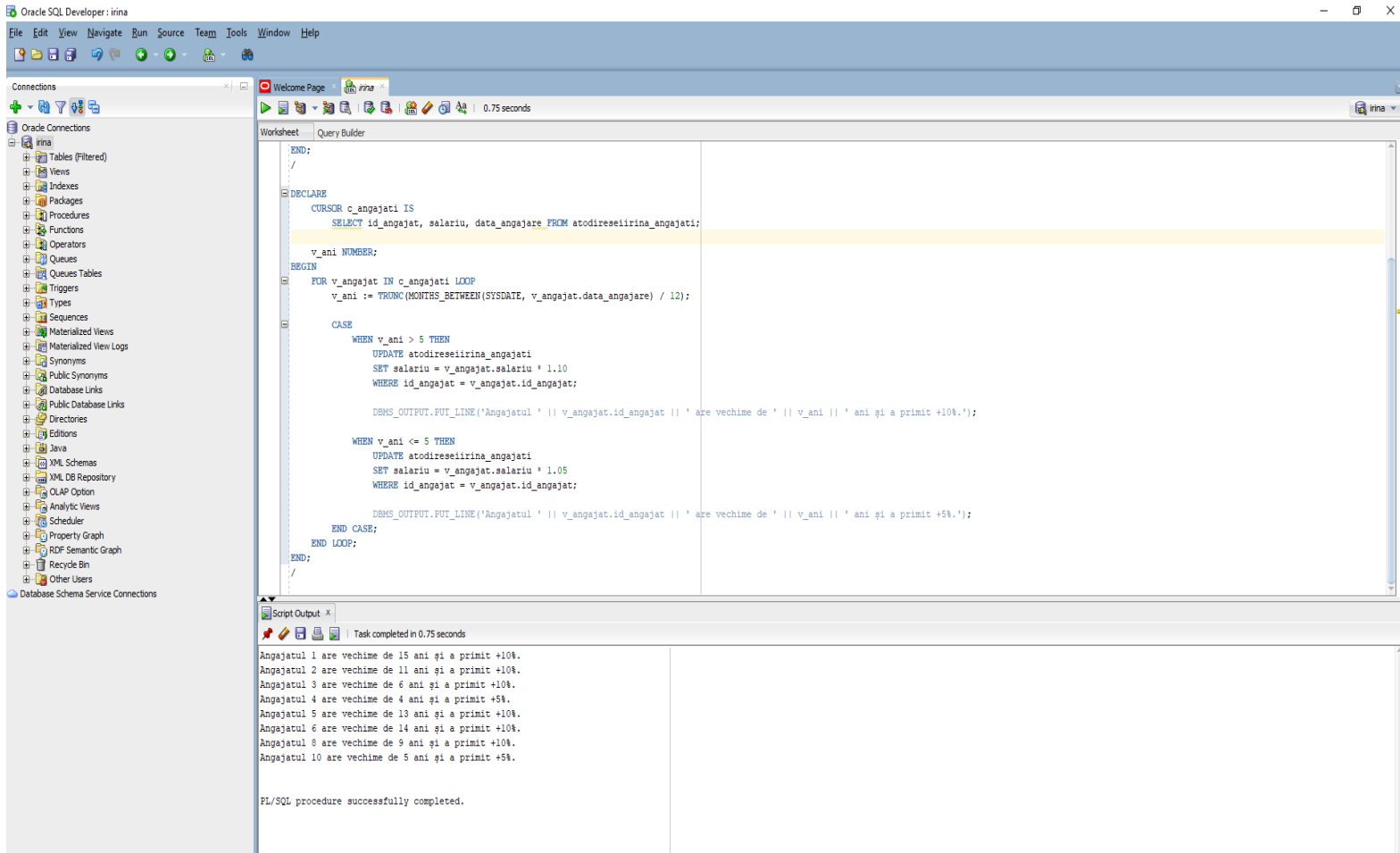
DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || ' are vechime de ' || v_ani || ' ani și
a primit +5%.');

END CASE;

END LOOP;

END;

/



-- 3. actualizarea comisionului angajaților în funcție de sex

BEGIN

FOR v_angajat IN (

SELECT id_angajat, sex FROM atodireseirina_angajati

) LOOP

CASE v_angajat.sex

WHEN 'F' THEN

UPDATE atodireseirina_angajati

SET comision = 0.12

WHERE id_angajat = v_angajat.id_angajat;

```
DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || 'femeie comision 0.12');
```

```
WHEN 'M' THEN
```

```
UPDATE atodireseirina_angajati
```

```
SET comision = 0.10
```

```
WHERE id_angajat = v_angajat.id_angajat;
```

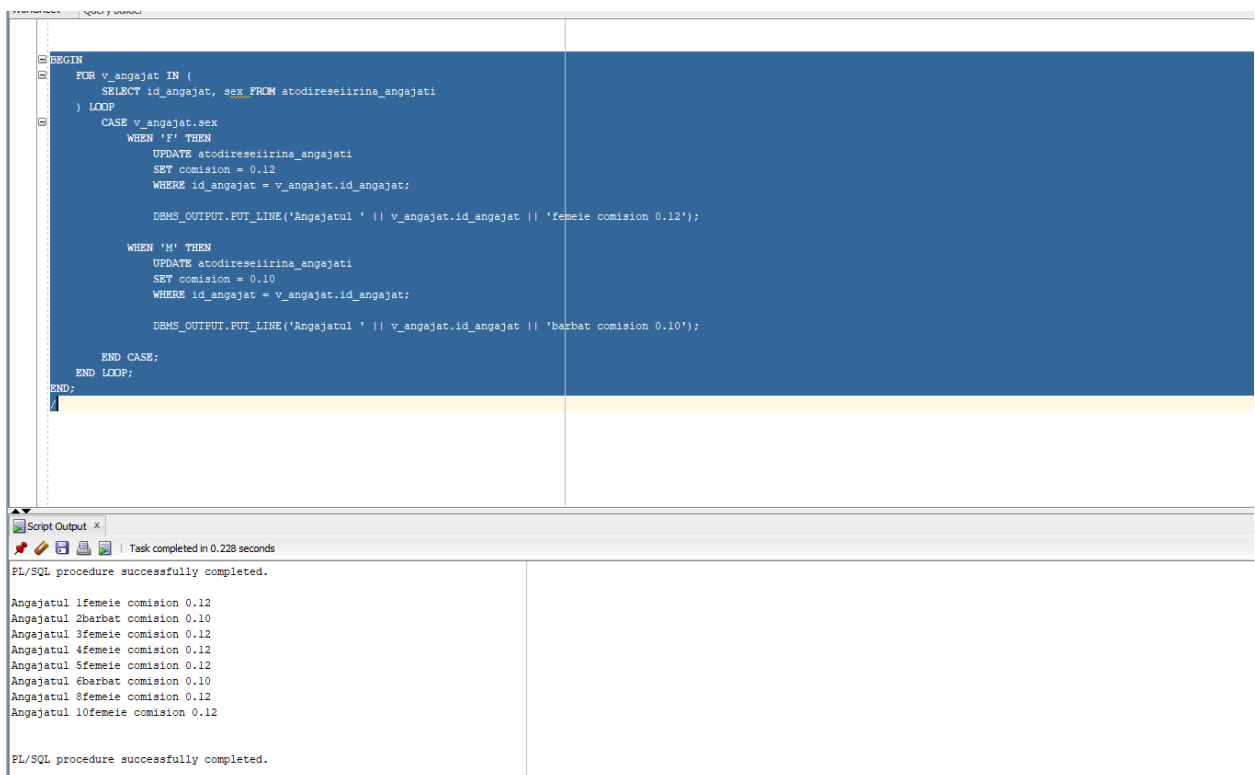
```
DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || 'barbat comision 0.10');
```

```
END CASE;
```

```
END LOOP;
```

```
END;
```

```
/
```



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

```
BEGIN
FOR v_angajat IN (
  SELECT id_angajat, sex FROM atodireseirina_angajati
) LOOP
  CASE v_angajat.sex
    WHEN 'F' THEN
      UPDATE atodireseirina_angajati
      SET comision = 0.12
      WHERE id_angajat = v_angajat.id_angajat;

      DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || 'femeie comision 0.12');

    WHEN 'M' THEN
      UPDATE atodireseirina_angajati
      SET comision = 0.10
      WHERE id_angajat = v_angajat.id_angajat;

      DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || 'barbat comision 0.10');
  END CASE;
END LOOP;
END;
```

Below the script editor, the 'Script Output' window is visible, showing the results of the execution:

```
Task completed in 0.228 seconds

PL/SQL procedure successfully completed.

Angajatul 1femeie comision 0.12
Angajatul 2barbat comision 0.10
Angajatul 3femeie comision 0.12
Angajatul 4femeie comision 0.12
Angajatul 5femeie comision 0.12
Angajatul 6barbat comision 0.10
Angajatul 8femeie comision 0.12
Angajatul 10femeie comision 0.12

PL/SQL procedure successfully completed.
```

-- 4. Actualizarea numărului de cărți împrumutate pentru cititori (cursor explicit + UPDATE)

DECLARE

CURSOR c_cititori IS

SELECT id_cititor FROM atodireseiirina_cititori;

v_nr_imprumuturi NUMBER;

BEGIN

FOR v_cititor IN c_cititori LOOP

SELECT COUNT(*) INTO v_nr_imprumuturi

FROM atodireseiirina_imprumuturi

WHERE id_cititor = v_cititor.id_cititor

AND data_restituire IS NULL;

UPDATE atodireseiirina_cititori

SET numar_carti_imprumutate = v_nr_imprumuturi

WHERE id_cititor = v_cititor.id_cititor;

DBMS_OUTPUT.PUT_LINE('Cititorul cu ID ' || v_cititor.id_cititor ||

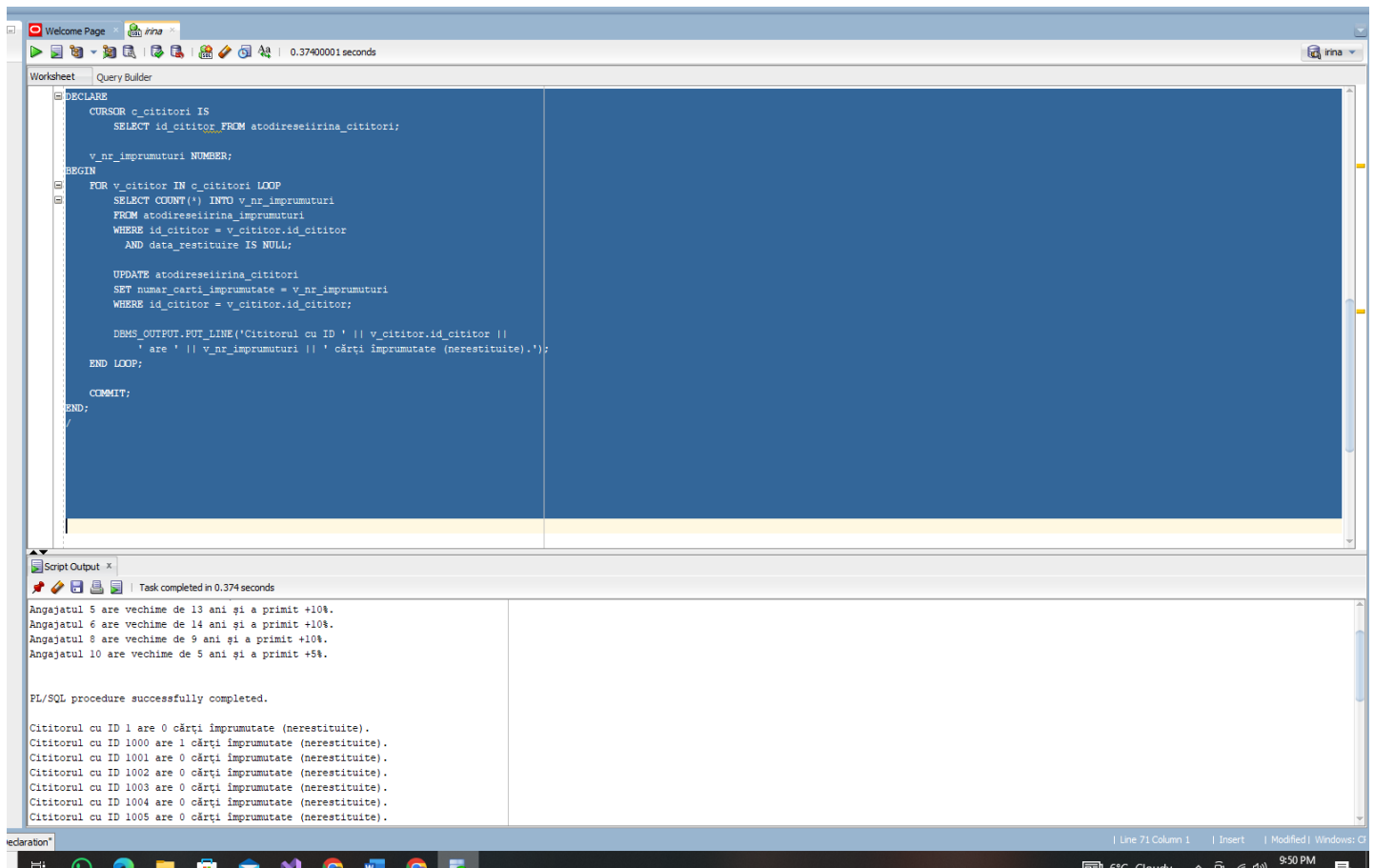
' are ' || v_nr_imprumuturi || ' cărți împrumutate (nerestituite).');

END LOOP;

COMMIT;

END;

/



-- 5. Ștergerea cititori care nu mai au împrumuturi înregistrate

BEGIN

FOR v_cititor IN (

SELECT id_cititor FROM atodireseirina_cititori

) LOOP

DECLARE

v_nr_imprumuturi NUMBER;

BEGIN

SELECT COUNT(*) INTO v_nr_imprumuturi

FROM atodireseirina_imprumuturi

WHERE id_cititor = v_cititor.id_cititor;

```

IF v_nr_imprumuturi = 0 THEN

DELETE FROM atodireseirina_cititori

WHERE id_cititor = v_cititor.id_cititor;

DBMS_OUTPUT.PUT_LINE('Cititorul ' || v_cititor.id_cititor || ' a fost șters (fără
împrumuturi).');

END IF;

END;

END LOOP;

ROLLBACK;

END;

```

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows the 'rma' connection. The 'Table Browser' pane lists various tables, including 'ATODIRESEIRINA_CITITORI'. The main 'Script Editor' pane contains the following PL/SQL code:

```

END LOOP;

COMMIT;

END;

/

BEGIN
FOR v_cititor IN (
  SELECT id_cititor FROM atodireseirina_cititori
) LOOP
  DECLARE
    v_nr_imprumuturi NUMBER;
  BEGIN
    SELECT COUNT(*) INTO v_nr_imprumuturi
    FROM atodireseirina_imprumuturi
    WHERE id_cititor = v_cititor.id_cititor;

    IF v_nr_imprumuturi = 0 THEN
      DELETE FROM atodireseirina_cititori
      WHERE id_cititor = v_cititor.id_cititor;

      DBMS_OUTPUT.PUT_LINE('Cititorul ' || v_cititor.id_cititor || ' a fost șters (fără împrumuturi).');
    END IF;
  END LOOP;

  ROLLBACK; -- sau COMMIT, dacă vrei să păstrezi ștergerile
END;

```

The 'Script Output' pane at the bottom shows the execution results, indicating that the task was completed in 0.648 seconds and listing the IDs of the deleted borrowers (Cititorul 1015 through Cititorul 1024).

```

Task completed in 0.648 seconds

Cititorul 1015 a fost șters (fără împrumuturi).
Cititorul 1016 a fost șters (fără împrumuturi).
Cititorul 1017 a fost șters (fără împrumuturi).
Cititorul 1018 a fost șters (fără împrumuturi).
Cititorul 1019 a fost șters (fără împrumuturi).
Cititorul 1020 a fost șters (fără împrumuturi).
Cititorul 1021 a fost șters (fără împrumuturi).
Cititorul 1022 a fost șters (fără împrumuturi).
Cititorul 1023 a fost șters (fără împrumuturi).
Cititorul 1024 a fost șters (fără împrumuturi).

PL/SQL procedure successfully completed.

```

Tema 2

1. Afișarea cărților publicate înainte de anul 1900

(Excepție implicită: NO_DATA_FOUND, excepție explicită definită de utilizator dacă cartea e prea veche)

```
set serveroutput on;
```

```
DECLARE
```

```
    CURSOR c_carti_vechi IS
```

```
        SELECT id_carte, titlu, an_publicare FROM atodireseirina_carti WHERE an_publicare < 1900;
```

```
    v_carte c_carti_vechi%ROWTYPE;
```

```
BEGIN
```

```
    OPEN c_carti_vechi;
```

```
    LOOP
```

```
        FETCH c_carti_vechi INTO v_carte;
```

```
        EXIT WHEN c_carti_vechi%NOTFOUND;
```

```
        IF v_carte.an_publicare < 1800 THEN
```

```
            RAISE_APPLICATION_ERROR(-20001, 'Cartea ' || v_carte.titlu || ' este prea veche pentru a  
fi păstrată în bibliotecă.');
```

```
        END IF;
```

```
        DBMS_OUTPUT.PUT_LINE('Carte veche: ' || v_carte.titlu || ' (' || v_carte.an_publicare || ');
```

```
    END LOOP;
```

```
    CLOSE c_carti_vechi;
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
```

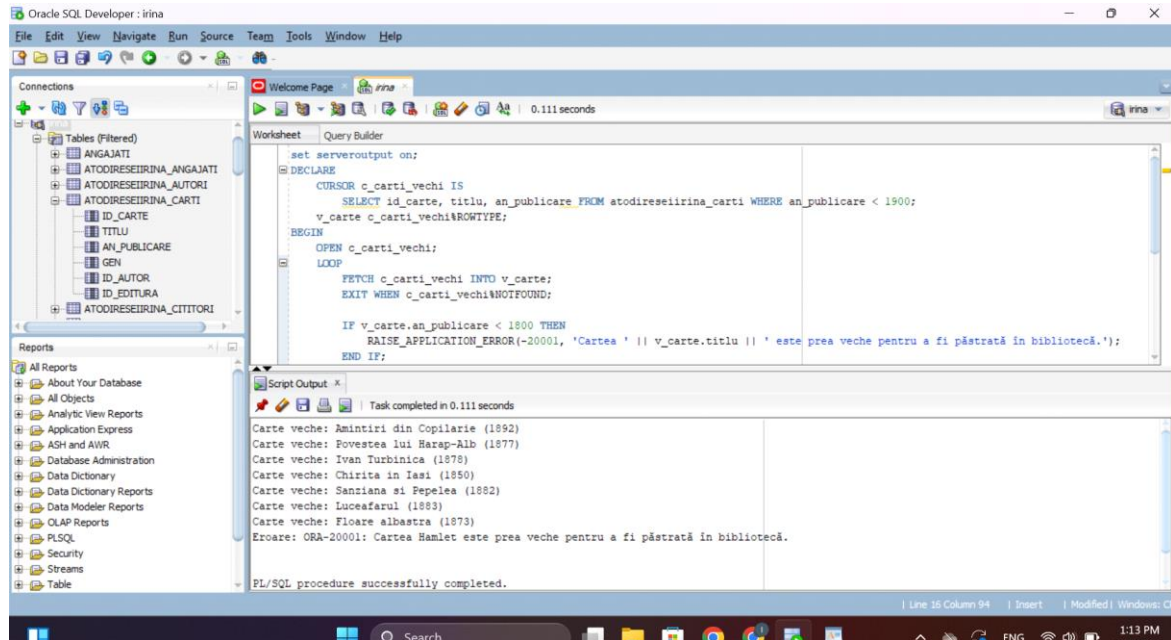
```
        DBMS_OUTPUT.PUT_LINE('Nu există cărți vechi în bibliotecă.');
```

```
    WHEN OTHERS THEN
```

```
DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
```

```
END;
```

```
/
```



2. Calculul mediei salariilor (cu tratament pentru împărțire la 0)

(Excepție implicită: ZERO_DIVIDE, excepție explicită: salariu inacceptabil)

```
set serveroutput on;
```

```
DECLARE
```

```
    v_total_salariu NUMBER := 0;
```

```
    v_nr_angajati NUMBER := 0;
```

```
    v_medie NUMBER;
```

```
BEGIN
```

```
    SELECT SUM(salariu), COUNT(*) INTO v_total_salariu, v_nr_angajati FROM  
    atodireseiirina_angajati;
```

```
    IF v_total_salariu > 10000000 THEN
```

```
        RAISE_APPLICATION_ERROR(-20002, 'Totalul salariilor este exagerat!');
```

```
    END IF;
```

```
v_medie := v_total_salariu / v_nr_angajati;
```

```
DBMS_OUTPUT.PUT_LINE('Media salariilor este: ' || TO_CHAR(v_medie, '9999.99'));
```

EXCEPTION

```
WHEN ZERO_DIVIDE THEN
```

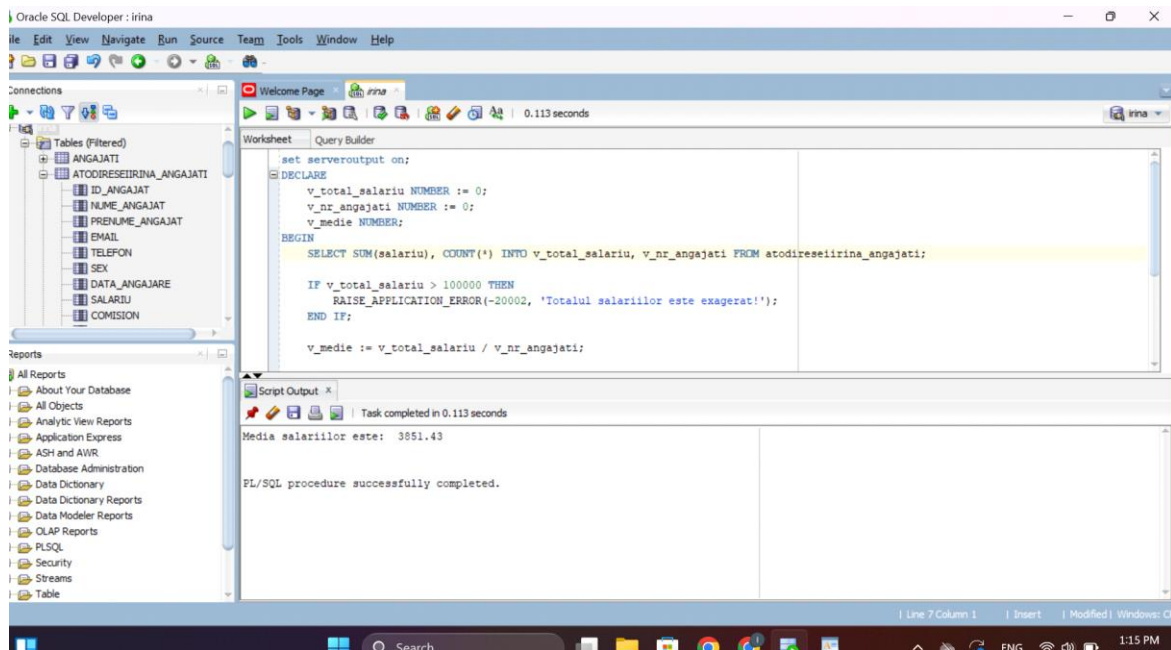
```
    DBMS_OUTPUT.PUT_LINE('Nu există angajați în baza de date.');
```

```
WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
```

```
END;
```

/



3. Găsirea unui autor unic după nume (cu TOO_MANY_ROWS)

(Excepție implicită: TOO_MANY_ROWS, excepție explicită: autor inexistent)

```
DECLARE
```

```
    v_id_autor NUMBER;
```

```
    v_nume_autor VARCHAR2(100) := 'Eminescu';
```

```
BEGIN
```

```
    SELECT id_autor INTO v_id_autor
```

```
    FROM atodireseirina_autori
```

```
WHERE nume_autor = v_nume_autor;
```

```
DBMS_OUTPUT.PUT_LINE('ID autor: ' || v_id_autor);
```

```
EXCEPTION
```

```
WHEN NO_DATA_FOUND THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Autorul ' || v_nume_autor || ' nu a fost găsit.');
```

```
WHEN TOO_MANY_ROWS THEN
```

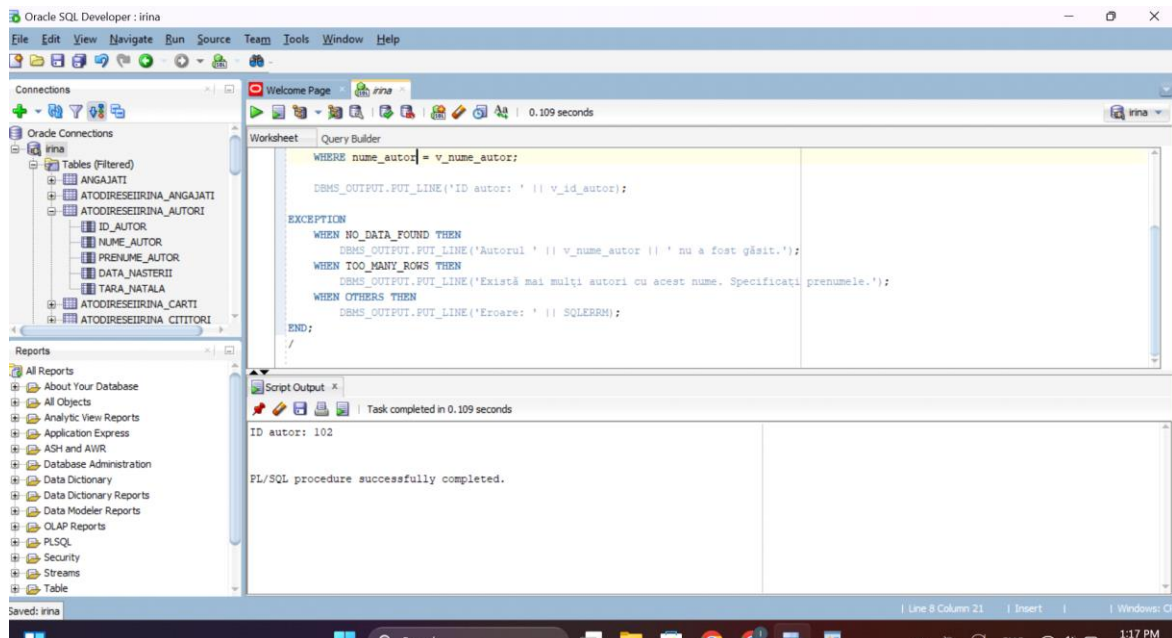
```
    DBMS_OUTPUT.PUT_LINE('Există mai mulți autori cu acest nume. Specificați prenumele.');
```

```
WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
```

```
END;
```

```
/
```



4. Încercarea de a adăuga un împrumut pentru o carte deja împrumutată (excepție explicită)

(Folosire de cursor + RAISE_APPLICATION_ERROR)

DECLARE

CURSOR c_imprumuturi IS

SELECT id_carte FROM atodireseirina_Imprumuturi WHERE data_restituire IS NULL;

v_carte_imprumutata c_imprumuturi%ROWTYPE;

v_id_carte NUMBER := 112;

v_gasit BOOLEAN := FALSE;

BEGIN

OPEN c_imprumuturi;

LOOP

FETCH c_imprumuturi INTO v_carte_imprumutata;

EXIT WHEN c_imprumuturi%NOTFOUND;

IF v_carte_imprumutata.id_carte = v_id_carte THEN

v_gasit := TRUE;

EXIT;

END IF;

END LOOP;

CLOSE c_imprumuturi;

IF v_gasit THEN

RAISE_APPLICATION_ERROR(-20003, 'Cartea cu ID-ul ' || v_id_carte || ' este deja împrumutată.');

ELSE

DBMS_OUTPUT.PUT_LINE('Cartea este disponibilă pentru împrumut.');

END IF;

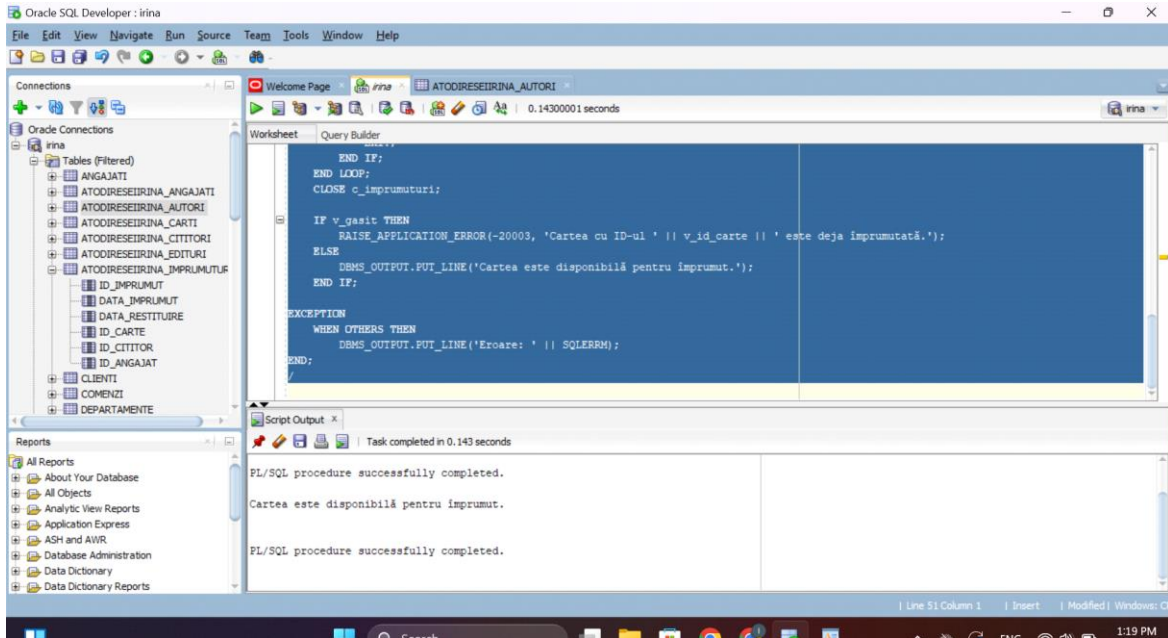
EXCEPTION

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);

END;

/



Tema 3

Pachet 1 Pachet_Biblioteca: 1 procedura, 3 functii:

CREATE OR REPLACE PACKAGE Pachet_Biblioteca AS

-- Proceduri

PROCEDURE Lista_Impurmuturi_Cititor(id_cititor NUMBER);

-- Functii

FUNCTION Titlul_Exista(titlu_carte VARCHAR2) RETURN NUMBER;

FUNCTION Numar_Impurmuturi(id_cititor NUMBER) RETURN NUMBER;

FUNCTION Carti_Editura(id_editura NUMBER) RETURN NUMBER;

END;

CREATE OR REPLACE PACKAGE BODY Pachet_Biblioteca AS

PROCEDURE Imprumuta_Carte(id_carte NUMBER, id_cititor NUMBER, id_angajat NUMBER)
IS

new_id NUMBER;

BEGIN


```
SELECT NVL(MAX(ID_IMPRUMUT), 0) + 1 INTO new_id FROM  
ATODIRESEIIRINA_IMPRUMUTURI;
```

```
INSERT INTO ATODIRESEIIRINA_IMPRUMUTURI VALUES (new_id, SYSDATE, NULL,  
id_carte, id_cititor, id_angajat);
```

```
END;
```

```
PROCEDURE Lista_Imprumuturi_Cititor(id_cititor NUMBER) IS
```

```
CURSOR imprumuturi IS
```

```
SELECT ID_CARTE, DATA_IMPRUMUT, DATA_RESTITUIRE
```

```
FROM ATODIRESEIIRINA_IMPRUMUTURI
```

```
WHERE ID_CITITOR = id_cititor;
```

```
BEGIN
```

```
FOR rec IN imprumuturi LOOP
```

```
DBMS_OUTPUT.PUT_LINE('Carte: ' || rec.ID_CARTE || ' | Imprumut: ' ||  
rec.DATA_IMPRUMUT || ' | Restituire: ' || rec.DATA_RESTITUIRE);
```

```
END LOOP;
```

```
END;
```

```
FUNCTION Titlul_Exista(titlu_carte VARCHAR2) RETURN NUMBER IS
```

```
nr NUMBER;
```

```
BEGIN
```

```
SELECT COUNT(*) INTO nr
```

```
FROM ATODIRESEIIRINA_CARTI
```

```
WHERE TITLU = titlu_carte;
```

```
RETURN nr;
```

```
END;
```

```
FUNCTION Numar_Imprumuturi(id_cititor NUMBER) RETURN NUMBER IS
```

```
nr NUMBER;
```

```
BEGIN
```

```
SELECT COUNT(*) INTO nr
```

```
FROM ATODIRESEIIRINA_IMPRUMUTURI
```

```
WHERE ID_CITITOR = id_cititor;
RETURN nr;
END;
```

```
FUNCTION Carti_Editura(id_editura NUMBER) RETURN NUMBER IS
    total NUMBER;
BEGIN
    SELECT COUNT(*) INTO total FROM ATODIRESEIIRINA_CARTI WHERE ID_EDITURA =
id_editura;
    RETURN total;
END;
```

```
END;
```

```
--apeluri pachet biblioteca
```

```
-- Lista_Imprumuturi_Cititor
```

```
BEGIN
```

```
    Pachet_Biblioteca.Lista_Imprumuturi_Cititor(2);
```

```
END;
```

```
/
```

```
-- Titlul_Exista
```

```
BEGIN
```

```
    IF Pachet_Biblioteca.Titlul_Exista('Baltagul') > 0 THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Cartea exista.');
```

```
    ELSE
```

```
        DBMS_OUTPUT.PUT_LINE('Cartea nu exista.');
```

```
    END IF;
```

```
END;
```

```
/
```

```
-- Numar_Imprumuturi
```

```
DECLARE
```

```

nr NUMBER;

BEGIN

nr := Pachet_Biblioteca.Numar_Imprumuturi(2);

DBMS_OUTPUT.PUT_LINE('Imprumuturi: ' || nr);

END;

/

```

-- Carti_Editura

```

DECLARE

nr NUMBER;

BEGIN

nr := Pachet_Biblioteca.Carti_Editura(1);

DBMS_OUTPUT.PUT_LINE('Carti publicate de editura: ' || nr);

END;

/

```

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows the 'PFA' schema with various tables like ANGAJATI, ATODORESEIRINA_ANGAJATI, etc. The main workspace shows a PL/SQL script with three sections: a check for book existence, a procedure to get the number of books, and a procedure to get the number of books published by a specific publisher. The script is executed, and the 'Script Output' pane at the bottom shows the results: 'Task completed in 0.257 seconds', 'PL/SQL procedure successfully completed.', and 'Cartea nu exista.'

```

-- Titlul_Existenta
/
-- Titlul_Existenta
IF Pachet_Biblioteca.Titlul_Existenta('Baltagul') > 0 THEN
  DBMS_OUTPUT.PUT_LINE('Cartea exista. ');
ELSE
  DBMS_OUTPUT.PUT_LINE('Cartea nu exista. ');
END IF;
END;

-- Numar_Imprumuturi
DECLARE
  nr NUMBER;
BEGIN
  nr := Pachet_Biblioteca.Numar_Imprumuturi(2);
  DBMS_OUTPUT.PUT_LINE('Imprumuturi: ' || nr);
END;

-- Carti_Editura
DECLARE
  nr NUMBER;
BEGIN
  nr := Pachet_Biblioteca.Carti_Editura(1);
  DBMS_OUTPUT.PUT_LINE('Carti publicate de editura: ' || nr);
END;

/

```

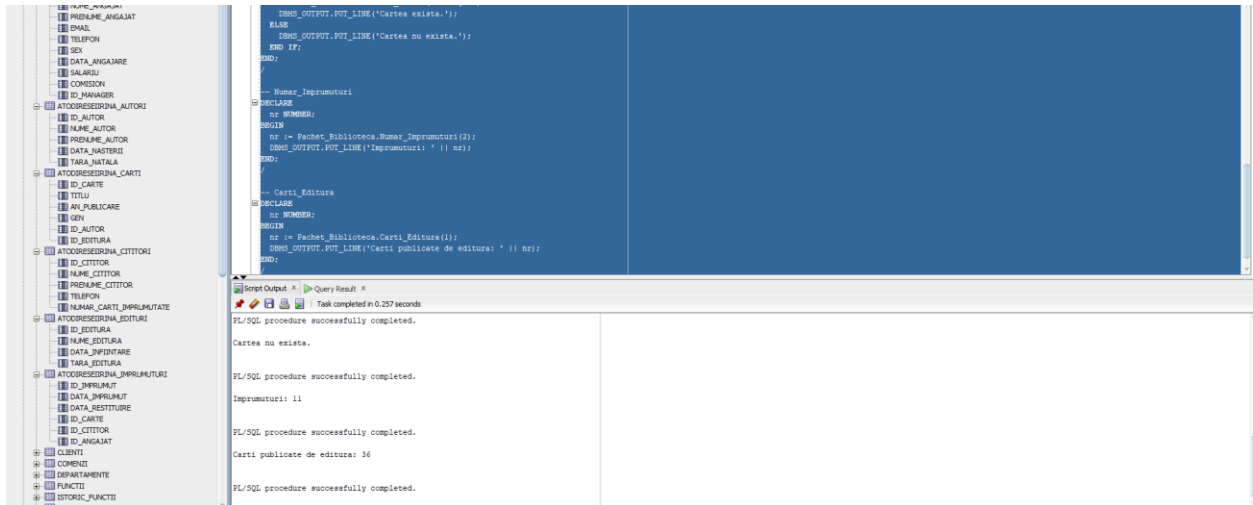
Script Output: Task completed in 0.257 seconds

Query Result:

Carte:	2001	Imprumut:	03-JUN-23	Restituit:	20-JUN-23
Carte:	2000	Imprumut:	05-SEP-23	Restituit:	30-SEP-23
Carte:	2002	Imprumut:	10-MAR-23	Restituit:	20-MAR-23
Carte:	2003	Imprumut:	11-MAY-23	Restituit:	28-FEB-23
Carte:	2004	Imprumut:	03-APR-23	Restituit:	17-APR-23
Carte:	2005	Imprumut:	04-APR-23	Restituit:	21-APR-23
Carte:	2006	Imprumut:	06-MAY-23	Restituit:	15-MAY-23
Carte:	2004	Imprumut:	27-JUL-23	Restituit:	08-AUG-23
Carte:	2007	Imprumut:	27-AUG-23	Restituit:	04-SEP-23
Carte:	2008	Imprumut:	09-JAN-23	Restituit:	25-JAN-23

PL/SQL procedure successfully completed.

Cartea nu exista.



Pachet 2 Pachet_Angajati: 2 proceduri, 2 functii:

CREATE OR REPLACE PACKAGE PACHET ANGAJATI AS

-- Proceduri

```
PROCEDURE Adauga_Cititor(nume VARCHAR2, prenume VARCHAR2, telefon VARCHAR2);
```

```
PROCEDURE Schimba Salariu(id angajat NUMBER, procent NUMBER);
```

-- Functii

FUNCTION Cititori Activi RETURN NUMBER;

```
FUNCTION Cel_Mai_Productiv_Angajat RETURN VARCHAR2;
```

END;

CREATE OR REPLACE PACKAGE BODY PACHET_ANGAJATI AS

```
PROCEDURE Adauga Cititor(num VARCHAR2, prenum VARCHAR2, telefon VARCHAR2) IS
```

```
new id NUMBER;
```

BEGIN

```
SELECT NVL(MAX(ID_CITITOR), 0) + 1 INTO new_id FROM
ATODIRESEIIRINA CITITORI;
```

INSERT INTO ATODIRESEIIRINA CITITORI VALUES (new id, nume, prenume, telefon, 0);

END;

```

PROCEDURE Schimba_Salariu(id_angajat NUMBER, procent NUMBER) IS
BEGIN
    UPDATE ATODIRESEIIRINA_ANGAJATI
    SET SALARIU = SALARIU + SALARIU * procent / 100
    WHERE ID_ANGAJAT = id_angajat;
END;

```

```

FUNCTION Cititori_Activi RETURN NUMBER IS
    nr NUMBER;
BEGIN
    SELECT COUNT(DISTINCT ID_CITITOR)
    INTO nr
    FROM ATODIRESEIIRINA_IMPRUMUTURI
    WHERE DATA_RESTITUIRE IS NULL;
    RETURN nr;
END;

```

```

FUNCTION Cel_Mai_Productiv_Angajat RETURN VARCHAR2 IS
    angajat VARCHAR2(100);
BEGIN
    SELECT NUME_ANGAJAT || ' ' || PRENUME_ANGAJAT
    INTO angajat
    FROM ATODIRESEIIRINA_ANGAJATI
    WHERE ID_ANGAJAT = (
        SELECT ID_ANGAJAT
        FROM (
            SELECT ID_ANGAJAT, COUNT(*) AS nr
            FROM ATODIRESEIIRINA_IMPRUMUTURI
            GROUP BY ID_ANGAJAT
            ORDER BY nr DESC
        )
    )

```

```

        WHERE ROWNUM = 1
    );
    RETURN angajat;
END;

END;
/

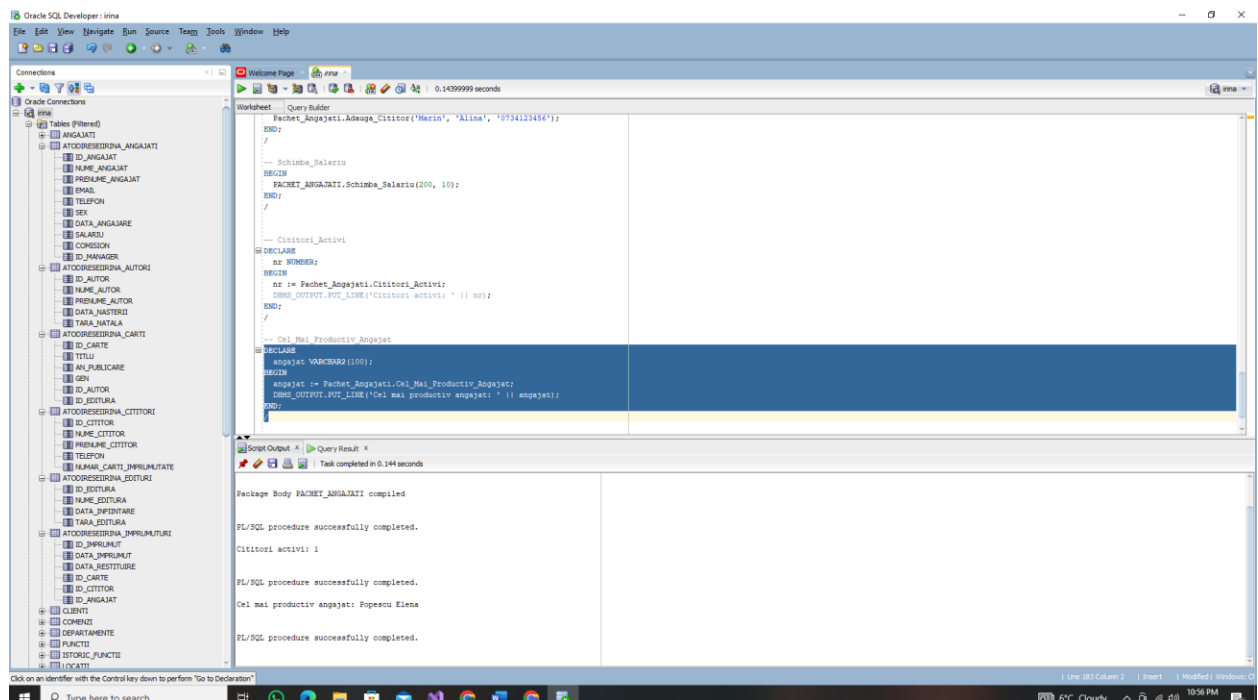
--apelare
-- Adauga_Cititor
BEGIN
    Pachet_Angajati.Adauga_Cititor('Marin', 'Alina', '0734123456');
END;
/

-- Schimba_Salariu
BEGIN
    PACHET_ANGAJATI.Schimba_Salariu(200, 10);
END;
/

-- Cititori_Activi
DECLARE
    nr NUMBER;
BEGIN
    nr := Pachet_Angajati.Cititori_Activi;
    DBMS_OUTPUT.PUT_LINE('Cititori activi: ' || nr);
END;
/

```

/



Procedura Arhiveaza_Carti_Inactive

Creăm o tabelă auxiliară atodireseiirina_carti_inactive, care va conține cărțile ce nu au fost niciodată împrumutate.

```
set serveroutput on;
```

```
CREATE TABLE atodireseiirina_carti_inactive AS
```

```
SELECT * FROM atodireseiirina_carti WHERE 1=0;
```

```
CREATE OR REPLACE PROCEDURE Arhiveaza_Carti_Inactive IS
```

```
    CURSOR c_carti IS
```

```
        SELECT * FROM atodireseiirina_carti c
```

```
        WHERE NOT EXISTS (
```

```
            SELECT 1 FROM atodireseiirina_imprumuturi i
```

```
            WHERE i.id_carte = c.id_carte
```

```
        );
```

```
    v_carte c_carti%ROWTYPE;
```

```
BEGIN
```

```
    OPEN c_carti;
```

```
    LOOP
```

```
        FETCH c_carti INTO v_carte;
```

```
        EXIT WHEN c_carti%NOTFOUND;
```

```
        -- Inserare în tabela de arhivă
```

```
        INSERT INTO atodireseiirina_carti_inactive
```

```
        VALUES v_carte;
```

```
        -- Ștergere din tabela principală
```

```
        DELETE FROM atodireseiirina_carti
```

```
        WHERE id_carte = v_carte.id_carte;
```

```
        DBMS_OUTPUT.PUT_LINE('Cartea "' || v_carte.titlu || '" a fost arhivată (nu a fost niciodată împrumutată).');
```


END LOOP;

CLOSE c_carti;

COMMIT;

END;

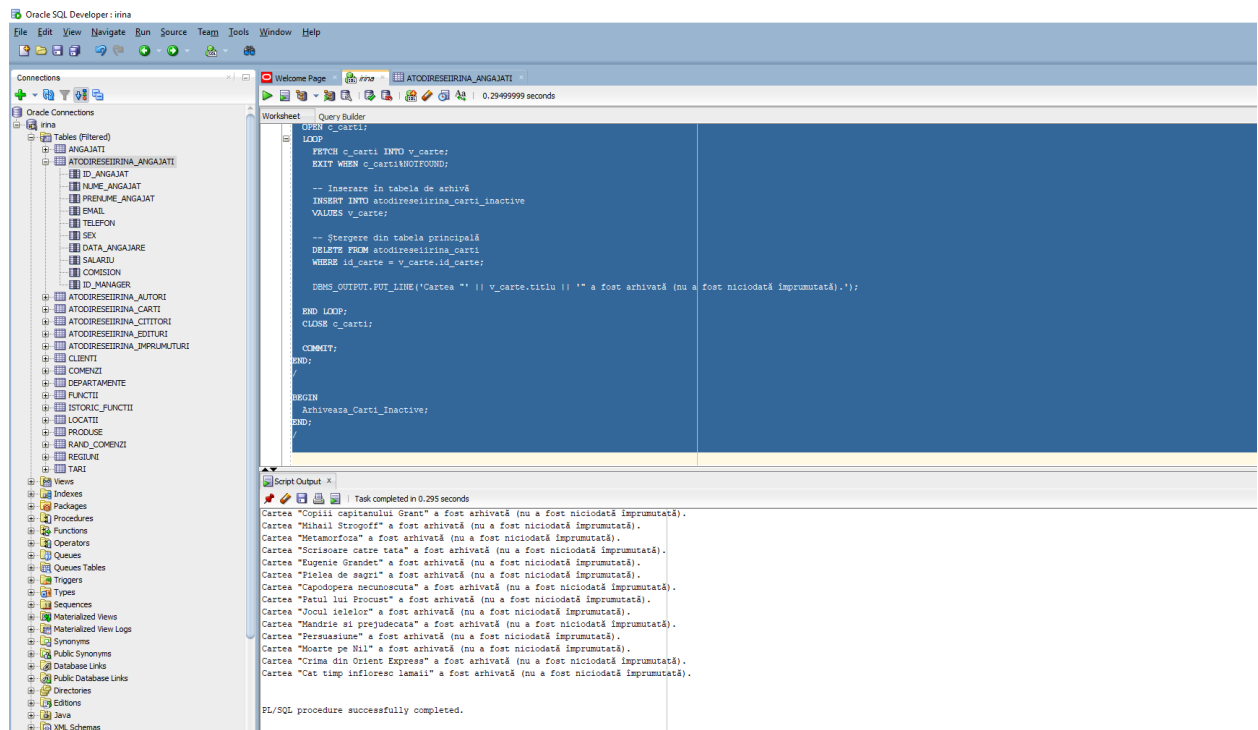
/

BEGIN

Arhiveaza_Carti_Inactive;

END;

/



Oracle SQL Developer: iina

Connections

Oracle Connections

iina

Tables (Filtered)

- ANGAJATI
- ATOORESEIRINA_ANGAJATI
 - ID_ANGAJAT
 - NUME_ANGAJAT
 - PRENUM_ANGAJAT
 - EMAIL
 - TELEFON
 - SEX
 - DATA_ANGAJARE
 - SALARIU
 - COMESION
 - ID_MANAGER
- ATOORESEIRINA_AUTORI
- ATOORESEIRINA_CARTI
- ATOORESEIRINA_CITITORI
- ATOORESEIRINA_EDITURI
- ATOORESEIRINA_IMPRUMUTURI
- CLIENTI
- COMENZI
- DEPARTAMENTE
- FUNCTII
- ISTORIC_FUNCTII
- LOCATII
- PRODUSE
- RAND_COMENZI
- RESOLUT
- TARI

Views

- Indexes
- Packages
- Procedures
- Functions
- Operators
- Queues
- Queues Tables
- Triggers
- Types
- Sequences
- Materialized Views
- Materialized View Logs
- Synonyms
- Public Synonyms
- Database Links
- Public Database Links
- Directories
- Editions
- Java
- Java Schemas

Worksheet

Query Builder

```
OPEN c_carti;  
LOOP  
FETCH c_carti INTO v_carte;  
EXIT WHEN c_carti%NOTFOUND;  
  
-- Inscrare in tabela de arhivă  
INSERT INTO atoareseirina_carti_inactive  
VALUES v_carte;  
  
-- Stergere din tabela principală  
DELETE FROM atoareseirina_carti  
WHERE id_carte = v_carte.id_carte;  
  
DBMS_OUTPUT.PUT_LINE('Cartea "' || v_carte.ciclu || '" a fost arhivată (nu a fost niciodată împrumutată).');  
  
END LOOP;  
CLOSE c_carti;  
  
COMMIT;  
  
END;  
  
BEGIN  
Arhiveaza_Carti_Inactive;  
END;
```

Script Output

Task completed in 0.295 seconds

Cartea "Copiii capitanului Grant" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Mihail Strogoff" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Memoriora" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Scrisoare catre tata" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Eugenie Grandet" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Pielea de saizi" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Capodopera necunoscuta" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Parul lui Proust" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Jocul lelelor" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Mandrie si prejudecata" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Persuasiune" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Moarte pe Nil" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Crima din Orient Express" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Cat timp infloresc lamail" a fost arhivată (nu a fost niciodată împrumutată).

PL/SQL procedure successfully completed.

Triggeri

Setare comision implicit în funcție de sex

```
CREATE OR REPLACE TRIGGER trg_set_comision_default
BEFORE INSERT ON atodireseiirina_angajati
FOR EACH ROW
BEGIN
    IF :NEW.sex = 'F' AND :NEW.comision IS NULL THEN
        :NEW.comision := 0.12;
    ELSIF :NEW.sex = 'M' AND :NEW.comision IS NULL THEN
        :NEW.comision := 0.10;
    END IF;
END;
/

SELECT id_angajat, nume_angajat, prenume_angajat, comision
FROM atodireseiirina_angajati
WHERE id_angajat = 998;
```

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows the 'hrna' database. The 'Tables (Filtered)' list includes 'ANGAJATI' and 'ATODIRESEIIRINA_ANGAJATI'. The main workspace shows the 'Query Builder' with the following SQL script:

```
CREATE OR REPLACE TRIGGER trg_set_comision_default
BEFORE INSERT ON atodireseiirina_angajati
FOR EACH ROW
BEGIN
    IF :NEW.sex = 'F' AND :NEW.comision IS NULL THEN
        :NEW.comision := 0.12;
    ELSIF :NEW.sex = 'M' AND :NEW.comision IS NULL THEN
        :NEW.comision := 0.10;
    END IF;
END;
/

SELECT id_angajat, nume_angajat, prenume_angajat, comision
FROM atodireseiirina_angajati
WHERE id_angajat = 998;
```

The 'Script Output' pane at the bottom shows the execution results:

```
Trigger TRG_SET_COMISION_DEFAULT compiled

ID_ANGAJAT NUME_ANGAJAT PRENUME_ANGAJAT COMISION
-----
998 Ionescu Maria .12
```

Verificare salariu minim (nu poate fi sub 2000)

BEGIN

```
INSERT INTO atodireseiirina_angajati (  
    id_angajat, nume, prenume, email, telefon, sex, data_angajare, salariu, comision, id_manager  
) VALUES (  
    999, 'Popescu', 'Ion', 'ion.popescu@example.com', '0712345678', 'M',  
    TO_DATE('2023-01-01', 'YYYY-MM-DD'), 1800, NULL, NULL  
);
```

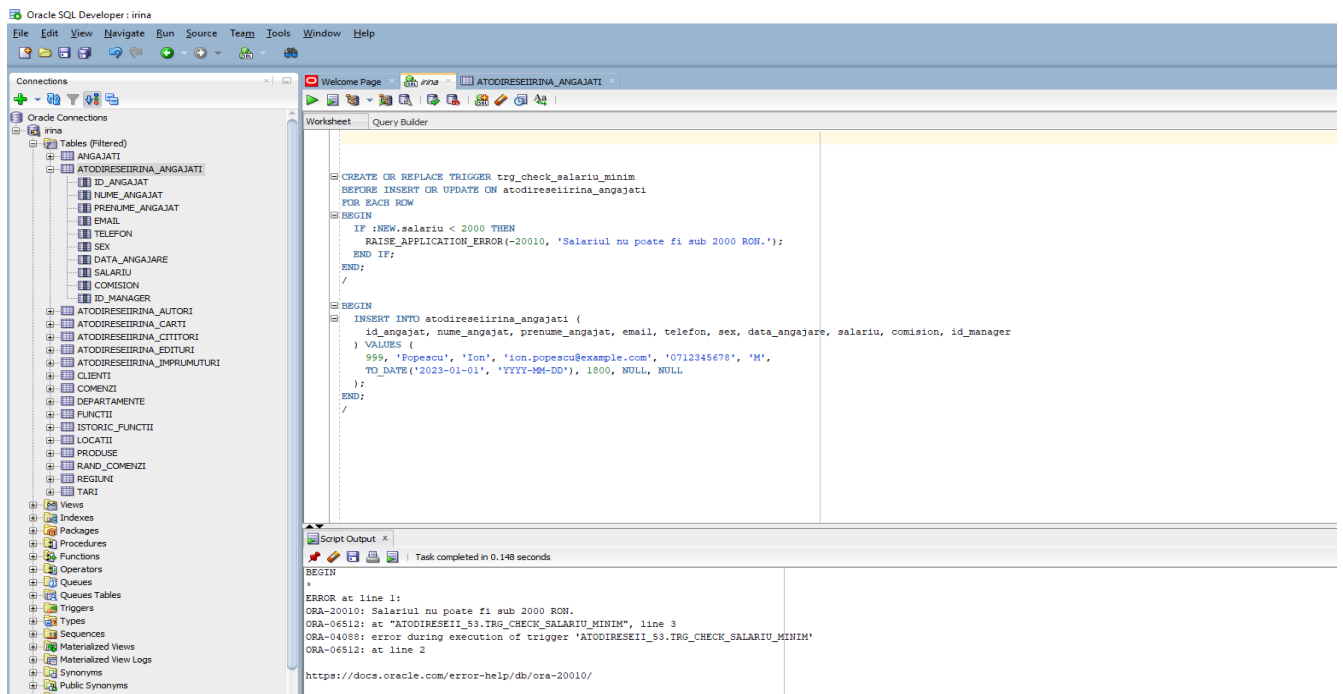
END;

/

BEGIN

```
INSERT INTO atodireseiirina_angajati (  
    id_angajat, nume_angajat, prenume_angajat, email, telefon, sex, data_angajare, salariu, comision,  
    id_manager  
) VALUES (  
    999, 'Popescu', 'Ion', 'ion.popescu@example.com', '0712345678', 'M',  
    TO_DATE('2023-01-01', 'YYYY-MM-DD'), 1800, NULL, NULL  
);
```

END;



Validare dată restituire: nu poate fi anterioară datei împrumutului

```
CREATE OR REPLACE TRIGGER trg_verificare_data_restituire
```

```
BEFORE INSERT OR UPDATE ON atodireseiirina_imprumuturi
```

```
FOR EACH ROW
```

```
BEGIN
```

```
IF :NEW.data_restituire IS NOT NULL AND :NEW.data_restituire < :NEW.data_imprumut THEN
```

```
    RAISE_APPLICATION_ERROR(-20005, 'Data restituirii nu poate fi anterioară datei de împrumut.');
```

```
END IF;
```

```
END;
```

```
/
```

```
-- Incerc să inserez o restituire in trecut
```

```
INSERT INTO atodireseiirina_imprumuturi VALUES (
```

```
    1502, TO_DATE('2025-05-20', 'YYYY-MM-DD'),
```

```
    TO_DATE('2025-05-19', 'YYYY-MM-DD'), -- invalid
```

```
    102, 2, 998
```

```
);
```

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows the 'inna' database connection. The 'Schema Browser' on the left lists various tables and views, including 'ATODIRESEIIRINA_ANGAJATI'. The main 'Worksheet' area contains the PL/SQL script defined above. The 'Script Output' pane at the bottom shows the execution results, including a task completion message and a detailed error report. The error report indicates a 'ORA-20005: Data restituirii nu poate fi anterioară datei de împrumut.' (Data of return cannot be earlier than the date of loan) at line 3 of the trigger 'ATODIRESEIIRINA_53.TRG_VERIFICARE_DATA_RESTITUIRE'.

```
COMMIT;
END;
/

BEGIN
    Arhiveaza_Carti_Inactive;
END;
/

CREATE OR REPLACE TRIGGER trg_verificare_data_restituire
BEFORE INSERT OR UPDATE ON atodireseiirina_imprumuturi
FOR EACH ROW
BEGIN
    IF :NEW.data_restituire IS NOT NULL AND :NEW.data_restituire < :NEW.data_imprumut THEN
        RAISE_APPLICATION_ERROR(-20005, 'Data restituirii nu poate fi anterioară datei de împrumut.');
```

```
END IF;
END;

-- Incerc să inserez o restituire in trecut
INSERT INTO atodireseiirina_imprumuturi VALUES (
    1502, TO_DATE('2025-05-20', 'YYYY-MM-DD'),
    TO_DATE('2025-05-19', 'YYYY-MM-DD'), -- invalid
    102, 2, 998
);
```

Task completed in 0.254 seconds

Error at Command Line : 53 Column : 13

Error report -

```
SQL Error: ORA-20005: Data restituirii nu poate fi anterioară datei de împrumut.
ORA-06512: at "ATODIRESEIIRINA_53.TRG_VERIFICARE_DATA_RESTITUIRE", line 3
ORA-04088: error during execution of trigger "ATODIRESEIIRINA_53.TRG_VERIFICARE_DATA_RESTITUIRE"
```

<https://docs.oracle.com/error-help/db/ora-20005/>

