

# **BAZĂ DE DATE PENTRU GESTIUNEA UNEI BIBLIOTECI**

ROGOZA RALUCA-IOANA

GRUPA 234

ANUL 2

# CUPRINS

1. Prezentarea bazei de date
2. Diagrama entitate-relație
3. Diagrama conceptuală
4. Crearea tabelelor și inserarea datelor
5. Exercițiul 6
6. Exercițiul 7
7. Exercițiul 8
8. Exercițiul 9
9. Exercițiul 10
10. Exercițiul 11
11. Exercițiul 12
12. Exercițiul 13
13. Exercițiul 14

# 1. PREZENTAREA BAZEI DE DATE

Acesta este o bază de date pentru o bibliotecă. Ea este utilă, deoarece permite stocarea informațiilor despre bibliotecari, cărți și copiile lor, autorii cărților, editurile de la care se găsesc publicații în bibliotecă și genurile în care se integrează cărțile, cititori, abonamentele posibile pe care le pot deține cititorii, cărțile împrumutate și toate tipurile de penalizări pe care le pot obține cititorii.

Biblioteca permite tuturor oamenilor să împrumute cărți. Bibliotecarii se ocupă de rafturile pe care se găsesc cărțile care sunt variate și aparțin diferitor domenii. Cărțile pot fi unicate sau pot avea mai multe copii. În bibliotecă se găsesc cărți de la mai multe edituri. Cititorii își pot face un abonament care are un preț pe lună diferit în funcție de tipul lui. Cu timpul, tipul de abonament al cititorului poate să difere față de cele pe care le-a avut deja, dar cititorul nu poate să dețină mai multe abonamente în același timp. Cititorii pot primi penalizări pentru nerespectarea regulilor impuse de bibliotecari.

# PREZENTAREA CONSTRÂNGERILOR IMPUSE ASUPRA MODELULUI

- Într-o bibliotecă există mai mulți bibliotecari și, prin urmare, mai mulți bibliotecari se ocupă de rafturi, iar de un raft pot să se ocupe mai mulți bibliotecari. Trebuie ca măcar un bibliotecar să se ocupe de un raft, iar de un raft trebuie să se ocupe măcar un bibliotecar.
- O carte poate avea unul sau mai mulți autori, iar fiecare autor poate să fi scris una sau mai multe cărți. Trebuie ca fiecare autor să fi scris măcar o carte, iar fiecare carte trebuie să fie scrisă de cel puțin un autor.
- O carte din bibliotecă poate să aparțină mai multor genuri și mai multe cărți pot avea același gen. Poate exista un gen de care să nu aparțină nicio carte din bibliotecă, dar o carte trebuie să se încadreze măcar într-un gen.
- O carte poate fi publicată de o singură editură, iar la o editură pot fi publicate mai multe cărți. O carte trebuie să fie publicată de o singură editură, iar o editură trebuie să publice măcar o carte.
- O carte poate să aibă mai multe copii, dar o copie poate să aparțină unei singure cărți. O carte trebuie să aibă cel puțin o copie (cartea însăși), iar o copie trebuie să aparțină unei singure cărți.
- O copie de carte se poate găsi pe un anumit raft, iar pe un raft se pot afla mai multe copii ale unei cărți. O copie de carte trebuie să se găsească pe un singur raft, iar pe un raft trebuie să se găsească cel puțin o copie de carte.
- Un cititor poate deține unul sau mai multe abonamente în funcție de data când îl achiziționează și numărul de luni cât va fi disponibil, iar un abonament identic poate fi deținut de unul sau mai mulți cititori. Pot exista abonamente care să nu fie deținute de niciun cititor, dar un cititor trebuie să dețină cel puțin un abonament.
- Un cititor poate obține una sau mai multe penalizări dacă nu respectă regulile bibliotecii, iar aceeași penalizare poate fi primită de mai mulți cititori. Un cititor nu trebuie neapărat să primească o penalizare, dar o penalizare trebuie să fie primită de cel puțin un cititor.
- Un cititor poate împrumuta una sau mai multe copii de cărți de la mai mulți autori. Un cititor trebuie să împrumute cel puțin o copie de carte scrisă de cel puțin un autor.

## DESCRIEREA ENTITĂȚILOR

### **BIBLIOTECAR**

Aici se memorează toți bibliotecarii și datele lor, ei fiind identificați după un cheia primară id\_bibliotecar.

### **CARTE**

Aici se memorează informații despre toate cărțile din bibliotecă, ele fiind identificate după cheia primară id\_carte, copiile unei cărți regăsindu-se la același id\_carte.

### **COPIE\_CARTE**

Aici se memorează pentru fiecare carte toate copiile ei, iar ele sunt identificate după cheia primară compusă formată din id\_copie și id\_carte, deoarece pentru fiecare carte există un anumit număr de copii și pot fi identificare în mod unic doar împreună.

### **CITITOR**

Aici sunt memorate informațiile despre toți cititorii care împrumută cărți de la bibliotecă și sunt identificați după cheia primară id\_cititor.

### **AUTOR**

Aici sunt memorate informații despre toți autorii ale căror cărți se găsesc în bibliotecă și sunt identificați după cheia primară id\_autor.

### **ABONAMENT**

Aici sunt memorate informații despre toate abonamentele pe care și le poate face un client și sunt identificate după cheia primară id\_abonament.

### **PENALIZARE**

Aici sunt memorate informații despre penalizările pe care le pot primi cititorii din varii motive, cum ar fi: deteriorarea cărților, restituirea după mai mult timp decât ar fi trebuit. Ele sunt identificate după cheia primară id\_penalizare care reprezintă numărul penalizării.

## **GEN**

Aici sunt memorate informații despre toate genurile de cărți disponibile în bibliotecă, iar ele sunt identificate după cheia primară `id_gen`.

## **EDITURA**

Aici sunt memorate informații despre toate editurile de la care se găsesc cărți în bibliotecă, iar ele sunt identificate după cheia primară `id_editura`.

## **RAFT**

Aici sunt memorate informații despre rafturile din bibliotecă, iar ele sunt identificate după cheia primară `id_raft`.

# **DESCRIEREA RELAȚIILOR**

### **BIBLIOTECAR se\_ocupa\_de RAFT**

Relația are cardinalitatea maximă many-many (n:m).

Relația are cardinalitatea minimă one-one (1:1).

### **CARTE scrisa\_de AUTOR**

Relația are cardinalitatea maximă many-many (n:m).

Relația are cardinalitatea minimă one-one (1:1).

### **CARTE aparține GEN**

Relația are cardinalitatea maximă many-many (n:m).

Relația are cardinalitatea minimă one-zero (1:0).

### **CARTE publicata\_de EDITURA**

Relația are cardinalitatea maximă many-one (n:1).

Relația are cardinalitatea minimă one-one (1:1).

### **CARTE are COPIE\_CARTE**

Relația are cardinalitatea maximă many-one (n:1).

Relația are cardinalitatea minimă one-one (1:1).

**COPIE\_CARTE se\_gaseste\_pe RAFT**

Relația are cardinalitatea maximă many-one (n:1).

Relația are cardinalitatea minimă one-one (1:1).

**CITITOR detine ABONAMENT**

Relația are cardinalitatea maximă many-many (n:m).

Relația are cardinalitatea minimă one-zero (1:0).

**CITITOR primește PENALIZARE**

Relația are cardinalitatea maximă many-many (n:m).

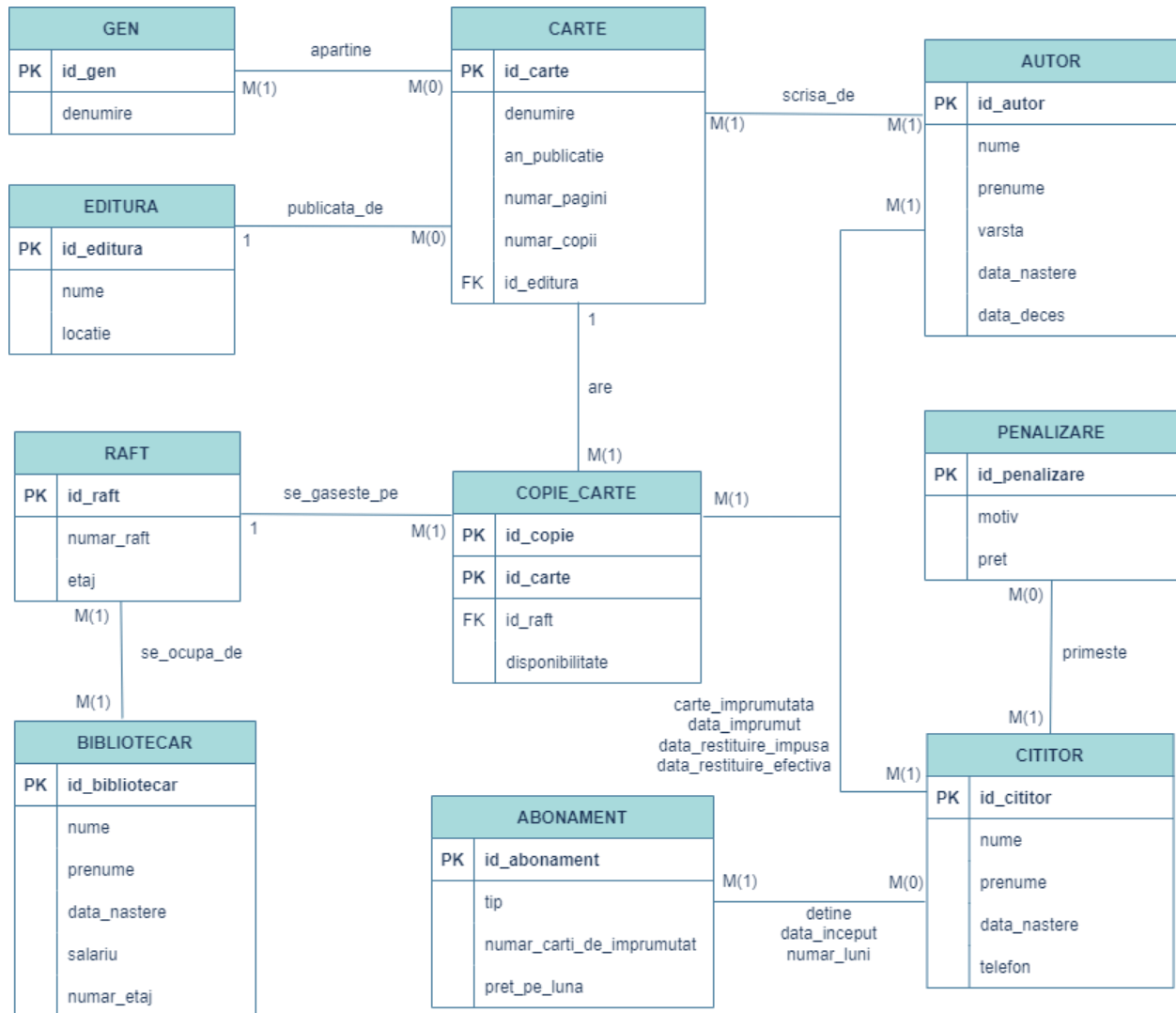
Relația are cardinalitatea minimă one-zero (1:0).

**COPIE\_CARTE****CITITOR****carte\_imprumutata****AUTOR**

Relația are cardinalitatea maximă many-many-many (n:m:n).

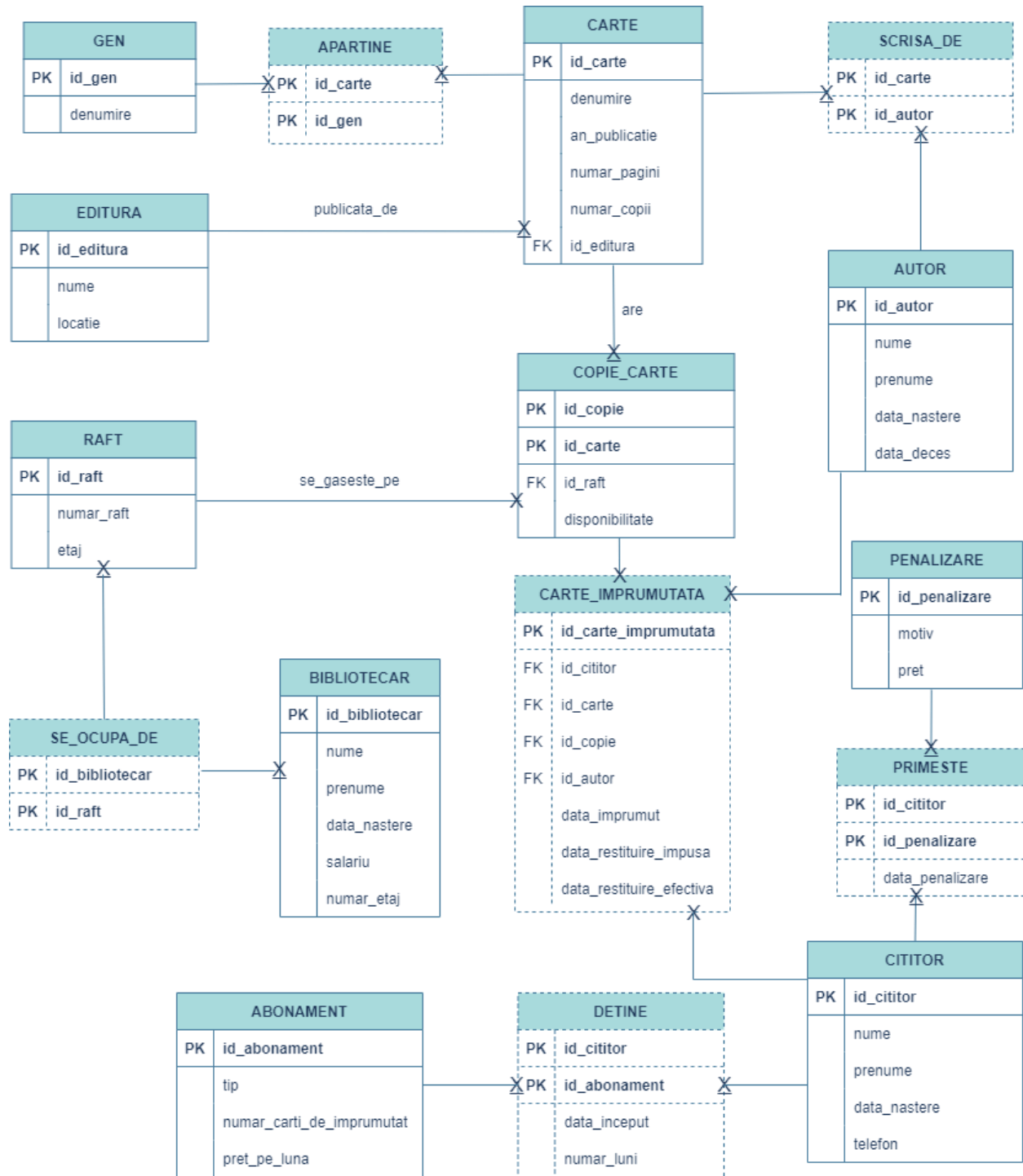
Relația are cardinalitatea minimă one-one-one (1:1:1).

## 2. DIAGRAMA ENTITATE-RELAȚIE





### 3. DIAGRAMA CONCEPTUALĂ



## 4. CREAREA TABELELOR ȘI INSERAREA DATELOR

### BIBLIOTECARI

create table bibliotecari

```
(id_bibliotecar number(3) constraint pk_bibliotecari primary key,  
  nume varchar2(20) constraint null_bibliotecar_nume not null,  
  prenume varchar2(20) constraint null_bibliotecar_prenume not null,  
  data_nastere date constraint null_bibliotecar_data_nastere not null,  
  salariu number(8,2) constraint null_bibliotecar_salariu not null,  
  numar_etaj number(2) constraint null_numar_etaj not null,  
  constraint unq_bibliotecar_nume_prenume unique(nume, prenume));
```

create sequence seq\_bibliotecari

start with 1

increment by 1

minvalue 0

maxvalue 100

nocycle;

```
insert into bibliotecari values(seq_bibliotecari.nextval, 'Teleaga', 'Mihaela', '17-MAR-1946',  
 2300, 1);
```

```
insert into bibliotecari values(seq_bibliotecari.nextval, 'Popa', 'Mariana', '28-JUN-1975', 2100,  
 2);
```

```
insert into bibliotecari values(seq_bibliotecari.nextval, 'Boghiu', 'Emilia', '21-SEP-1940', 2500, 3);
```

```
insert into bibliotecari values(seq_bibliotecari.nextval, 'Fornica', 'Gabriel', '17-NOV-1940', 2400,  
 1);
```

```
insert into bibliotecari values(seq_bibliotecari.nextval, 'Botezatu', 'Raluca', '22-APR-1982', 2100,  
 2);
```

```
insert into bibliotecari values(seq_bibliotecari.nextval, 'Tudor', 'Lavinia', '1-JAN-1971', 2300, 1);
```

```
insert into bibliotecari values(seq_bibliotecari.nextval, 'Panfil', 'Otilia', '5-MAR-1990', 2700, 3);
```

```
insert into bibliotecari values(seq_bibliotecari.nextval, 'Rafaila', 'Camelia', '9-MAY-1960', 1900,  
 4);
```

select * from bibliotecari;						
Script Output x Query Result x						
SQL   All Rows Fetched: 8 in 0.002 seconds						
	ID_BIBLIOTECAR	NUME	PRENUME	DATA_NASTERE	SALARIU	NUMAR_ETAJ
1	1	Teleaga	Mihaela	17-MAR-46	2300	1
2	2	Popa	Mariana	28-JUN-75	2100	2
3	3	Boghiu	Emilia	21-SEP-40	2500	3
4	4	Fornica	Gabriel	17-NOV-40	2400	1
5	5	Botezatu	Raluca	22-APR-82	2100	2
6	6	Tudor	Lavinia	01-JAN-71	2300	1
7	7	Panfil	Otilia	05-MAR-90	2700	3
8	8	Rafaila	Camelia	09-MAY-60	1900	4

## CARTI

create table carti

(id\_carte number(3) constraint pk\_carti primary key,  
denumire varchar2(30) constraint null\_carte\_denumire not null,  
an\_publicatie number(4) constraint null\_carte\_an\_publicatie not null,  
numar\_pagini number(4) constraint null\_carte\_numar\_pagini not null,  
numar\_copii number(2) default 1,  
id\_editura number(2) constraint fk\_carte\_editura references edituri(id\_editura));

create sequence seq\_carti

start with 1

increment by 1

minvalue 0

maxvalue 100

nocycle;

insert into carti values(seq\_carti.nextval, 'Harry Potter', 2005, 566, 4, 1);

insert into carti values(seq\_carti.nextval, 'Hotul', 2012, 600, 6, 2);

insert into carti values(seq\_carti.nextval, 'Baltagul', 1930, 123, 3, 2);

insert into carti values(seq\_carti.nextval, 'Ion', 1920, 289, 5, 3);

insert into carti values(seq\_carti.nextval, 'Enigma Otiliei', 1938, 400, 1, 4);

select * from carti;						
Script Output x Query Result x						
SQL   All Rows Fetched: 5 in 0.001 seconds						
ID_CARTE	DENUMIRE	AN_PUBLICATIE	NUMAR_PAGINI	NUMAR_COPII	ID_EDITURA	
1	1 Harry Potter	2005	566	4	1	
2	2 Hotul	2012	600	6	2	
3	3 Baltagul	1930	123	3	2	
4	4 Ion	1920	289	5	3	
5	5 Enigma Otiliei	1938	400	1	4	

## CITITORI

create table cititori

(id\_cititor number(3) constraint pk\_cititor primary key,  
 nume varchar2(20) constraint null\_cititor\_nume not null,  
 prenume varchar2(20) constraint null\_cititor\_prenume not null,  
 data\_nastere date constraint null\_cititor\_data\_nastere not null,  
 telefon varchar2(12) constraint null\_cititor\_telefon not null,  
 constraint unq\_cititor\_nume\_prenume unique(nume, prenume));

create sequence seq\_cititori

start with 1

increment by 1

minvalue 0

maxvalue 100

nocycle;

insert into cititori values(seq\_cititori.nextval, 'Rogoza', 'Raluca', '8-JAN-2002', '0741837264');

insert into cititori values(seq\_cititori.nextval, 'Stoica', 'Elena', '7-NOV-2008', '0778347287');

insert into cititori values(seq\_cititori.nextval, 'Cojoc', 'Georgiana', '17-JAN-2000', '0756123656');

insert into cititori values(seq\_cititori.nextval, 'Obreja', 'Carina', '5-DEC-1980', '0723457846');

insert into cititori values(seq\_cititori.nextval, 'Talpalariu', 'Iulia', '3-JUL-1965', '0789648247');

insert into cititori values(seq\_cititori.nextval, 'Bejan', 'Mara', '25-AUG-2000', '0746590478');

select * from cititori;					
Script Output x Query Result x					
SQL   All Rows Fetched: 6 in 0.006 seconds					
ID_CITITOR	NUME	PRENUME	DATA_NASTERE	TELEFON	
1	1 Rogoza	Raluca	08-JAN-02	0741837264	
2	2 Stoica	Elena	07-NOV-08	0778347287	
3	3 Cojoc	Georgiana	17-JAN-00	0756123656	
4	4 Obreja	Carina	05-DEC-80	0723457846	
5	5 Talpalariu	Iulia	03-JUL-65	0789648247	
6	6 Bejan	Mara	25-AUG-00	0746590478	

## RAFTURI

create table rafturi

(id\_raft number(3) constraint pk\_raft primary key,  
 numar\_raft number(2) constraint null\_raft\_numarraft not null,  
 etaj number(1) constraint null\_raft\_etaj not null);

create sequence seq\_rafturi

start with 1

increment by 1

minvalue 0

maxvalue 100

nocycle;

insert into rafturi values(seq\_rafturi.nextval, 1, 1);

insert into rafturi values(seq\_rafturi.nextval, 2, 1);

insert into rafturi values(seq\_rafturi.nextval, 3, 1);

insert into rafturi values(seq\_rafturi.nextval, 4, 1);

insert into rafturi values(seq\_rafturi.nextval, 1, 2);

insert into rafturi values(seq\_rafturi.nextval, 2, 2);

insert into rafturi values(seq\_rafturi.nextval, 3, 2);

insert into rafturi values(seq\_rafturi.nextval, 1, 3);

insert into rafturi values(seq\_rafturi.nextval, 2, 3);

insert into rafturi values(seq\_rafturi.nextval, 3, 3);

insert into rafturi values(seq\_rafturi.nextval, 1, 4);

insert into rafturi values(seq\_rafturi.nextval, 2, 4);

select * from rafturi;			
Script Output x Query Result x			
SQL   All Rows Fetched: 12 in			
ID_RAFT	NUMAR_RAFT	ETAJ	
1	1	1	
2	2	1	
3	3	1	
4	4	1	
5	5	1	2
6	6	2	2
7	7	3	2
8	8	1	3
9	9	2	3
10	10	3	3
11	11	1	4
12	12	2	4

## AUTORI

create table autori

(id\_autor number(3) constraint pk\_autor primary key,  
 nume varchar2(20) constraint null\_autor\_nume not null,  
 prenume varchar2(20) constraint null\_autor\_prenume not null,  
 data\_nastere date constraint null\_data\_nastere not null,  
 data\_deces date,  
 constraint unq\_autor\_nume\_prenume unique(nume, prenume));

create sequence seq\_autori

start with 1

increment by 1

minvalue 0

maxvalue 100

nocycle;

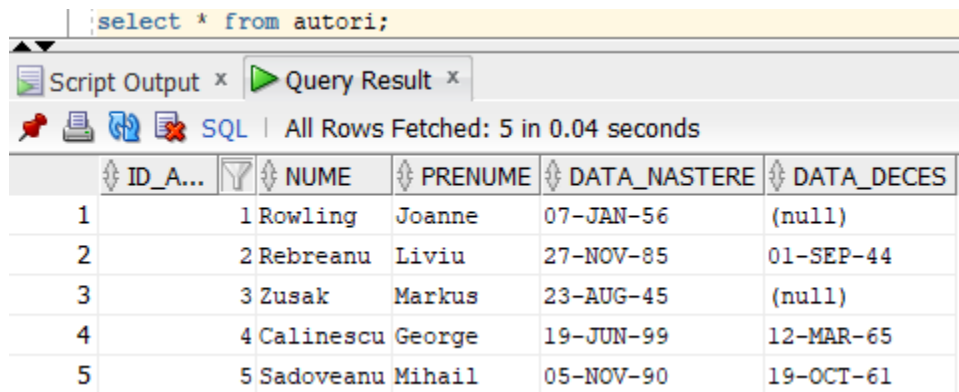
insert into autori(id\_autor, nume, prenume, data\_nastere) values(seq\_autori.nextval, 'Rowling', 'Joanne', '7-JAN-1956');

insert into autori values(seq\_autori.nextval, 'Rebreanu', 'Liviu', '27-NOV-1885', '1-SEP-1944');

insert into autori(id\_autor, nume, prenume, data\_nastere) values(seq\_autori.nextval, 'Zusak', 'Markus', '23-AUG-1945');

insert into autori values(seq\_autori.nextval, 'Calinescu', 'George', '19-JUN-1899', '12-MAR-1965');

```
insert into autori values(seq_autori.nextval, 'Sadoveanu', 'Mihail', '5-NOV-1890', '19-OCT-1961');
```



ID_A...	NUME	PRENUME	DATA_NASTERE	DATA_DECES
1	Rowling	Joanne	07-JAN-56	(null)
2	Rebreanu	Liviu	27-NOV-85	01-SEP-44
3	Zusak	Markus	23-AUG-45	(null)
4	Calinescu	George	19-JUN-99	12-MAR-65
5	Sadoveanu	Mihail	05-NOV-90	19-OCT-61

## GENURI

```
create table genuri
```

```
(id_gen number(3) constraint pk_gen primary key,  
denumire varchar(20) constraint null_gen_denumire not null);
```

```
create sequence seq_genuri
```

```
start with 1
```

```
increment by 1
```

```
minvalue 0
```

```
maxvalue 100
```

```
nocycle;
```

```
insert into genuri values(seq_genuri.nextval, 'Aventura');
```

```
insert into genuri values(seq_genuri.nextval, 'Mister');
```

```
insert into genuri values(seq_genuri.nextval, 'Fictiune');
```

```
insert into genuri values(seq_genuri.nextval, 'Psihologie');
```

```
insert into genuri values(seq_genuri.nextval, 'Dragoste');
```

select \* from genuri;

ID_GEN	DENUMIRE
1	1 Aventura
2	2 Mister
3	3 Fictiune
4	4 Psihologie
5	5 Dragoste

## EDITURI

create table edituri

(id\_editura number(3) constraint pk\_editura primary key,  
 nume varchar(20) constraint null\_editura\_nume not null,  
 locatie varchar(40) constraint null\_editura\_locatie not null);

create sequence seq\_edituri

start with 1

increment by 1

minvalue 0

maxvalue 100

nocycle;

insert into edituri values(seq\_edituri.nextval, 'Humanitas', 'Sector 1 Bucuresti');

insert into edituri values(seq\_edituri.nextval, 'Trei', 'Sector 3 Bucuresti');

insert into edituri values(seq\_edituri.nextval, 'Polirom', 'Sector 1 Bucuresti');

insert into edituri values(seq\_edituri.nextval, 'Arthur', 'Sector 4 Bucuresti');

insert into edituri values(seq\_edituri.nextval, 'YoungArt', 'Sector 1 Bucuresti');

select \* from edituri;

ID_EDITURA	NUME	LOCATIE
1	1 Humanitas	Sector 1 Bucuresti
2	2 Trei	Sector 3 Bucuresti
3	3 Polirom	Sector 1 Bucuresti
4	4 Arthur	Sector 4 Bucuresti
5	5 YoungArt	Sector 1 Bucuresti



## PENALIZARI

create table penalizari

```
(id_penalizare number(3) constraint pk_penalizare primary key,  
motiv varchar2(40),  
pret number(4, 2) default 0);
```

create sequence seq\_penalizari

start with 1

increment by 1

minvalue 0

maxvalue 100

nocycle;

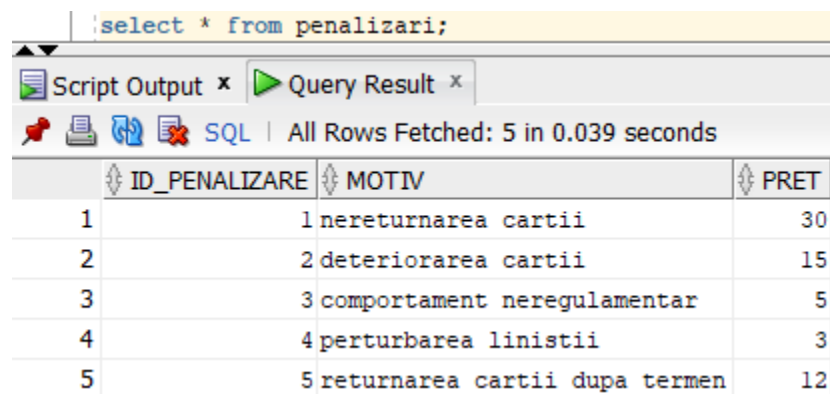
insert into penalizari values(seq\_penalizari.nextval, 'nereturnarea cartii', 30);

insert into penalizari values(seq\_penalizari.nextval, 'deteriorarea cartii', 15);

insert into penalizari values(seq\_penalizari.nextval, 'comportament neregulamentar', 5);

insert into penalizari values(seq\_penalizari.nextval, 'perturbarea linistii', 3);

insert into penalizari values(seq\_penalizari.nextval, 'returnarea cartii dupa termen', 12);



The screenshot shows a SQL query result in a database client. The query is 'select \* from penalizari;'. The result is displayed in a table with 5 rows and 3 columns: ID\_PENALIZARE, MOTIV, and PRET. The data is as follows:

ID_PENALIZARE	MOTIV	PRET
1	1 nereturnarea cartii	30
2	2 deteriorarea cartii	15
3	3 comportament neregulamentar	5
4	4 perturbarea linistii	3
5	5 returnarea cartii dupa termen	12

## ABONAMENTE

create table abonamente

```
(id_abonament number(3) constraint pk_abonament primary key,  
tip varchar2(10) constraint null_abonament_tip not null,  
numar_carti_de_imprumutat number(2) constraint null_abonament_numar_carti not null,  
pret_pe_luna number(5, 2) constraint null_abonament_pret not null);
```

```

create sequence seq_abonamente
start with 1
increment by 1
minvalue 0
maxvalue 100
nocycle;

```

```

insert into abonamente values(seq_abonamente.nextval, 'student', 15, 50);
insert into abonamente values(seq_abonamente.nextval, 'elev', 12, 40);
insert into abonamente values(seq_abonamente.nextval, 'adult', 18, 80);
insert into abonamente values(seq_abonamente.nextval, 'pensionar', 16, 50);
insert into abonamente values(seq_abonamente.nextval, 'student', 8, 25);
insert into abonamente values(seq_abonamente.nextval, 'premium', 15, 45);

```

`select * from abonamente;`

Script Output x Query Result x

SQL | All Rows Fetched: 6 in 0.036 seconds

	ID_ABONAMENT	TIP	NUMAR_CARTI_DE_IMPRUMUTAT	PRET_PE_LUNA
1	1	student	15	50
2	2	elev	12	40
3	3	adult	18	80
4	4	pensionar	16	50
5	5	student	8	25
6	6	premium	15	45

## SE\_OCUPA\_DE

```

create table se_ocupa_de
(id_bibliotecar number(3),
id_raft number(3),
constraint pk_bibliotecar_carte primary key(id_bibliotecar, id_raft),
constraint fk_bibliotecar_carte_1 foreign key (id_bibliotecar) references
bibliotecari(id_bibliotecar),
constraint fk_bibliotecar_carte_2 foreign key (id_raft) references rafturi(id_raft));

```

```

insert into se_ocupa_de values(1, 1);
insert into se_ocupa_de values(2, 2);
insert into se_ocupa_de values(3, 1);
insert into se_ocupa_de values(4, 2);

```

```

insert into se_ocupa_de values(3, 2);
insert into se_ocupa_de values(7, 3);
insert into se_ocupa_de values(3, 3);
insert into se_ocupa_de values(1, 2);
insert into se_ocupa_de values(1, 4);
insert into se_ocupa_de values(5, 1);
insert into se_ocupa_de values(4, 3);
insert into se_ocupa_de values(6, 1);
insert into se_ocupa_de values(8, 2);
insert into se_ocupa_de values(8, 1);

```

SQL | All Rows Fetched:

	ID_BIBLIOTECAR	ID_RAFT
1	1	1
2	1	2
3	1	4
4	2	2
5	3	1
6	3	2
7	3	3
8	4	2
9	4	3
10	5	1
11	6	1
12	7	3
13	8	1
14	8	2

## APARTINE

```

create table apartine
(id_carte number(3),
 id_gen number(3),
constraint pk_gen_carte primary key(id_carte, id_gen),
constraint fk_gen_carte_1 foreign key (id_carte) references carti(id_carte),
constraint fk_gen_carte_2 foreign key (id_gen) references genuri(id_gen));

```

```

insert into apartine values(4, 1);
insert into apartine values(1, 3);
insert into apartine values(2, 3);
insert into apartine values(3, 4);
insert into apartine values(1, 1);
insert into apartine values(1, 5);
insert into apartine values(3, 2);
insert into apartine values(4, 3);
insert into apartine values(5, 2);
insert into apartine values(3, 5);

```

select \* from apartine;

Script Output x Query Result x

SQL | All Rows Fetched:

	ID_CARTE	ID_GEN
1	4	1
2	1	3
3	2	3
4	3	4
5	1	1
6	1	5
7	3	2
8	4	3
9	5	2
10	3	5

## COPII\_CARTI

```

create table copii_carti
(id_copie number(3),
 id_carte number(3),
 id_raft number(3) constraint fk_copie_carte_raft references rafturi(id_raft),
 disponibilitate varchar2(4) default 'da',
 constraint pk_copie_carte primary key(id_copie, id_carte),
 constraint fk_copie_carte_carte foreign key (id_carte) references carti(id_carte));

alter table copii_carti drop constraint fk_copie_carte_carte;
alter table copii_carti add constraint fk_copie_carte_carte foreign key (id_carte) references
carti(id_carte) on delete cascade;

```

```

insert into copii_carti values(3, 1, 3, 'da');
insert into copii_carti values(1, 5, 2, 'da');
insert into copii_carti values(2, 1, 6, 'da');
insert into copii_carti values(5, 3, 4, 'da');
insert into copii_carti values(4, 2, 5, 'da');
insert into copii_carti values(2, 4, 7, 'da');
insert into copii_carti values(4, 1, 7, 'da');
insert into copii_carti values(1, 2, 5, 'da');
insert into copii_carti values(1, 3, 1, 'da');
insert into copii_carti values(5, 1, 2, 'da');
insert into copii_carti values(1, 1, 2, 'da');
insert into copii_carti values(3, 2, 4, 'da');
insert into copii_carti values(2, 5, 8, 'da');

```

select \* from copii\_carti;

Script Output x Query Result x

SQL | All Rows Fetched: 13 in 0.002 seconds

	ID_COPIE	ID_CARTE	ID_RAFT	DISPONIBILITATE
1	3	1	3	da
2	1	5	2	da
3	2	1	6	da
4	5	3	4	da
5	4	2	5	da
6	2	4	7	da
7	4	1	7	da
8	1	2	5	da
9	1	3	1	da
10	5	1	2	da
11	1	1	2	da
12	3	2	4	da
13	2	5	8	da

## SCRISA\_DE

```

create table scrisa_de
(id_carte number(3),
id_autor number(3),
constraint pk_carte_autor primary key(id_carte, id_autor),
constraint fk_carte_autor_1 foreign key (id_carte) references carti(id_carte),
constraint fk_carte_autor_2 foreign key (id_autor) references autori(id_autor));

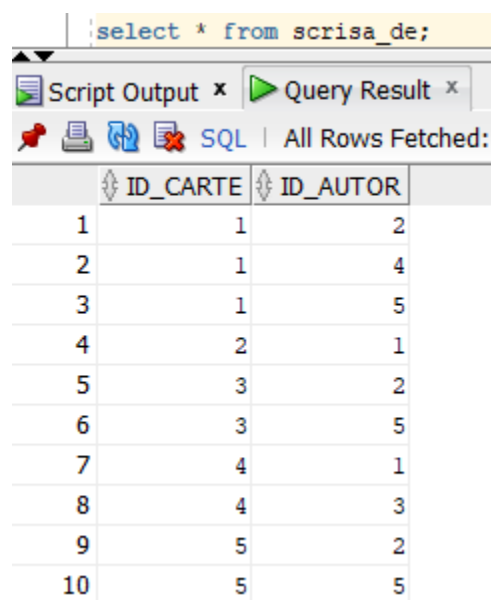
```

```

insert into scrisa_de values(3, 2);
insert into scrisa_de values(1, 2);
insert into scrisa_de values(5, 2);
insert into scrisa_de values(1, 4);
insert into scrisa_de values(4, 3);
insert into scrisa_de values(3, 5);
insert into scrisa_de values(1, 5);
insert into scrisa_de values(4, 1);
insert into scrisa_de values(5, 5);
insert into scrisa_de values(2, 1);

```

select \* from scrisa\_de;



	ID_CARTE	ID_AUTOR
1	1	2
2	1	4
3	1	5
4	2	1
5	3	2
6	3	5
7	4	1
8	4	3
9	5	2
10	5	5

## DETINE

create table detine

```

(id_cititor number(3),
 id_abonament number(3),
 data_inceput date constraint null_data_inceput not null,
 numar_luni number(1) constraint null_numar_luni not null,
 constraint pk_cititor_abonament primary key(id_cititor, id_abonament, data_inceput),
 constraint fk_cititor_abonament_1 foreign key (id_cititor) references cititori(id_cititor),
 constraint fk_cititor_abonament_2 foreign key (id_abonament) references
 abonamente(id_abonament));

```

```

insert into detine values(2, 2, '02-MAY-22', 2);

```

```

insert into detine values(1, 1, '08-JAN-22', 1);
insert into detine values(3, 6, '12-MAR-22', 1);
insert into detine values(5, 4, '17-FEB-20', 4);
insert into detine values(2, 6, '02-JAN-21', 2);
insert into detine values(1, 5, '20-MAR-21', 1);
insert into detine values(4, 6, '22-SEP-21', 3);
insert into detine values(5, 6, '01-MAY-21', 2);
insert into detine values(3, 1, '15-MAY-22', 2);
insert into detine values(4, 3, '02-APR-22', 3);
insert into detine values(1, 1, '12-DEC-22', 1);
insert into detine values(5, 4, '06-NOV-22', 2);
insert into detine values(6, 1, '08-JAN-23', 2);

```

select * from detine;				
Script Output x Query Result x				
SQL   All Rows Fetched: 13 in 0.002 seconds				
	ID_CITITOR	ID_ABONAMENT	DATA_INCEPUT	NUMAR_LUNI
1	2	2	02-MAY-22	2
2	1	1	08-JAN-22	1
3	3	6	12-MAR-22	1
4	5	4	17-FEB-20	4
5	2	6	02-JAN-21	2
6	1	5	20-MAR-21	1
7	4	6	22-SEP-21	3
8	5	6	01-MAY-21	2
9	3	1	15-MAY-22	2
10	4	3	02-APR-22	3
11	1	1	12-DEC-22	1
12	5	4	06-NOV-22	2
13	6	1	08-JAN-23	2

## PRIMESTE

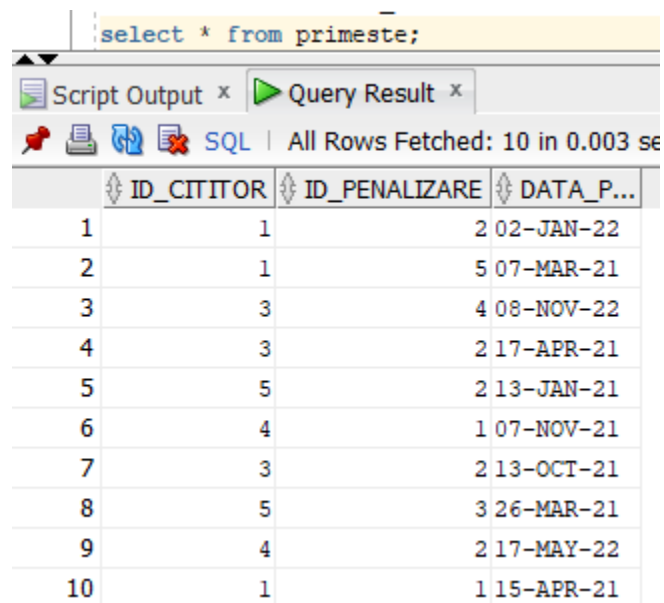
```

create table primeste
(id_cititor number(3),
 id_penalizare number(3),
 data_penalizare date constraint null_data_penalizare not null,
 constraint pk_cititor_penalizare primary key(id_cititor, id_penalizare, data_penalizare),
 constraint fk_cititor_penalizare_1 foreign key (id_cititor) references cititori(id_cititor),

```

```
constraint fk_cititor_penalizare_2 foreign key (id_penalizare) references  
penalizari(id_penalizare));
```

```
insert into primeste values(1, 2, '2-JAN-2022');  
insert into primeste values(1, 5, '7-MAR-2021');  
insert into primeste values(3, 4, '8-NOV-2022');  
insert into primeste values(3, 2, '17-APR-2021');  
insert into primeste values(5, 2, '13-JAN-2021');  
insert into primeste values(4, 1, '7-NOV-2021');  
insert into primeste values(3, 2, '13-OCT-2021');  
insert into primeste values(5, 3, '26-MAR-2021');  
insert into primeste values(4, 2, '17-MAY-2022');  
insert into primeste values(1, 1, '15-APR-2021');
```



	ID_CITITOR	ID_PENALIZARE	DATA_P...
1	1	2	02-JAN-22
2	1	5	07-MAR-21
3	3	4	08-NOV-22
4	3	2	17-APR-21
5	5	2	13-JAN-21
6	4	1	07-NOV-21
7	3	2	13-OCT-21
8	5	3	26-MAR-21
9	4	2	17-MAY-22
10	1	1	15-APR-21

## CARTI\_IMPRUMUTATE

```
create table carti_imprumutate  
(id_carte_imprumutata number(3) constraint pk_carte_imprumutata primary key,  
id_cititor number(3),  
id_carte number(3),  
id_copie number(3),  
id_autor number(3),  
data_imprumut date constraint null_data_imprumut not null,  
data_restituire_impusa date constraint null_data_restituire_impusa not null,
```



```
data_restituire_efectiva date,  
constraint unq_carte_imprumutata unique(id_cititor, id_carte, id_copie, id_autor),  
constraint fk_carte_imprumutata_1 foreign key (id_cititor) references cititori(id_cititor),  
constraint fk_carte_imprumutata_2 foreign key (id_copie, id_carte) references  
copii_carti(id_copie, id_carte),  
constraint fk_carte_imprumutata_3 foreign key (id_autor) references autori(id_autor));
```

```
create sequence seq_carte_imprumutata  
start with 1  
increment by 1  
minvalue 0  
maxvalue 100  
nocycle;
```

```
insert into carti_imprumutate values(seq_carte_imprumutata.nextval, 1, 1, 3, 1, '08-JAN-22',  
'20-JAN-22', '30-JAN-22');  
insert into carti_imprumutate values(seq_carte_imprumutata.nextval, 1, 5, 1, 2, '21-MAR-22',  
'30-MAR-22', '5-APR-22');  
insert into carti_imprumutate values(seq_carte_imprumutata.nextval, 2, 1, 2, 2, '03-JAN-22',  
'13-JAN-22', '10-JAN-22');  
insert into carti_imprumutate values(seq_carte_imprumutata.nextval, 2, 3, 5, 5, '05-MAY-22',  
'11-MAY-22', '10-MAY-22');  
insert into carti_imprumutate values(seq_carte_imprumutata.nextval, 3, 2, 4, 3, '14-MAR-22',  
'25-MAR-22', '22-MAR-22');  
insert into carti_imprumutate(id_carte_imprumutata, id_cititor, id_carte, id_copie, id_autor,  
data_imprumut, data_restituire_impusa)  
values(seq_carte_imprumutata.nextval, 3, 4, 2, 1, '20-MAY-22', '30-MAY-22');  
insert into carti_imprumutate values(seq_carte_imprumutata.nextval, 4, 2, 1, 2, '24-SEP-21',  
'01-OCT-21', '06-OCT-22');  
insert into carti_imprumutate values(seq_carte_imprumutata.nextval, 4, 3, 1, 4, '01-NOV-22',  
'14-NOV-22', '05-NOV-22');  
insert into carti_imprumutate values(seq_carte_imprumutata.nextval, 5, 1, 5, 2, '05-MAY-21',  
'10-MAY-21', '07-MAY-22');  
insert into carti_imprumutate(id_carte_imprumutata, id_cititor, id_carte, id_copie, id_autor,  
data_imprumut, data_restituire_impusa)  
values(seq_carte_imprumutata.nextval, 5, 1, 4, 2, '19-FEB-20', '27-FEB-20');  
insert into carti_imprumutate values(seq_carte_imprumutata.nextval, 1, 2, 1, 1, '23-DEC-22',  
'31-DEC-22', '01-JAN-23');  
insert into carti_imprumutate(id_carte_imprumutata, id_cititor, id_carte, id_copie, id_autor,  
data_imprumut, data_restituire_impusa)  
values(seq_carte_imprumutata.nextval, 5, 2, 3, 1, '26-DEC-22', '03-JAN-23');
```

select * from carti_imprumutate;								
Script Output x Query Result x								
SQL   All Rows Fetched: 12 in 0.009 seconds								
	ID_CARTE_IMPRUMUTATA	ID_CITITOR	ID_CARTE	ID_COPIE	ID_AUTOR	DATA_IMPRUMUT	DATA_RESTITUIRE_IMPUSA	DATA_RESTITUIRE_EFECTIVA
1	1	1	1	3	1	08-JAN-22	20-JAN-22	30-JAN-22
2	2	1	5	1	2	21-MAR-22	30-MAR-22	05-APR-22
3	3	2	1	2	2	03-JAN-22	13-JAN-22	10-JAN-22
4	4	2	3	5	5	05-MAY-22	11-MAY-22	10-MAY-22
5	5	3	2	4	3	14-MAR-22	25-MAR-22	22-MAR-22
6	6	3	4	2	1	20-MAY-22	30-MAY-22	(null)
7	7	4	2	1	2	24-SEP-21	01-OCT-21	06-OCT-22
8	8	4	3	1	4	01-NOV-22	14-NOV-22	05-NOV-22
9	9	5	1	5	2	05-MAY-21	10-MAY-21	07-MAY-22
10	10	5	1	4	2	19-FEB-20	27-FEB-20	(null)
11	11	1	2	1	1	23-DEC-22	31-DEC-22	01-JAN-23
12	12	5	2	3	1	26-DEC-22	03-JAN-23	(null)

## 5. EXERCITIUL 6

Cerință: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un **subprogram stocat independent** care să utilizeze două tipuri diferite de colecții studiate. Apelați subprogramul.

Să se afișeze cărțile care nu au fost returnate(denumirea cărții) și autorii care au scris-o. Să se afișeze persoanele care nu au returnat cărțile respective și să li se aplice penalizarea corespunzătoare. Dacă au mai multe cărți nereturnate, li se aplică o singură dată pedeapsa.

create or replace procedure exercitiul\_6

is

```
type tablou_indexat is table of carti_imprumutate%rowtype index by pls_integer;
```

```
tablou tablou_indexat;
```

```
cod_carte carti_imprumutate.id_carte%type;
```

```
type vector is varray(20) of number;
```

```
coduri_autori vector := vector();
```

```
cititori_penalizati vector := vector();
```

```
v_nume_autor autori.nume%type;
```

```
v_prenume_autor autori.prenume%type;
```

```
cod_cititor cititori.id_cititor%type;
```

```
numar_cititor_penalizat number := 0;
```

```
nume_carte carti.denumire%type;
```

```
nume_cititor cititori.nume%type;
```

```
prenume_cititor cititori.prenume%type;
```

```
gasit number := 0;
```

begin

```
select *
```

```
bulk collect into tablou
```

```
from carti_imprumutate
```

```
where data_restituire_efectiva is null;
```

```
for i in tablou.first..tablou.last loop
```

```
cod_carte := tablou(i).id_carte;
```

```
cod_cititor := tablou(i).id_cititor;
```

```
gasit := 0;
```

```
if numar_cititor_penalizat != 0 then
```

```
for j in cititori_penalizati.first..cititori_penalizati.last loop
```

```
if cititori_penalizati(j) = cod_cititor then
```

```

        gasit := 1;
    end if;
end loop;
end if;
if gasit = 0 then
    numar_cititor_penalizat := numar_cititor_penalizat + 1;
    cititori_penalizati.extend;
    cititori_penalizati(numar_cititor_penalizat):= cod_cititor;
end if;
select id_autor
bulk collect into coduri_autori
from scrisa_de
where id_carte = cod_carte;
select denumire
into nume_carte
from carti
where id_carte = cod_carte;
dbms_output.put_line('Cartea: ' || nume_carte);
dbms_output.put_line('Autorii cartii: ');
for j in coduri_autori.first..coduri_autori.last loop
    select nume, prenume
    into v_nume_autor, v_prenume_autor
    from autori
    where id_autor = coduri_autori(j);
    dbms_output.put_line( v_nume_autor || ' ' || v_prenume_autor);
end loop;
dbms_output.new_line();
end loop;
dbms_output.put_line('Au fost penalizati ' || numar_cititor_penalizat || ' cititori.');
```

```

for i in cititori_penalizati.first..cititori_penalizati.last loop
    insert into primeste values(cititori_penalizati(i), 1, sysdate);
    select prenume, nume
    into nume_cititor, prenume_cititor
    from cititori
    where id_cititor = cititori_penalizati(i);
    dbms_output.put_line(nume_cititor || ' ' || prenume_cititor);
end loop;
end exercitiul_6;
/

execute exercitiul_6;
```

```

    dbms_output.new_line();
end loop;
dbms_output.put_line('Au fost penalizati ' || numar_cititor_penalizat || ' cititori.');
```

```

for i in cititori_penalizati.first..cititori_penalizati.last loop
    insert into primeste values(cititori_penalizati(i), 1, sysdate);
    select prenume, nume
    into nume_cititor, prenume_cititor
    from cititori
    where id_cititor = cititori_penalizati(i);
    dbms_output.put_line(nume_cititor || ' ' || prenume_cititor);
end loop;
end exercitiul_6;
/
execute exercitiul_6;
```

Script Output x Query Result x

Task completed in 0.045 seconds

Procedure EXERCITIUL\_6 compiled

Cartea: Ion  
 Autorii cartii:  
 Rowling Joanne  
 Zusak Markus

Cartea: Harry Potter  
 Autorii cartii:  
 Rebreanu Liviu  
 Calinescu George  
 Sadoveanu Mihail

Cartea: Notul  
 Autorii cartii:  
 Rowling Joanne

Au fost penalizati 2 cititori.  
 Georgiana Cojoc  
 Iulia Talpalaru

PL/SQL procedure successfully completed.

Tabelul PRIMESTE înainte de insert:

```
select * from primeste;
```

	ID_CITITOR	ID_PENALIZARE	DATA_P...
1	1	2	02-JAN-22
2	1	5	07-MAR-21
3	3	4	08-NOV-22
4	3	2	17-APR-21
5	5	2	13-JAN-21
6	4	1	07-NOV-21
7	3	2	13-OCT-21
8	5	3	26-MAR-21
9	4	2	17-MAY-22
10	1	1	15-APR-21

Tabelul PRIMESTE după insert:

```

for i in cititori_penalizati.first..cititori_penalizati.last loop
insert into primeste values(cititori_penalizati(i), 1, sysdate);
select prenume, nume
into nume_cititor, prenume_cititor
from cititori
where id_cititor = cititori_penalizati(i);
dbms_output.put_line(nume_cititor || ' ' || prenume_cititor);
end loop;
end exercitiul_6;
/

execute exercitiul_6;

select * from primeste;

```

Script Output x Query Result x

SQL All Rows Fetched: 12 in 0.001 seconds

ID_CITITOR	ID_PENALIZARE	DATA_P...
1	1	2-02-JAN-22
2	1	5-07-MAR-21
3	3	4-09-NOV-22
4	3	2-17-APR-21
5	5	2-13-JAN-21
6	4	1-07-NOV-21
7	3	2-13-OCT-21
8	5	3-26-MAR-21
9	4	2-17-MAY-22
10	1	1-15-APR-21
11	3	1-13-JAN-23
12	5	1-13-JAN-23

## 6. EXERCITIUL 7

Cerință: 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. Apelați subprogramul.

Pentru fiecare etaj din bibliotecă afișați bibliotecarii și pentru fiecare etaj să se menționeze în dreptul lor cine are salariul maxim, cine are salariul minim și cine lucrează singur la acel etaj.

create or replace procedure exercitiul\_7

is

cursor c\_etaje is

select distinct(etaj)

from rafturi

group by etaj

order by 1;

cursor colegi\_palier (v\_numar\_etaj bibliotecari.numar\_etaj%type) is

select id\_bibliotecar, nume, prenume, salariu

from bibliotecari

where numar\_etaj = v\_numar\_etaj

order by 2, 3;

b\_numar\_etaj bibliotecari.numar\_etaj%type;

b\_numar\_angajati\_etaj number;

type b\_record is record (id bibliotecari.id\_bibliotecar%type,

```

        nume bibliotecari.nume%type,
        prenume bibliotecari.prenume%type,
        salariu bibliotecari.salariu%type);
record_bibliotecari b_record;
salariu_minim_etaj bibliotecari.salariu%type;
salariu_maxim_etaj bibliotecari.salariu%type;
begin
    open c_etaje;
    loop
        fetch c_etaje into b_numar_etaj;
        exit when c_etaje%notfound;

        --calculez max si minim pe etaj
        select max(salariu), min(salariu), count(id_bibliotecar)
        into salariu_maxim_etaj, salariu_minim_etaj, b_numar_angajati_etaj
        from bibliotecari
        where numar_etaj = b_numar_etaj;

        dbms_output.put_line('Etajul ' || b_numar_etaj);
        dbms_output.new_line;
        dbms_output.put_line('Colegi: ');

        open colegi_palier(b_numar_etaj);

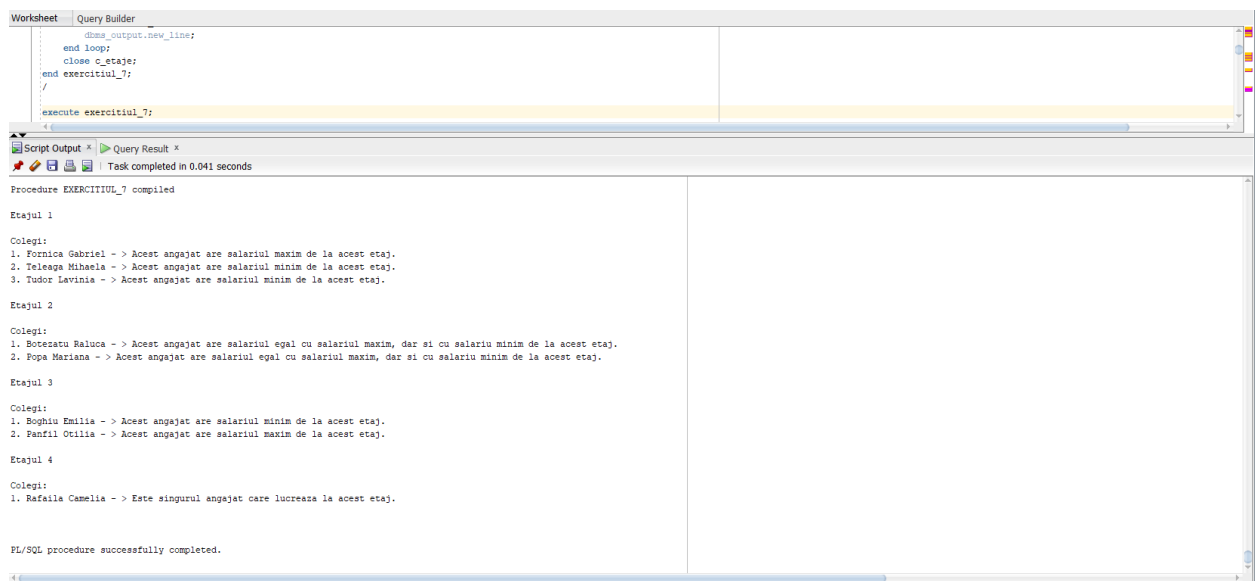
        loop
            fetch colegi_palier into record_bibliotecari;
            exit when colegi_palier%notfound;
            dbms_output.put(colegi_palier%rowcount || '. ' || record_bibliotecari.nume || ' ' ||
record_bibliotecari.prenume);
            if b_numar_angajati_etaj = 1 then
                dbms_output.put_line(' - > Este singurul angajat care lucreaza la acest etaj.');
```

```

        if colegi_palier%rowcount = 0 then
            dbms_output.put_line('La acest etaj nu lucreaza nimeni momentan.');
```

```
        end if;
        close colegi_palier;
        dbms_output.new_line;
    end loop;
    close c_etaje;
end exercitiul_7;
/
```

```
execute exercitiul_7;
```



## 7. EXERCITIUL 8

Cerință: 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. Definiți minim 2 excepții. Apelați subprogramul astfel încât să evidențiați **toate** cazurile tratate.

Pentru un cititor cu prenumele dat să se afișeze numele autorilor celei mai recente cărți împrumutate de către acesta și să se calculeze media autorilor contemporani.



[illegible]

```

                                where upper(cititori.prenume) =
upper(prenume_cititor));
--caut autorii contemporani
dbms_output.put_line('Autorii celei mai recente carti imprumutate de ' || prenume_cititor
|| ': ');
for i in autori_carte.first..autori_carte.last loop
    --dbms_output.put_line(autori_carte(i));
    select data_deces, nume, prenume
    into deces, nume_autor, prenume_autor
    from autori
    where id_autor = autori_carte(i);
    dbms_output.put('Nume si prenume: ' || nume_autor || ' ' || prenume_autor);
    if deces is null then
        select extract(year from sysdate) - extract(year from data_nastere)
        into varsta
        from autori
        where id_autor = autori_carte(i);
        dbms_output.put_line(' Varsta: ' || varsta);
        numar_autori_contemporani := numar_autori_contemporani + 1;
        varsta_totala_autori := varsta_totala_autori + varsta;
        autori_contemporani.extend();
        autori_contemporani(numar_autori_contemporani) := autori_carte(i);
    else
        dbms_output.put_line(' Deces ' || deces);
    end if;

end loop;
dbms_output.new_line;

if numar_autori_contemporani = 0 then
    raise exceptie_autori;
else
    medie_varste := varsta_totala_autori/numar_autori_contemporani;
end if;

return ('Media varstelor autorilor contemporani ai cartii: ' || to_char(medie_varste));
exception
when no_data_found then
    return('Nu exista niciun cititor cu prenumele ' || prenume_cititor || '.');
when too_many_rows then
    return('Exista mai multi cititori cu prenumele ' || prenume_cititor || '.');

```

```

when exceptie_autori then
    return('Toti autorii cartii au murit.');
```

```

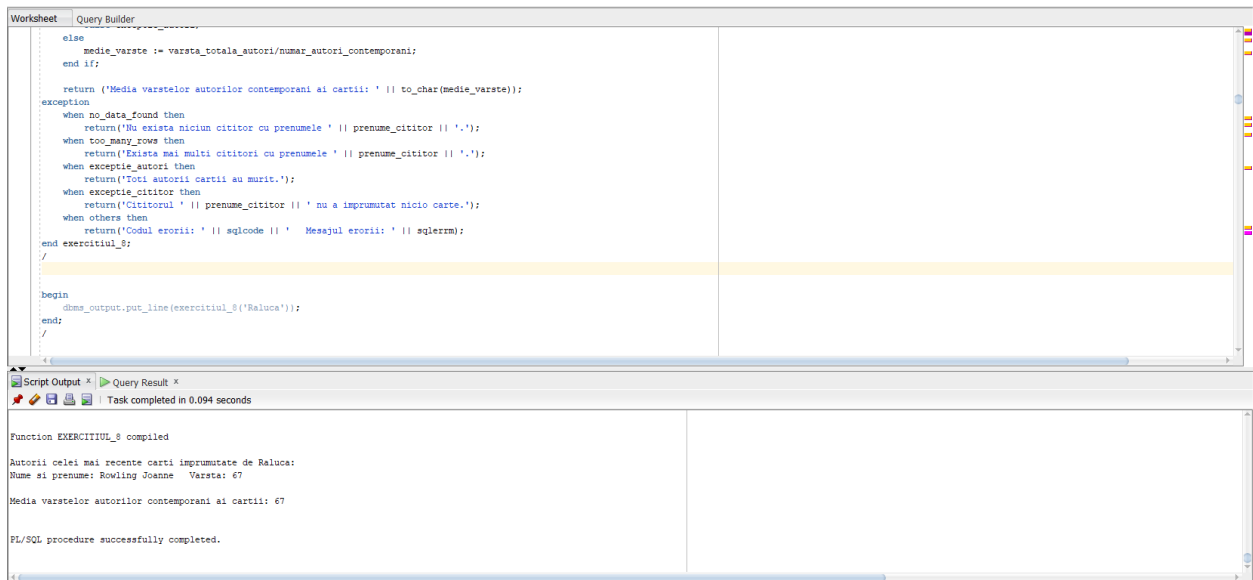
when exceptie_cititor then
    return('Cititorul ' || prenume_cititor || ' nu a imprumutat nicio carte.');
```

```

end exercitiul_8;
/
```

```

begin
    dbms_output.put_line(exercitiul_8('Raluca'));
end;
/
```



The screenshot displays the Oracle SQL Developer environment. The 'Query Builder' tab contains the following PL/SQL code:

```

else
    medie_varste := varsta_totala_autori/numar_autori_contemporani;
end if;

return ('Media varstelor autorilor contemporani ai cartii: ' || to_char(medie_varste));
exception
when no_data_found then
    return('Nu exista niciun cititor cu prenumele ' || prenume_cititor || '.');
when too_many_rows then
    return('Exista mai multi cititori cu prenumele ' || prenume_cititor || '.');
when exceptie_autori then
    return('Toti autorii cartii au murit.');
```

```

when exceptie_cititor then
    return('Cititorul ' || prenume_cititor || ' nu a imprumutat nicio carte.');
```

```

when others then
    return('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);
end exercitiul_8;
/

begin
    dbms_output.put_line(exercitiul_8('Raluca'));
end;
/
```

The 'Script Output' tab shows the execution results:

```

Function EXERCITIUL_8 compiled
Autorii celei mai recente carti imprumutate de Raluca:
Nume si prenume: Rowling Joanne  Varsta: 67
Media varstelor autorilor contemporani ai cartii: 67
PL/SQL procedure successfully completed.
```

The 'Query Result' tab is empty, indicating no data was returned from a query.

```

begin
    dbms_output.put_line(exercitiul_8('Elena'));
end;
/
```

The screenshot shows the SQL Developer interface with a worksheet titled 'Query Builder'. The main editor contains a PL/SQL procedure named `exercitiul_8`. The procedure has an `exception` block with several `when` clauses: `when no_data_found then`, `when too_many_rows then`, `when exceptie_autori then`, `when exceptie_cititor then`, and `when others then`. Each clause returns a specific message. The procedure ends with `end exercitiul_8;`. Below the procedure, there is a `begin` block with a call to `dbms_output.put_line(exercitiul_8('Raluca'));`, followed by `end;` and `/`. The bottom pane shows the 'Script Output' tab with the message 'Task completed in 0.146 seconds'. The 'Query Result' tab shows the output of the procedure: 'Autorii celei mai recente carti imprumutate de Elena: Nume si prenume: Rebreanu Liviu Deces 01-SEP-44 Nume si prenume: Sadoveanu Mihail Deces 19-OCT-61 Toti autorii cartii au murit. PL/SQL procedure successfully completed.'

```
exception
when no_data_found then
    return('Nu exista niciun cititor cu prenumele ' || prenume_cititor || '.');
when too_many_rows then
    return('Exista mai multi cititori cu prenumele ' || prenume_cititor || '.');
when exceptie_autori then
    return('Toti autorii cartii au murit.');
```

```
begin
    dbms_output.put_line(exercitiul_8('Raluca'));
end;
/
```

Task completed in 0.146 seconds

PL/SQL procedure successfully completed.

Autorii celei mai recente carti imprumutate de Elena:  
Nume si prenume: Rebreanu Liviu Deces 01-SEP-44  
Nume si prenume: Sadoveanu Mihail Deces 19-OCT-61

Toti autorii cartii au murit.

PL/SQL procedure successfully completed.

```
begin
    dbms_output.put_line(exercitiul_8('Mara'));
end;
/
```

The screenshot shows the SQL Developer interface with a worksheet titled 'Query Builder'. The main editor contains a PL/SQL procedure named `exercitiul_8`. The procedure has an `exception` block with several `when` clauses: `when too_many_rows then`, `when exceptie_autori then`, `when exceptie_cititor then`, and `when others then`. Each clause returns a specific message. The procedure ends with `end exercitiul_8;`. Below the procedure, there is a `begin` block with three calls to `dbms_output.put_line(exercitiul_8('Raluca'));`, `dbms_output.put_line(exercitiul_8('Elena'));`, and `dbms_output.put_line(exercitiul_8('Mara'));`, each followed by `end;` and `/`. The bottom pane shows the 'Script Output' tab with the message 'Task completed in 0.107 seconds'. The 'Query Result' tab shows the output of the procedure: 'Nume si prenume: Sadoveanu Mihail Deces 19-OCT-61 Toti autorii cartii au murit. PL/SQL procedure successfully completed. Cititorul Mara nu a imprumutat nicio carte. PL/SQL procedure successfully completed.'

```
when too_many_rows then
    return('Exista mai multi cititori cu prenumele ' || prenume_cititor || '.');
when exceptie_autori then
    return('Toti autorii cartii au murit.');
```

```
begin
    dbms_output.put_line(exercitiul_8('Raluca'));
end;
/
begin
    dbms_output.put_line(exercitiul_8('Elena'));
end;
/
begin
    dbms_output.put_line(exercitiul_8('Mara'));
end;
/
```

Task completed in 0.107 seconds

Nume si prenume: Sadoveanu Mihail Deces 19-OCT-61

Toti autorii cartii au murit.

PL/SQL procedure successfully completed.

Cititorul Mara nu a imprumutat nicio carte.

PL/SQL procedure successfully completed.

Am adăugat un nou cititor cu prenumele Raluca pentru a verifica excepția `too_many_rows`:

```
insert into cititori values(7, 'Ioana', 'Raluca', '02-JAN-2009', '095847584');
```

	ID_CIT...	NUME	PRENUME	DATA_NASTERE	TELEFON
1		1 Rogoza	Raluca	08-JAN-02	0741837264
2		2 Stoica	Elena	07-NOV-08	0778347287
3		3 Cojoc	Georgiana	17-JAN-00	0756123656
4		4 Obreja	Carina	05-DEC-80	0723457846
5		5 Talpalariu	Iulia	03-JUL-65	0789648247
6		6 Bejan	Mara	25-AUG-00	0746590478
7		7 Ioana	Raluca	02-JAN-09	095847584

```

begin
    dbms_output.put_line(exercitiul_8('Raluca'));
end;
/

```

The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Query Builder', contains a PL/SQL script. The script includes a function 'exercitiul\_8' that checks if a borrower has borrowed a book, followed by three calls to 'dbms\_output.put\_line' for the names 'Raluca', 'Elena', and 'Mara'. It also includes an 'insert' statement for a new borrower 'Ioana' with the name 'Raluca' and a date of birth '02-JAN-2009'. The bottom pane, titled 'Script Output', shows the execution results: '1 row inserted.', 'Exista mai multi cititori cu prenumele Raluca.', and 'PL/SQL procedure successfully completed.' The status bar indicates the task was completed in 0.087 seconds.

```

Worksheet  Query Builder
exercitiul_8:
when exceptie_cititor then
    return('Cititorul ' || prenume_cititor || ' nu a imprumutat nicio carte.');
```

```

end exercitiul_8;
/

begin
    dbms_output.put_line(exercitiul_8('Raluca'));
end;
/

begin
    dbms_output.put_line(exercitiul_8('Elena'));
end;
/

begin
    dbms_output.put_line(exercitiul_8('Mara'));
end;
/

insert into cititori values(7, 'Ioana', 'Raluca', '02-JAN-2009', '095847584');
begin
    dbms_output.put_line(exercitiul_8('Raluca'));
end;
/

```

Script Output \* Query Result \*

Task completed in 0.087 seconds

```

1 row inserted.

Exista mai multi cititori cu prenumele Raluca.

PL/SQL procedure successfully completed.

```

```

begin
    dbms_output.put_line(exercitiul_8('Ana'));
end;
/

```

The screenshot shows the Oracle SQL Developer interface. The top pane, titled 'Worksheet', contains a PL/SQL script with three blocks. The first block calls `doma_output.put_line(exercitiul_9('Elena'));`. The second block calls `doma_output.put_line(exercitiul_9('Mara'));`. The third block inserts a row into the `cititori` table and then calls `doma_output.put_line(exercitiul_9('Raluca'));`. The bottom pane, titled 'Script Output', shows the execution results. It indicates that 1 row was inserted, lists the existing readers (Elena, Mara, Raluca), and confirms the successful completion of the PL/SQL procedure.

```

begin
  doma_output.put_line(exercitiul_9('Elena'));
end;
/

begin
  doma_output.put_line(exercitiul_9('Mara'));
end;
/

insert into cititori values(7, 'Ioana', 'Raluca', '02-JAN-2009', '055847584');
begin
  doma_output.put_line(exercitiul_9('Raluca'));
end;
/

begin
  doma_output.put_line(exercitiul_9('Ana'));
end;
/

```

Script Output x Query Result x  
Task completed in 0.137 seconds

```

1 row inserted.

Exista mai multi cititori cu prenumele Raluca.

PL/SQL procedure successfully completed.

Nu exista niciun cititor cu prenumele Ana.

PL/SQL procedure successfully completed.

```

## 8. EXERCITIUL 9

Cerință: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un **subprogram stocat independent de tip procedură** care să utilizeze **într-o singură comandă SQL** 5 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.

Pentru o carte cu denumirea dată să se afle editura ei, numărul de împrumuturi (luându-se în calcul orice copie a cărții), numărul de cititori care au împrumutat-o, numărul de copii ale cărții care au fost împrumutate, id-urile copiilor cărții care sunt disponibile la momentul actual și raportul dintre numărul de copii disponibile care au fost împrumutate și numărul total de copii împrumutate ale cărții.

```
create or replace procedure exercitiul_9(denumire_carte carti.denumire%type)
is
```

```

  type vector is varray(100) of number;
  v_copii_carti_disponibile vector := vector();
  numar_cititori number;
  numar_copii_carti_imprumutate number := 0;
  numar_imprumuturi number;
  editura varchar2(20);
  data date;
```

```

copii_disponibile number;
raport float;
imprumuturi number;
begin
select count(id_carte_imprumutata)
into numar_imprumuturi
from carti_imprumutate
where id_carte = (select id_carte
                  from carti
                  where upper(carti.denumire) = upper(denumire_carte));
if numar_imprumuturi != 0 then
select e.ume, count(distinct(c.id_cititor)), count(cc.id_copie), min(data_nastere)
into editura, numar_cititori, numar_copii_carti_imprumutate, data
from carti_imprumutate ci, cititori c, carti ca, edituri e, copii_carti cc
where ca.id_carte = (select id_carte
                    from carti
                    where upper(carti.denumire) = upper(denumire_carte)) and ci.id_cititor =
c.id_cititor and ci.id_carte = ca.id_carte and ca.id_editura = e.id_editura and cc.id_carte =
ci.id_carte and ci.id_copie = cc.id_copie
group by e.ume;
dbms_output.put_line('Denumirea cartii este: ' || denumire_carte);
dbms_output.put_line('Editura la care se gaseste cartea este: ' || editura);
dbms_output.put_line('Numarul de cititori care au imprumutat cartea este: ' ||
numar_cititori);
dbms_output.put_line('Numarul de imprumuturi este: ' || numar_imprumuturi);
dbms_output.put_line('Numarul de copii de carte imprumutate este: ' ||
numar_copii_carti_imprumutate);

select ci.id_copie
bulk collect into v_copii_carti_disponibile
from copii_carti cc, carti_imprumutate ci
where ci.id_carte = (select id_carte
                    from carti
                    where upper(carti.denumire) = upper('Harry Potter')) and disponibilitate = 'da' and
data_restituire_efectiva is not null
and cc.id_carte = ci.id_carte and cc.id_copie = ci.id_copie;

dbms_output.put('Id-urile copiilor cartilor care sunt disponibile: ');
for i in v_copii_carti_disponibile.first..v_copii_carti_disponibile.last loop
    dbms_output.put(v_copii_carti_disponibile(i) || ' ');
end loop;

```

```

        dbms_output.new_line();
    end if;
    --nr copii imprumutate disponibile / numar copii total imprumutate
    raport := v_copii_carti_disponibile.count/numar_copii_carti_imprumutate;

    dbms_output.put_line('Raportul dintre numarul de copii disponibile care au fost
imprumutate si numarul total de copii imprumutate ale cartii date este egal cu ' || raport || '.');

exception
    when no_data_found then
        dbms_output.put_line('Nu exista nicio carte cu aceasta denumire.');
```

```

    when too_many_rows then
        dbms_output.put_line('Exista mai multe carti cu aceasta denumire.');
```

```

    when zero_divide then
        dbms_output.put_line('Ai facut o impartire la 0! Nu a fost imprumutata nicio copie a
cartii.');
```

```

    when others then
        dbms_output.put_line('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);
end exercitiul_9;
/

begin
    exercitiul_9('Harry Potter');
end;
/
```

The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Worksheet - Query Builder', contains the PL/SQL script for 'exercitiul\_9'. The script calculates the ratio of available copies to total borrowed copies for a given book title and handles various exceptions. The bottom pane, titled 'Script Output \*', shows the execution results. It indicates that the procedure was compiled successfully and then executed, displaying the output for the book 'Harry Potter'.

```

Worksheet - Query Builder
dbms_output.new_line();
--nr copii imprumutate disponibile / numar copii total imprumutate
raport := v_copii_carti_disponibile.count/numar_copii_carti_imprumutate;

dbms_output.put_line('Raportul dintre numarul de copii disponibile care au fost imprumutate si numarul total de copii imprumutate ale cartii date este egal cu ' || raport || '.');

exception
    when no_data_found then
        dbms_output.put_line('Nu exista nicio carte cu aceasta denumire.');
```

```

    when too_many_rows then
        dbms_output.put_line('Exista mai multe carti cu aceasta denumire.');
```

```

    when zero_divide then
        dbms_output.put_line('Ai facut o impartire la 0! Nu a fost imprumutata nicio copie a cartii.');
```

```

    when others then
        dbms_output.put_line('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);
end exercitiul_9;
/

begin
    exercitiul_9('Harry Potter');
end;
/

Script Output *
Task completed in 0.079 seconds

Procedure EXERCITIUL_9 compiled
Denumirea cartii este: Harry Potter
Editura la care se gaseste cartea este: Humanitas
Numarul de cititori care au imprumutat cartea este: 3
Numarul de imprumuturi este: 4
Numarul de copii de carte imprumutate este: 4
Id-urile copiilor cartilor care sunt disponibile: 3 2 5
Raportul dintre numarul de copii disponibile care au fost imprumutate si numarul total de copii imprumutate ale cartii date este egal cu .75.

PL/SQL procedure successfully completed.
```

```
begin
```



```

exercitiul_9('Hobbitul');
end;
/

```

The screenshot shows a database query builder interface. The top pane displays a PL/SQL procedure named `exercitiul_9` that calculates the number of copies of books borrowed from a table named `carti`. The procedure uses a cursor to iterate over the books and calculate the total number of copies. The bottom pane shows the output of the procedure, which includes the number of books borrowed, the number of copies, and the total number of copies for the book 'Hobbitul'.

```

Worksheet  Query Builder
report := v_copii_carti_disponibile.count/numar_copii_carti_imprumutate;
dms_output.put_line('Raportul dintre numarul de copii disponibile care au fost imprumutate si numarul total de copii imprumutate ale cartii date este egal cu ' || raport || '.');

exception
when no_data_found then
dms_output.put_line('Nu exista nicio carte cu aceasta denumire.');
```

Script Output x Query Result x  
Task completed in 0.122 seconds

Numarul de cititori care au imprumutat cartea este: 3  
Numarul de imprumuturi este: 4  
Numarul de copii de carte imprumutate este: 4  
Id-urile copiilor cartilor care sunt disponibile: 3 2 5  
Raportul dintre numarul de copii disponibile care au fost imprumutate si numarul total de copii imprumutate ale cartii date este egal cu .75.

PL/SQL procedure successfully completed.  
Nu exista nicio carte cu aceasta denumire.  
PL/SQL procedure successfully completed

Am inserat în tabelul carti o carte, fără a însera în tabela CARTI\_IMPRUMUTATE nimic.

```

insert into carti values(6, 'Hobbitul', 2009, 467, 2, 2);
select * from carti;

```

	ID_CARTE	DENUMIRE	AN_PUBLICATIE	NUMAR_PAGINI	NUMAR_COPII	ID_EDITURA
1	1	Harry Potter	2005	566	4	1
2	2	Hotul	2012	600	6	2
3	3	Baltagul	1930	123	3	2
4	4	Ion	1920	289	5	3
5	5	Enigma Otiliei	1938	400	1	4
6	6	Hobbitul	2009	467	2	2

```

begin
    exercitiul_9('Hobbitul');
end;
/

```

Worksheet Query Builder

```

when no_data_found then
    dms_output.put_line('Nu exista nicio carte cu aceasta denumire.');
```

```

when too_many_rows then
    dms_output.put_line('Exista mai multe carti cu aceasta denumire.');
```

```

when zero_divide then
    dms_output.put_line('Ai facut o impartire la 0! Nu a fost imprumutata nicio copie a cartii.');
```

```

when others then
    dms_output.put_line('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);
end exercitiul_9;
/

begin
    exercitiul_9('Harry Potter');
end;
/

begin
    exercitiul_9('Hobbitul');
end;
/

insert into carti values(6, 'Hobbitul', 2009, 467, 2, 2);
select * from carti;
begin
    exercitiul_9('Hobbitul');
end;
/

```

Script Output x

Query Result x

Task completed in 0.123 seconds

```

PL/SQL procedure successfully completed.

Ai facut o impartire la 0! Nu a fost imprumutata nicio copie a cartii.

PL/SQL procedure successfully completed.

```

insert into carti values(7, 'Hobbitul', 2009, 467, 2, 2);  
 select \* from carti;

	ID_CARTE	DENUMIRE	AN_PUBLICATIE	NUMAR_PAGINI	NUMAR_COPII	ID_EDITURA
1	1	Harry Potter	2005	566	4	1
2	2	Hotul	2012	600	6	2
3	3	Baltagul	1930	123	3	2
4	4	Ion	1920	289	5	3
5	5	Enigma Otiliei	1938	400	1	4
6	6	Hobbitul	2009	467	2	2
7	7	Hobbitul	2009	467	2	2

```

begin
    exercitiul_9('Hobbitul');
end;
/

```

```

Worksheet | Query Builder
dms_output.put_line('Ai rasut o impartire la u! Nu a fost imprumutata nicio copie a cartii.');
```

```

when
dms_output.put_line('Codul erorii: ' || sqlcode || '   Mesajul erorii: ' || sqlerrm);
end exercitiul_9;
/

begin
exercitiul_9('Harry Potter');
end;
/

begin
exercitiul_9('Hobbitul');
end;
/

insert into carti values(6, 'Hobbitul', 2009, 467, 2, 2);
select * from carti;
begin
exercitiul_9('Hobbitul');
end;
/

insert into carti values(7, 'Hobbitul', 2009, 467, 2, 2);
select * from carti;
exercitiul_9('Hobbitul');
end;
/

```

Script Output x | Query Result x

Task completed in 0.079 seconds

```

PL/SQL procedure successfully completed.

Exista mai multe carti cu aceasta denumire.

PL/SQL procedure successfully completed.

```

## 9. EXERCIȚIUL 10

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

Creați un trigger astfel încât numai utilizatorul 'raluca' să aibă permisiunea de a insera în tabela BIBLIOTECARI.

```

create or replace trigger exercitiul_10
before insert on bibliotecari
begin
    if user != upper('raluca') then
        raise_application_error(-20004,'Doar userul RALUCA are voie sa introduca date in tabela
bibliotecari!');
    end if;
end;
/

```

Userul principal:

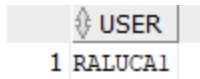
USER
1 RALUCA

Am creat un nou user 'raluca1' și i-am dat permisiunea să insereze în tabelul BIBLIOTECARI.

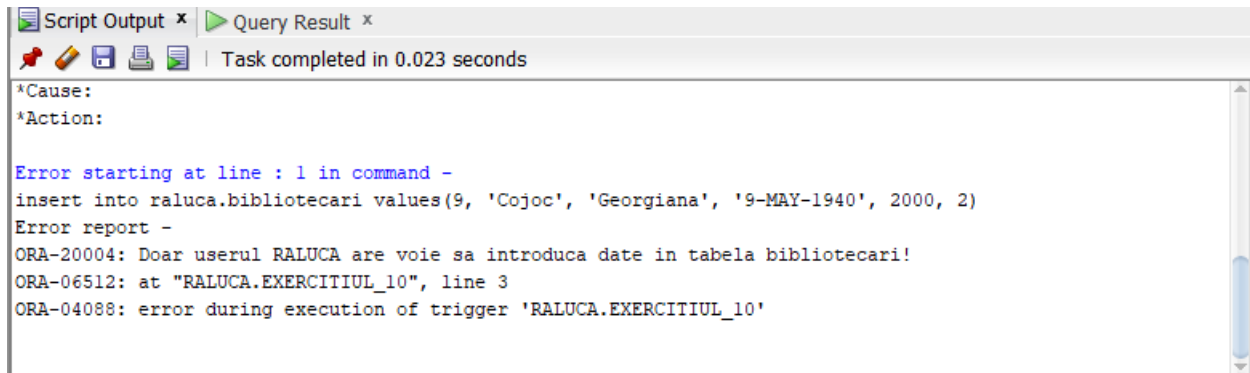
```

select user from dual;

```



```
insert into raluca.bibliotecari values(9, 'Cojoc', 'Georgiana', '9-MAY-1940', 2000, 2);
```



```
drop trigger exercitiul_10;
```

## 10. EXERCITIUL 11

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

Creați un trigger astfel încât la fiecare insert, delete, update pe coloana *data\_restituire\_efectiva* din tabelul *CARTI\_IMPRUMUTATE* să se actualizeze și atributul *disponibilitate* din tabela *COPII\_CARTI* (chiar dacă pentru unele copii nu a fost setată disponibilitatea înainte, este setată cu această ocazie).

```
create or replace trigger exercitiul_11
after insert or delete or update of data_restituire_efectiva on carti_imprumutate
for each row
begin
    if inserting then
        if :new.data_restituire_efectiva is null then
            update copii_carti
            set disponibilitate = 'nu'
            where id_carte = :new.id_carte and id_copie = :new.id_copie;
        else
            update copii_carti
            set disponibilitate = 'da'
```

```

        where id_carte = :old.id_carte and id_copie = :old.id_copie;
    end if;
elsif updating then
    if :new.data_restituire_efectiva is null then
        update copii_carti
        set disponibilitate = 'nu'
        where id_carte = :old.id_carte and id_copie = :old.id_copie;
    else
        update copii_carti
        set disponibilitate = 'da'
        where id_carte = :old.id_carte and id_copie = :old.id_copie;
    end if;
else
    if :old.data_restituire_efectiva is null then
        update copii_carti
        set disponibilitate = 'nu'
        where id_carte = :old.id_carte and id_copie = :old.id_copie;
    else
        update copii_carti
        set disponibilitate = 'da'
        where id_carte = :old.id_carte and id_copie = :old.id_copie;
    end if;
end if;
end exercitiul_11;
/

```

Tabela CARTI\_IMPRUMUTATE înainte de insert:

ID_CARTE_IMPRUMUTATA	ID_CITITOR	ID_CARTE	ID_COPIE	ID_AUTOR	DATA_IMPRUMUT	DATA_RESTITUIRE_IMPUSA	DATA_RESTITUIRE_EFECTIVA
1	1	1	1	3	1 08-JAN-22	20-JAN-22	30-JAN-22
2	2	1	5	1	2 21-MAR-22	30-MAR-22	05-APR-22
3	3	2	1	2	2 03-JAN-22	13-JAN-22	10-JAN-22
4	4	2	3	5	5 05-MAY-22	11-MAY-22	10-MAY-22
5	5	3	2	4	3 14-MAR-22	25-MAR-22	22-MAR-22
6	6	3	4	2	1 20-MAY-22	30-MAY-22	(null)
7	7	4	2	1	2 24-SEP-21	01-OCT-21	06-OCT-22
8	8	4	3	1	4 01-NOV-22	14-NOV-22	05-NOV-22
9	9	5	1	5	2 05-MAY-21	10-MAY-21	07-MAY-22
10	10	5	1	4	2 19-FEB-20	27-FEB-20	(null)
11	11	1	2	1	1 23-DEC-22	31-DEC-22	01-JAN-23
12	12	5	2	3	1 26-DEC-22	03-JAN-23	(null)

Tabela COPII\_CARTI înainte de insert:

	ID_COPIE	ID_CARTE	ID_RAFT	DISPONIBILITATE
1	3	1	3	da
2	1	5	2	da
3	2	1	6	da
4	5	3	4	da
5	4	2	5	da
6	2	4	7	da
7	4	1	7	da
8	1	2	5	da
9	1	3	1	da
10	5	1	2	da
11	1	1	2	da
12	3	2	4	da
13	2	5	8	da

insert into carti\_imprumutate(id\_carte\_imprumutata, id\_cititor, id\_carte, id\_copie, id\_autor, data\_imprumut, data\_restituire\_impusa) values(13, 1, 5, 2, 2, '5-JAN-23', '10-JAN-23');

Worksheet

Query Builder

```

elseif updating then
    if :new.data_restituire_efectiva is null then
        update copii_carti
        set disponibilitate = 'nu'
        where id_carte = :old.id_carte and id_copie = :old.id_copie;
    else
        update copii_carti
        set disponibilitate = 'da'
        where id_carte = :old.id_carte and id_copie = :old.id_copie;
    end if;
else
    if :old.data_restituire_efectiva is null then
        update copii_carti
        set disponibilitate = 'nu'
        where id_carte = :old.id_carte and id_copie = :old.id_copie;
    else
        update copii_carti
        set disponibilitate = 'da'
        where id_carte = :old.id_carte and id_copie = :old.id_copie;
    end if;
end if;
end exercitiul_11;
/

drop trigger exercitiul_11;

select * from carti_imprumutate;
select * from copii_carti;

--teste pt insert
insert into carti_imprumutate(id_carte_imprumutata, id_cititor, id_carte, id_copie, id_autor, data_imprumut, data_restituire_impusa) values(13, 1, 5, 2, 2, '5-JAN-23', '10-JAN-23');
insert into carti_imprumutate(id_carte_imprumutata, id_cititor, id_carte, id_copie, id_autor, data_imprumut, data_restituire_impusa, data_restituire_efectiva) values(14, 1, 1, 3, 2, '14-DEC-22',

```

Script Output

Query Result

Task completed in 0.031 seconds

Trigger EXERCITIUL\_11 compiled

1 row inserted.

Actualizarea tabeli COPII\_CARTI:

ID_COPIE	ID_CARTE	ID_RAFT	DISPONIBILITATE
1	3	1	3 da
2	1	5	2 da
3	2	1	6 da
4	5	3	4 da
5	4	2	5 da
6	2	4	7 da
7	4	1	7 da
8	1	2	5 da
9	1	3	1 da
10	5	1	2 da
11	1	1	2 da
12	3	2	4 da
13	2	5	8 nu

insert into carti\_imprumutate(id\_carte\_imprumutata, id\_cititor, id\_carte, id\_copie, id\_autor, data\_imprumut, data\_restituire\_impusa, data\_restituire\_efectiva) values(14, 1, 1, 3, 2, '14-DEC-22', '23-DEC-21', '28-DEC-22');

Tabela COPII\_CARTI după cel de-al doilea insert:

ID_COPIE	ID_CARTE	ID_RAFT	DISPONIBILITATE
1	3	1	3 da
2	1	5	2 da
3	2	1	6 da
4	5	3	4 da
5	4	2	5 da
6	2	4	7 da
7	4	1	7 da
8	1	2	5 da
9	1	3	1 da
10	5	1	2 da
11	1	1	2 da
12	3	2	4 da
13	2	5	8 nu

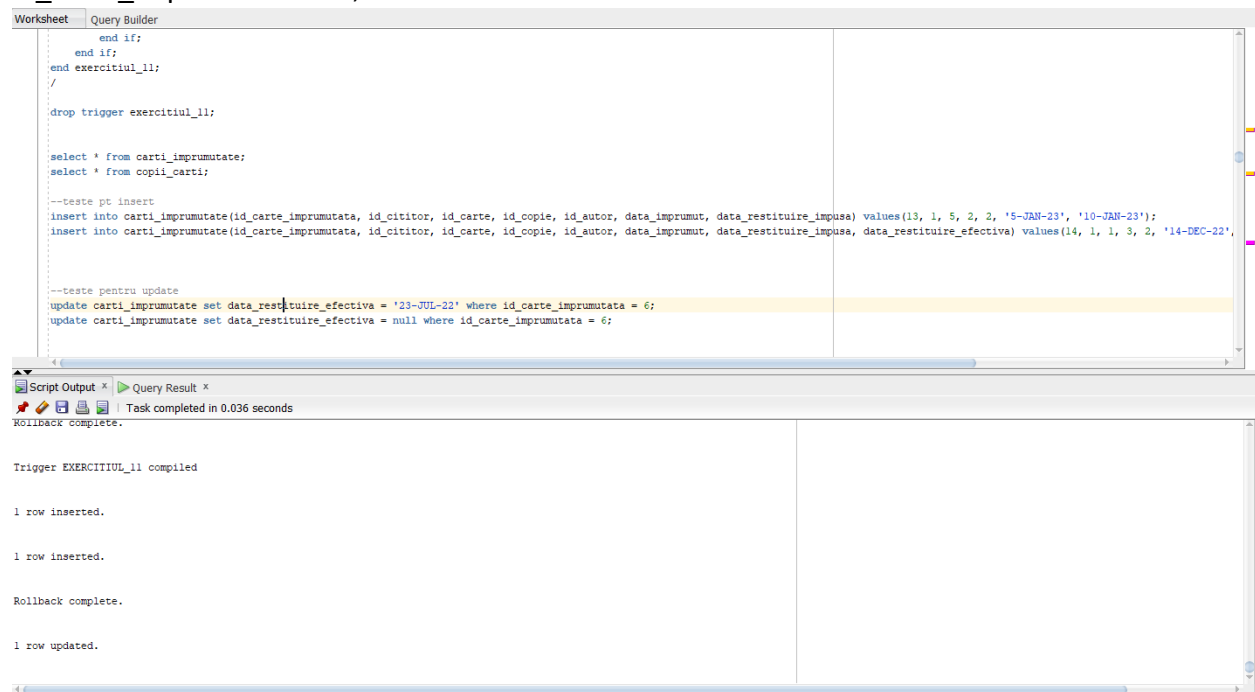
Tabela CARTI\_IMPRUMUTATE după cele 2 insert-uri:

ID_CARTE_IMPRUMUTATA	ID_CITITOR	ID_CARTE	ID_COPIE	ID_AUTOR	DATA_IMPRUMUT	DATA_RESTITUIRE_IMPUSA	DATA_RESTITUIRE_EFECTIVA
1	13	1	5	2	2 05-JAN-23	10-JAN-23	(null)
2	14	1	1	3	2 14-DEC-22	23-DEC-21	28-DEC-22
3	1	1	1	3	1 08-JAN-22	20-JAN-22	30-JAN-22
4	2	1	5	1	2 21-MAR-22	30-MAR-22	05-APR-22
5	3	2	1	2	2 03-JAN-22	13-JAN-22	10-JAN-22
6	4	2	3	5	5 05-MAY-22	11-MAY-22	10-MAY-22
7	5	3	2	4	3 14-MAR-22	25-MAR-22	22-MAR-22
8	6	3	4	2	1 20-MAY-22	30-MAY-22	(null)
9	7	4	2	1	2 24-SEP-21	01-OCT-21	06-OCT-22
10	8	4	3	1	4 01-NOV-22	14-NOV-22	05-NOV-22
11	9	5	1	5	2 05-MAY-21	10-MAY-21	07-MAY-22
12	10	5	1	4	2 19-FEB-20	27-FEB-20	(null)
13	11	1	2	1	1 23-DEC-22	31-DEC-22	01-JAN-23
14	12	5	2	3	1 26-DEC-22	03-JAN-23	(null)

După cele 2 insert-uri am făcut rollback.

Primul update:

update carti\_imprumutate set data\_restituire\_efectiva = '23-JUL-22' where  
id\_carte\_imprumutata = 6;



```
end if;
end if;
end exercitiul_11;
/

drop trigger exercitiul_11;

select * from carti_imprumutate;
select * from copii_carti;

--teste pt insert
insert into carti_imprumutate(id_carte_imprumutata, id_cititor, id_carte, id_copie, id_autor, data_imprumut, data_restituire_impusa) values(13, 1, 5, 2, 2, '5-JAN-23', '10-JAN-23');
insert into carti_imprumutate(id_carte_imprumutata, id_cititor, id_carte, id_copie, id_autor, data_imprumut, data_restituire_impusa, data_restituire_efectiva) values(14, 1, 1, 3, 2, '14-DEC-22', null);

--teste pentru update
update carti_imprumutate set data_restituire_efectiva = '23-JUL-22' where id_carte_imprumutata = 6;
update carti_imprumutate set data_restituire_efectiva = null where id_carte_imprumutata = 6;
```

Script Output x Query Result x

Task completed in 0.036 seconds

Rollback complete.

Trigger EXERCITIUL\_11 compiled

1 row inserted.

1 row inserted.

Rollback complete.

1 row updated.

Tabela CARTI\_IMPRUMUTATE după update:

ID_CARTE_IMPRUMUTATA	ID_CITITOR	ID_CARTE	ID_COPIE	ID_AUTOR	DATA_IMPRUMUT	DATA_RESTITUIRE_IMPUSA	DATA_RESTITUIRE_EFECTIVA
1	1	1	1	3	1 08-JAN-22	20-JAN-22	30-JAN-22
2	2	1	5	1	2 21-MAR-22	30-MAR-22	05-APR-22
3	3	2	1	2	2 03-JAN-22	13-JAN-22	10-JAN-22
4	4	2	3	5	5 05-MAY-22	11-MAY-22	10-MAY-22
5	5	3	2	4	3 14-MAR-22	25-MAR-22	22-MAR-22
6	6	3	4	2	1 20-MAY-22	30-MAY-22	23-JUL-22
7	7	4	2	1	2 24-SEP-21	01-OCT-21	06-OCT-22
8	8	4	3	1	4 01-NOV-22	14-NOV-22	05-NOV-22
9	9	5	1	5	2 05-MAY-21	10-MAY-21	07-MAY-22
10	10	5	1	4	2 19-FEB-20	27-FEB-20	(null)
11	11	1	2	1	1 23-DEC-22	31-DEC-22	01-JAN-23
12	12	5	2	3	1 26-DEC-22	03-JAN-23	(null)

Tabela COPII\_CARTI după update:



	ID_COPIE	ID_CARTE	ID_RAFT	DISPONIBILITATE
1	3	1	3	da
2	1	5	2	da
3	2	1	6	da
4	5	3	4	da
5	4	2	5	da
6	2	4	7	da
7	4	1	7	da
8	1	2	5	da
9	1	3	1	da
10	5	1	2	da
11	1	1	2	da
12	3	2	4	da
13	2	5	8	nu

Al doilea update:

update carti\_imprumutate set data\_restituire\_efectiva = null where id\_carte\_imprumutata = 6;

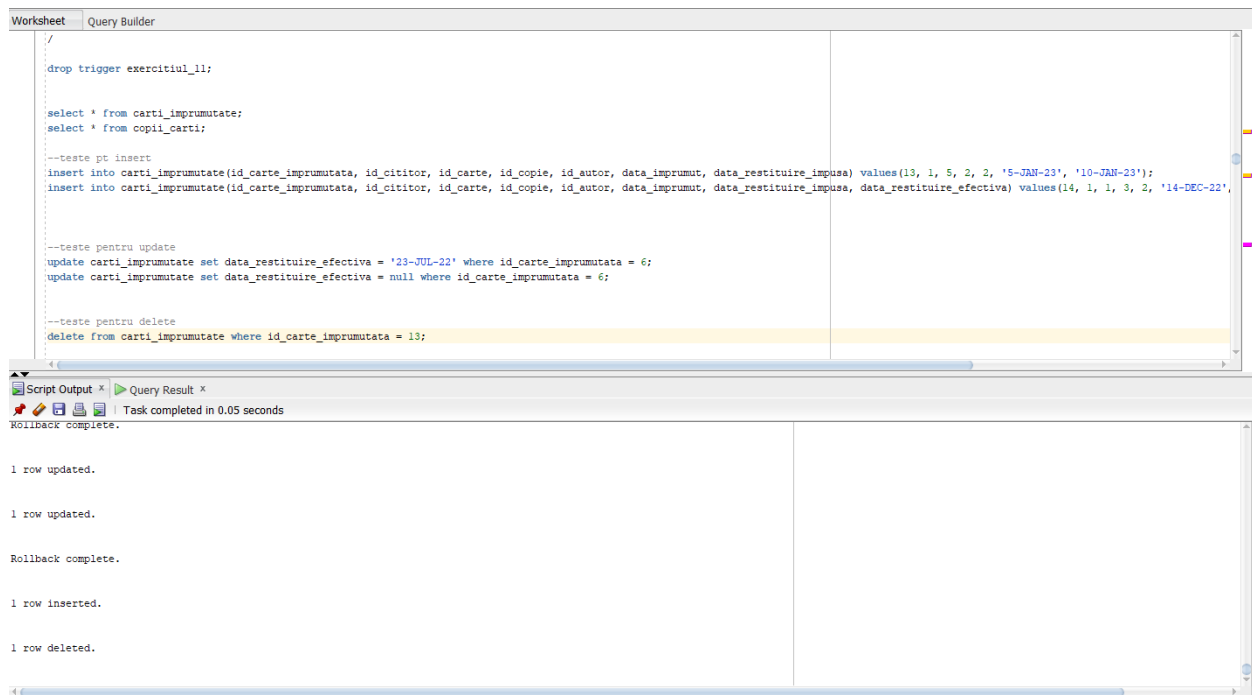
Tabela COPII\_CARTI după al doilea update:

	ID_C...	ID_CARTE	ID_RAFT	DISPONIBILITATE
1	3	1	3	da
2	1	5	2	da
3	2	1	6	da
4	5	3	4	da
5	4	2	5	da
6	2	4	7	nu
7	4	1	7	da
8	1	2	5	da
9	1	3	1	da
10	5	1	2	da
11	1	1	2	da
12	3	2	4	da
13	2	5	8	nu

După cele 2 update-uri am făcut rollback.

Am inserat din nou un rând pentru a intra trigger-ul și pe delete.

insert into carti\_imprumutate(id\_carte\_imprumutata, id\_cititor, id\_carte, id\_copie, id\_autor, data\_imprumut, data\_restituire\_impusa) values(13, 1, 5, 2, 2, '5-JAN-23', '10-JAN-23');



```
drop trigger exercitiul_11;

select * from carti_imprumutate;
select * from copii_carti;

--teste pt insert
insert into carti_imprumutate(id_carte_imprumutata, id_cititor, id_carte, id_copie, id_autor, data_imprumut, data_restituire_impusa) values(13, 1, 5, 2, '5-JAN-23', '10-JAN-23');
insert into carti_imprumutate(id_carte_imprumutata, id_cititor, id_carte, id_copie, id_autor, data_imprumut, data_restituire_impusa, data_restituire_efectiva) values(14, 1, 1, 3, 2, '14-DEC-22', null);

--teste pentru update
update carti_imprumutate set data_restituire_efectiva = '23-JUL-22' where id_carte_imprumutata = 6;
update carti_imprumutate set data_restituire_efectiva = null where id_carte_imprumutata = 6;

--teste pentru delete
delete from carti_imprumutate where id_carte_imprumutata = 13;
```

Script Output x Query Result x

Task completed in 0.05 seconds

Rollback complete.

1 row updated.

1 row updated.

Rollback complete.

1 row inserted.

1 row deleted.

Cele 2 tabele rămân la fel ca la început.

```
drop trigger exercitiul_11;
```

## 11. EXERCIȚIUL 12

Cerință: Definiți un *trigger* de tip LDD. Declanșați *trigger*-ul.

Creați un *trigger* care pune în tabelul istoric doar comenzile care au abut loc fără erori, celelalte fiind semnalate prin mesaj.

```
create table istoric(utilizator varchar2(30),
                    nume_baza_de_date varchar2(40),
                    eveniment varchar2(30),
                    nume_obiect varchar2(30),
                    data date);
```

```
create or replace trigger exercitiul_12
after create or drop or alter or servererror on database
begin
    if dbms_utility.format_error_stack is null then
```

```

insert into istoric values(sys.login_user, sys.database_name, sys.sysevent,
sys.dictionary_obj_name, sysdate);
else
    raise_application_error(-20010, 'S-a declanșat o eroare!');
end if;
end exercitiul_12;
/

```

```
create table test(id number(2));
```

```
select * from istoric;
```

The screenshot displays the Oracle SQL Developer environment. The top pane shows a SQL script with the following content:

```

--end exercitiul_12;
--/

create or replace trigger exercitiul_12
after create or drop or alter or servererror on database
begin
    if dbms_utility.format_error_stack is null then
        insert into istoric values(sys.login_user, sys.database_name, sys.sysevent, sys.dictionary_obj_name, sysdate);
    else
        raise_application_error(-20010, 'S-a declanșat o eroare!');
    end if;
end exercitiul_12;
/

drop trigger exercitiul_12;

create table test(id number(2));

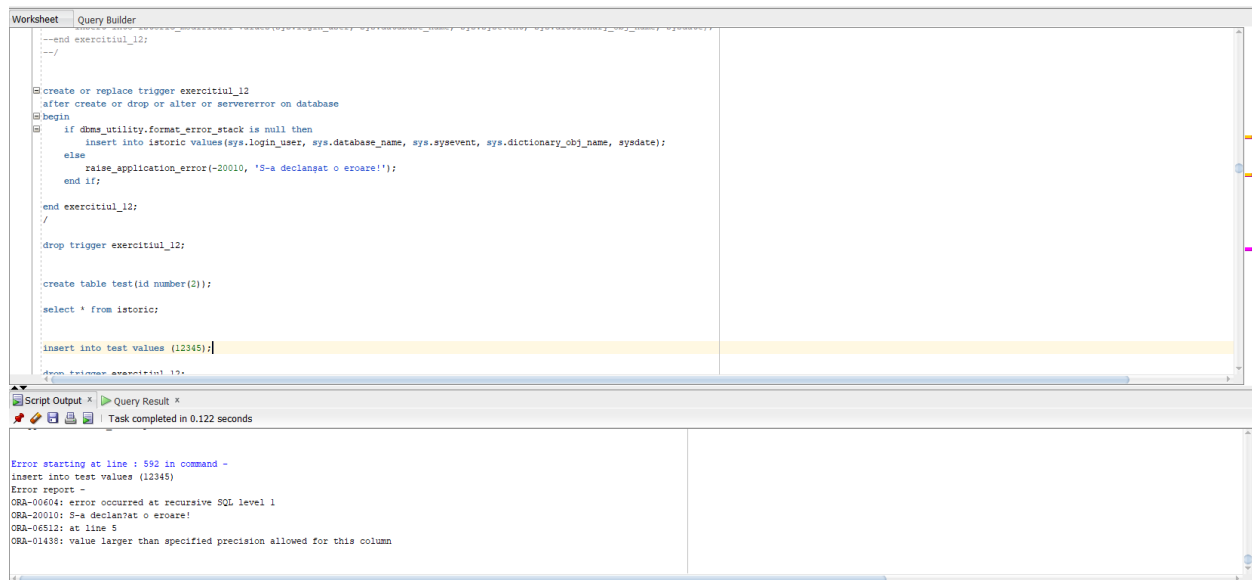
select * from istoric;

```

The bottom pane shows the 'Query Result' tab, indicating that 1 row was fetched in 0.043 seconds. The result is displayed in a table with the following columns and data:

UTILIZATOR	NUME_BAZA_DE_DATE	EVENTIMENT	NUME_OBIECT	DATA
1 RALUCA	XE	CREATE	TEST	13-JAN-23

```
insert into test values (12345);
```



drop table test;  
 drop trigger exercitiul\_12;  
 drop table istoric;

## 12. EXERCIȚIUL 13

Cerință: Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.

```
create or replace package exercitiul_13 as
  procedure exercitiul_6;
  procedure exercitiul_7;
  function exercitiul_8(prenume_cititor cititori.prenume%type) return varchar2;
  procedure exercitiul_9(denumire_carte carti.denumire%type);
end exercitiul_13;
/
```

```
create or replace package body exercitiul_13 as
  procedure exercitiul_6
  is
    type tablou_indexat is table of carti_imprumutate%rowtype index by pls_integer;
    tablou tablou_indexat;
    cod_carte carti_imprumutate.id_carte%type;
    type vector is varray(20) of number;
    coduri_autori vector := vector();
    cititori_penalizati vector := vector();
```

```

v_nume_autor autori.nume%type;
v_prenume_autor autori.prenume%type;
cod_cititor cititori.id_cititor%type;
numar_cititor_penalizat number := 0;
nume_carte carti.denumire%type;
nume_cititor cititori.nume%type;
prenume_cititor cititori.prenume%type;
gasit number := 0;
begin
    select *
    bulk collect into tablou
    from carti_imprumutate
    where data_restituire_efectiva is null;
    for i in tablou.first..tablou.last loop
        cod_carte := tablou(i).id_carte;
        cod_cititor := tablou(i).id_cititor;
        gasit := 0;
        if numar_cititor_penalizat != 0 then
            for j in cititori_penalizati.first..cititori_penalizati.last loop
                if cititori_penalizati(j) = cod_cititor then
                    gasit := 1;
                end if;
            end loop;
        end if;
        if gasit = 0 then
            numar_cititor_penalizat := numar_cititor_penalizat + 1;
            cititori_penalizati.extend;
            cititori_penalizati(numar_cititor_penalizat):= cod_cititor;
        end if;
        select id_autor
        bulk collect into coduri_autori
        from scrisa_de
        where id_carte = cod_carte;
        select denumire
        into nume_carte
        from carti
        where id_carte = cod_carte;
        dbms_output.put_line('Cartea: ' || nume_carte);
        dbms_output.put_line('Autorii cartii: ');
        for j in coduri_autori.first..coduri_autori.last loop
            select nume, prenume

```

```

        into v_numa_auor, v_prenume_auor
        from autori
        where id_auor = coduri_auori(j);
        dbms_output.put_line( v_numa_auor || ' ' || v_prenume_auor);
    end loop;
    dbms_output.new_line();
end loop;
dbms_output.put_line('Au fost penalizati ' || numar_cititor_penalizat || ' cititori.');
```

```

for i in cititori_penalizati.first..cititori_penalizati.last loop
    insert into primeste values(cititori_penalizati(i), 1, sysdate);
    select prenume, nume
    into nume_cititor, prenume_cititor
    from cititori
    where id_cititor = cititori_penalizati(i);
    dbms_output.put_line(nume_cititor || ' ' || prenume_cititor);
end loop;
end exercitiul_6;
```

procedure exercitiul\_7

is

```

    cursor c_etaje is
        select distinct(etaj)
        from rafturi
        group by etaj
        order by 1;
    cursor colegi_palier (v_numar_etaj bibliotecari.numar_etaj%type) is
        select id_bibliotecar, nume, prenume, salariu
        from bibliotecari
        where numar_etaj = v_numar_etaj
        order by 2, 3;
    b_numar_etaj bibliotecari.numar_etaj%type;
    b_numar_angajati_etaj number;
    type b_record is record (id bibliotecari.id_bibliotecar%type,
                             nume bibliotecari.nume%type,
                             prenume bibliotecari.prenume%type,
                             salariu bibliotecari.salariu%type);
    record_bibliotecari b_record;
    salariu_minim_etaj bibliotecari.salariu%type;
    salariu_maxim_etaj bibliotecari.salariu%type;
begin
```

```

open c_etaje;
loop
    fetch c_etaje into b_numar_etaj;
    exit when c_etaje%notfound;

    --calculez max si minim pe etaj
    select max(salariu), min(salariu), count(id_bibliotecar)
    into salariu_maxim_etaj, salariu_minim_etaj, b_numar_angajati_etaj
    from bibliotecari
    where numar_etaj = b_numar_etaj;

    dbms_output.put_line('Etajul ' || b_numar_etaj);
    dbms_output.new_line;
    dbms_output.put_line('Colegi: ');

    open colegi_palier(b_numar_etaj);

    loop
        fetch colegi_palier into record_bibliotecari;
        exit when colegi_palier%notfound;
        dbms_output.put(colegi_palier%rowcount || '. ' || record_bibliotecari.numa || ' ' ||
record_bibliotecari.prenume);
        if b_numar_angajati_etaj = 1 then
            dbms_output.put_line(' - > Este singurul angajat care lucreaza la acest etaj.');
```

```

            elsif record_bibliotecari.salariu = salariu_minim_etaj and record_bibliotecari.salariu =
salariu_maxim_etaj then
                dbms_output.put_line(' - > Acest angajat are salariul egal cu salariul maxim, dar si cu
salariu minim de la acest etaj.');
```

```

            elsif record_bibliotecari.salariu = salariu_maxim_etaj then
                dbms_output.put_line(' - > Acest angajat are salariul maxim de la acest etaj.');
```

```

            elsif record_bibliotecari.salariu = salariu_minim_etaj then
                dbms_output.put_line(' - > Acest angajat are salariul minim de la acest etaj.');
```

```

            end if;
        end loop;
        if colegi_palier%rowcount = 0 then
            dbms_output.put_line('La acest etaj nu lucreaza nimeni momentan.');
```

```

        end if;
        close colegi_palier;
        dbms_output.new_line;
    end loop;
close c_etaje;

```

```

end exercitiul_7;

function exercitiul_8(prenume_cititor cititori.prenume%type)
return varchar2
is
    medie_varste float;
    type vector is varray(100) of autori.nume%type;
    autori_contemporani vector := vector();
    autori_carte vector := vector();
    numar_autori_contemporani number := 0;
    varsta_totala_autori number := 0 ;
    varsta number := 0;
    deces date;
    nume_autor autori.nume%type;
    prenume_autor autori.prenume%type;
    exceptie_autori exception;
    exceptie_cititor exception;
    numar_carti_imprumutate number;
    prenume_cititor_cautat cititori.prenume%type;
begin
    select prenume
    into prenume_cititor_cautat
    from cititori
    where upper(cititori.prenume) = upper(prenume_cititor);
    select count(id_carte_imprumutate)
    into numar_carti_imprumutate
    from carti_imprumutate
    where id_cititor = (select id_cititor
                        from cititori
                        where upper(cititori.prenume) = upper(prenume_cititor));
    if numar_carti_imprumutate = 0 then
        raise exceptie_cititor;
    end if;
    --aflu cati autori are cartea
    select scria_de.id_autor
    bulk collect into autori_carte
    from carti_imprumutate join cititori on cititori.id_cititor = carti_imprumutate.id_cititor join
scria_de on carti_imprumutate.id_carte = scria_de.id_carte
    where carti_imprumutate.data_imprumut = (select max(data_imprumut)
                                             from carti_imprumutate
                                             where carti_imprumutate.id_cititor = (select id_cititor

```



```

                                from cititori
                                where upper(cititori.prenume) =
upper(prenume_cititor));
    --caut autorii contemporani
    dbms_output.put_line('Autorii celei mai recente carti imprumutate de ' || prenume_cititor
|| ': ');
    for i in autori_carte.first..autori_carte.last loop
        --dbms_output.put_line(autori_carte(i));
        select data_deces, nume, prenume
        into deces, nume_autor, prenume_autor
        from autori
        where id_autor = autori_carte(i);
        dbms_output.put('Nume si prenume: ' || nume_autor || ' ' || prenume_autor);
        if deces is null then
            select extract(year from sysdate) - extract(year from data_nastere)
            into varsta
            from autori
            where id_autor = autori_carte(i);
            dbms_output.put_line(' Varsta: ' || varsta);
            numar_autori_contemporani := numar_autori_contemporani + 1;
            varsta_totala_autori := varsta_totala_autori + varsta;
            autori_contemporani.extend();
            autori_contemporani(numar_autori_contemporani) := autori_carte(i);
        else
            dbms_output.put_line(' Deces ' || deces);
        end if;

    end loop;
    dbms_output.new_line;

    if numar_autori_contemporani = 0 then
        raise exceptie_autori;
    else
        medie_varste := varsta_totala_autori/numar_autori_contemporani;
    end if;

    return ('Media varstelor autorilor contemporani ai cartii: ' || to_char(medie_varste));
exception
    when no_data_found then
        return('Nu exista niciun cititor cu prenumele ' || prenume_cititor || '.');
    when too_many_rows then

```

```

        return('Exista mai multi cititori cu prenumele ' || prenume_cititor || '.');
when exceptie_autori then
    return('Toti autorii cartii au murit.');
```

```

when exceptie_cititor then
    return('Cititorul ' || prenume_cititor || ' nu a imprumutat nicio carte.');
```

```

end exercitiul_8;
```

```

procedure exercitiul_9(denumire_carte carti.denumire%type)
is
```

```

    type vector is varray(100) of number;
    v_copii_carti_disponibile vector := vector();
    numar_cititori number;
    numar_copii_carti_imprumutate number := 0;
    numar_imprumuturi number;
    editura varchar2(20);
    data date;
    copii_disponibile number;
    raport float;
    imprumuturi number;
    cod_carte carti.id_carte%type;
```

```

begin
```

```

    select id_carte
    into cod_carte
    from carti
    where upper(carti.denumire) = upper(denumire_carte);
    select count(id_carte_imprumutata)
    into numar_imprumuturi
    from carti_imprumutate
    where id_carte = (select id_carte
                      from carti
                      where upper(carti.denumire) = upper(denumire_carte));
    if numar_imprumuturi != 0 then
        select e.ume, count(distinct(c.id_cititor)), count(cc.id_copie), min(data_nastere)
        into editura, numar_cititori, numar_copii_carti_imprumutate, data
        from carti_imprumutate ci, cititori c, carti ca, edituri e, copii_carti cc
        where ca.id_carte = (select id_carte
                            from carti
                            where upper(carti.denumire) = upper(denumire_carte)) and ci.id_cititor
        = c.id_cititor and ci.id_carte = ca.id_carte and ca.id_editura = e.id_editura and cc.id_carte =
        ci.id_carte and ci.id_copie = cc.id_copie
```

```

group by e.num;
dbms_output.put_line('Denumirea cartii este: ' || denumire_carte);
dbms_output.put_line('Editura la care se gaseste cartea este: ' || editura);
dbms_output.put_line('Numarul de cititori care au imprumutat cartea este: ' ||
numar_cititori);
dbms_output.put_line('Numarul de imprumuturi este: ' || numar_imprumuturi);
dbms_output.put_line('Numarul de copii de carte imprumutate este: ' ||
numar_copii_carti_imprumutate);

select ci.id_copie
bulk collect into v_copii_carti_disponibile
from copii_carti cc, carti_imprumutate ci
where ci.id_carte = (select id_carte
                    from carti
                    where upper(carti.denumire) = upper('Harry Potter')) and disponibilitate = 'da'
and data_restituire_efectiva is not null
and cc.id_carte = ci.id_carte and cc.id_copie = ci.id_copie;

dbms_output.put('Id-urile copiilor cartilor care sunt disponibile: ');
for i in v_copii_carti_disponibile.first..v_copii_carti_disponibile.last loop
    dbms_output.put(v_copii_carti_disponibile(i) || ' ');
end loop;
dbms_output.new_line();
end if;
--nr copii imprumutate disponibile / numar copii total imprumutate
raport := v_copii_carti_disponibile.count/numar_copii_carti_imprumutate;

dbms_output.put_line('Raportul dintre numarul de copii disponibile care au fost
imprumutate si numarul total de copii imprumutate ale cartii date este egal cu ' || raport || '.');

exception
when no_data_found then
    dbms_output.put_line('Nu exista nicio carte cu aceasta denumire.');
```

```

when too_many_rows then
    dbms_output.put_line('Exista mai multe carti cu aceasta denumire.');
```

```

when zero_divide then
    dbms_output.put_line('Ai facut o impartire la 0! Nu a fost imprumutata nicio copie a
cartii.');
```

```

when others then
    dbms_output.put_line('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);
end exercitiul_9;
```

```
end exercitiul_13;
```

```
/
```

```
begin
```

```
    exercitiul_13.exercitiul_6();
```

```
end;
```

```
/
```

The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Worksheet' and 'Query Builder', contains the following PL/SQL code:

```
when others then
    dbms_output.put_line('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);
end exercitiul_9;

end exercitiul_13;
/

begin
    exercitiul_13.exercitiul_6();
end;
/
```

The bottom pane shows the 'Script Output' window with the following text:

Task completed in 0.035 seconds  
Package EXERCITIUL\_13 compiled

Package Body EXERCITIUL\_13 compiled

PL/SQL procedure successfully completed.

Cartea: Ion  
Autorii cartii:  
Rowling Joanne  
Zusak Markus

Cartea: Harry Potter  
Autorii cartii:  
Rebreanu Liviu  
Calinescu George  
Sadoveanu Mihail

Cartea: Hotul  
Autorii cartii:  
Rowling Joanne

Au fost penalizati 2 cititori.  
Georgiana Cojoc  
Iulia Talpalaru

```
begin
```

```
    exercitiul_13.exercitiul_7();
```

```
end;
```

```
/
```

```

Worksheet | Query Builder
begin
    exercitiul_13.exercitiul_6();
end;
/

begin
    exercitiul_13.exercitiul_7();
end;
/

Script Output x | Query Result x
Task completed in 0.045 seconds

Etajul 1
Colegi:
1. Fornica Gabriel - > Acest angajat are salariul maxim de la acest etaj.
2. Teleaga Mihaela - > Acest angajat are salariul minim de la acest etaj.
3. Tudor Lavinia - > Acest angajat are salariul minim de la acest etaj.

Etajul 2
Colegi:
1. Botesatu Raluca - > Acest angajat are salariul egal cu salariul maxim, dar si cu salariu minim de la acest etaj.
2. Popa Mariana - > Acest angajat are salariul egal cu salariul maxim, dar si cu salariu minim de la acest etaj.

Etajul 3
Colegi:
1. Boghiu Emilia - > Acest angajat are salariul minim de la acest etaj.
2. Penfil Otilia - > Acest angajat are salariul maxim de la acest etaj.

Etajul 4
Colegi:
1. Rafaela Camelia - > Este singurul angajat care lucreaza la acest etaj.

PL/SQL procedure successfully completed.

```

```

begin
    dbms_output.put_line(exercitiul_13.exercitiul_8('Raluca'));
end;
/

```

```

Worksheet | Query Builder

exception
when no_data_found then
    dbms_output.put_line('Nu exista nicio carte cu aceasta denumire.');
```

```

when too_many_rows then
    dbms_output.put_line('Exista mai multe carti cu aceasta denumire.');
```

```

when zero_divide then
    dbms_output.put_line('Ai facut o impartire la 0! Nu a fost imprumutata nicio copie a cartii.');
```

```

when others then
    dbms_output.put_line('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);
end exercitiul_9;

end exercitiul_13;
/

begin
    exercitiul_13.exercitiul_6();
end;
/

begin
    exercitiul_13.exercitiul_7();
end;
/

begin
    dbms_output.put_line(exercitiul_13.exercitiul_8('Raluca'));
end;
/

Script Output x | Query Result x
Task completed in 0.034 seconds

Autorii celei mai recente carti imprumutate de Raluca:
Nume si prenume: Rowling Joanne Varsta: 67

Media varstelor autorilor contemporani ai cartii: 67

PL/SQL procedure successfully completed.

```

```

begin
    exercitiul_13.exercitiul_9('Harry Potter');

```

end;

/

```
Worksheet | Query Builder
-----|-----
dms_output.put_line('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);
end exercitiul_9;

end exercitiul_13;
/

begin
  exercitiul_13.exercitiul_6();
end;
/

begin
  exercitiul_13.exercitiul_7();
end;
/

begin
  dms_output.put_line(exercitiul_13.exercitiul_8('Raluca'));
end;
/

begin
  exercitiul_13.exercitiul_9('Harry Potter');
end;
/
```

Script Output x | Query Result x

Task completed in 0.056 seconds

PL/SQL procedure successfully completed.

Denumirea cartii este: Harry Potter  
Editura la care se gaseste cartea este: Humanitas  
Numarul de cititori care au imprumutat cartea este: 3  
Numarul de imprumuturi este: 4  
Numarul de copii de carte imprumutate este: 4  
Id-urile copiilor cartilor care sunt disponibile: 3 2 5  
Raportul dintre numarul de copii disponibile care au fost imprumutate si numarul total de copii imprumutate ale cartii date este egal cu .75.

PL/SQL procedure successfully completed.

## 13. EXERCIȚIUL 14

Cerință: Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

Pentru fiecare cititor să se stabilească dacă este activ sau inactiv(este activ dacă a împrumutat cel puțin 2 cărți în ultimul an care este considerat a fi anul 2022), iar dacă este activ și a împrumutat mai mult de 3 cărți de când și-a făcut primul abonament, să i se înmâneze premiul pentru cititor devotat. Dacă este inactiv, dar are o vechime ca abonat la bibliotecă de mai mult de 2 ani, să i se creeze un nou abonament premium pe perioadă de o lună, având ca dată de început ziua curentă. Să se înmâneze și marele premiu celui mai tânăr cititor. Să se afișeze și cărțile împrumutate de fiecare cititor.

- procedure populare\_matrice - procedură prin care populez cele 2 matrice
  - lista\_cititori o matrice în care am pentru fiecare cititor nume, prenume, numărul de cărți împrumutate, un text dacă este cititor devotat;

- lista\_carti\_imprumutate o matrice în care pentru fiecare cititor am toate cărțile împrumutate de acesta;

- procedure este\_cel\_mai\_tanar\_cititor – procedură care verifică dacă un cititor dat este cel mai tânăr și memorează asta într-un parametru de ieșire;
- procedure cel\_mai\_tanar\_cititor – procedură care pune în record datele celui mai tânăr cititor;
- function cititor\_activ – funcție care returnează ,activ' dacă un cititor a împrumutat cel puțin 2 cărți în anul 2022 și ,inactiv' în caz contrar;
- function vechime – funcție care returnează ,true' dacă un cititor dat este abonat de cel puțin 2 ani la bibliotecă(dacă a avut primul abonament cu cel puțin 2 ani în urmă) , iar ,false' în caz contrar;
- procedure cititor\_devotat – procedură prin care se stabilește dacă un cititor dat este devotat sau nu; se consideră devotat dacă a împrumutat cel puțin 3 cărți în total;
- procedure stimulare\_cititor – procedură prin care stimulăm un cititor dat să citească prin crearea unui abonament premium pe o perioadă de o lună, având ca dată de început ziua curentă;
- procedure afisare – procedură unde afișăm toate datele memorate.

create or replace package exercitiul\_14 as

```
procedure populare_matrice;
procedure este_cel_mai_tanar_cititor(cod_cititor in cititori.id_cititor%type,
                                     tanar out boolean);
procedure cel_mai_tanar_cititor(cod_cititor in cititori.id_cititor%type);
procedure cititor_devotat(cod_cititor cititori.id_cititor%type);
function cititor_activ(cod_cititor cititori.id_cititor%type) return varchar2;
function vechime(cod_cititor cititori.id_cititor%type) return boolean;
procedure stimulare_cititor(cod_cititor cititori.id_cititor%type);
procedure afisare;
end exercitiul_14;
/
```

create or replace package body exercitiul\_14 as

```
type vector is varray(200) of varchar2(250);
type matrice is varray(200) of vector;
lista_carti_imprumutate matrice := matrice();
lista_cititori matrice := matrice();
```

```
type cititor_record is record(
    nume cititori.nume%type,
    prenume cititori.prenume%type,
```

```

        data_nastere cititori.data_nastere%type);
cititor_tanar cititor_record;
este_cititor_tanar boolean;

type tablou_imbricat is table of number;
coduri_cititori tablou_imbricat := tablou_imbricat();

procedure populare_matrice
is
    numar_cititori number;
begin
    select count(id_cititor)
    into numar_cititori
    from cititori;

    coduri_cititori.extend(numar_cititori);
    lista_carti_imprumutate.extend(numar_cititori);
    lista_cititori.extend(numar_cititori);

    select id_cititor
    bulk collect into coduri_cititori
    from cititori;
    for i in coduri_cititori.first..coduri_cititori.last loop
        lista_carti_imprumutate(i) := vector();
        lista_carti_imprumutate(i).extend();
        lista_cititori(i) := vector();
        lista_cititori(i).extend(6);

        select prenume, nume
        into lista_cititori(i)(1), lista_cititori(i)(2)
        from cititori
        where id_cititor = coduri_cititori(i);

        select to_char(count(id_carte_imprumutata))
        into lista_cititori(i)(3)
        from carti_imprumutate
        where id_cititor = coduri_cititori(i);

        if lista_cititori(i)(3) = '0' then
            lista_carti_imprumutate(i)(1) := 'Nu a imprumutat nicio carte momentan.';
        else

```



```

        select distinct(denumire)
        bulk collect into lista_carti_imprumutate(i)
        from carti_imprumutate, carti
        where carti_imprumutate.id_carte = carti.id_carte and id_cititor = coduri_cititori(i);
    end if;
end loop;
end populare_matrice;

```

```

procedure este_cel_mai_tanar_cititor(cod_cititor in cititori.id_cititor%type,
                                     tanar out boolean)

```

```

is

```

```

    data_minima date;
    data_cititor date;

```

```

begin

```

```

    select max(data_nastere)
    into data_minima
    from cititori;

```

```

    select data_nastere
    into data_cititor
    from cititori
    where cod_cititor = id_cititor;

```

```

    if data_cititor = data_minima then
        tanar := true;
    else
        tanar := false;
    end if;

```

```

exception

```

```

    when no_data_found then
        dbms_output.put_line('Nu exista niciun cititor cu acest id. ');
    when others then
        dbms_output.put_line('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);

```

```

end este_cel_mai_tanar_cititor;

```

```

procedure cel_mai_tanar_cititor(cod_cititor in cititori.id_cititor%type)

```

```

is

```

```

begin

```

```

    select nume, prenume, data_nastere

```

```

        into cititor_tanar
        from cititori
        where id_cititor = cod_cititor;
exception
    when no_data_found then
        dbms_output.put_line('Nu exista niciun cititor cu acest id.');
```

```

    when others then
        dbms_output.put_line('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);
end cel_mai_tanar_cititor;

function cititor_activ(cod_cititor cititori.id_cititor%type)
return varchar2
is
    activitate number;
    id_max_cititor number;
    exceptie exception;
begin
    select max(id_cititor)
    into id_max_cititor
    from cititori;
    if cod_cititor > id_max_cititor then
        raise exceptie;
    else
        select count(distinct(id_carte))
        into activitate
        from carti_imprumutate
        where id_cititor = cod_cititor and extract(year from data_imprumut) = '2022';

        if activitate >= 2 then
            return 'activ';
        else
            return 'inactiv';
        end if;
    end if;
exception
    when exceptie then
        return('Nu exista niciun cititor cu acest id. Id-ul trebuie sa fie mai mic decat ' ||
id_max_cititor || '.');
```

```

    when others then
        return('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);

```

```
end cititor_activ;
```

```
function vechime(cod_cititor cititori.id_cititor%type)
return boolean
is
    ani number;
begin
    select extract(year from sysdate) - extract(year from min(data_inceput))
    into ani
    from detine
    where id_cititor = cod_cititor;

    if ani >= 2 then
        return true;
    else
        return false;
    end if;
exception
    when no_data_found then
        dbms_output.put_line('Nu exista niciun cititor cu acest id.');
```

```
    when others then
        dbms_output.put_line('Codul erorii: ' || sqlcode || ' Mesajul erorii: ' || sqlerrm);
end vechime;

procedure cititor_devotat(cod_cititor cititori.id_cititor%type)
is
begin
    if to_number(lista_cititori(cod_cititor)(3)) >= 3 then
        lista_cititori(cod_cititor)(4) := 'PREMIU: CITITOR DEVOTAT';
    end if;
end cititor_devotat;
```

```
procedure stimulare_cititor(cod_cititor cititori.id_cititor%type)
is
    cod_abonament_premium number;
begin
    select id_abonament
    into cod_abonament_premium
```

```
from abonamente
where lower(tip) = 'premium';
```

```
insert into detine
values(cod_cititor, cod_abonament_premium, sysdate, 1);
end stimulare_cititor;
```

```
procedure afisare
```

```
is
```

```
begin
```

```
    populare_matrice();
```

```
for i in coduri_cititori.first..coduri_cititori.last loop
```

```
    este_cititor_tanar := false;
```

```
    este_cel_mai_tanar_cititor(coduri_cititori(i), este_cititor_tanar);
```

```
    if este_cititor_tanar = true then
```

```
        cel_mai_tanar_cititor(coduri_cititori(i));
```

```
    end if;
```

```
    if cititor_activ(coduri_cititori(i)) = 'activ' then
```

```
        lista_cititori(coduri_cititori(i))(2) := lista_cititori(coduri_cititori(i))(2) || ' activ';
```

```
        cititor_devotat(coduri_cititori(i));
```

```
    elsif cititor_activ(coduri_cititori(i)) = 'inactiv' and vechime(coduri_cititori(i)) = true then
```

```
        lista_cititori(coduri_cititori(i))(2) := lista_cititori(coduri_cititori(i))(2) || ' inactiv';
```

```
        stimulare_cititor(coduri_cititori(i));
```

```
    elsif cititor_activ(coduri_cititori(i)) = 'inactiv' then
```

```
        lista_cititori(coduri_cititori(i))(2) := lista_cititori(coduri_cititori(i))(2) || ' inactiv';
```

```
    end if;
```

```
end loop;
```

```
for i in coduri_cititori.first..coduri_cititori.last loop
```

```
    dbms_output.put_line('Cititorul ' || lista_cititori(coduri_cititori(i))(1) || ' ' ||
```

```
lista_cititori(coduri_cititori(i))(2));
```

```
    dbms_output.put_line(lista_cititori(coduri_cititori(i))(4));
```

```
    dbms_output.put_line('Carti imprumutate: ');
```

```
    for j in
```

```
lista_carti_imprumutate(coduri_cititori(i)).first..lista_carti_imprumutate(coduri_cititori(i)).last
loop
```

```
    dbms_output.put_line(lista_carti_imprumutate(coduri_cititori(i))(j));
```

```

        end loop;
        dbms_output.new_line;
        dbms_output.put_line('-----');
    end loop;
    dbms_output.new_line;
    dbms_output.new_line;
    dbms_output.put_line('MARELE PREMIU SE ACORDA: ');
    dbms_output.put_line('Citorului ' || cititor_tanar.prenume || ' ' || cititor_tanar.num);
    dbms_output.put_line('PREMIU: CEL MAI TANAR CITITOR');
end afisare;
end exercitiul_14;
/

```

```

-- Package Source Code
create or replace package exercitiul_14 as
    procedure populare_matrice;
    procedure este_cel_mai_tanar_cititor(cod_cititor in cititori.id_cititor%type,
                                         tanar out boolean);
    procedure cel_mai_tanar_cititor(cod_cititor in cititori.id_cititor%type);
    procedure cititor_devotat(cod_cititor cititori.id_cititor%type);
    function cititor_activ(cod_cititor cititori.id_cititor%type) return varchar2;
    function vechime(cod_cititor cititori.id_cititor%type) return boolean;
    procedure stimulare_cititor(cod_cititor cititori.id_cititor%type);
    procedure afisare;
end exercitiul_14;
/

-- Package Body Source Code
create or replace package body exercitiul_14 as
    type vector is varray(200) of varchar2(250);
    type matrice is varray(200) of vector;
    lista_carti_imprumutate matrice := matrice();
    lista_cititori matrice := matrice();

    type cititor_record is record(nume cititori.nume%type,
                                   prenume cititori.prenume%type,
                                   data_nastere cititori.data_nastere%type);
    cititor_tanar cititor_record;
    este_cititor_tanar boolean;

    type tablou_imbricat is table of number;
    coduri_cititori tablou_imbricat := tablou_imbricat();

    procedure populare_matrice

```

Script Output x Query Result x

Task completed in 0.152 seconds

```

Package EXERCITIUL_14 compiled

Package Body EXERCITIUL_14 compiled

```

```

begin
    exercitiul_14.afisare();
end;
/

```

Cititorul Raluca Rogoza activ  
PREMIU: CITITOR DEVOTAT  
Carti imprumutate:  
Enigma Otiliei  
Hotul  
Harry Potter

-----  
Cititorul Elena Stoica activ

Carti imprumutate:  
Baltagul  
Harry Potter

-----  
Cititorul Georgiana Cojoc activ

Carti imprumutate:  
Ion  
Hotul

-----  
Cititorul Carina Obreja inactiv

Carti imprumutate:  
Baltagul  
Hotul

-----  
Cititorul Iulia Talpalariu inactiv

Carti imprumutate:  
Hotul  
Harry Potter

-----  
Cititorul Mara Bejan inactiv

Carti imprumutate:  
Nu a imprumutat nicio carte momentan.

-----  
MARELE PREMIU SE ACORDA:  
Cititorului Elena Stoica  
PREMIU: CEL MAI TANAR CITITOR

PL/SQL procedure successfully completed.

Tabela DETINE înainte:

	ID_CITITOR	ID_ABONAMENT	DATA_INCEPUT	NUMAR_LUNI
1	2	2	02-MAY-22	2
2	1	1	08-JAN-22	1
3	3	6	12-MAR-22	1
4	5	4	17-FEB-20	4
5	2	6	02-JAN-21	2
6	1	5	20-MAR-21	1
7	4	6	22-SEP-21	3
8	5	6	01-MAY-21	2
9	3	1	15-MAY-22	2
10	4	3	02-APR-22	3
11	1	1	12-DEC-22	1
12	5	4	06-NOV-22	2
13	6	1	08-JAN-23	2

Tabela DETINE după:

	ID_CITITOR	ID_ABONAMENT	DATA_INCEPUT	NUMAR_LUNI
1	2	2	02-MAY-22	2
2	1	1	08-JAN-22	1
3	3	6	12-MAR-22	1
4	5	4	17-FEB-20	4
5	2	6	02-JAN-21	2
6	1	5	20-MAR-21	1
7	4	6	22-SEP-21	3
8	5	6	01-MAY-21	2
9	3	1	15-MAY-22	2
10	4	3	02-APR-22	3
11	1	1	12-DEC-22	1
12	5	4	06-NOV-22	2
13	6	1	08-JAN-23	2
14	4	6	13-JAN-23	1
15	5	6	13-JAN-23	1

```

declare
    tanar boolean;
begin
    exercitiul_14.este_cel_mai_tanar_cititor(2, tanar);
    if tanar = true then
        dbms_output.put_line('E cel mai tanar cititor. ');
    else

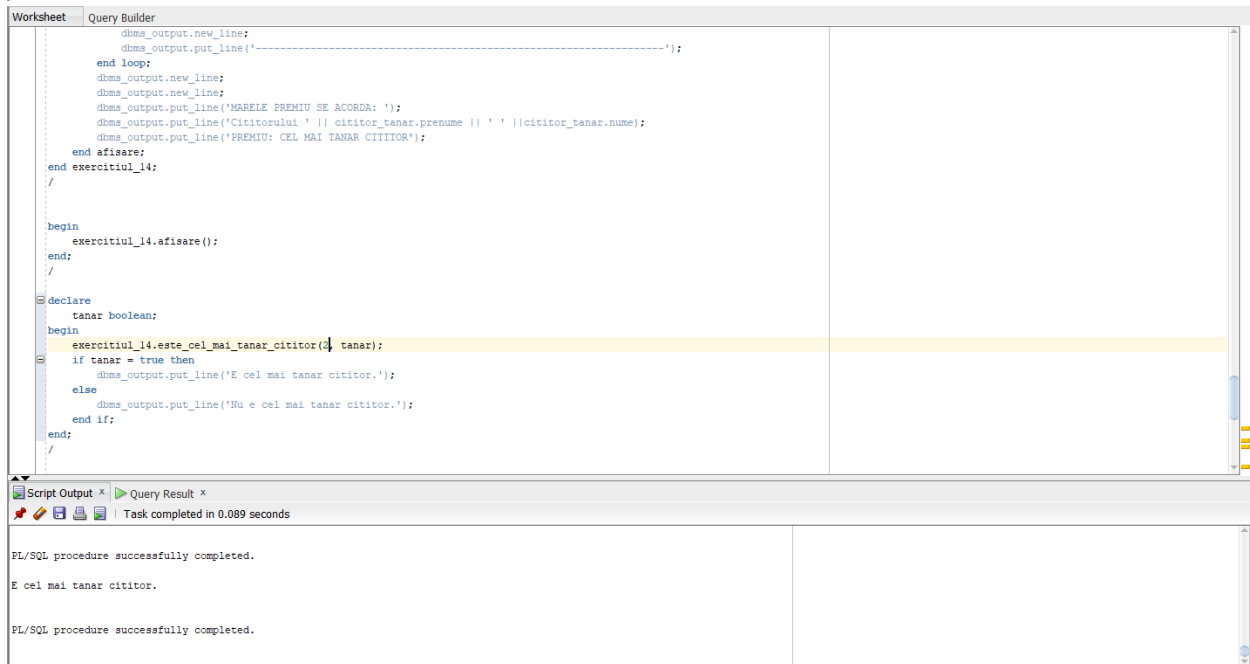
```

```

        dbms_output.put_line('Nu e cel mai tanar cititor.');
```

```

    end if;
end;
/
```



```

declare
    tanar boolean;
begin
    exercitiul_14.este_cel_mai_tanar_cititor(8, tanar);
    if tanar = true then
        dbms_output.put_line('E cel mai tanar cititor.');
```

```

    else
        dbms_output.put_line('Nu e cel mai tanar cititor.');
```

```

    end if;
end;
/
```



Worksheet Query Builder

```
dbms_output.new_line;
dbms_output.put_line('-----');
end loop;
dbms_output.new_line;
dbms_output.new_line;
dbms_output.put_line('HARELE PREMIU SE ACORDA: ');
dbms_output.put_line('Cititorului ' || cititor_tanar.prenume || ' ' || cititor_tanar.nume);
dbms_output.put_line('PREMIU: CEL MAI TANAR CITITOR');
end afisare;
end exercitiul_14;
/

begin
exercitiul_14.afisare();
end;
/

declare
tanar boolean;
begin
exercitiul_14.este_cel_mai_tanar_cititor(0, tanar);
if tanar = true then
dbms_output.put_line('E cel mai tanar cititor.');
```

Script Output x Query Result x

Task completed in 0.128 seconds

PL/SQL procedure successfully completed.

Nu exista niciun cititor cu acest id.  
Nu e cel mai tanar cititor.

PL/SQL procedure successfully completed.

```
begin
  dbms_output.put_line(exercitiul_14.cititor_activ(1));
end;
/
```

```
begin
  dbms_output.put_line(exercitiul_14.cititor_activ(1));
end;
/
```

Script Output x Query Result x

Task completed in 0.136 seconds

PL/SQL procedure successfully completed.

activ

PL/SQL procedure successfully completed.