

SISTEM DE GESTIUNE ȘI BAZE DE DATE PENTRU LIBRĂRIE ONLINE

Dobromirescu Mihaela

Seria 24

Grupa 242

An Universitar 2023-2024

Cerinta 1.	3
Cerinta 2.	4
Cerinta 3.	5
Cerinta 4.	7
Cerinta 5.	10
Cerinta 6.	18
Cerinta 7.	22
Cerinta 8.	25
Cerinta 9.	28
Cerinta 10.	32
Cerinta 11.	34
Cerinta 12.	36

Cerinta 1.

Prezentați pe scurt baza de date (utilitatea ei).

Baza de date prezentată este un sistem de management al unei librării online. Scopul acestei baze de date este de a organiza și gestiona informațiile referitoare la cărți, autori, clienți, comenzi, edituri și categorii.

Infrastructura utilizată în implementarea acestui proiect:

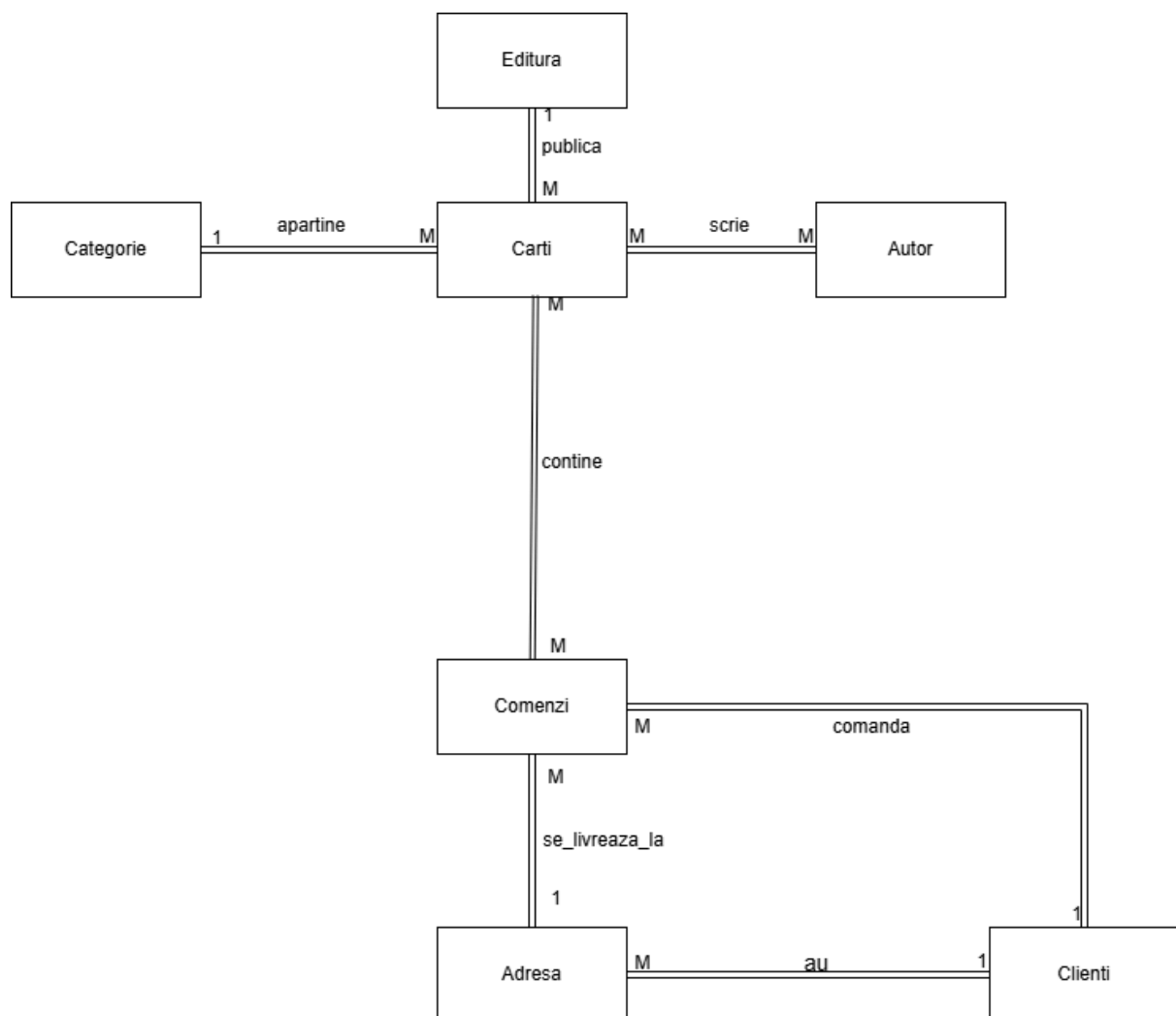
Versiune SGBD - Oracle Database 21c Express Edition Release 21.0.0.0.0

Configurația Software - Sistem de operare Version 23H2 (OS Build 22631.4571)

Nu am utilizat mașina virtuală în realizarea acestui proiect.

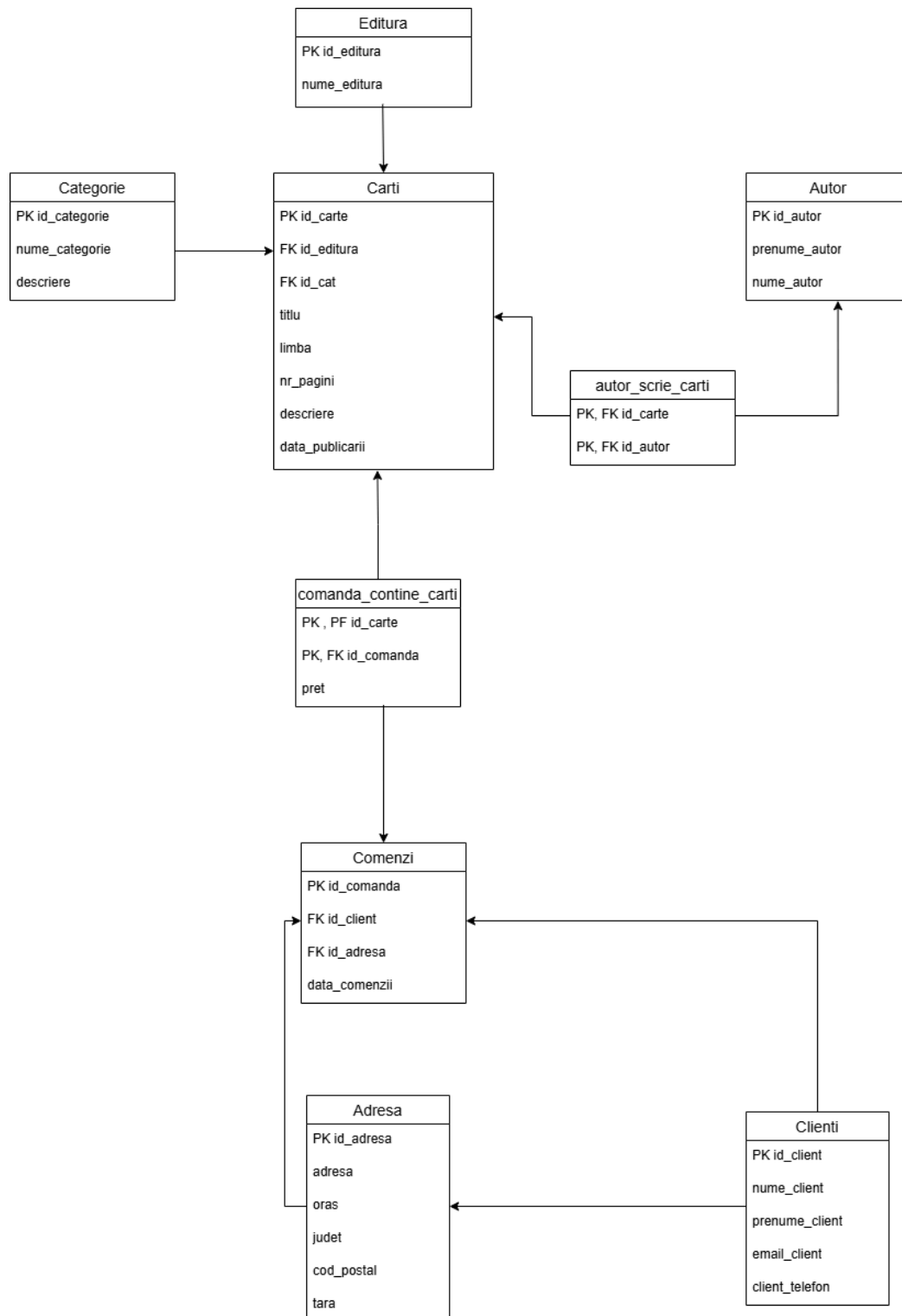
Cerinta 2.

Realizați diagrama entitate-relatie (ERD)



Cerinta 3.

Pornind de la diagrama entitate-relatie realizați diagrama conceptuală a modelului propus



AUTOR(id_autor#, nume_autor, prenume_autor)

CATEGORIE(id_cat#, nume_categorie, descriere)

CLIENTI(id_client#, nume_client, prenume_client, email_client, telefon_client)

ADRESA(id_adresa#, adresa, oras, judet, cod_postal, tara)

EDITURA(id_editura#, nume_editura)

CARTI(id_carte#, titlu, limba, nr_pagini, descriere, data_publicarii, id_editura, id_cat)

COMENZI(id_comanda, data_comenzii, id_client, id_adresa)

COMANDA_CONTINE_CARTI(id_carte#, id_comanda, id_carte, id_comanda, pret)

AUTOR_SCRIE_CARTI(id_carte#, id_autor#, id_carte, id_autor)

Cerinta 4.

**Implementați în Oracle diagrama conceptuală realizată:
definiți toate tabelele, definind toate constrângerile de
integritate necesare (chei primare, cheile externe etc).**

```
CREATE TABLE Autor (  
  id_autor INT PRIMARY KEY,  
  prenume_autor VARCHAR2(50) NOT NULL,  
  nume_autor VARCHAR2(50) NOT NULL  
);
```

```
CREATE TABLE Categorie (  
  id_cat INT PRIMARY KEY,  
  nume_categorie VARCHAR2(50) NOT NULL,  
  descriere VARCHAR2(255) NOT NULL  
);
```

```
CREATE TABLE Clienti (  
  id_client INT PRIMARY KEY,
```

```
    nume_client VARCHAR2(100) NOT NULL,  
    prenume_client VARCHAR2(100) NOT NULL,  
    email_client VARCHAR2(255) UNIQUE NOT NULL,  
    telefon_client VARCHAR2(10) NOT NULL  
);
```

```
CREATE TABLE Adresa (  
    id_adresa INT PRIMARY KEY,  
    adresa VARCHAR2(255) NOT NULL,  
    oras VARCHAR2(100) NOT NULL,  
    judet VARCHAR2(100) NOT NULL,  
    cod_postal VARCHAR2(10) NOT NULL,  
    tara VARCHAR2(100) NOT NULL  
);
```

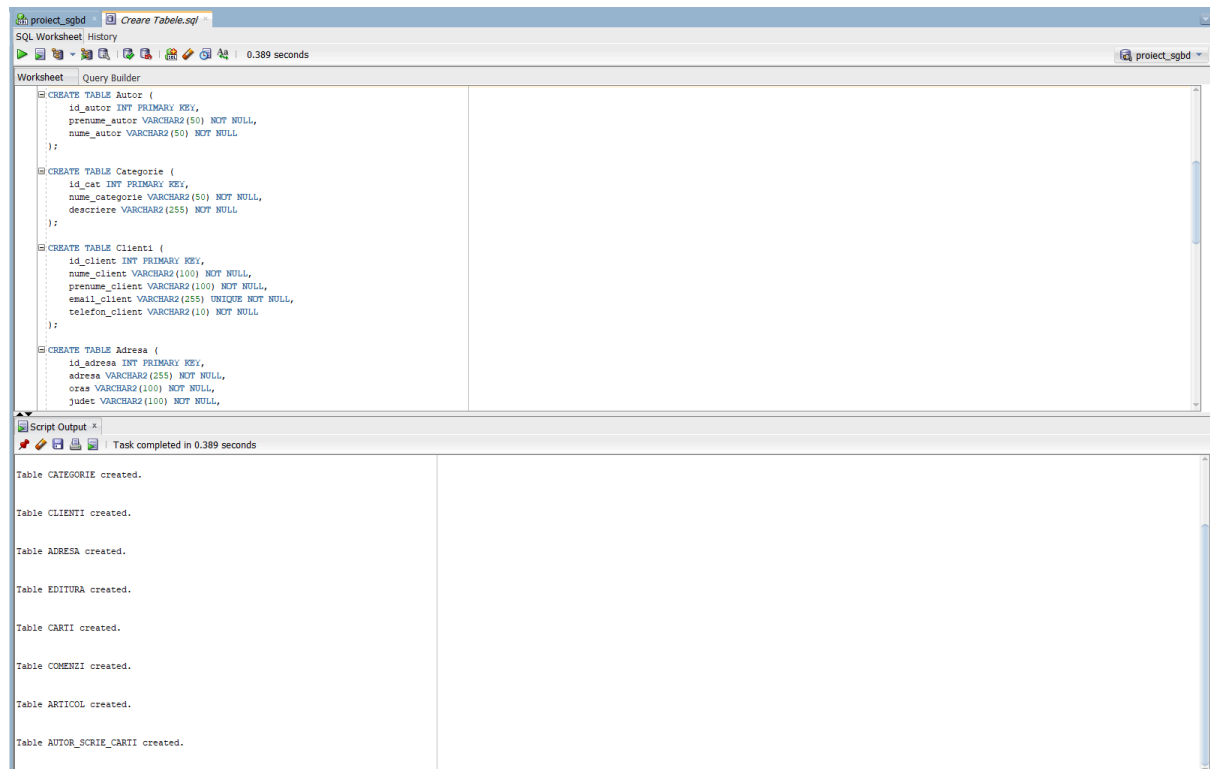
```
CREATE TABLE Editura (  
    id_editura INT PRIMARY KEY,  
    nume_editura VARCHAR2(255) NOT NULL  
);
```

```
CREATE TABLE Carti (  
    id_carte INT PRIMARY KEY,  
    titlu VARCHAR2(255) NOT NULL,  
    limba VARCHAR2(50) NOT NULL,  
    nr_pagini INT NOT NULL,  
    descriere VARCHAR2(255) NOT NULL,  
    data_publicarii DATE NOT NULL,  
    id_editura INT,  
    id_cat INT,  
    FOREIGN KEY (id_editura) REFERENCES Editura(id_editura),  
    FOREIGN KEY (id_cat) REFERENCES Categorie(id_cat)  
);
```

```
CREATE TABLE Comenzi (  
    id_comanda INT PRIMARY KEY,  
    data_comenzii DATE NOT NULL,  
    id_client INT,  
    id_adresa INT,  
    FOREIGN KEY (id_client) REFERENCES Clienti(id_client),  
    FOREIGN KEY (id_adresa) REFERENCES Adresa(id_adresa)  
);
```

```
CREATE TABLE autor_serie_carti (  
    id_carte INT,  
    id_autor INT,  
    PRIMARY KEY (id_carte, id_autor),  
    FOREIGN KEY (id_carte) REFERENCES Carti(id_carte),  
    FOREIGN KEY (id_autor) REFERENCES Autor(id_autor)  
);
```

```
CREATE TABLE comanda_contine_carti (
    id_carte INT,
    id_comanda INT,
    pret DECIMAL(10,2),
    PRIMARY KEY (id_carte, id_comanda),
    FOREIGN KEY (id_carte) REFERENCES Carti(id_carte)
    FOREIGN KEY (id_comanda) REFERENCES Comenzi(id_comanda)
);
```



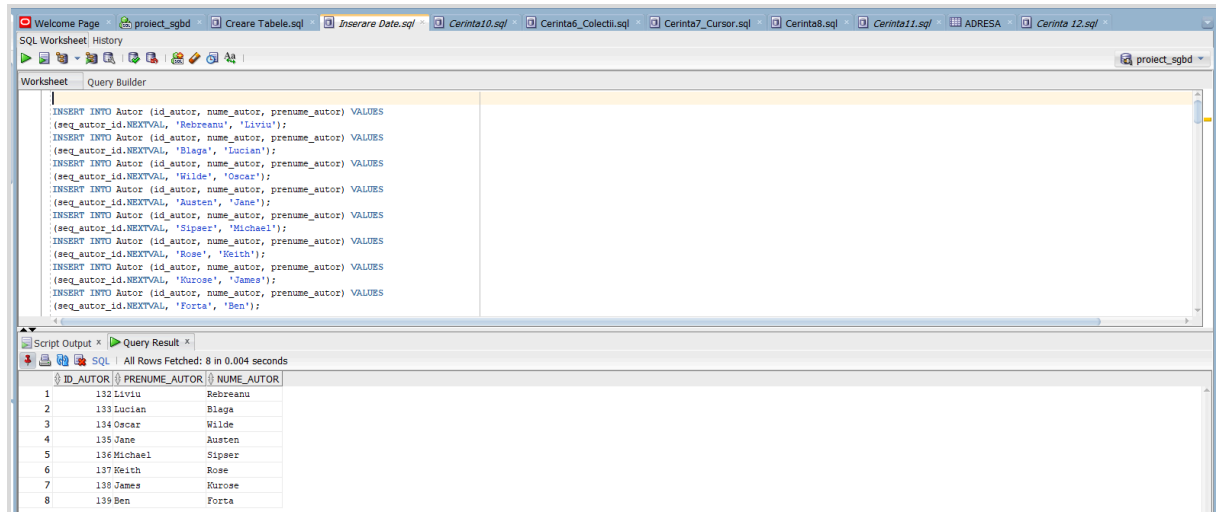
Cerinta 5.

Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

```
INSERT INTO Autor (id_autor, nume_autor, prenume_autor) VALUES
(seq_autor_id.NEXTVAL, 'Rebreanu', 'Liviu');
INSERT INTO Autor (id_autor, nume_autor, prenume_autor) VALUES
(seq_autor_id.NEXTVAL, 'Blaga', 'Lucian');
INSERT INTO Autor (id_autor, nume_autor, prenume_autor) VALUES
(seq_autor_id.NEXTVAL, 'Wilde', 'Oscar');
INSERT INTO Autor (id_autor, nume_autor, prenume_autor) VALUES
(seq_autor_id.NEXTVAL, 'Austen', 'Jane');
INSERT INTO Autor (id_autor, nume_autor, prenume_autor) VALUES
(seq_autor_id.NEXTVAL, 'Sipser', 'Michael');
INSERT INTO Autor (id_autor, nume_autor, prenume_autor) VALUES
(seq_autor_id.NEXTVAL, 'Rose', 'Keith');
```



```
INSERT INTO Autor (id_autor, nume_autor, prenume_autor) VALUES
(seq_autor_id.NEXTVAL, 'Kurose', 'James');
INSERT INTO Autor (id_autor, nume_autor, prenume_autor) VALUES
(seq_autor_id.NEXTVAL, 'Forta', 'Ben');
```



```
INSERT INTO Categorie (id_cat, nume_categorie, descriere) VALUES
(seq_cat_id.NEXTVAL, 'Fictiune', 'Descopera lumi imaginare si povesti captivante care trezesc emotii si
stimuleaza imaginatia, de la romane epice la povestiri fascinante.');
```

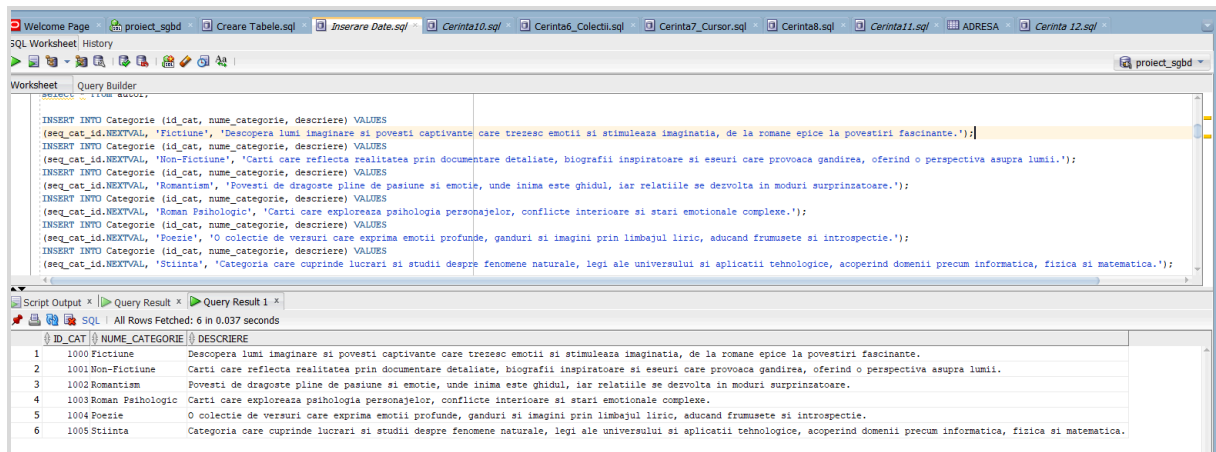
```
INSERT INTO Categorie (id_cat, nume_categorie, descriere) VALUES
(seq_cat_id.NEXTVAL, 'Non-Fictiune', 'Carti care reflecta realitatea prin documentare detaliate, biografii
inspiratoare si eseuri care provoaca gandirea, oferind o perspectiva asupra lumii.');
```

```
INSERT INTO Categorie (id_cat, nume_categorie, descriere) VALUES
(seq_cat_id.NEXTVAL, 'Romantism', 'Povesti de dragoste pline de pasiune si emotie, unde inima este ghidul,
iar relatiile se dezvoltă in moduri surprinzătoare.');
```

```
INSERT INTO Categorie (id_cat, nume_categorie, descriere) VALUES
(seq_cat_id.NEXTVAL, 'Roman Psihologic', 'Carti care exploreaza psihologia personajelor, conflicte interioare
si stari emotionale complexe.');
```

```
INSERT INTO Categorie (id_cat, nume_categorie, descriere) VALUES
(seq_cat_id.NEXTVAL, 'Poezie', 'O colectie de versuri care exprima emotii profunde, ganduri si imagini prin
limbajul liric, aducand frumuseti si introspectie.');
```

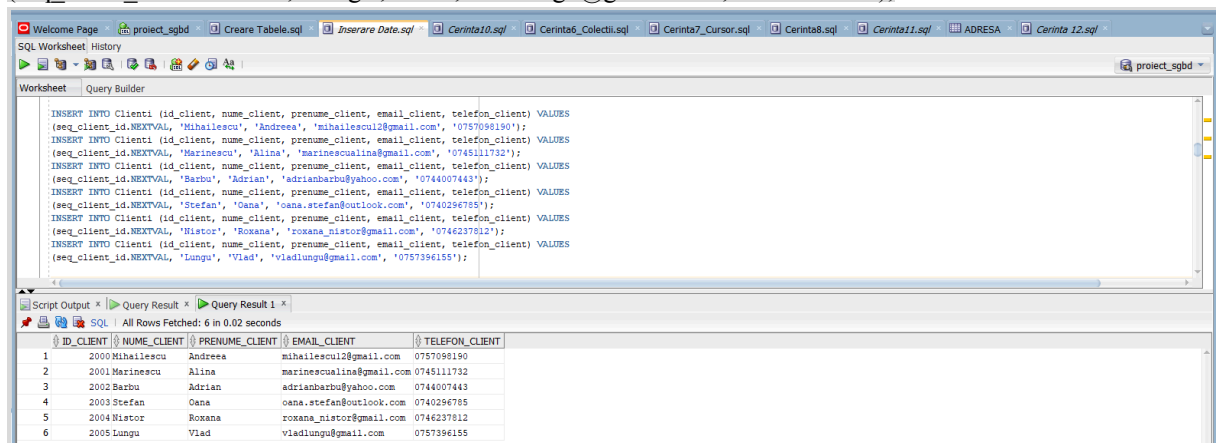
```
INSERT INTO Categorie (id_cat, nume_categorie, descriere) VALUES
(seq_cat_id.NEXTVAL, ' Stiinta', 'Categorii care cuprind lucrari si studii despre fenomene naturale, legi ale
universului si aplicatii tehnologice, acoperind domenii precum informatica, fizica si matematica.);
```



```

INSERT INTO Clienti (id_client, nume_client, prenume_client, email_client, telefon_client) VALUES
(seq_client_id.NEXTVAL, 'Mihailescu', 'Andreea', 'mihailescu12@gmail.com', '0757098190');
INSERT INTO Clienti (id_client, nume_client, prenume_client, email_client, telefon_client) VALUES
(seq_client_id.NEXTVAL, 'Marinescu', 'Alina', 'marinescualina@gmail.com', '0745111732');
INSERT INTO Clienti (id_client, nume_client, prenume_client, email_client, telefon_client) VALUES
(seq_client_id.NEXTVAL, 'Barbu', 'Adrian', 'adrianbarbu@yahoo.com', '0744007443');
INSERT INTO Clienti (id_client, nume_client, prenume_client, email_client, telefon_client) VALUES
(seq_client_id.NEXTVAL, 'Stefan', 'Oana', 'oana.stefan@outlook.com', '0740296785');
INSERT INTO Clienti (id_client, nume_client, prenume_client, email_client, telefon_client) VALUES
(seq_client_id.NEXTVAL, 'Nistor', 'Roxana', 'roxana_nistor@gmail.com', '0746237812');
INSERT INTO Clienti (id_client, nume_client, prenume_client, email_client, telefon_client) VALUES
(seq_client_id.NEXTVAL, 'Lungu', 'Vlad', 'vladlungu@gmail.com', '0757396155');

```



```

INSERT INTO Adresa (id_adresa, adresa, oras, judet, cod_postal, tara) VALUES
(seq_adresa_id.NEXTVAL, 'Strada Ion Creangă, nr 11', 'Râmnicu Vâlcea', 'Vâlcea', '240001', 'România');
INSERT INTO Adresa (id_adresa, adresa, oras, judet, cod_postal, tara) VALUES
(seq_adresa_id.NEXTVAL, 'Strada Aurel Vlaicu, nr 10', 'București', 'București', '010202', 'România');
INSERT INTO Adresa (id_adresa, adresa, oras, judet, cod_postal, tara) VALUES
(seq_adresa_id.NEXTVAL, 'Strada Calea Moșilor, nr 45', 'București', 'București', '020303', 'România');

```

```

INSERT INTO Adresa (id_adresa, adresa, oras, judet, cod_postal, tara) VALUES
(seq_adresa_id.NEXTVAL, 'Strada Tomis, nr 7', 'Constanța', 'Constanța', '900003', 'România');
INSERT INTO Adresa (id_adresa, adresa, oras, judet, cod_postal, tara) VALUES
(seq_adresa_id.NEXTVAL, 'Strada Grivița, nr 12', 'Tulcea', 'Tulcea', '820001', 'România');
INSERT INTO Adresa (id_adresa, adresa, oras, judet, cod_postal, tara) VALUES
(seq_adresa_id.NEXTVAL, 'Strada Spitalului, nr 25', 'Tulcea', 'Tulcea', '820063', 'România');

```

ID_ADRESA	ADRESA	ORAS	JUDET	COD_POSTAL	TARA
1	3000 Strada Ion Creanga, nr 11	Râmnicu Vâlcea Vâlcea	240001	România	
2	3001 Strada Aurel Vlaicu, nr 10	București	010202	România	
3	3002 Strada Calea Mosilor, nr 45	București	020303	România	
4	3003 Strada Tomis, nr 7	Constanța	Constanța	900003	România
5	3004 Strada Grivița, nr 12	Tulcea	Tulcea	820001	România
6	3005 Strada Spitalului, nr 25	Tulcea	Tulcea	820063	România

```

INSERT INTO Editura (id_editura, nume_editura) VALUES
(seq_editura_id.NEXTVAL, 'Humanitas');
INSERT INTO Editura (id_editura, nume_editura) VALUES
(seq_editura_id.NEXTVAL, 'Polirom');
INSERT INTO Editura (id_editura, nume_editura) VALUES
(seq_editura_id.NEXTVAL, 'RAO');
INSERT INTO Editura (id_editura, nume_editura) VALUES
(seq_editura_id.NEXTVAL, 'Litera');
INSERT INTO Editura (id_editura, nume_editura) VALUES
(seq_editura_id.NEXTVAL, 'Nemira');

```

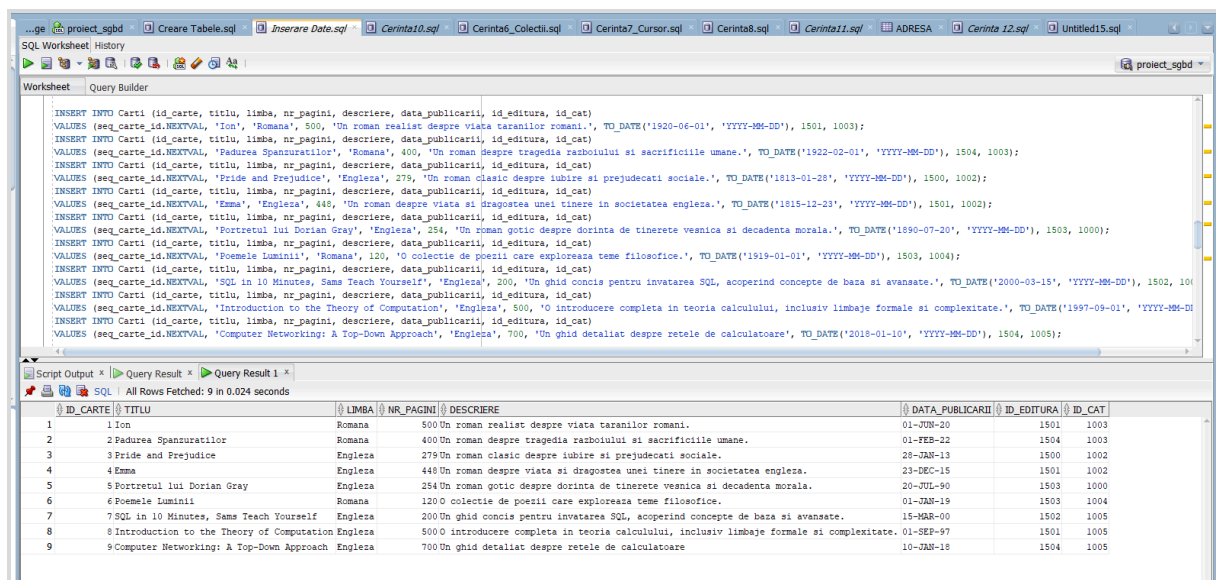
ID_EDITURA	NUME_EDITURA
1	1500 Humanitas
2	1501 Polirom
3	1502 RAO
4	1503 Litera
5	1504 Nemira

```

INSERT INTO Carti (id_carte, titlu, limba, nr_pagini, descriere, data_publicarii, id_editura, id_cat)
VALUES (seq_carte_id.NEXTVAL, 'Ion', 'Romana', 500, 'Un roman realist despre viata taranilor romani.',
TO_DATE('1920-06-01', 'YYYY-MM-DD'), 1501, 1003);
INSERT INTO Carti (id_carte, titlu, limba, nr_pagini, descriere, data_publicarii, id_editura, id_cat)

```

```
VALUES (seq_carte_id.NEXTVAL, 'Padurea Spanzuratiilor', 'Romana', 400, 'Un roman despre tragedia
razboiului si sacrificiile umane.', TO_DATE('1922-02-01', 'YYYY-MM-DD'), 1504, 1003);
INSERT INTO Carti (id_carte, titlu, limba, nr_pagini, descriere, data_publicarii, id_editura, id_cat)
VALUES (seq_carte_id.NEXTVAL, 'Pride and Prejudice', 'Engleza', 279, 'Un roman clasic despre iubire si
prejudicati sociale.', TO_DATE('1813-01-28', 'YYYY-MM-DD'), 1500, 1002);
INSERT INTO Carti (id_carte, titlu, limba, nr_pagini, descriere, data_publicarii, id_editura, id_cat)
VALUES (seq_carte_id.NEXTVAL, 'Emma', 'Engleza', 448, 'Un roman despre viata si dragostea unei tinere in
societatea engleza.', TO_DATE('1815-12-23', 'YYYY-MM-DD'), 1501, 1002);
INSERT INTO Carti (id_carte, titlu, limba, nr_pagini, descriere, data_publicarii, id_editura, id_cat)
VALUES (seq_carte_id.NEXTVAL, 'Portretul lui Dorian Gray', 'Engleza', 254, 'Un roman gotic despre dorinta
de tinerețe vesnica si decadenta morala.', TO_DATE('1890-07-20', 'YYYY-MM-DD'), 1503, 1000);
INSERT INTO Carti (id_carte, titlu, limba, nr_pagini, descriere, data_publicarii, id_editura, id_cat)
VALUES (seq_carte_id.NEXTVAL, 'Poemele Luminii', 'Romana', 120, 'O colectie de poezii care exploreaza
teme filosofice.', TO_DATE('1919-01-01', 'YYYY-MM-DD'), 1503, 1004);
INSERT INTO Carti (id_carte, titlu, limba, nr_pagini, descriere, data_publicarii, id_editura, id_cat)
VALUES (seq_carte_id.NEXTVAL, 'SQL in 10 Minutes, Sams Teach Yourself', 'Engleza', 180, 'Un ghid concis
pentru invatarea SQL, acoperind concepte de baza si avansate.', TO_DATE('2000-03-15', 'YYYY-MM-DD'),
1502, 1005);
INSERT INTO Carti (id_carte, titlu, limba, nr_pagini, descriere, data_publicarii, id_editura, id_cat)
VALUES (seq_carte_id.NEXTVAL, 'Introduction to the Theory of Computation', 'Engleza', 500, 'O introducere
completa in teoria calculului, inclusiv limbaje formale si complexitate.', TO_DATE('1997-09-01',
'YYYY-MM-DD'), 1501, 1005);
INSERT INTO Carti (id_carte, titlu, limba, nr_pagini, descriere, data_publicarii, id_editura, id_cat)
VALUES (seq_carte_id.NEXTVAL, 'Computer Networking: A Top-Down Approach', 'Engleza', 700, 'Un ghid
detaliat despre retele de calculatoare', TO_DATE('2018-01-10', 'YYYY-MM-DD'), 1504, 1005);
```



The screenshot shows a SQL IDE interface. The top window is the 'Query Builder' for a table named 'Carti'. It contains an INSERT statement with 9 rows of data. The bottom window is the 'Query Result' showing the same 9 rows of data in a table format.

ID_CARTE	TITLU	LIMBA	NR_PAGINI	DESCRIERE	DATA_PUBLICARII	ID_EDITURA	ID_CAT
1	Ion	Romana	500	Un roman realist despre viata taranilor romani.	01-JUN-20	1501	1003
2	Padurea Spanzuratiilor	Romana	400	Un roman despre tragedia razboiului si sacrificiile umane.	01-FEB-22	1504	1003
3	Pride and Prejudice	Engleza	279	Un roman clasic despre iubire si prejudicati sociale.	28-JAN-13	1500	1002
4	Emma	Engleza	448	Un roman despre viata si dragostea unei tinere in societatea engleza.	23-DEC-15	1501	1002
5	Portretul lui Dorian Gray	Engleza	254	Un roman gotic despre dorinta de tinerețe vesnica si decadenta morala.	20-JUL-90	1503	1000
6	Poemele Luminii	Romana	120	O colectie de poezii care exploreaza teme filosofice.	01-JAN-19	1503	1004
7	SQL in 10 Minutes, Sams Teach Yourself	Engleza	200	Un ghid concis pentru invatarea SQL, acoperind concepte de baza si avansate.	15-MAR-00	1502	1005
8	Introduction to the Theory of Computation	Engleza	500	O introducere completa in teoria calculului, inclusiv limbaje formale si complexitate.	01-SEP-97	1501	1005
9	Computer Networking: A Top-Down Approach	Engleza	700	Un ghid detaliat despre retele de calculatoare	10-JAN-18	1504	1005

```
INSERT INTO Comenzi (id_comanda, data_comenzii, id_client, id_adresa)
VALUES (seq_comanda_id.NEXTVAL, TO_DATE('2024-01-15', 'YYYY-MM-DD'), 2003, 3002);
INSERT INTO Comenzi (id_comanda, data_comenzii, id_client, id_adresa)
VALUES (seq_comanda_id.NEXTVAL, TO_DATE('2024-02-20', 'YYYY-MM-DD'), 2001, 3001);
INSERT INTO Comenzi (id_comanda, data_comenzii, id_client, id_adresa)
```

```
VALUES (seq_comanda_id.NEXTVAL, TO_DATE('2024-03-25', 'YYYY-MM-DD'), 2005, 3005);
INSERT INTO Comenzi (id_comanda, data_comenzii, id_client, id_adresa)
VALUES (seq_comanda_id.NEXTVAL, TO_DATE('2024-04-30', 'YYYY-MM-DD'), 2000, 3003);
INSERT INTO Comenzi (id_comanda, data_comenzii, id_client, id_adresa)
VALUES (seq_comanda_id.NEXTVAL, TO_DATE('2024-05-10', 'YYYY-MM-DD'), 2002, 3004);
INSERT INTO Comenzi (id_comanda, data_comenzii, id_client, id_adresa)
VALUES (seq_comanda_id.NEXTVAL, TO_DATE('2024-06-12', 'YYYY-MM-DD'), 2003, 3000);
INSERT INTO Comenzi (id_comanda, data_comenzii, id_client, id_adresa)
VALUES (seq_comanda_id.NEXTVAL, TO_DATE('2024-07-22', 'YYYY-MM-DD'), 2003, 3001);
```

The screenshot shows a SQL Worksheet with a query in the 'Query Builder' tab. The query is an INSERT statement for the 'Comenzi' table. The 'Query Result' tab shows the results of the query, which are 7 rows of data. The columns are ID_COMANDA, DATA_COMENZII, ID_CLIENT, and ID_ADRESA.

ID_COMANDA	DATA_COMENZII	ID_CLIENT	ID_ADRESA
1	6000 15-JAN-24	2003	3002
2	6001 20-FEB-24	2001	3001
3	6002 25-MAR-24	2005	3005
4	6003 30-APR-24	2000	3003
5	6004 10-MAY-24	2002	3004
6	6005 12-JUN-24	2003	3000
7	6006 22-JUL-24	2003	3001

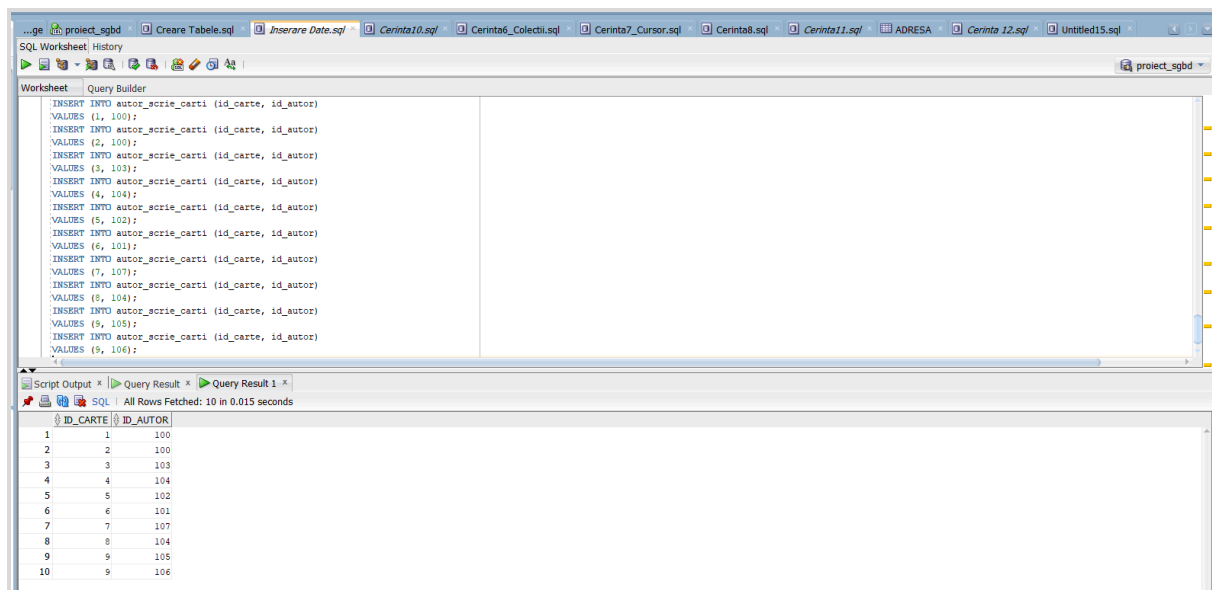
```
INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret)
VALUES (1, 6000, 55.00);
INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret)
VALUES (2, 6001, 60.00);
INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret)
VALUES (3, 6003, 75.00);
INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret)
VALUES (4, 6006, 70.00);
INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret)
VALUES (5, 6001, 45.00);
INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret)
VALUES (6, 6002, 40.00);
INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret)
VALUES (7, 6004, 85.00);
INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret)
VALUES (8, 6005, 110.00);
INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret)
VALUES (9, 6006, 95.00);
```

<pre> INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret) VALUES (1, 6000, 55.00); INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret) VALUES (2, 6001, 60.00); INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret) VALUES (3, 6003, 75.00); INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret) VALUES (4, 6006, 70.00); INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret) VALUES (5, 6001, 45.00); INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret) VALUES (6, 6002, 40.00); INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret) VALUES (7, 6004, 85.00); INSERT INTO comanda_contine_carti (id_carte, id_comanda, pret) </pre>			
Script Output x Query Result x			
SQL All Rows Fetched: 9 in 0.006 seconds			
ID_CARTE	ID_COMANDA	PRET	
1	1	6000	55
2	2	6001	60
3	3	6003	75
4	4	6006	70
5	5	6001	45
6	6	6002	40
7	7	6004	85
8	8	6005	110
9	9	6006	95

```

INSERT INTO autor_scrie_carti (id_carte, id_autor)
VALUES (1, 100);
INSERT INTO autor_scrie_carti (id_carte, id_autor)
VALUES (2, 100);
INSERT INTO autor_scrie_carti (id_carte, id_autor)
VALUES (3, 103);
INSERT INTO autor_scrie_carti (id_carte, id_autor)
VALUES (4, 104);
INSERT INTO autor_scrie_carti (id_carte, id_autor)
VALUES (5, 102);
INSERT INTO autor_scrie_carti (id_carte, id_autor)
VALUES (6, 101);
INSERT INTO autor_scrie_carti (id_carte, id_autor)
VALUES (7, 107);
INSERT INTO autor_scrie_carti (id_carte, id_autor)
VALUES (8, 104);
INSERT INTO autor_scrie_carti (id_carte, id_autor)
VALUES (9, 105);
INSERT INTO autor_scrie_carti (id_carte, id_autor)
VALUES (9, 106);

```



Cerinta 6.

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

Implementati un subprogram stocat independent care să analizeze comenzile unui client și să returneze:

1. Lista cărților comandate de acel client .
2. Totalul valorii comenzilor plasate de client
3. Lista unică a editurilor de la care au fost cumpărate cărți.

```
SET SERVEROUTPUT ON;
```

```
-- definirea tipurilor de colectii
```

```
CREATE OR REPLACE TYPE lista_carti AS TABLE OF VARCHAR2(100); --varray pentru a stoca  
titlurile cartilor comandate de client
```

```
/
```

```
CREATE OR REPLACE TYPE total_comenzi AS TABLE OF NUMBER; --tablou indexat pentru a mapa  
id-urile comenzilor cu totalul acestora
```

```
/
```

```
CREATE OR REPLACE TYPE lista_edituri AS TABLE OF VARCHAR2(100); --nested table pentru a retine  
editurile de la care au fost comandate cartile
```

```
/
```

```
CREATE OR REPLACE PROCEDURE analiza_comenzi_client(
```

```
    p_nume_client IN VARCHAR2 --parametru de intrare
```

```
) AS
```

```
    v_id_client Clienti.id_client%TYPE; --var pentru a stoca id-ul clientului
```

```
    v_nr_comenzi NUMBER; --var pentru nr total de comenzi plasate de comenzi
```

```
    -- varray pentru lista cartilor comandate
```

```
    v_carti lista_carti := lista_carti();
```

```
    -- asociative array pentru a stoca totalul comenzilor, indexate dupa id-ul comenzii
```

```
    TYPE mapare_comenzi IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
```

```
    v_total_comenzi mapare_comenzi;
```

```
    -- initializarea unui nested table pentru a stoca numele editurilor de la care au fost comandate carti
```

```
    v_edituri lista_edituri := lista_edituri();
```

```
BEGIN
```

```
    -- preluam ID-ul clientului dupa nume
```

```
    SELECT id_client INTO v_id_client
```

```
    FROM Clienti
```

```
    WHERE nume_client = p_nume_client;
```

```
    -- verificam daca clientul are comenzi plasate
```

```
    SELECT COUNT(*) INTO v_nr_comenzi
```

```
    FROM Comenzi
```

```
    WHERE id_client = v_id_client;
```

```
    IF v_nr_comenzi = 0 THEN --daca clientul nu are comenzi plasate , afisam un mesaj si iesim din procedura
```

```
        DBMS_OUTPUT.PUT_LINE('Clientul ' || p_nume_client || ' nu a plasat nicio comanda.');
```

```
        RETURN;
```

```
    END IF;
```

```
    -- colectam cartile comandate intr-un varray
```

```
    SELECT DISTINCT c.titlu BULK COLLECT INTO v_carti
```

```
    FROM Carti c
```

```
    JOIN comanda_contine_carti cc ON c.id_carte = cc.id_carte
```

```
    JOIN Comenzi co ON cc.id_comanda = co.id_comanda
```

```
    WHERE co.id_client = v_id_client;
```

```
    -- colectam totalul comenzilor in asociative array
```

```
    FOR rec IN (SELECT co.id_comanda, SUM(cc.pret) AS total
```



```

        FROM Comenzi co
        JOIN comanda_contine_carti cc ON co.id_comanda = cc.id_comanda
        WHERE co.id_client = v_id_client
        GROUP BY co.id_comanda)
    LOOP
        v_total_comenzi(rec.id_comanda) := rec.total;
        --asociem fiecarei id de comanda valoarea totala a comenzii respective
    END LOOP;

    -- colectam editurile unice intr-un nested table
    SELECT DISTINCT e.ume_editura BULK COLLECT INTO v_edituri
    FROM Editura e
    JOIN Carti c ON e.id_editura = c.id_editura
    JOIN comanda_contine_carti cc ON c.id_carte = cc.id_carte
    JOIN Comenzi co ON cc.id_comanda = co.id_comanda
    WHERE co.id_client = v_id_client;

    -- afisam lista cartilor comandate
    DBMS_OUTPUT.PUT_LINE('Carti comandate de clientul ' || p_nume_client || ':');
    FOR i IN 1 .. v_carti.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(' - ' || v_carti(i));
    END LOOP;

    -- afisam totalul comenzilor
    DBMS_OUTPUT.PUT_LINE('Totalul comenzilor plasate:');
    FOR i IN v_total_comenzi.FIRST .. v_total_comenzi.LAST LOOP
        IF v_total_comenzi.EXISTS(i) THEN
            DBMS_OUTPUT.PUT_LINE(' - Comanda ' || i || ': ' || v_total_comenzi(i) || ' RON');
        END IF;
    END LOOP;

    -- afisam lista editurilor
    DBMS_OUTPUT.PUT_LINE('Edituri de la care s-au comandat carti:');
    FOR i IN 1 .. v_edituri.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(' - ' || v_edituri(i));
    END LOOP;

END analiza_comenzi_client;
/

BEGIN
    analiza_comenzi_client('Stefan');
END;
/

```

```
END LOOP;

-- colectam editurile unice intr-un nested table
SELECT DISTINCT e.num_e editura BULK COLLECT INTO v_edituri
FROM Editura e
JOIN Carti c ON e.id_editura = c.id_editura
JOIN comanda_contine_carti cc ON c.id_carte = cc.id_carte
JOIN Comenzi co ON cc.id_comanda = co.id_comanda
WHERE co.id_client = v_id_client;

-- afisam lista cartilor comandate
DBMS_OUTPUT.PUT_LINE('Carti comandate de clientul ' || p_num_client || ':');
FOR i IN 1 .. v_carti.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(' - ' || v_carti(i));
END LOOP;

-- afisam totalul comenzilor
DBMS_OUTPUT.PUT_LINE('Totalul comenzilor plasate:');
FOR i IN v_total_comenzi.FIRST .. v_total_comenzi.LAST LOOP
    IF v_total_comenzi.EXISTS(i) THEN
```

Script Output x

Task completed in 0.927 seconds

PL/SQL procedure successfully completed.

Dbms Output x

Buffer Size: 2000

proiect_sgbd x

Carti comandate de clientul Stefan:

- Ion
- Emma
- Introduction to the Theory of Computation
- Computer Networking: A Top-Down Approach

Totalul comenzilor plasate:

- Comanda 6000: 55 RON
- Comanda 6005: 110 RON
- Comanda 6006: 165 RON

Edituri de la care s-au comandat carti:

- Polirom
- Nemira

Cerinta 7.

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.

Pentru fiecare dintre editurile cu codul 1501, 1503 obtineti prețul, precum și lista titlurilor cărților care fac parte din acele edituri.

```
SET SERVEROUTPUT ON;
```

```
CREATE OR REPLACE PROCEDURE cursor_carti(v_id_editura NUMBER)IS
```

```
CURSOR c_editura IS --cursorul principal
```

```

SELECT id_editura, nume_editura
FROM Editura --selecteaza id_editura si nume_editura din tabelul Editura
WHERE id_editura=v_id_editura ;

```

```

CURSOR c_carti(p_id_editura Editura.id_editura%TYPE) IS --cursorul parametrizat, are ca parametru
p_id_editura
SELECT DISTINCT c.titlu, a.pret, c.limba --selecteaza titlul si pretul pentru fiecare carte din tabelul Carti si
Articol
FROM Carti c
JOIN comanda_contine_carti a ON c.id_carte = a.id_carte
WHERE c.id_editura = p_id_editura; --filtreaza cartile pentru care id_editura din tabelul Carti este egal cu
valoarea parametrului p_id_editura

```

```

v_editura_id Editura.id_editura%TYPE; --variabila folosita pentru a stoca id-ul editurii
v_editura_nume Editura.nume_editura%TYPE; --variabila folosita pentru a stoca numele editurii

```

```

BEGIN

```

```

FOR r_editura IN c_editura LOOP --folosind cursorul principal itereaza prin edituri
v_editura_id := r_editura.id_editura; --asigneaza variabilelor valorile
v_editura_nume := r_editura.nume_editura;

```

```

DBMS_OUTPUT.PUT_LINE('Editura' || ' ' || v_editura_nume); --afiseaza numele editurii

```

```

FOR r_carti IN c_carti(v_editura_id) LOOP --itereaza prin cartile din editura curenta si foloseste ca
parametru id-ul editurii curente
DBMS_OUTPUT.PUT_LINE(' Titlu: ' || r_carti.titlu || ', Pret: ' || r_carti.pret); --afiseaza titlul si pretul
cartilor corespunzatoare
END LOOP;

```

```

DBMS_OUTPUT.PUT_LINE('-----');
END LOOP;

```

```

END cursor_carti;
/
execute cursor_carti(1501);
/
execute cursor_carti(1504);
/

```

```
BEGIN
FOR r_editura IN c_editura LOOP --folosind cursorul principal itereaza prin edituri
v_editura_id := r_editura.id_editura; --asigneaza variabilelor valorile
v_editura_nume := r_editura.nume_editura;

DBMS_OUTPUT.PUT_LINE('Editura' || ' ' || v_editura_nume); --afiseaza numele editurii

FOR r_carti IN c_carti(v_editura_id) LOOP --itereaza prin cartile din editura curenta si foloseste ca parametru id-ul editurii curente
DBMS_OUTPUT.PUT_LINE(' Titlu: ' || r_carti.titlu || ', Pret: ' || r_carti.pret); --afiseaza titlul si pretul cartilor corespunzatoare
END LOOP;

DBMS_OUTPUT.PUT_LINE('-----');
END LOOP;

END cursor_carti;
/
execute cursor_carti(1501);
/
execute cursor_carti(1504);
/
```

Script Output x

Task completed in 0.048 seconds

Dbms Output x

Buffer Size:2000

proiect_sgbd x

Editura Polirom
Titlu: Ion, Pret: 55
Titlu: Emma, Pret: 70
Titlu: Introduction to the Theory of Computation, Pret: 110

Editura Nemira
Titlu: Padurea Spanzuratiilor, Pret: 60
Titlu: Computer Networking: A Top-Down Approach, Pret: 95

Cerinta 8.

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile `NO_DATA_FOUND` și `TOO_MANY_ROWS`. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

O librărie online gestionează un sistem de vânzare a cărților, unde fiecare client poate plasa una sau mai multe comenzi. Fiecare comandă conține una sau mai multe cărți, fiecare având un preț individual.

Librăria își dorește să implementeze un sistem care să permită calcularea valorii totale a comenzilor unui client, pe baza numelui acestuia.

Tratarea exceptiilor:

NO_DATA_FOUND – dacă clientul nu există.

TOO_MANY_ROWS – dacă există mai mulți clienți cu același nume.

```
SET SERVEROUTPUT ON;
```

```
-- definim functia care returneaza valoarea comenzilor unui client
```

```
CREATE OR REPLACE FUNCTION valoare_totala_comenzi(
```

```
  p_nume_client IN VARCHAR2 --parametrul de intrare adica numele clientului
```

```
) RETURN NUMBER IS
```

```
  v_total NUMBER := 0; -- variabila pentru suma totala a comenzilor
```

```
  v_count NUMBER; -- variabila pentru a verifica numarul de clienti gasiti
```

```
  v_client_id Clienti.id_client%TYPE; -- var pentru a stoca id-ul clientului
```

```
BEGIN
```

```
  -- verificam cate inregistrari exista pentru acest client
```

```
  SELECT COUNT(*)
```

```
  INTO v_count
```

```
  FROM Clienti
```

```
  WHERE nume_client = p_nume_client;
```

```
  -- daca clientul nu exista, tratam exceptia
```

```
  IF v_count = 0 THEN
```

```
    RAISE NO_DATA_FOUND; --exceptia NO_DATA_FOUND daca nu exista niciun client cu numele dat
```

```
  ELSIF v_count > 1 THEN
```

```
    RAISE TOO_MANY_ROWS; --exceptia TOO_MANY_ROWS daca exista mai multi clienti cu numele
```

```
  dat
```

```
  END IF;
```

```
  -- obtinem ID-ul clientului
```

```
  SELECT id_client
```

```
  INTO v_client_id
```

```
  FROM Clienti
```

```
  WHERE nume_client = p_nume_client;
```

```

-- calculam valoarea totala a comenzilor clientului intr-o singura interogare SQL
SELECT COALESCE(SUM(cc.pret), 0) --evitam situatia cand suma ar fi nula
INTO v_total
FROM Clienti c
JOIN Comenzi co ON c.id_client = co.id_client
JOIN Comanda_contine_carti cc ON co.id_comanda = cc.id_comanda
WHERE c.num_e_client = p_num_e_client;

-- returnam valoarea totala a comenzilor
RETURN v_total;

EXCEPTION
-- exceptie in cazul in care clientul nu este gasit in baza de date
WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Eroare: clientul "' || p_num_e_client || '" nu exista.');
```

```

    RETURN 0;

WHEN TOO_MANY_ROWS THEN
--exceptie in cazul in care exisra mai multi clienti cu acelasi numar
    DBMS_OUTPUT.PUT_LINE('Eroare: exista mai multi clienti cu numele "' || p_num_e_client || '"');
    RETURN 0;

WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Eroare neasteptata: ' || SQLERRM);
    RETURN NULL;
END valoare_totala_comenzi;
/

BEGIN
--apel functie pentru un client care exista in baza de date si are comenzi
    DBMS_OUTPUT.PUT_LINE('Valoarea totala a comenzilor pentru Mihailescu: ' ||
valoare_totala_comenzi('Mihailescu') || ' RON');
```

```

END;
/

BEGIN
--apel functie pentru un client care nu exista in baza de date, se va declansa NO_DATA_FOUND
    DBMS_OUTPUT.PUT_LINE('Valoarea totala a comenzilor pentru Dobromirescu: ' ||
valoare_totala_comenzi('Dobromirescu'));
```

```

END;
/

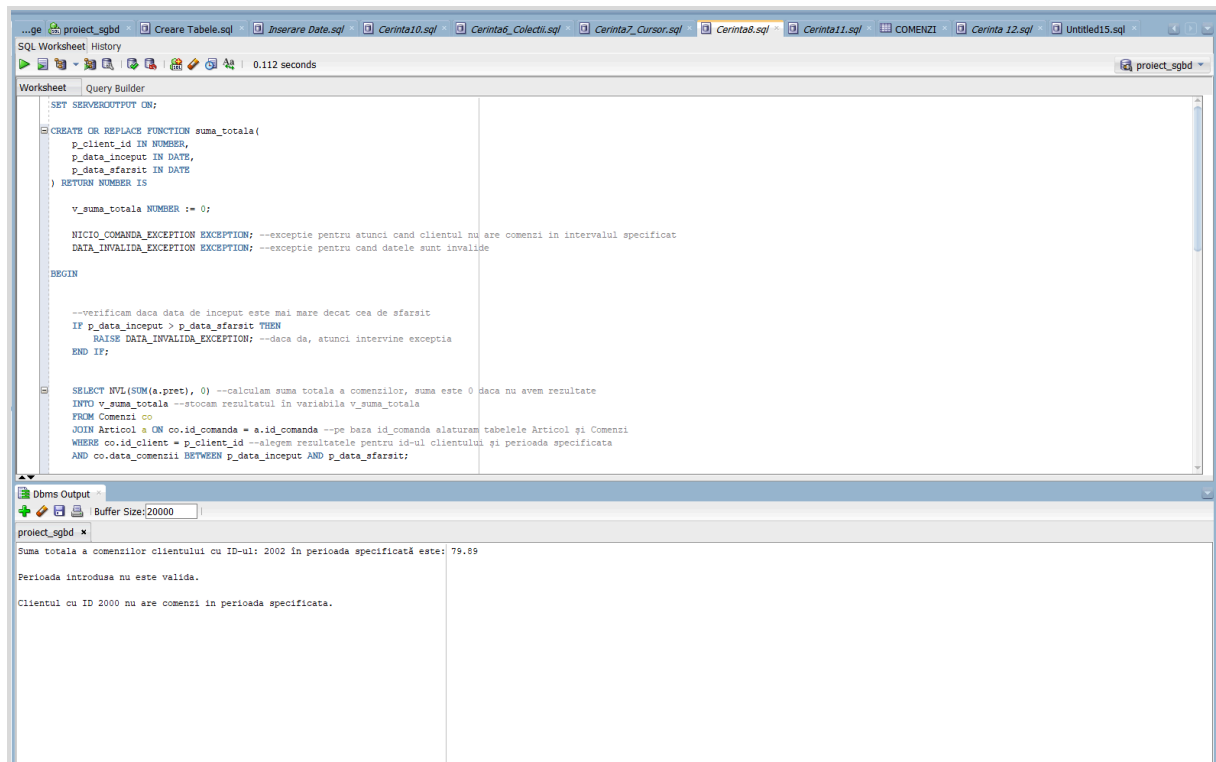
BEGIN
--apel functie pentru un client al carui nume apare de mai multe ori, se va declansa TOO_MANY_ROWS
    DBMS_OUTPUT.PUT_LINE('Valoarea totala a comenzilor pentru Lungu: ' ||
valoare_totala_comenzi('Lungu'));
```

```

END;
/

/

```



Cerinta 9.

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.

. Fiecare comandă conține detalii despre client, adresa de livrare și cărțile comandate. Se dorește implementarea unui sistem care verifică existența și conținutul unei comenzi, afișând detalii relevante.

Se va crea o procedură stocată care primește două argumente: ID-ul comenzii care trebuie verificată și ID-ul clientului care a plasat comanda. Procedura verifică dacă există comanda pentru clientul specificat. Dacă

aceasta nu există, va genera o eroare personalizată și va afișa un mesaj corespunzător.

În cazul în care comanda există, se extrag informațiile despre client, inclusiv numele și prenumele său, adresa unde trebuie livrată comanda și data în care aceasta a fost plasată. Ulterior, se obține lista cărților comandate, editura care a publicat fiecare carte și prețul aferent fiecărei cărți.

```
SET SERVEROUTPUT ON;
```

```
CREATE OR REPLACE PROCEDURE verificare_comanda(
  p_id_comanda IN Comenzi.id_comanda%TYPE, -- id-ul comenzii de verificat
  p_id_client IN Clienti.id_client%TYPE -- id-ul clientului care a plasat comanda
) IS
  -- variabile pentru informatiile despre client si comanda
  v_num_client Clienti.num_client%TYPE; --var pentru numele clientului
  v_prenume_client Clienti.prenume_client%TYPE; --var prenume client
  v_adresa Adresa.adresa%TYPE; --var pentru adresa comenzii
  v_data_comenzii Comenzi.data_comenzii%TYPE; --data comenzii

  -- colectii pentru cartile comandate
  TYPE t_titlu_carti IS TABLE OF Carti.titlu%TYPE; --tabel indexat pentru titlurile cartilor
  TYPE t_editura IS TABLE OF Editura.num_editura%TYPE; --tabel indexat pentru numele editurilor
  TYPE t_pret IS TABLE OF Comanda_contine_carti.pret%TYPE; --tabel pentru preturi

  v_titluri_carti t_titlu_carti;
  v_edituri t_editura;
  v_preturi t_pret;

  -- verificam existenta comenzii
  v_count NUMBER;

  -- definim exceptiile personalizate
  ex_comanda_neexistenta EXCEPTION; --exceptia daca comanda nu exista
  ex_comanda_fara_produce EXCEPTION; --comanda este in baza de date, dar nu are nicio carte

BEGIN
  -- Verificam daca exista comanda
  SELECT COUNT(*) INTO v_count
  FROM Comenzi
  WHERE id_comanda = p_id_comanda AND id_client = p_id_client;

  -- Daca nu exista, declansam exceptia personalizata ex_comanda_neexistenta
  IF v_count = 0 THEN
    RAISE ex_comanda_neexistenta;
  END IF;
```



```

-- Preluam informatiile comenzilor (client, adresa, data)
SELECT c.ume_client, c.prenume_client, a.adresa, co.data_comenzii
INTO v_ume_client, v_prenume_client, v_adresa, v_data_comenzii
FROM Comenzi co
JOIN Clienti c ON co.id_client = c.id_client
JOIN Adresa a ON co.id_adresa = a.id_adresa
WHERE co.id_comanda = p_id_comanda;

--preluam informatiile cartilor asociatie comenzii impreuna cu editura si pretul fiecarui produs
SELECT ca.titlu, e.ume_editura, cc.pret
BULK COLLECT INTO v_titluri_carti, v_edituri, v_preturi
FROM Comanda_contine_carti cc
JOIN Carti ca ON cc.id_carte = ca.id_carte
JOIN Editura e ON ca.id_editura = e.id_editura
WHERE cc.id_comanda = p_id_comanda;

-- daca nu sunt carti in comanda, declansam exceptia ex_comanda_fara_produce
IF v_titluri_carti.COUNT = 0 THEN
    RAISE ex_comanda_fara_produce;
END IF;

-- Afisam informatiile despre comanda
DBMS_OUTPUT.PUT_LINE('Client: ' || v_ume_client || ' ' || v_prenume_client);
DBMS_OUTPUT.PUT_LINE('Adresa livrarii: ' || v_adresa);
DBMS_OUTPUT.PUT_LINE('Data comenzii: ' || TO_CHAR(v_data_comenzii, 'DD-MON-YYYY'));
DBMS_OUTPUT.PUT_LINE('Cartile din comanda: ');

-- Afisam lista cartilor comandate si editura lor
FOR i IN 1..v_titluri_carti.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(' - ' || v_titluri_carti(i) || ' (Editura: ' || v_edituri(i) || ') - Pret: ' || v_preturi(i) ||
' RON');
END LOOP;

EXCEPTION
-- Tratatam exceptia pentru comenzi inexistente
WHEN ex_comanda_neexistenta THEN
    DBMS_OUTPUT.PUT_LINE('Eroare: comanda cu ID-ul ' || p_id_comanda || ' nu exista pentru clientul
specificat.');
```

```

-- Tratatam exceptia pentru comenzi fara carti
WHEN ex_comanda_fara_produce THEN
    DBMS_OUTPUT.PUT_LINE('Eroare: comanda cu ID-ul ' || p_id_comanda || ' nu contine carti.');
```

```

-- Tratatam orice alta eroare SQL
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Eroare neasteptata: ' || SQLERRM);
END verificare_comanda;
/

BEGIN
--comanda normala

```

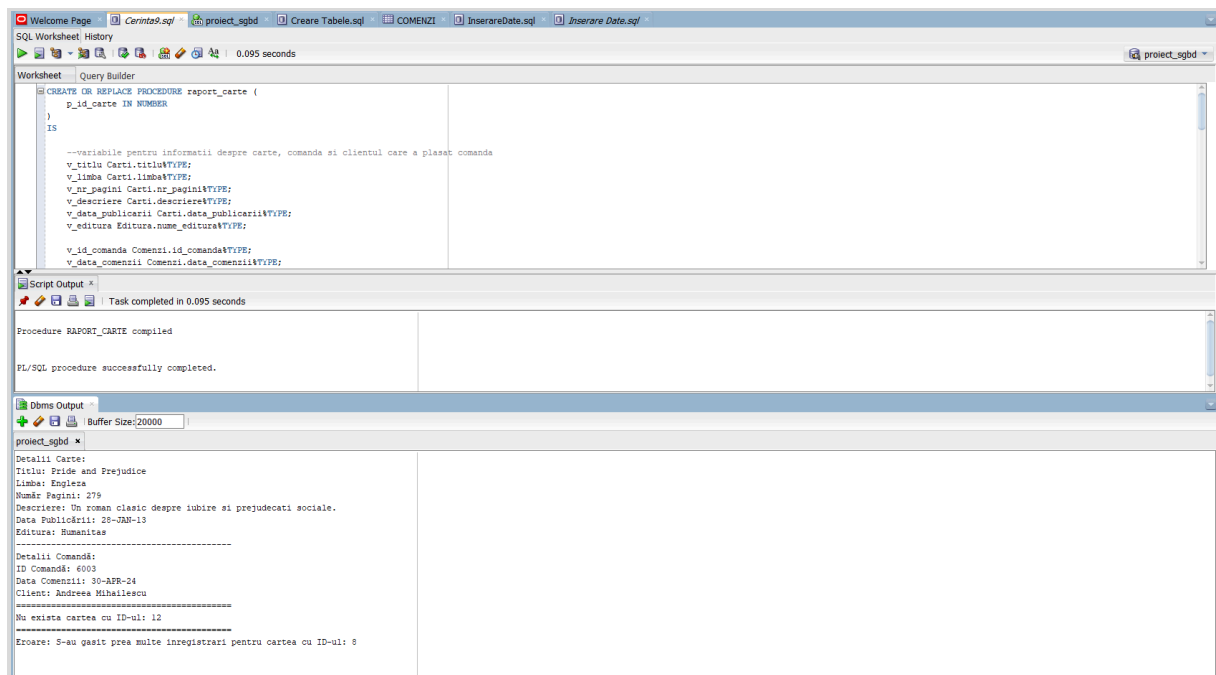
```

    verificare_comanda(6000, 2003);
END;
/

BEGIN
--comanda inexistentă
    verificare_comanda(9999, 2001);
END;
/

BEGIN
--comanda fara produse
    verificare_comanda(6007, 2003);
END;
/

```



Cerinta 10.

**Definiți un trigger de tip LMD la nivel de comandă.
Declanșați trigger-ul.**

Definiți un trigger care previne ștergerea din tabelul Autor.

```

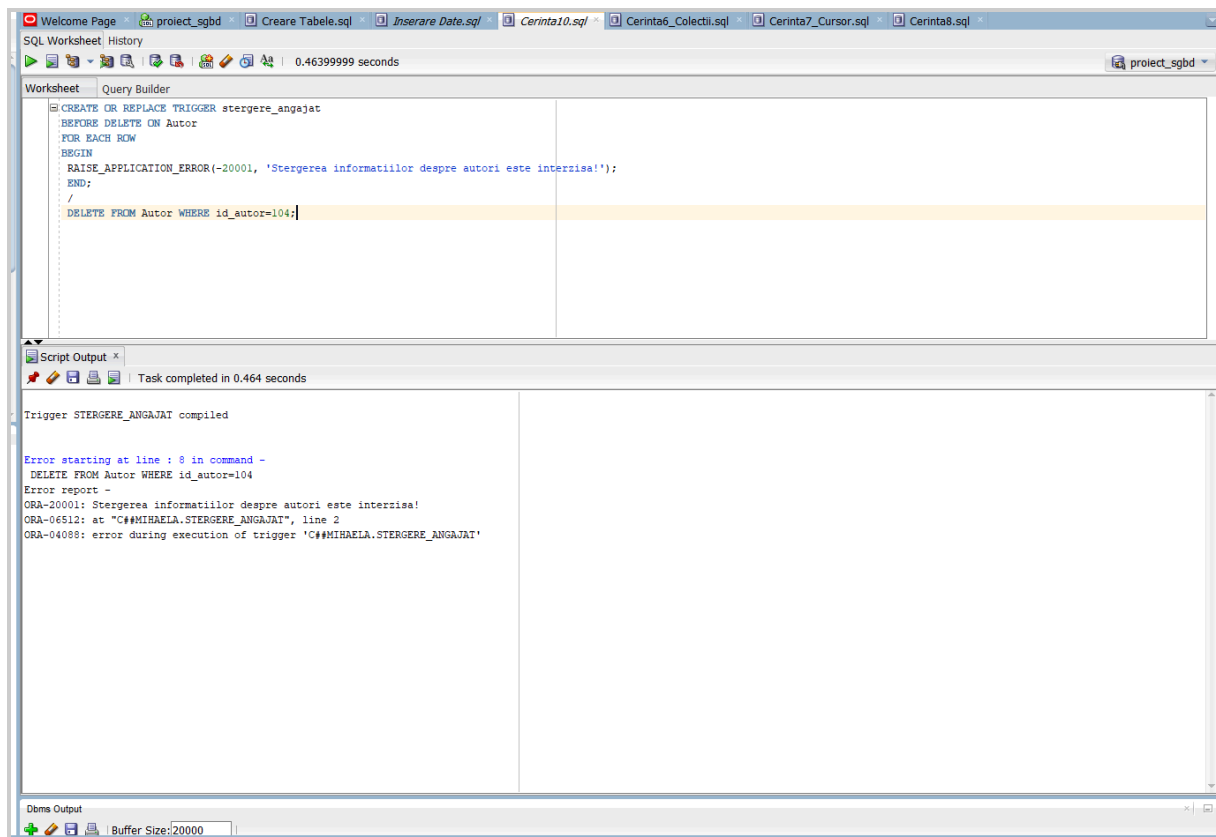
CREATE OR REPLACE TRIGGER stergere_autor --crearea triggerului
BEFORE DELETE ON Autor --triggerul se va declansa inainte de orice op de stergere asupra tablei Autor

```

```

FOR EACH ROW --daca se incearca stergerea mai multor ramduri simultan, triggerul se va executa pentru
fiecare rand
BEGIN
RAISE_APPLICATION_ERROR(-20001, 'Stergerea informatiilor despre autori este interzisa!'); --mesajul care
va fi afisat daca se incearca stergerea datelor din Autor
END;
/
DELETE FROM Autor WHERE id_autor=104;

```



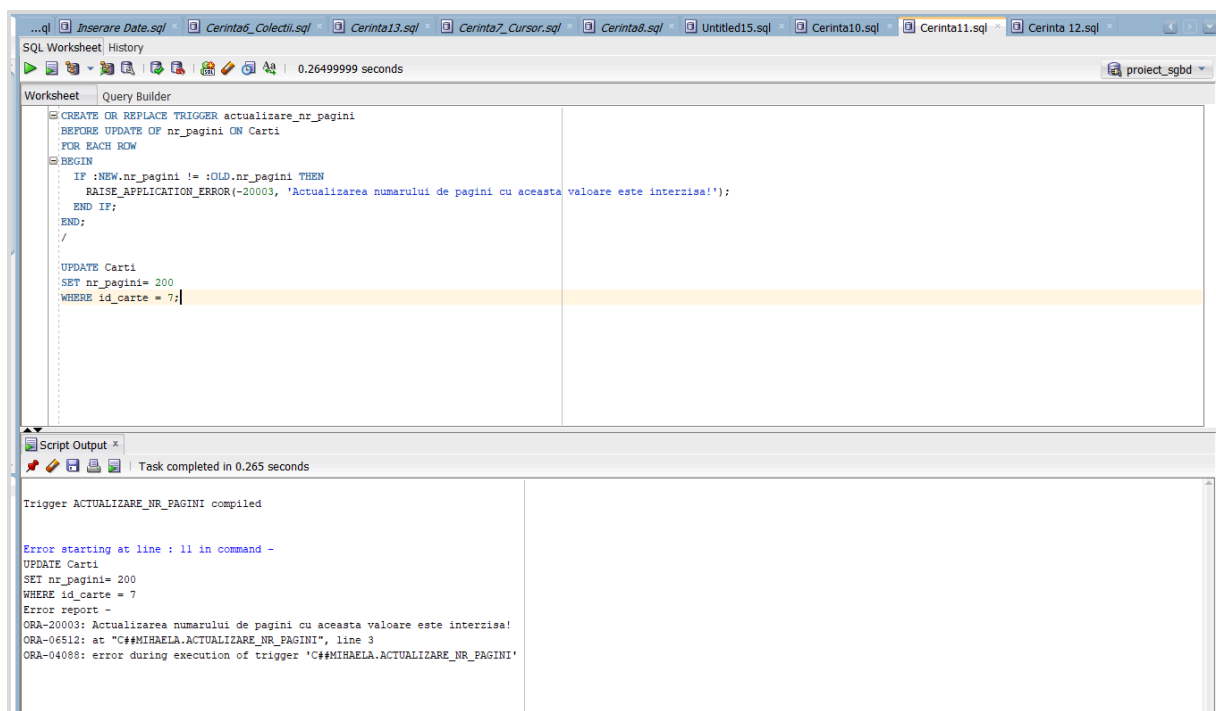
Cerinta 11.

Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

Definiți un trigger care sa prevină actualizarea numărului de pagini al unei cărți cu o valoare mai mica sau mai mare decat cea actuala.

```
CREATE OR REPLACE TRIGGER actualizare_nr_pagini --crearea triggerului
BEFORE UPDATE OF nr_pagini ON Carti --triggerul se declanseaza inainte de o op de actualizare asupra
coloanei nr_pagini
FOR EACH ROW
BEGIN
    IF :NEW.nr_pagini != :OLD.nr_pagini THEN --daca valoarea noua nu corespunde cu cea deja existenta in
tabel, triggerul genereaza eroare
        RAISE_APPLICATION_ERROR(-20003, 'Actualizarea numarului de pagini cu aceasta valoare este
interzisa!'); --mesajul de eroare
    END IF;
END;
/
```

```
UPDATE Carti
SET nr_pagini= 200
WHERE id_carte = 7;
```



Cerinta 12.

Definiți un trigger de tip LDD. Declanșați trigger-ul

Definiți un trigger care înregistrează într-un tabel toate încercările de a șterge o tabela din baza de date.

```
DROP TABLE jurnal_stergeri;
CREATE TABLE jurnal_stergeri ( --tabela unde se vor inregistra incercarile

    operatiune VARCHAR2(50), --denumirea operatiunii efectuate, drop table
    tabela VARCHAR2(50), --numele tablei
    data_operatiunii DATE -- data si ora la care a avut loc operatiunea

);
CREATE OR REPLACE TRIGGER trg_jurnal_stergeri --crearea triggerului
BEFORE DROP ON SCHEMA --triggerul se delanseaza inainte de orice op de stergere a unui table din baza
de date
BEGIN
    IF ORA_DICT_OBJ_TYPE = 'TABLE' THEN --daca obiectul care urmeaza sa fie sters este tabela
        INSERT INTO jurnal_stergeri ( operatiune, tabela, data_operatiunii) --insereaza in jurnal_stergeri
informatiile aferente)
        VALUES ('DROP TABLE', ORA_DICT_OBJ_NAME, SYSDATE);
    END IF;
END;
/

DROP TABLE Autor CASCADE CONSTRAINTS;
DROP TABLE Articol CASCADE CONSTRAINTS;
DROP TABLE Editura CASCADE CONSTRAINTS;

SELECT * FROM jurnal_stergeri;

ROLLBACK;

/

DROP TRIGGER trg_jurnal_stergeri;
```

SQL Worksheet: History

Worksheet: Query Builder

```

DROP TABLE jurnal_stergeri;
CREATE TABLE jurnal_stergeri (
    operatiune VARCHAR2(50),
    tabela VARCHAR2(50),
    data_operatiunii DATE
);
CREATE OR REPLACE TRIGGER trg_jurnal_stergeri
BEFORE DROP ON SCHEMA
BEGIN
    IF ORA_DICT_OBJ_TYPE = 'TABLE' THEN
        INSERT INTO jurnal_stergeri (operatiune, tabela, data_operatiunii)
        VALUES ('DROP TABLE', ORA_DICT_OBJ_NAME, SYSDATE);
    END IF;
END;
/
DROP TABLE Autor CASCADE CONSTRAINTS;
DROP TABLE Articol CASCADE CONSTRAINTS;
DROP TABLE Editura CASCADE CONSTRAINTS;

```

Script Output x Query Result x

SQL | All Rows Fetched: 3 in 0.001 seconds

	OPERATIUNE	TABELA	DATA_OPERATIUNII
1	DROP TABLE	AUTOR	25-AUG-24
2	DROP TABLE	ARTICOL	25-AUG-24
3	DROP TABLE	EDITURA	25-AUG-24