

Proiectare unui sistem de gestiune
a bazelor de date
-gestionarea unui teatru-

CUPRINS

INTRODUCERE	3
PREZENTAREA MODELULUI DE BAZE DE DATE.....	3
DIAGRAMĂ E/R	5
DIAGRAMĂ CONCEPTUALĂ	6
CREAREA TABELELOR.....	7
INSERAREA DATELOR	23
SUBPROGRAM STOCAT CU 2 TIPURI DE COLECȚII.....	42
SUBPROGRAM STOCAT CU UN TIP DE CURSOR	47
SUBPROGRAM STOCAT DE TIP FUNCȚIE	50
SUBPROGRAM STOCAT DE TIP PROCEDURA	54
TRIGGER LMD LA NIVEL DE COMANDĂ	59
TRIGGER LMD LA NIVEL DE LINIE	61
TRIGGER DE TIP LDD.....	64
CREAREA UNUI PACHET DE BAZĂ	66
CREAREA UNUI PACHET PENTRU UN FLUX DE ACȚIUNI	74

INTRODUCERE:

Lucrarea de față propune un model de sistem de gestiune a bazelor de date. Se va folosi o bază de date a unui teatru pentru a exemplifica cum se pot manipula datele stocate prin intermediul SGBD-ului creat. Exemplele folosite în această lucrare au și utilitate practică, fiind menționate să rezolve diverse operații frecvent întâlnite de utilizatorii bazei de date și care ajută la buna organizare a teatrului.

PREZENTAREA MODELULUI DE BAZE DE DATE:

Teatrul dispune de mai multe săli și de mai mulți angajați. Fiecare sală de teatru este special amenajată pentru a asigura buna desfășurare a actorilor și confortul spectatorilor. Acestea sunt dotate cu aparatură pentru efecte speciale și sisteme audio, precum și spațiu pentru depozitarea recuzitei aduse de trupele de teatru și a costumelor actorilor. Angajații se împart în: casieri (se ocupă cu vânzarea de bilete), îngrijitori (asigură păstrarea curățeniei în săli) și tehnicieni (operează aparatura în timpul spectacolelor). Fiecare angajat are câte un manager în funcție de tipul jobului său, iar cei trei manageri au la rândul lor un manager – administratorul teatrului. Fiecare piesă de teatru este jucată într-o anumită sală de o trupă de actori. Actorii, deși au un anumit salariu pe perioada de colaborare cu teatrul, nu sunt angajați ai teatrului deoarece s-a considerat că trupele de teatru au un anumit grad de autonomie și se schimbă periodic. Persoanele care merg des la teatru sunt invitate să completeze un chestionar online sau la casierie pentru a deveni spectatori înregistrați, putând beneficia de o reducere la anumite spectacole, cât și de posibilitatea de a face o rezervare online sau telefonică în avans.

Utilitatea modelului de baze de date propus provine din capacitatea acestuia de a ține evidența programului teatrului, a ierarhiei angajaților și de a face un inventar al recuzitei și a aparaturii din sălile de teatru. Mai mult, baza de date va reține și evidența билетelor vândute la fiecare spectacol, cât și anumite date despre spectatori, iar o analiză a acestor informații poate determina gradul de popularitate al anumitor piese de teatru, aspect deosebit de important pentru managerii teatrului care doresc să-și maximizeze profitul.

Restricții de funcționare:

- Un spectacol poate fi organizat într-o singură sală la un anumit interval orar.
- Același spectacol poate fi jucat de două trupe de teatru diferite.
- Fiecare angajat are un manager, mai puțin administratorul teatrului.
- Un tehnician poate folosi aparatura din mai multe săli de teatru.

- În timpul colaborării cu un grup de actori, angajații teatrului sunt responsabili de păstrarea recuzitei și a costumelor aduse de actori la teatru.
- Numărul de săli curățate de un anumit îngrijitor este variabil.
- Un îngrijitor nu este repartizat să curețe de două ori aceeași sală într-o zi.
- Fiecare tehnician are datoria de a raporta starea în care se află dispozitivele după fiecare utilizare.
- Se consideră că recuzita și costumele pot fi mutate frecvent dintr-o sală în alta.
- Fiecare actor dintr-o trupă de teatru parteneră este implicat în cel puțin un spectacol.
- Un actor poate face parte dintr-o trupă de teatru sau niciuna, cazul în care este un actor independent.
- La teatru se acceptă colaborările dintre mai multe trupe de actori pentru reprezentarea unei piese.
- Un spectator este înregistrat în baza de date doar dacă a completat formularul de înregistrare.
- Nu pot exista mai multe bilete la o anumită piesă decât locuri din sala în care aceasta este jucată.

Entități:

Modelul de date prezentat cuprinde entitățile: SALĂ, ANGAJAT, PIESĂ, TRUPĂ, ACTOR, COSTUM, RECUZITĂ, APARATURĂ, BILET, SPECTATOR, REPREZENTAȚIE și subentitățile: MANAGER, CASIER, ÎNGRIJITOR, TEHNICIAN.

Entitățile COSTUM și RECUZITĂ sunt dependente, iar restul entităților sunt independente.

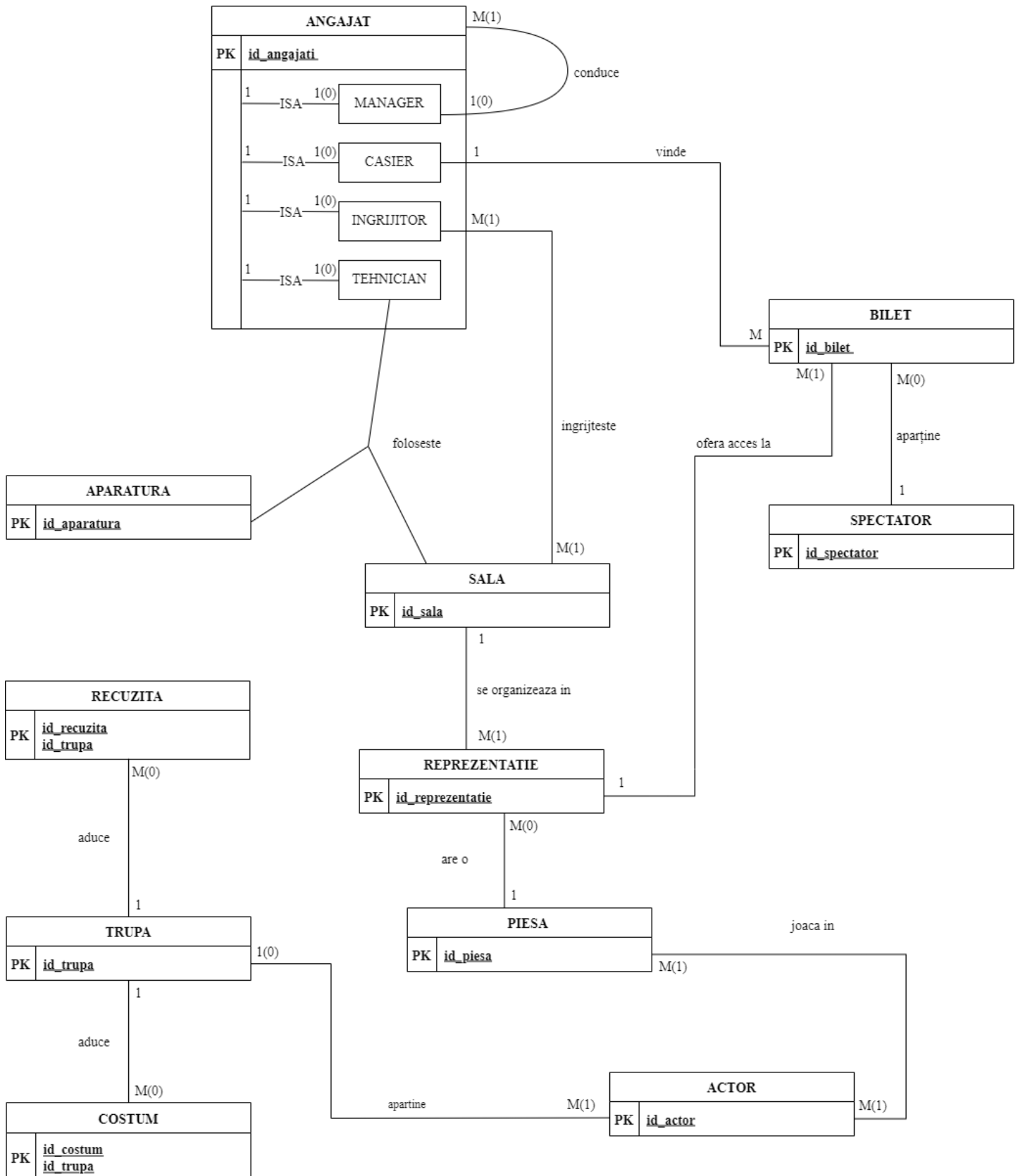
Tabele asociative:

FOLOSEȘTE: modelează relația de tip 3 dintre TEHNICIAN, SALĂ și APARATURĂ;

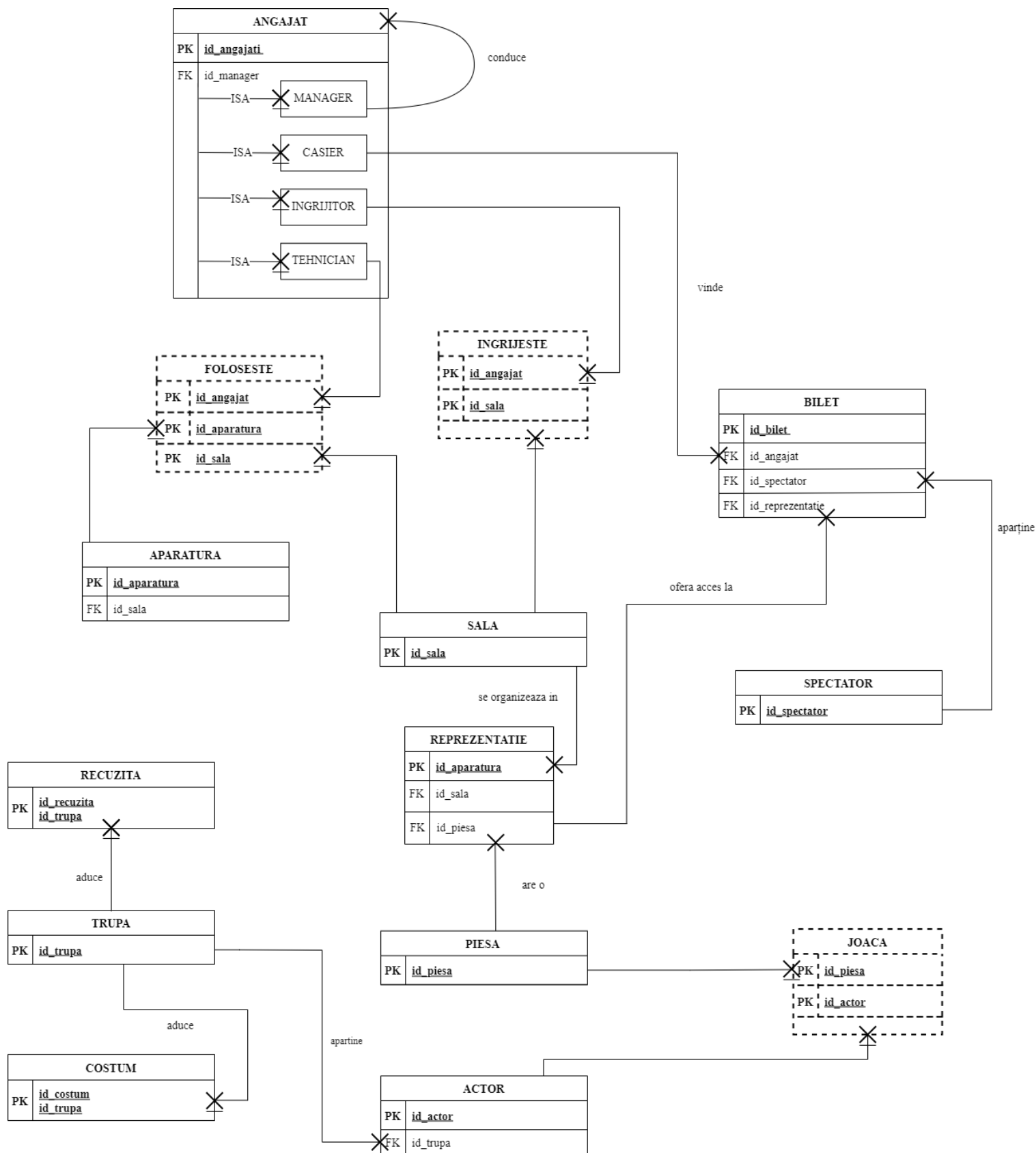
ÎNGRIJEȘTE: modelează relația de many to many dintre ÎNGRIJITOR și SALĂ;

JOACĂ: modelează relația de many to many dintre ACTOR și PIESĂ.

DIAGRAMĂ E/R:



DIAGRAMĂ CONCEPTUALĂ:



CREAREA TABELELOR:

a) Tabelul SALĂ:

```
CREATE TABLE SALA (  
  
    id_sala NUMBER(2, 0) PRIMARY KEY,  
  
    nume VARCHAR(40) UNIQUE,  
  
    capacitate NUMBER(4, 0) NOT NULL,  
  
    lungime_scena NUMBER(3, 1) NOT NULL,  
  
    latime_scena NUMBER(3, 1) NOT NULL,  
  
    data_verificare DATE );
```

The screenshot shows a database query builder interface. The top section, labeled 'Query Builder', contains the SQL code for creating the SALA table and a query to select all data from it. The bottom section, labeled 'Query Result', displays the results of the SELECT query as a table with 5 rows and 6 columns.

```
CREATE TABLE SALA (  
    id_sala NUMBER(2, 0) PRIMARY KEY,  
    nume VARCHAR(40) UNIQUE,  
    capacitate NUMBER(4, 0) NOT NULL,  
    lungime_scena NUMBER(3, 1) NOT NULL,  
    latime_scena NUMBER(3, 1) NOT NULL,  
    data_verificare DATE  
);  
  
DESCRIBE SALA;  
  
SELECT * FROM SALA;
```

	ID_SALA	NUME	CAPACITATE	LUNGIME_SCENA	LATIME_SCENA	DATA_VERIFICARE
1	15	Sala I. L. Caragiale	292	10	4	10-MAR-17
2	16	Sala Foarte Mica	100	6	3	12-APR-18
3	17	Sala Eugen Ionesco	147	9	5	20-JAN-15
4	18	Sala Rotunda	256	12	5	30-NOV-18
5	19	Sala Victoria	300	9	5	17-JUL-15

b) Tabelul ANGAJAT:

```
CREATE TABLE ANGAJAT (  
  
    id_angajat NUMBER(4, 0) PRIMARY KEY,  
  
    nume VARCHAR(20) NOT NULL,  
  
    prenume VARCHAR(30),
```

```

data_nasterii DATE,

sex CHAR(1),

numar_telefon VARCHAR(10) CONSTRAINT verifica_numar CHECK (numar_telefon LIKE '07%'),

data_angajare DATE,

salariu NUMBER(6,0) DEFAULT 2000

---de adaugat manager_id

);

ALTER TABLE ANGAJAT

ADD id_manager NUMBER(4, 0);

ALTER TABLE ANGAJAT

ADD CONSTRAINT FK_manager FOREIGN KEY(id_manager) REFERENCES MANAGER(id_angajat);

ALTER TABLE ANGAJAT MODIFY id_manager DEFAULT 1;

ALTER TABLE ANGAJAT MODIFY salariu DEFAULT 2000;

```

SQL Worksheet History

Worksheet Query Builder

```

CREATE TABLE ANGAJAT (
  id_angajat NUMBER(4, 0) PRIMARY KEY,
  nume VARCHAR(20) NOT NULL,
  prenume VARCHAR(30),
  data_nasterii DATE,
  sex CHAR(1),
  numar_telefon VARCHAR(10) CONSTRAINT verifica_numar CHECK (numar_telefon LIKE '07%'),
  data_angajare DATE,
  salariu NUMBER(6,0) DEFAULT 2000
  ---de adaugat job_id si manager_id
);

DESCRIBE ANGAJAT;

SELECT * FROM ANGAJAT;

```

Query Result x

All Rows Fetched: 19 in 0.043 seconds

ID_ANGAJAT	NUME	PRENUME	DATA_NASTERII	SEX	NUMAR_TELEFON	DATA_ANGAJARE	SALARIU	ID_MANAGER
1	2 Mill	Frances	12-SEP-97	M	0724985301	12-OCT-16	2200	7
2	1 Bryan	Loki	30-NOV-73	M	0733513210	(null)	55000	(null)
3	7 Fulton	Mimi	30-SEP-99	F	0738340140	05-SEP-10	12000	1
4	9 Denzel	Wyatt	11-SEP-87	M	0778985301	12-OCT-16	2250	7
5	10 Marshall	Conrad	30-NOV-98	M	0724956001	19-OCT-19	2600	7
6	11 Mill	Frances	12-SEP-97	F	0711287889	17-OCT-19	1950	7
7	12 Tait	Emilia	03-MAR-92	F	0729570001	16-APR-20	1800	7
8	13 Ross	Dru	20-MAY-81	M	0736420640	05-MAR-10	23600	1
9	14 Owen	Zoe	10-JAN-94	F	0736555640	02-MAR-15	18300	1
10	15 Franklin	Faiza	30-MAR-97	F	0768421998	05-SEP-10	7500	13
11	16 Cooke	Ariya	30-MAR-97	F	0738563211	05-SEP-10	7200	13
12	17 Cole	Seth	20-APR-85	M	0755316736	05-SEP-10	6500	13

c) Tabelul MANAGER

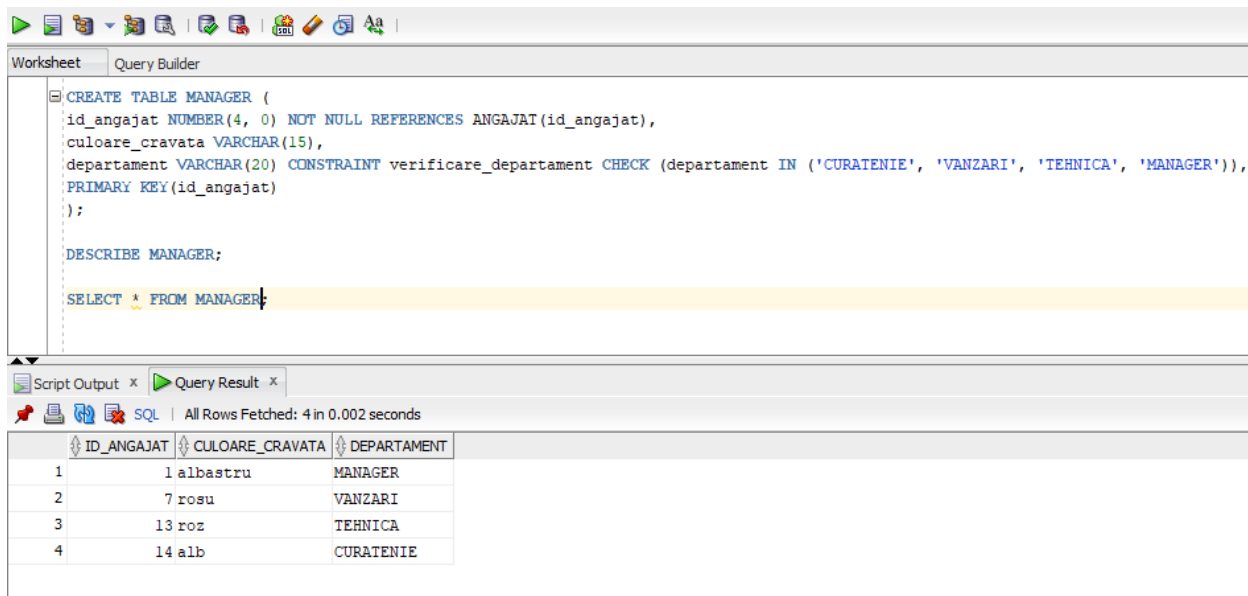
CREATE TABLE MANAGER (

id_angajat NUMBER(4, 0) NOT NULL REFERENCES ANGAJAT(id_angajat),

culoare_cravata VARCHAR(15),

departament VARCHAR(20) CONSTRAINT verificare_departament CHECK (departament IN ('CURATENIE', 'VANZARI', 'TEHNICA', 'MANAGER')),

PRIMARY KEY(id_angajat));



The screenshot shows a database query editor with a 'Query Builder' tab. The SQL script in the editor is as follows:

```
CREATE TABLE MANAGER (  
  id_angajat NUMBER(4, 0) NOT NULL REFERENCES ANGAJAT(id_angajat),  
  culoare_cravata VARCHAR(15),  
  departament VARCHAR(20) CONSTRAINT verificare_departament CHECK (departament IN ('CURATENIE', 'VANZARI', 'TEHNICA', 'MANAGER')),  
  PRIMARY KEY(id_angajat)  
);  
  
DESCRIBE MANAGER;  
  
SELECT * FROM MANAGER;
```

Below the script, the 'Query Result' tab is active, showing the results of the 'SELECT * FROM MANAGER;' query. The results are displayed in a table with 4 rows and 3 columns: ID_ANGAJAT, CULOARE_CRAVATA, and DEPARTAMENT.

ID_ANGAJAT	CULOARE_CRAVATA	DEPARTAMENT
1	1 albastru	MANAGER
2	7 rosu	VANZARI
3	13 roz	TEHNICA
4	14 alb	CURATENIE

d) Tabelul CASIER:

CREATE TABLE CASIER (

id_angajat NUMBER(4, 0) NOT NULL REFERENCES ANGAJAT(id_angajat),

nr_ghiseu NUMBER(2) NOT NULL,

comision NUMBER(3, 2) CONSTRAINT verifica_comision CHECK (comision <= 1.00),

PRIMARY KEY(id_angajat)

);

Worksheet Query Builder

```

CREATE TABLE CASIER (
  id_angajat NUMBER(4, 0) NOT NULL REFERENCES ANGAJAT(id_angajat),
  nr_ghiseu NUMBER(2) NOT NULL,
  comision NUMBER(3, 2) CONSTRAINT verifica_comision CHECK (comision <= 1.00),
  PRIMARY KEY(id_angajat)
);

DESCRIBE CASIER;

SELECT * FROM CASIER;

```

Script Output x Query Result x

SQL | All Rows Fetched: 5 in 0.005 seconds

ID_ANGAJAT	NR_GHISEU	COMISION
1	2	0.15
2	9	0.1
3	10	0
4	11	0.35
5	12	0.37

e) Tabelul INGRIJITOR:

```
CREATE TABLE INGRIJITOR (
```

```
  id_angajat NUMBER(4, 0) NOT NULL REFERENCES ANGAJAT(id_angajat),
```

```
  ore_suplimentare NUMBER(2, 0),
```

```
  marime VARCHAR(3) CONSTRAINT verifica_marime_ingrijitor CHECK (marime IN ('S', 'XS', 'XXS', 'M', 'L', 'XL', 'XXL')),
```

```
  PRIMARY KEY(id_angajat)
```

```
);
```

Worksheet Query Builder

```

CREATE TABLE INGRIJITOR (
  id_angajat NUMBER(4, 0) NOT NULL REFERENCES ANGAJAT(id_angajat),
  ore_suplimentare NUMBER(2, 0),
  marime VARCHAR(3) CONSTRAINT verifica_marime_ingrijitor CHECK (marime IN ('S', 'XS', 'XXS', 'M', 'L', 'XL', 'XXL')),
  PRIMARY KEY(id_angajat)
);

DESCRIBE INGRIJITOR;

SELECT * FROM INGRIJITOR;

```

Query Result x

SQL | All Rows Fetched: 5 in 0.002 seconds

	ID_ANGAJAT	ORE_SUPLIMENTARE	MARIME
1	20	10	M
2	21	0	S
3	22	22	L
4	23	40	M
5	24	5	XXL

f) Tabelul TEHNICIAN:

CREATE TABLE TEHNICIAN (

id_angajat NUMBER(4, 0) NOT NULL REFERENCES ANGAJAT(id_angajat),

specializare VARCHAR(15),

dispozitiv_preferat VARCHAR(30),

PRIMARY KEY(id_angajat)

);

Worksheet Query Builder

```

CREATE TABLE TEHNICIAN (
  id_angajat NUMBER(4, 0) NOT NULL REFERENCES ANGAJAT(id_angajat),
  specializare VARCHAR(15),
  dispozitiv_preferat VARCHAR(30),
  PRIMARY KEY(id_angajat)
);

DESCRIBE TEHNICIAN;

SELECT * FROM TEHNICIAN;

```

Query Result x

SQL | All Rows Fetched: 5 in 0.009 seconds

	ID_ANGAJAT	SPECIALIZARE	DISPOZITIV_PREFERAT
1	15	inginer	microfon
2	16	inginer	laser
3	17	inginer	aparat de facut baloane
4	18	programator	laser
5	19	fizician	proiector

g) Tabelul PIESĂ:

```
CREATE TABLE PIESA (
```

```
id_piesa NUMBER(5, 0) PRIMARY KEY,
```

```
titlu VARCHAR(50) NOT NULL UNIQUE,
```

```
gen VARCHAR(15),
```

```
durata NUMBER(4, 0) NOT NULL,
```

```
zi VARCHAR(10) NOT NULL CONSTRAINT verifica_zi CHECK (zi IN ('Luni', 'Marti', 'Miercuri', 'Joi', 'Vineri', 'Sambata', 'Duminica')),
```

```
ora VARCHAR(6) NOT NULL,
```

```
pret NUMBER(4, 0) NOT NULL
```

```
);
```

```
ALTER TABLE PIESA DROP COLUMN zi;
```

```
ALTER TABLE PIESA DROP COLUMN ora;
```

```
ALTER TABLE PIESA ADD autor VARCHAR(60);
```

```
ALTER TABLE PIESA ADD pauza NUMBER(1, 0) CONSTRAINT verifica_pauza CHECK (pauza = 1 or pauza = 0);
```

The screenshot shows a database management tool interface. The top section displays a list of SQL queries. The bottom section shows the results of the last query, which is a SELECT statement. The results are displayed in a table with 7 columns: ID_PIESA, TITLU, GEN, DURATA, PRET, PAUZA, and AUTOR. The table contains 7 rows of data.

```
CREATE TABLE PIESA (  
id_piesa NUMBER(5, 0) PRIMARY KEY,  
titlu VARCHAR(50) NOT NULL UNIQUE,  
gen VARCHAR(15),  
durata NUMBER(4, 0) NOT NULL,  
zi VARCHAR(10) NOT NULL CONSTRAINT verifica_zi CHECK (zi IN ('Luni', 'Marti', 'Miercuri', 'Joi', 'Vineri', 'Sambata', 'Duminica')),  
ora VARCHAR(6) NOT NULL,  
pret NUMBER(4, 0) NOT NULL  
);  
  
ALTER TABLE PIESA DROP COLUMN zi;  
ALTER TABLE PIESA DROP COLUMN ora;  
ALTER TABLE PIESA ADD autor VARCHAR(60);  
ALTER TABLE PIESA ADD pauza NUMBER(1, 0) CONSTRAINT verifica_pauza CHECK (pauza = 1 or pauza = 0);  
  
DESCRIBE PIESA;  
SELECT * FROM PIESA;  
COMMIT;
```

ID_PIESA	TITLU	GEN	DURATA	PRET	PAUZA	AUTOR
1	1 Iona	tragedie	70	25	0	Marin Sorescu
2	20 noapte furtunoasa	comedie	86	50	1	Ion Luca Caragiale
3	30 scrisoare pierduta	comedie	133	83	0	Ion Luca Caragiale
4	4 Hamlet	tragedie	151	47	1	William Shakespeare
5	5 Adio, domnule Haffmann	drama	105	44	1	Jean-Philippe Daguerre
6	6 Scufita Rosie	comedie	70	15	0	(null)
7	7 ActOrchestra	muzical	94	35	0	(null)

h) Tabelul TRUPĂ:

```
CREATE TABLE TRUPA (  
id_trupa NUMBER(3, 0) PRIMARY KEY,  
nume VARCHAR(50) NOT NULL,  
oras VARCHAR(15),  
numar_telefon VARCHAR(10) CONSTRAINT verifica_numar_trupa CHECK (numar_telefon LIKE '07%'),  
email VARCHAR(50) CONSTRAINT verifica_email CHECK (email LIKE '%@%'),  
data_colab DATE);
```

The screenshot shows a database management tool interface. The top section is the 'Query Builder' tab, displaying the SQL script for creating the TRUPA table. The script includes constraints for the phone number and email fields. Below the script, the 'Query Result' tab is active, showing the results of a DESCRIBE and SELECT query. The results are displayed in a table with 6 columns: ID_TRUPA, NUME, ORAS, NUMAR_TELEFON, EMAIL, and DATA_COLAB. There are 5 rows of data.

```
CREATE TABLE TRUPA (  
id_trupa NUMBER(3, 0) PRIMARY KEY,  
nume VARCHAR(50) NOT NULL,  
oras VARCHAR(15),  
numar_telefon VARCHAR(10) CONSTRAINT verifica_numar_trupa CHECK (numar_telefon LIKE '07%'),  
email VARCHAR(50) CONSTRAINT verifica_email CHECK (email LIKE '%@%'),  
data_colab DATE  
);  
  
DESCRIBE TRUPA;  
SELECT * FROM TRUPA;
```

ID_TRUPA	NUME	ORAS	NUMAR_TELEFON	EMAIL	DATA_COLAB
1	3 Gasca Zurli	Bucuresti	0733033522	gasca@zurli.com	12-SEP-16
2	4 Trupa Chaplin	Iasi	0740445400	chaplin05@gmail.com	02-OCT-17
3	5 Trupa TNB	Bucuresti	0733338899	t_n_b@gmail.com	11-SEP-20
4	6 Asociatia Teta	Craiova	0722121233	asctt33@yahoo.com	02-FEB-20
5	7 Trupa EXCELSIOR	Brasov	0744340202	exc_22@gmail.com	27-MAR-19

i) Tabelul ACTOR:

```
CREATE TABLE ACTOR (  
id_actor NUMBER(5, 0) PRIMARY KEY,  
id_trupa NUMBER(3, 0),
```

nume VARCHAR(20) NOT NULL,

prenume VARCHAR(30),

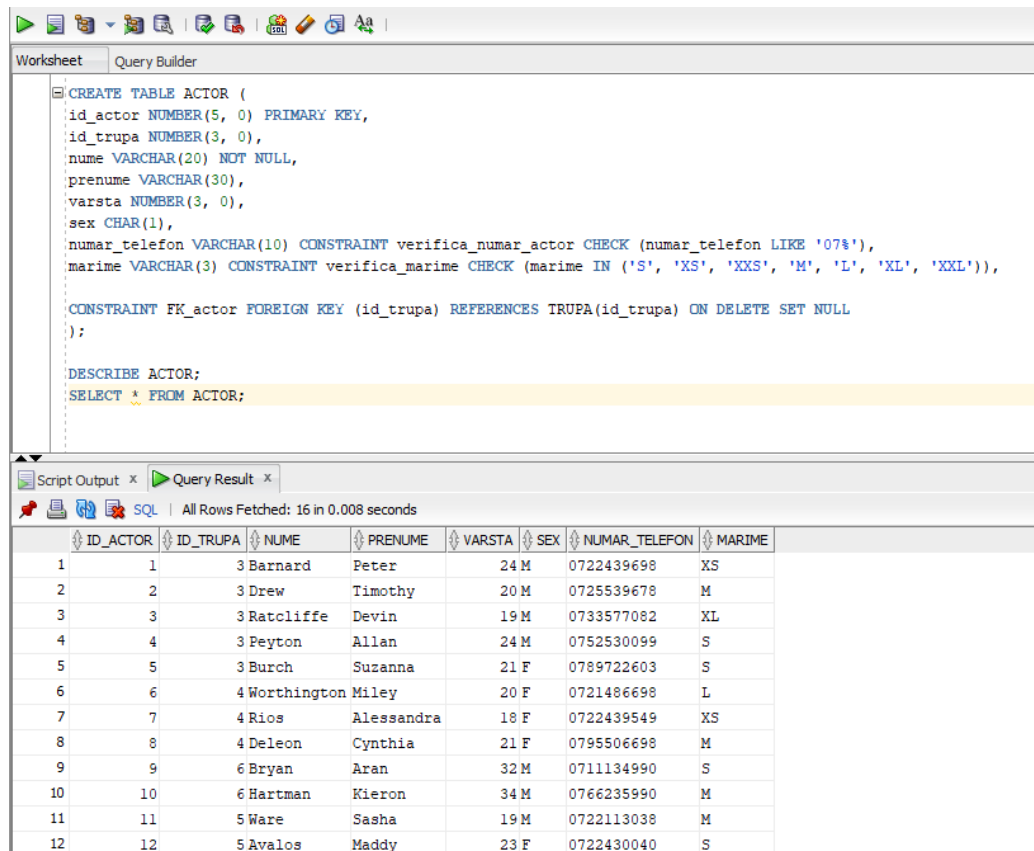
varsta NUMBER(3, 0),

sex CHAR(1),

numar_telefon VARCHAR(10) CONSTRAINT verifica_numar_actor CHECK (numar_telefon LIKE '07%'),

marime VARCHAR(3) CONSTRAINT verifica_marime CHECK (marime IN ('S', 'XS', 'XXS', 'M', 'L', 'XL', 'XXL')),

CONSTRAINT FK_actor FOREIGN KEY (id_trupa) REFERENCES TRUPA(id_trupa) ON DELETE SET NULL);



The screenshot shows a database query editor with a 'Query Builder' tab. The SQL code defines the 'ACTOR' table with columns: id_actor (NUMBER(5, 0) PRIMARY KEY), id_trupa (NUMBER(3, 0)), nume (VARCHAR(20) NOT NULL), prenume (VARCHAR(30)), varsta (NUMBER(3, 0)), sex (CHAR(1)), numar_telefon (VARCHAR(10) with a CHECK constraint), marime (VARCHAR(3) with a CHECK constraint), and a foreign key constraint FK_actor. Below the code, the 'Query Result' tab displays 16 rows of data.

ID_ACTOR	ID_TRUPA	NUME	PRENUME	VARSTA	SEX	NUMAR_TELEFON	MARIME
1	1	3 Barnard	Peter	24	M	0722439698	XS
2	2	3 Drew	Timothy	20	M	0725539678	M
3	3	3 Ratcliffe	Devin	19	M	0733577082	XL
4	4	3 Peyton	Allan	24	M	0752530099	S
5	5	3 Burch	Suzanna	21	F	0789722603	S
6	6	4 Worthington	Miley	20	F	0721486698	L
7	7	4 Rios	Alessandra	18	F	0722439549	XS
8	8	4 Deleon	Cynthia	21	F	0795506698	M
9	9	6 Bryan	Aran	32	M	0711134990	S
10	10	6 Hartman	Kieron	34	M	0766235990	M
11	11	5 Ware	Sasha	19	M	0722113038	M
12	12	5 Avalos	Maddy	23	F	0722430040	S

j) Tabelul COSTUM:

CREATE TABLE COSTUM(

id_costum NUMBER(4, 0),

id_trupa NUMBER(3, 0),

denumire VARCHAR(30) NOT NULL,

marime VARCHAR(3) NOT NULL CONSTRAINT verifica_marime_costum CHECK (marime IN ('S', 'XS', 'XXS', 'M', 'L', 'XL', 'XXL')),

material VARCHAR(15),

accesorii NUMBER(1, 0) CONSTRAINT verifica_accesorii CHECK (accesorii = 0 or accesorii = 1 or accesorii is NULL),

pret NUMBER(4, 0),

CONSTRAINT PK_costum PRIMARY KEY(id_costum, id_trupa),

CONSTRAINT FK_costum FOREIGN KEY (id_trupa) REFERENCES TRUPA(id_trupa) ON DELETE CASCADE
);

The screenshot shows a database query editor with a toolbar at the top. The main window is titled 'Query Builder' and contains the following SQL code:

```
CREATE TABLE COSTUM(  
  id_costum NUMBER(4, 0),  
  id_trupa NUMBER(3, 0),  
  denumire VARCHAR(30) NOT NULL,  
  marime VARCHAR(3) NOT NULL CONSTRAINT verifica_marime_costum CHECK (marime IN ('S', 'XS', 'XXS', 'M', 'L', 'XL', 'XXL')),  
  material VARCHAR(15),  
  accesorii NUMBER(1, 0) CONSTRAINT verifica_accesorii CHECK (accesorii = 0 or accesorii = 1 or accesorii is NULL),  
  pret NUMBER(4, 0),  
  CONSTRAINT PK_costum PRIMARY KEY(id_costum, id_trupa),  
  CONSTRAINT FK_costum FOREIGN KEY (id_trupa) REFERENCES TRUPA(id_trupa) ON DELETE CASCADE  
);  
  
DROP TABLE COSTUM;  
  
SELECT * FROM COSTUM;
```

Below the query editor, there is a 'Script Output' tab and a 'Query Result' tab. The 'Query Result' tab is active, showing a table with 7 columns: ID_COSTUM, ID_TRUPA, DENUMIRE, MARIME, MATERIAL, ACCESORII, and PRET. The table contains 6 rows of data.

ID_COSTUM	ID_TRUPA	DENUMIRE	MARIME	MATERIAL	ACCESORII	PRET
1	1	6 Zana cea buna	S	in	1	340
2	2	6 Vrajitoare	L	bumbac	0	420
3	3	7 Cersetor	XS	bumbac	0	430
4	4	7 Fantoma	M	bumbac	0	140
5	5	7 Pirat	S	in	1	220
6	6	6 Imparat	XL	piele	1	500

k) Tabelul RECUZITĂ:

CREATE TABLE RECUZITA(

id_recuzita NUMBER(4, 0),

id_trupa NUMBER(3, 0),

denumire VARCHAR(30) NOT NULL,

material VARCHAR(15),

lungime NUMBER(3, 0) NOT NULL,

latime NUMBER(3, 0) NOT NULL,

inaltime NUMBER(3, 0) NOT NULL,

pret NUMBER(4, 0),

CONSTRAINT PK_recuzita PRIMARY KEY(id_recuzita, id_trupa),

CONSTRAINT FK_recuzita FOREIGN KEY (id_trupa) REFERENCES TRUPA(id_trupa) ON DELETE CASCADE
);

The screenshot shows a SQL query editor with the following SQL code:

```
CREATE TABLE RECUZITA(  
  id_recuzita NUMBER(4, 0),  
  id_trupa NUMBER(3, 0),  
  denumire VARCHAR(30) NOT NULL,  
  material VARCHAR(15),  
  lungime NUMBER(3, 0) NOT NULL,  
  latime NUMBER(3, 0) NOT NULL,  
  inaltime NUMBER(3, 0) NOT NULL,  
  pret NUMBER(4, 0),  
  CONSTRAINT PK_recuzita PRIMARY KEY(id_recuzita, id_trupa),  
  CONSTRAINT FK_recuzita FOREIGN KEY (id_trupa) REFERENCES TRUPA(id_trupa) ON DELETE CASCADE  
);  
  
DESCRIBE RECUZITA;  
  
SELECT * FROM RECUZITA;
```

The query results are displayed in a table with 8 columns: ID_RECUZITA, ID_TRUPA, DENUMIRE, MATERIAL, LUNGIME, LATIME, INALTIME, and PRET. The results show 7 rows of data.

ID_RECUZITA	ID_TRUPA	DENUMIRE	MATERIAL	LUNGIME	LATIME	INALTIME	PRET
1	1	3 sabie	plastic	70	30	10	100
2	2	3 castel	carton	200	150	300	3000
3	3	4 pistol	metal	40	30	10	100
4	4	5 arc cu sageti	plastic	40	10	10	250
5	5	5 corabie	carton	250	100	310	3200
6	6	5 copac	carton	20	30	180	70
7	7	5 caine	plus	50	40	20	50

1) Tabelul APARATURĂ:

CREATE TABLE APARATURA (

id_aparatura NUMBER(4, 0) PRIMARY KEY,

id_sala NUMBER(2, 0),


```

denumire VARCHAR(30),

firma VARCHAR(15),

an_achizitie DATE,

pret NUMBER(4, 0) DEFAULT 0,

garantie NUMBER(1, 0) CONSTRAINT verifica_garantie CHECK (garantie = 0 or garantie = 1),

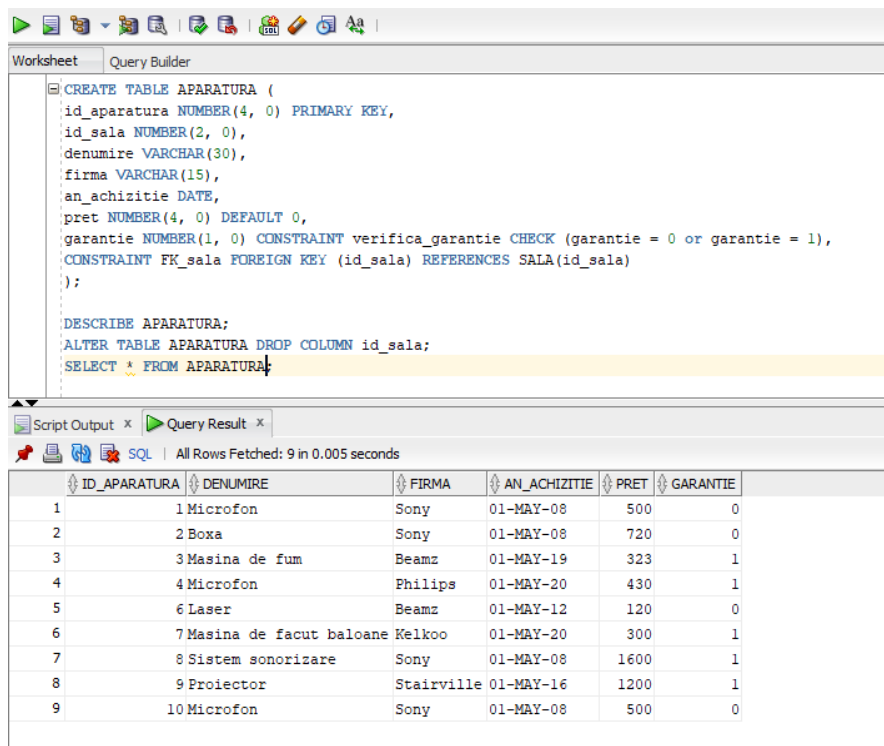
CONSTRAINT FK_sala FOREIGN KEY (id_sala) REFERENCES SALA(id_sala)

);

DESCRIBE APARATURA;

ALTER TABLE APARATURA DROP COLUMN id_sala;

```



The screenshot shows a database query editor with the following SQL script:

```

CREATE TABLE APARATURA (
  id_aparatura NUMBER(4, 0) PRIMARY KEY,
  id_sala NUMBER(2, 0),
  denumire VARCHAR(30),
  firma VARCHAR(15),
  an_achizitie DATE,
  pret NUMBER(4, 0) DEFAULT 0,
  garantie NUMBER(1, 0) CONSTRAINT verifica_garantie CHECK (garantie = 0 or garantie = 1),
  CONSTRAINT FK_sala FOREIGN KEY (id_sala) REFERENCES SALA(id_sala)
);

DESCRIBE APARATURA;
ALTER TABLE APARATURA DROP COLUMN id_sala;
SELECT * FROM APARATURA;

```

Below the script, the query result is displayed as a table with 9 rows and 6 columns:

ID_APARATURA	DENUMIRE	FIRMA	AN_ACHIZITIE	PRET	GARANTIE
1	1 Microfon	Sony	01-MAY-08	500	0
2	2 Boxa	Sony	01-MAY-08	720	0
3	3 Masina de fum	Beamz	01-MAY-19	323	1
4	4 Microfon	Philips	01-MAY-20	430	1
5	6 Laser	Beamz	01-MAY-12	120	0
6	7 Masina de facut baloane	Kelkoo	01-MAY-20	300	1
7	8 Sistem sonorizare	Sony	01-MAY-08	1600	1
8	9 Projector	Stairville	01-MAY-16	1200	1
9	10 Microfon	Sony	01-MAY-08	500	0

m) Tabelul BILET:

```

CREATE TABLE BILET (

id_bilet NUMBER(7, 0) PRIMARY KEY,

id_angajat NUMBER(5, 0),

id_reprezentatie NUMBER(10, 0),

```

id_spectator NUMBER(7, 0),

rand NUMBER(3) NOT NULL,

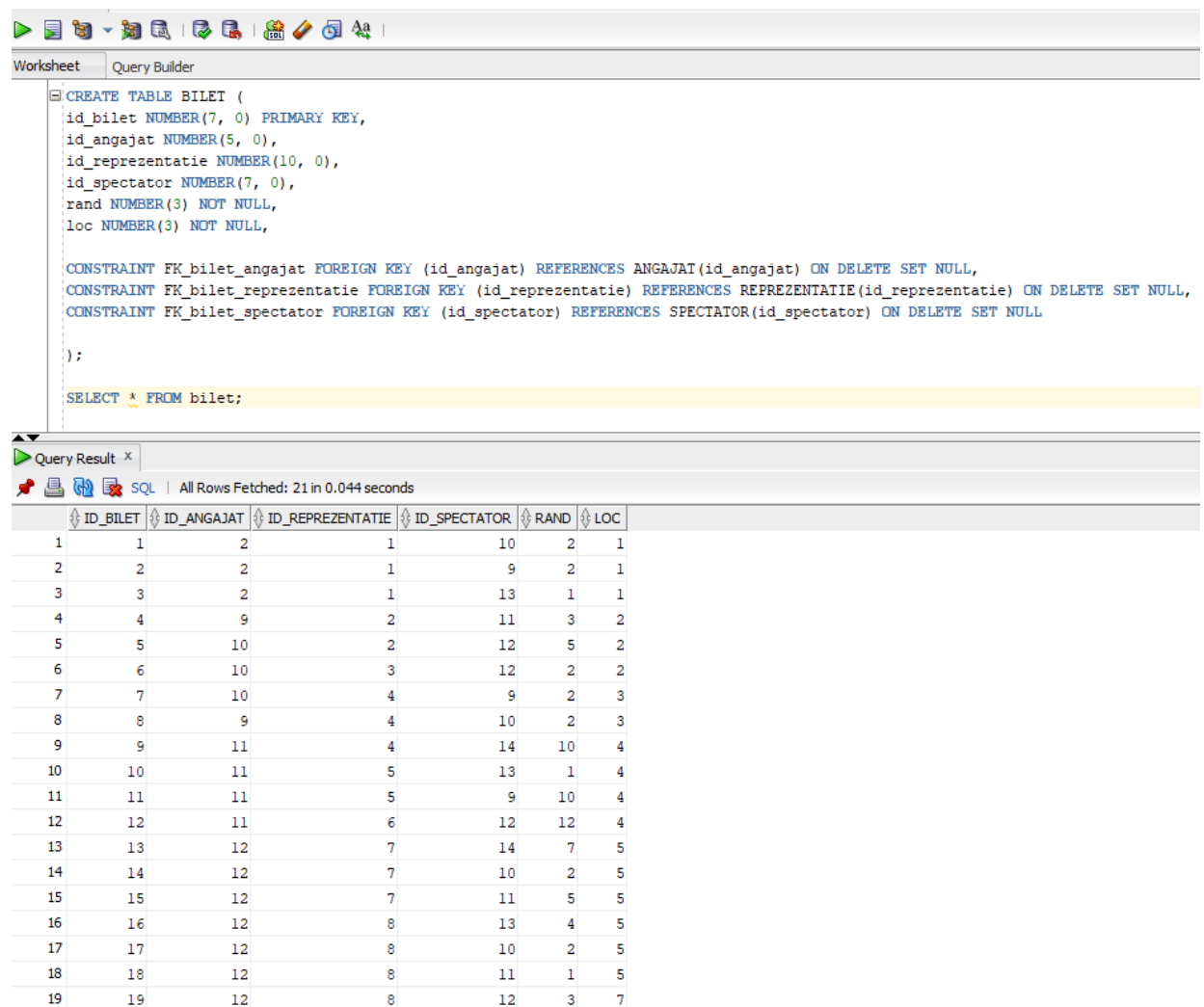
loc NUMBER(3) NOT NULL,

CONSTRAINT FK_bilet_angajat FOREIGN KEY (id_angajat) REFERENCES ANGAJAT(id_angajat) ON DELETE SET NULL,

CONSTRAINT FK_bilet_reprezentatie FOREIGN KEY (id_reprezentatie) REFERENCES REPREZENTATIE(id_reprezentatie) ON DELETE SET NULL,

CONSTRAINT FK_bilet_spectator FOREIGN KEY (id_spectator) REFERENCES SPECTATOR(id_spectator) ON DELETE SET NULL

);



```
CREATE TABLE BILET (  
  id_bilet NUMBER(7, 0) PRIMARY KEY,  
  id_angajat NUMBER(5, 0),  
  id_reprezentatie NUMBER(10, 0),  
  id_spectator NUMBER(7, 0),  
  rand NUMBER(3) NOT NULL,  
  loc NUMBER(3) NOT NULL,  
  
  CONSTRAINT FK_bilet_angajat FOREIGN KEY (id_angajat) REFERENCES ANGAJAT(id_angajat) ON DELETE SET NULL,  
  CONSTRAINT FK_bilet_reprezentatie FOREIGN KEY (id_reprezentatie) REFERENCES REPREZENTATIE(id_reprezentatie) ON DELETE SET NULL,  
  CONSTRAINT FK_bilet_spectator FOREIGN KEY (id_spectator) REFERENCES SPECTATOR(id_spectator) ON DELETE SET NULL  
);  
  
SELECT * FROM bilet;
```

ID_BILET	ID_ANGAJAT	ID_REPREZENTATIE	ID_SPECTATOR	RAND	LOC
1	1	2	1	10	2
2	2	2	1	9	2
3	3	2	1	13	1
4	4	9	2	11	3
5	5	10	2	12	5
6	6	10	3	12	2
7	7	10	4	9	2
8	8	9	4	10	2
9	9	11	4	14	10
10	10	11	5	13	1
11	11	11	5	9	10
12	12	11	6	12	12
13	13	12	7	14	7
14	14	12	7	10	2
15	15	12	7	11	5
16	16	12	8	13	4
17	17	12	8	10	2
18	18	12	8	11	1
19	19	12	8	12	3

n) Tabelul SPECTATOR:

CREATE TABLE SPECTATOR (

id_spectator NUMBER(7, 0) PRIMARY KEY,

nume VARCHAR(20) NOT NULL,

prenume VARCHAR(30),

varsta NUMBER(3, 0),

sex CHAR(1),

numar_telefon VARCHAR(10) CONSTRAINT verifica_numar_spectator CHECK (numar_telefon LIKE '07%'),

email VARCHAR(30) CONSTRAINT verifica_email_spectator CHECK (email LIKE '%@%'));

The screenshot shows a database query editor with a toolbar at the top. The main window is divided into two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, displaying the following SQL code:

```
CREATE TABLE SPECTATOR (  
  id_spectator NUMBER(7, 0) PRIMARY KEY,  
  nume VARCHAR(20) NOT NULL,  
  prenume VARCHAR(30),  
  varsta NUMBER(3, 0),  
  sex CHAR(1),  
  numar_telefon VARCHAR(10) CONSTRAINT verifica_numar_spectator CHECK (numar_telefon LIKE '07%'),  
  email VARCHAR(30) CONSTRAINT verifica_email_spectator CHECK (email LIKE '%@%')  
);  
  
DESCRIBE SPECTATOR;  
SELECT * FROM SPECTATOR;
```

Below the code editor, there is a 'Script Output' tab and a 'Query Result' tab. The 'Query Result' tab is active, showing the results of the 'SELECT * FROM SPECTATOR;' query. The results are displayed in a table with 7 columns: ID_SPECTATOR, NUME, PRENUME, VARSTA, SEX, NUMAR_TELEFON, and EMAIL. There are 6 rows of data.

ID_SPECTATOR	NUME	PRENUME	VARSTA	SEX	NUMAR_TELEFON	EMAIL
1	9 Herman	Jim	20	M	0748222989	jimmy.jim@yahoo.com
2	10 Hinton	Caitlin	18	F	0734341110	caittyhint@gmail.com
3	11 Bull	Jack	30	M	0778993212	redbull@gmail.com
4	12 McLaughlin	Gabriel	45	F	0703031156	gabbyl@gmail.com
5	13 Horner	Mia	22	F	0799458921	mama_mia@yahoo.com
6	14 Hill	Emilio	23	M	0744953329	em_hill@gmail.com

o) Tabelul REPREZENTAȚIE:

CREATE TABLE REPREZENTATIE(

id_reprezentatie NUMBER(10, 0) PRIMARY KEY,

id_piesa NUMBER(5, 0) NOT NULL REFERENCES PIESA(id_piesa),

id_sala NUMBER(2, 0) NOT NULL REFERENCES SALA(id_sala),

succes NUMBER(3, 2) NOT NULL CONSTRAINT verifica_rata_succes CHECK (succes <= 1.00),

data_org DATE,

ora VARCHAR(6)

);

The screenshot shows a database query builder interface. The top section is the 'Query Builder' tab, which contains the following SQL code:

```
CREATE TABLE REPREZENTATIE(  
  id_reprezentatie NUMBER(10, 0) PRIMARY KEY,  
  id_piesa NUMBER(5, 0) NOT NULL REFERENCES PIESA(id_piesa),  
  id_sala NUMBER(2, 0) NOT NULL REFERENCES SALA(id_sala),  
  succes NUMBER(3, 2) NOT NULL CONSTRAINT verifica_rata_succes CHECK (succes <= 1.00),  
  data_org DATE,  
  ora VARCHAR(6)  
);  
  
SELECT * FROM reprezentatie;
```

The bottom section is the 'Query Result' tab, which displays the results of the query. The results are shown in a table with 7 columns: ID_REPREZENTATIE, ID_PIESA, ID_SALA, SUCCES, DATA_ORG, and ORA. The table contains 17 rows of data.

ID_REPREZENTATIE	ID_PIESA	ID_SALA	SUCCES	DATA_ORG	ORA
1	1	1	15	0.8	25-MAR-20 19:00
2	2	1	16	0.67	27-MAR-20 18:15
3	3	2	17	0.8	25-MAR-20 18:30
4	4	2	18	0.92	25-APR-20 12:15
5	5	3	19	0.83	05-APR-20 12:30
6	6	3	15	0.75	20-APR-20 12:15
7	7	3	15	0.69	20-MAR-20 14:00
8	8	4	16	0.56	30-MAR-20 16:30
9	9	4	16	0.95	05-APR-20 20:00
10	10	5	17	0.4	05-APR-20 19:00
11	11	6	18	0.8	10-APR-20 20:00
12	12	6	19	0.2	10-APR-20 18:15
13	13	7	15	0.9	15-APR-20 19:00
14	14	7	15	0.78	16-APR-20 20:00
15	15	7	15	0.84	25-APR-20 19:00
16	16	1	15	0.3	24-MAY-21 15:00
17	17	2	15	0.4	24-MAY-21 19:00

p) Tabelul FOLOSEȘTE:

CREATE TABLE FOLOSESTE(
 id_angajat NUMBER(4, 0) NOT NULL REFERENCES TEHNICIAN(id_angajat),
 id_aparatura NUMBER(4, 0) NOT NULL REFERENCES APARATURA(id_aparatura),
 id_sala NUMBER(2, 0) NOT NULL REFERENCES SALA(id_sala),
 stare VARCHAR(15),
 PRIMARY KEY(id_angajat, id_sala, id_aparatura));

Worksheet Query Builder				
<pre> CREATE TABLE FOLOSESTE(id_angajat NUMBER(4, 0) NOT NULL REFERENCES TEHNICIAN(id_angajat), id_aparatura NUMBER(4, 0) NOT NULL REFERENCES APARATURA(id_aparatura), id_sala NUMBER(2, 0) NOT NULL REFERENCES SALA(id_sala), stare VARCHAR(15), PRIMARY KEY(id_angajat, id_sala, id_aparatura)); DESCRIBE FOLOSESTE; SELECT * FROM FOLOSESTE; </pre>				
Script Output x Query Result x				
SQL All Rows Fetched: 15 in 0.009 seconds				
ID_ANGAJAT	ID_APARATURA	ID_SALA	STARE	
1	15	1	15 buna	
2	15	2	16 buna	
3	15	2	17 buna	
4	16	3	18 buna	
5	16	3	19 foarte buna	
6	17	3	15 foarte buna	
7	18	4	16 buna	
8	18	6	18 buna	
9	19	7	19 mici defectiuni	
10	16	9	17 mici defectiuni	
11	18	6	17 buna	
12	19	8	15 buna	

q) Tabelul JOACĂ:

```
CREATE TABLE JOACA(
```

```
  id_piesa NUMBER(5, 0) NOT NULL REFERENCES PIESA(id_piesa),
```

```
  id_actor NUMBER(2, 0) NOT NULL REFERENCES ACTOR(id_actor),
```

```
  rol VARCHAR(20)
```

```
);
```

```
ALTER TABLE JOACA
```

```
MODIFY rol DEFAULT 'dublura';
```

```
ALTER TABLE JOACA
```

```
ADD CONSTRAINT PK_joaca PRIMARY KEY(id_actor, id_piesa);
```

Worksheet Query Builder

```

CREATE TABLE JOACA(
  id_piesa NUMBER(5, 0) NOT NULL REFERENCES PIESA(id_piesa),
  id_actor NUMBER(2, 0) NOT NULL REFERENCES ACTOR(id_actor),
  nota NUMBER(2, 0) NOT NULL CONSTRAINT verifica_nota CHECK (nota >= 0 and nota <= 10)
);

DESCRIBE JOACA;
COMMIT;

ALTER TABLE JOACA
ADD data DATE NOT NULL;

ALTER TABLE JOACA
ADD CONSTRAINT PK_joaca PRIMARY KEY(id_actor, id_piesa);

ALTER TABLE JOACA
MODIFY id_actor NUMBER(5, 0);
COMMIT;

SELECT * FROM JOACA;

```

Script Output x Query Result x

SQL | All Rows Fetched: 15 in 0.004 seconds

ID_PIESA	ID_ACTOR	NOTA
1	1	7
2	1	7
3	1	8
4	1	10
5	1	10
6	1	4
7	2	10
8	2	9
9	3	4
10	4	9

r) Tabelul ÎNGRIJEȘTE:

CREATE TABLE ÎNGRIJEȘTE(

id_angajat NUMBER(4, 0) NOT NULL REFERENCES ÎNGRIJITOR(id_angajat),

id_sala NUMBER(2, 0) NOT NULL REFERENCES SALA(id_sala),

durata NUMBER(3, 0),

PRIMARY KEY(id_angajat, id_sala)

);

The screenshot shows a database query tool interface. The top section is the 'Query Builder' tab, which contains the following SQL code:

```
CREATE TABLE INGRIJESTE(
  id_angajat NUMBER(4, 0) NOT NULL REFERENCES INGRIJITOR(id_angajat),
  id_sala NUMBER(2, 0) NOT NULL REFERENCES SALA(id_sala),
  durata NUMBER(3, 0),
  PRIMARY KEY(id_angajat, id_sala)
);

DESCRIBE INGRIJESTE;

SELECT * FROM INGRIJESTE;
```

The bottom section is the 'Query Result' tab, which displays the results of the 'SELECT * FROM INGRIJESTE;' query. The results are shown in a table with 4 columns: ID_ANGAJAT, ID_SALA, DURATA, and an unnamed column. The table contains 11 rows of data.

ID_ANGAJAT	ID_SALA	DURATA	
1	20	16	90
2	20	17	90
3	20	18	30
4	20	19	15
5	21	15	100
6	21	16	25
7	22	17	35
8	23	18	45
9	23	19	65
10	23	15	65
11	23	17	105

INSERAREA DATELOR:

Pentru generarea mai ușoară a cheilor primare am creat pentru fiecare cheie primară câte o secvență:

```
CREATE SEQUENCE pk_sala
```

```
INCREMENT BY 1
```

```
START WITH 1
```

```
MAXVALUE 99;
```

```
CREATE SEQUENCE pk_angajat
```

```
INCREMENT BY 1
```

```
START WITH 1
```

```
MAXVALUE 9999;
```

```
CREATE SEQUENCE pk_actor  
INCREMENT BY 1  
START WITH 1  
MAXVALUE 99999;
```

```
CREATE SEQUENCE pk_aparatura  
INCREMENT BY 1  
START WITH 1  
MAXVALUE 9999;
```

```
CREATE SEQUENCE pk_bilet  
INCREMENT BY 1  
START WITH 1  
MAXVALUE 9999999;
```

```
CREATE SEQUENCE pk_piesa  
INCREMENT BY 1  
START WITH 1  
MAXVALUE 99999;
```

```
CREATE SEQUENCE pk_trupa  
INCREMENT BY 1  
START WITH 1  
MAXVALUE 999;
```

```
CREATE SEQUENCE pk_spectator  
INCREMENT BY 1  
START WITH 1  
MAXVALUE 9999999;
```

```
CREATE SEQUENCE pk_recuzita  
INCREMENT BY 1  
START WITH 1
```


MAXVALUE 9999;

CREATE SEQUENCE pk_costum

INCREMENT BY 1

START WITH 1

MAXVALUE 9999;

CREATE SEQUENCE pk_reprezentatie

INCREMENT BY 1

START WITH 1

MAXVALUE 9999999999;

CREATE SEQUENCE pk_bilet

INCREMENT BY 1

START WITH 1

MAXVALUE 99999999;

Inserarea în tabele:

a) Tabelul SALĂ:

INSERT INTO SALA

VALUES (pk_sala.NEXTVAL, 'Sala I. L. Caragiale', 292, 10, 4, TO_DATE('2017-03-10', 'YYYY-MM-DD'));

INSERT INTO SALA

VALUES (pk_sala.NEXTVAL, 'Sala Foarte Mica', 100, 6, 3, TO_DATE('2018-04-12', 'YYYY-MM-DD'));

INSERT INTO SALA

VALUES (pk_sala.NEXTVAL, 'Sala Eugen Ionesco', 147, 9, 5, TO_DATE('2015-01-20', 'YYYY-MM-DD'));

INSERT INTO SALA

VALUES (pk_sala.NEXTVAL, 'Sala Rotunda', 256, 12, 5, TO_DATE('2018-11-30', 'YYYY-MM-DD'));

INSERT INTO SALA

VALUES (pk_sala.NEXTVAL, 'Sala Victoria', 300, 9, 5, TO_DATE('2015-7-17', 'YYYY-MM-DD'));

b) Tabelul ANGAJAT:

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Mill', 'Frances', TO_DATE('1997-9-12', 'YYYY-MM-DD'), 'M', '0724985301', TO_DATE('2016-10-12', 'YYYY-MM-DD'), 2200, NULL);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Denzel', 'Wyatt', TO_DATE('1987-9-11', 'YYYY-MM-DD'), 'M', '0778985301', TO_DATE('2016-10-12', 'YYYY-MM-DD'), 2250, NULL);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Marshall', 'Conrad', TO_DATE('2020-4-16', 'YYYY-MM-DD'), 'M', '0724956001', TO_DATE('2019-10-19', 'YYYY-MM-DD'), 2600, NULL);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Mill', 'Frances', TO_DATE('1997-9-12', 'YYYY-MM-DD'), 'F', '0711287889', TO_DATE('2019-10-17', 'YYYY-MM-DD'), 1950, NULL);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Tait', 'Emilia', TO_DATE('1992-3-3', 'YYYY-MM-DD'), 'F', '0729570001', TO_DATE('2020-4-16', 'YYYY-MM-DD'), 1800, NULL);

INSERT INTO ANGAJAT

VALUES (1, 'Bryan', 'Loki', TO_DATE('1973-11-30', 'YYYY-MM-DD'), 'M', '0733513210', NULL, 55000, NULL);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Fulton', 'Mimi', TO_DATE('1999-9-30', 'YYYY-MM-DD'), 'F', '0738340140', TO_DATE('2010-9-5', 'YYYY-MM-DD'), 12000, 1);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Ross', 'Dru', TO_DATE('2020-4-16', 'YYYY-MM-DD'), 'M', '0736420640', TO_DATE('2010-3-5', 'YYYY-MM-DD'), 23600, 1);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Owen', 'Zoe', TO_DATE('1994-1-10', 'YYYY-MM-DD'), 'F', '0736555640', TO_DATE('2015-3-2', 'YYYY-MM-DD'), 18300, 1);

UPDATE ANGAJAT

SET id_manager = 7

WHERE id_angajat >= 9 and id_angajat <= 12 or id_angajat = 2;

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Franklin', 'Faiza', TO_DATE('1997-3-30', 'YYYY-MM-DD'), 'F', '0738563211', TO_DATE('2010-9-5', 'YYYY-MM-DD'), 7500, 13);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Cole', 'Seth', TO_DATE('1985-4-20', 'YYYY-MM-DD'), 'M', '0755316736', TO_DATE('2010-9-5', 'YYYY-MM-DD'), 6500, 13);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Cooke', 'Ariya', TO_DATE('1997-3-30', 'YYYY-MM-DD'), 'F', '0768421998', TO_DATE('2010-9-5', 'YYYY-MM-DD'), 7200, 13);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Kemp', 'Darren', TO_DATE('1992-4-12', 'YYYY-MM-DD'), 'M', '0712630064', TO_DATE('2010-9-5', 'YYYY-MM-DD'), 8200, 13);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Rios', 'Kobie', TO_DATE('1986-3-10', 'YYYY-MM-DD'), 'M', '0789553211', TO_DATE('2009-8-2', 'YYYY-MM-DD'), 5500, 13);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Ramsey', 'Sianna', TO_DATE('1992-4-18', 'YYYY-MM-DD'), 'F', '0712630064', TO_DATE('2010-9-5', 'YYYY-MM-DD'), 3200, 14);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Hook', 'Taya', TO_DATE('1998-8-18', 'YYYY-MM-DD'), 'F', '0718840064', TO_DATE('2004-8-1', 'YYYY-MM-DD'), 1800, 14);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Mair', 'Christy', TO_DATE('2020-4-16', 'YYYY-MM-DD'), 'M', '0712944685', TO_DATE('2004-7-7', 'YYYY-MM-DD'), 1695, 14);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Cruz', 'Jamelia ', TO_DATE('1992-7-18', 'YYYY-MM-DD'), 'F', '0712635564', TO_DATE('2010-9-5', 'YYYY-MM-DD'), 3200, 14);

INSERT INTO ANGAJAT

VALUES (pk_angajat.NEXTVAL, 'Alford', 'Bob', TO_DATE('1973-1-10', 'YYYY-MM-DD'), 'M', '0715347754', TO_DATE('2003-8-4', 'YYYY-MM-DD'), 1950, 14);

c) Tabelul MANAGER:

INSERT INTO MANAGER

VALUES(1, 'albastru', 'MANAGER');

INSERT INTO MANAGER

VALUES(7, 'rosu', 'VANZARI');

INSERT INTO MANAGER

VALUES(13, 'roz', 'TEHNICA');

INSERT INTO MANAGER

VALUES(14, 'alb', 'CURATENIE');

d) Tabelul CASIER:

INSERT INTO CASIER

VALUES (pk_angajat.CURRVAL, 1, 0.15);

INSERT INTO CASIER

VALUES (pk_angajat.CURRVAL, 1, 0.10);

INSERT INTO CASIER

VALUES (pk_angajat.CURRVAL, 2, 0);

INSERT INTO CASIER

VALUES (pk_angajat.CURRVAL, 3, 0.35);

INSERT INTO CASIER

VALUES (pk_angajat.CURRVAL, 3, 0.37);

e) Tabelul INGRIJITOR:

INSERT INTO INGRIJITOR

VALUES(pk_angajat.CURRVAL, 10, 'M');

INSERT INTO INGRIJITOR

VALUES(pk_angajat.CURRVAL, 0, 'S');

```
INSERT INTO INGRIJITOR  
VALUES(pk_angajat.CURRVAL, 40, 'M');  
INSERT INTO INGRIJITOR  
VALUES(pk_angajat.CURRVAL, 22, 'L');  
INSERT INTO INGRIJITOR  
VALUES(pk_angajat.CURRVAL, 5, 'XXL');
```

f) Tabelul TEHNICIAN:

```
INSERT INTO TEHNICIAN  
VALUES(pk_angajat.CURRVAL, 'inginer', 'microfon');  
INSERT INTO TEHNICIAN  
VALUES(pk_angajat.CURRVAL, 'inginer', 'aparat de facut baloane');  
INSERT INTO TEHNICIAN  
VALUES(pk_angajat.CURRVAL, 'inginer', 'laser');  
INSERT INTO TEHNICIAN  
VALUES(pk_angajat.CURRVAL, 'programator', 'laser');  
INSERT INTO TEHNICIAN  
VALUES(pk_angajat.CURRVAL, 'fizician', 'proiector');
```

g) Tabelul PIESĂ:

```
INSERT INTO PIESA  
VALUES (pk_piesa.NEXTVAL, 'Iona', 'tragedie', 70, 'Luni', '10:30', 25);  
INSERT INTO PIESA  
VALUES (pk_piesa.NEXTVAL, 'O noapte furtunoasa', 'comedie', 86, 'Luni', '18:00', 50);  
INSERT INTO PIESA  
VALUES (pk_piesa.NEXTVAL, 'O scrisoare pierduta', 'comedie', 133, 'Sambata', '13:15', 53);  
INSERT INTO PIESA
```

VALUES (pk_piesa.NEXTVAL, 'Hamlet', 'tragedie', 151, 'Joi', '15:00', 47);

INSERT INTO PIESA

VALUES (pk_piesa.NEXTVAL, 'Adio, domnule Haffmann', 'drama', 105, 'Vineri', '19:00', 44);

INSERT INTO PIESA

VALUES (pk_piesa.NEXTVAL, 'Scufita Rosie', 'comedie', 70, 'Sambata', '10:30', 15);

INSERT INTO PIESA

VALUES (pk_piesa.NEXTVAL, 'ActOrchestra', 'muzical', 94, 'Vineri', '14:30', 35);

UPDATE PIESA

SET pauza = 0, autor = 'Marin Sorescu'

WHERE id_piesa = 1;

UPDATE PIESA

SET pauza = 1, autor = 'Ion Luca Caragiale'

WHERE id_piesa = 2;

UPDATE PIESA

SET pauza = 0, autor = 'Ion Luca Caragiale'

WHERE id_piesa = 3;

UPDATE PIESA

SET pauza = 1, autor = 'William Shakespeare'

WHERE id_piesa = 4;

UPDATE PIESA

SET pauza = 1, autor = 'Jean-Philippe Daguerre'

WHERE id_piesa = 5;

UPDATE PIESA

SET pauza = 0

WHERE id_piesa = 6;

UPDATE PIESA

SET pauza = 0

WHERE id_piesa = 7;

h) Tabelul TRUPĂ

INSERT INTO TRUPA

VALUES (pk_trupa.NEXTVAL, 'Gasca Zurli', 'Bucuresti', '0733033522', 'gasca@zurli.com', TO_DATE('2016-9-12', 'YYYY-MM-DD'));

INSERT INTO TRUPA

VALUES (pk_trupa.NEXTVAL, 'Trupa Chaplin', 'Iasi', '0740445400', 'chaplin05@gmail.com', TO_DATE('2017-10-2', 'YYYY-MM-DD'));

INSERT INTO TRUPA

VALUES (pk_trupa.NEXTVAL, 'Trupa TNB', 'Bucuresti', '0733338899', 't_n_b@gmail.com', TO_DATE('2020-9-11', 'YYYY-MM-DD'));

INSERT INTO TRUPA

VALUES (pk_trupa.NEXTVAL, 'Asociatia Teta', 'Craiova', '0722121233', 'asctt33@yahoo.com', TO_DATE('2020-2-2', 'YYYY-MM-DD'));

INSERT INTO TRUPA

VALUES (pk_trupa.NEXTVAL, 'Trupa EXCELSIOR', 'Brasov', '0744340202', 'exc_22@gmail.com', TO_DATE('2019-3-27', 'YYYY-MM-DD'));

i) Tabelul ACTOR:

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 3, 'Barnard', 'Peter', 24, 'M', '0722439698', 'XS');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 3, 'Drew', 'Timothy', 20, 'M', '0725539678', 'M');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 3, 'Burch', 'Suzanna', 21, 'F', '0789722603', 'S');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 3, 'Ratcliffe', 'Devin', 19 , 'M', '0733577082', 'XL');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 3, 'Peyton', 'Allan', 24 , 'M', '0752530099', 'S');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 4, 'Rios', 'Alessandra', 18 , 'F', '0722439549', 'XS');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 4, 'Worthington', 'Miley', 20 , 'F', '0721486698', 'L');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 4, 'Deleon', 'Cynthia', 21 , 'F', '0795506698', 'M');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 6, 'Bryan', 'Aran', 32 , 'M', '0711134990', 'S');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 6, 'Hartman', 'Kieron', 34 , 'M', '0766235990', 'M');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 5, 'Ware', 'Sasha', 19 , 'M', '0722113038', 'M');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 5, 'Avalos', 'Maddy', 23 , 'F', '0722430040', 'S');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 5, 'Burch', 'Eliot', 30 , 'M', '078749205', 'XS');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 5, 'Leal', 'Susie', 25 , 'F', '0769114037', 'M');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 7, 'Curtis', 'Steffan', 37 , 'M', '0734346890', 'M');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, 7, 'Nichollas', 'Hughie', 35 , 'M', '0736183055', 'L');

INSERT INTO ACTOR

VALUES (pk_actor.NEXTVAL, NULL, 'Swift', 'Martin', 23 , 'M', '076799205', 'L')

j) Tabelul COSTUM:

```
INSERT INTO COSTUM
```

```
VALUES(pk_costum.NEXTVAL, 6, 'Zana cea buna', 'S', 'in', 1, 340);
```

```
INSERT INTO COSTUM
```

```
VALUES(pk_costum.NEXTVAL, 6, 'Vrajitoare', 'L', 'bumbac', 0, 420);
```

```
INSERT INTO COSTUM
```

```
VALUES(pk_costum.NEXTVAL, 6, 'Imparat', 'XL', 'piele', 1, 500);
```

```
INSERT INTO COSTUM
```

```
VALUES(pk_costum.NEXTVAL, 7, 'Cersetor', 'XS', 'bumbac', 0, 430);
```

```
INSERT INTO COSTUM
```

```
VALUES(pk_costum.NEXTVAL, 7, 'Fantoma', 'M', 'bumbac', 0, 140);
```

```
INSERT INTO COSTUM
```

```
VALUES(pk_costum.NEXTVAL, 7, 'Pirat', 'S', 'in', 1, 220);
```

```
INSERT INTO COSTUM
```

```
VALUES(pk_costum.NEXTVAL, 7, 'Zmeu', 'XXL', NULL, 0, 450);
```

k) Tabelul RECUZITĂ:

```
INSERT INTO RECUZITA
```

```
VALUES(pk_recuzita.NEXTVAL, 3, 'sabie', 'plastic', 70, 30, 10, 100);
```

```
INSERT INTO RECUZITA
```

```
VALUES(pk_recuzita.NEXTVAL, 3, 'castel', 'carton', 200, 150, 300, 3000);
```

```
INSERT INTO RECUZITA
```

```
VALUES(pk_recuzita.NEXTVAL, 4, 'pistol', 'metal', 40, 30, 10, 100);
```

```
INSERT INTO RECUZITA
```

```
VALUES(pk_recuzita.NEXTVAL, 5, 'arc cu sageti', 'plastic', 40, 10, 10, 250);
```

```
INSERT INTO RECUZITA
```

VALUES(pk_recuzita.NEXTVAL, 5, 'corabie', 'carton', 250, 100, 310, 3200);

INSERT INTO RECUZITA

VALUES(pk_recuzita.NEXTVAL, 5, 'copac', 'carton', 20, 30, 180, 70);

INSERT INTO RECUZITA

VALUES(pk_recuzita.NEXTVAL, 5, 'caine', 'plus', 50, 40, 20, 50);

1) Tabelul APARATURĂ:

INSERT INTO APARATURA

VALUES (pk_aparatura.NEXTVAL, 'Microfon', 'Sony', TO_DATE('2008','YYYY'), 500, 0);

INSERT INTO APARATURA

VALUES (pk_aparatura.NEXTVAL, 'Boxa', 'Sony', TO_DATE('2008','YYYY'), 720, 0);

INSERT INTO APARATURA

VALUES (pk_aparatura.NEXTVAL, 'Masina de fum', 'Beamz', TO_DATE('2019','YYYY'), 323, 1);

INSERT INTO APARATURA

VALUES (pk_aparatura.NEXTVAL, 'Microfon', 'Philips', TO_DATE('2020','YYYY'), 430, 1);

INSERT INTO APARATURA

VALUES (pk_aparatura.NEXTVAL, 'Laser', 'Beamz', TO_DATE('2012','YYYY'), 120, 0);

INSERT INTO APARATURA

VALUES (pk_aparatura.NEXTVAL, 'Masina de facut baloane', 'Kelkoo', TO_DATE('2020','YYYY'), 300, 1);

INSERT INTO APARATURA

VALUES (pk_aparatura.NEXTVAL, 'Sistem sonorizare', 'Sony', TO_DATE('2008','YYYY'), 1600, 1);

INSERT INTO APARATURA

VALUES (pk_aparatura.NEXTVAL, 'Proiector', 'Stairville', TO_DATE('2016','YYYY'), 1200, 1);

INSERT INTO APARATURA

VALUES (pk_aparatura.NEXTVAL, 'Microfon', 'Sony', TO_DATE('2008','YYYY'), 500, 0);

m) Tabelul BILET:

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 2, 9, 10, 2, 1);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 2, 10, 4, 2, 1);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 2, 11, 10, 1, 1);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 9, 12, 3, 3, 2);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 10, 13, 3, 5, 2);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 10, 9, 15, 2, 2);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 10, 10, 20, 2, 3);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 9, 11, 11, 2, 3);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 11, 12, 10, 10, 4);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 11, 13, 14, 1, 4);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 11, 14, 1, 10, 4);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 11, 9, 12, 12, 4);

INSERT INTO BILET\

VALUES (pk_bilet.NEXTVAL, 12, 14, 30, 7, 5);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 12, 13, 8, 2, 5);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 12, 12, 19, 5, 5);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 12, 11, 13, 4, 5);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 12, 10, 24, 2, 5);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 12, 9, 11, 1, 5);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 12, 13, 21, 3, 7);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 12, 11, 3, 10, 6);

INSERT INTO BILET

VALUES (pk_bilet.NEXTVAL, 12, 12, 1, 1, 6);

n) Tabelul SPECTATOR:

INSERT INTO SPECTATOR

VALUES(pk_spectator.NEXTVAL, 'Herman', 'Jim', 20, 'M', '0748222989', 'jimmy.jim@yahoo.com');

INSERT INTO SPECTATOR

VALUES (pk_spectator.NEXTVAL, 'Hinton', 'Caitlin', 18, 'F', '0734341110', 'caittyhint@gmail.com');

INSERT INTO SPECTATOR

VALUES (pk_spectator.NEXTVAL, 'Bull', 'Jack', 30, 'M', '0778993212', 'redbull@gmail.com');

INSERT INTO SPECTATOR

VALUES (pk_spectator.NEXTVAL, 'McLaughlin', 'Gabriel', 45, 'F', '0703031156', 'gabbyl@gmail.com');

INSERT INTO SPECTATOR

VALUES (pk_spectator.NEXTVAL, 'Horner', 'Mia', 22, 'F', '0799458921', 'mama_mia@yahoo.com');

INSERT INTO SPECTATOR

VALUES (pk_spectator.NEXTVAL, 'Hill', 'Emilio', 23, 'M', '0744953329', 'em_hill@gmail.com');

o) Tabelul REPREZENTATIE:

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 1, 15, 0.8, TO_DATE('2020-3-25', 'YYYY-MM-DD'), '19:00');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 1, 16, 0.67, TO_DATE('2020-3-27', 'YYYY-MM-DD'), '18:15');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 2, 17, 0.8, TO_DATE('2020-3-25', 'YYYY-MM-DD'), '18:30');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 2, 18, 0.92, TO_DATE('2020-4-25', 'YYYY-MM-DD'), '12:15');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 3, 19, 0.83, TO_DATE('2020-4-5', 'YYYY-MM-DD'), '12:30');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 3, 15, 0.75, TO_DATE('2020-4-20', 'YYYY-MM-DD'), '12:15');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 3, 15, 0.69, TO_DATE('2020-3-20', 'YYYY-MM-DD'), '14:00');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 4, 16, 0.56, TO_DATE('2020-3-30', 'YYYY-MM-DD'), '16:30');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 4, 16, 0.95, TO_DATE('2020-4-5', 'YYYY-MM-DD'), '20:00');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 5, 17, 0.40, TO_DATE('2020-4-5', 'YYYY-MM-DD'), '19:00');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 6, 18, 0.8, TO_DATE('2020-4-10', 'YYYY-MM-DD'), '20:00');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 6, 19, 0.2, TO_DATE('2020-4-10', 'YYYY-MM-DD'), '18:15');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 7, 15, 0.9, TO_DATE('2020-4-15', 'YYYY-MM-DD'), '19:00');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 7, 15, 0.78, TO_DATE('2020-4-16', 'YYYY-MM-DD'), '20:00');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 7, 15, 0.84, TO_DATE('2020-4-25', 'YYYY-MM-DD'), '19:00');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 1, 15, 0.3, TO_DATE('2021-5-24', 'YYYY-MM-DD'), '15:00');

INSERT INTO REPREZENTATIE

VALUES (pk_reprezentatie.NEXTVAL, 2, 15, 0.4, TO_DATE('2021-5-24', 'YYYY-MM-DD'), '19:00');

p) Tabelul FOLOSEȘTE:

INSERT INTO FOLOSESTE

VALUES (15, 1, 15, 'buna');

INSERT INTO FOLOSESTE

VALUES (15, 2, 16, 'buna');

INSERT INTO FOLOSESTE

VALUES (15, 2, 17, 'buna');

INSERT INTO FOLOSESTE

VALUES (16, 3, 18, 'buna');

INSERT INTO FOLOSESTE

VALUES (16, 3, 19, 'foarte buna');

INSERT INTO FOLOSESTE

VALUES (17, 3, 15, 'foarte buna');

INSERT INTO FOLOSESTE

```
VALUES (18, 4, 16, 'buna');  
  
INSERT INTO FOLOSESTE  
  
VALUES (18, 6, 17, 'buna');  
  
INSERT INTO FOLOSESTE  
  
VALUES (18, 6, 18, 'buna');  
  
INSERT INTO FOLOSESTE  
  
VALUES (19, 7, 19, 'mici defectiuni');  
  
INSERT INTO FOLOSESTE  
  
VALUES (19, 8, 15, 'buna');  
  
INSERT INTO FOLOSESTE  
  
VALUES (16, 8, 16, 'buna');  
  
INSERT INTO FOLOSESTE  
  
VALUES (16, 9, 17, 'mici defectiuni');  
  
INSERT INTO FOLOSESTE  
  
VALUES (16, 10, 18, 'foarte buna');  
  
INSERT INTO FOLOSESTE  
  
VALUES (15, 10, 15, 'foarte buna');
```

q) Tabelul JOACĂ:

```
INSERT INTO JOACA  
  
VALUES (1, 1, 'principal');  
  
INSERT INTO JOACA  
  
VALUES (1, 2, 'secundar');  
  
INSERT INTO JOACA  
  
VALUES (1, 3, 'figurant');  
  
INSERT INTO JOACA  
  
VALUES (1, 4, 'episodic');
```

```
INSERT INTO JOACA
VALUES (1, 5, 'dublura');
INSERT INTO JOACA
VALUES (1, 4, 'principal');
INSERT INTO JOACA
VALUES (2, 6, 'secundar');
INSERT INTO JOACA
VALUES (2, 7, 'dublura');
INSERT INTO JOACA
VALUES (3, 16, 'episodic');
INSERT INTO JOACA
VALUES (4, 15, 'episodic');
INSERT INTO JOACA
VALUES (5, 1, 'principal');
INSERT INTO JOACA
VALUES (6, 11, 'principal');
INSERT INTO JOACA
VALUES (6, 12, 'secundar');
INSERT INTO JOACA
VALUES (7, 12, 'secundar');
INSERT INTO JOACA
VALUES (7, 13, 'secundar');
INSERT INTO JOACA
VALUES (7, 21, 'figurant');
INSERT INTO JOACA
VALUES (2, 16, 'principal');
INSERT INTO JOACA
VALUES (3, 7, 'secundar');
```


r) Tabelul ÎNGRIJEȘTE:

```
INSERT INTO INGRIJESTE
```

```
VALUES (20, 15, 60);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (20, 16, 90);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (20, 17, 90);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (20, 18, 30);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (20, 19, 15);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (21, 15, 100);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (21, 16, 25);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (22, 17, 35);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (23, 18, 45);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (23, 19, 65);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (23, 15, 65);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (23, 17, 105);
```

```
INSERT INTO INGRIJESTE
```

```
VALUES (24, 15, 100);
```

```
INSERT INTO INGRIJESE
```

```
VALUES (24, 19, 95);
```

```
INSERT INTO INGRIJESE
```

```
VALUES (24, 18, 90);
```

SUBPROGRAM STOCAT CU 2 TIPURI DE COLECȚII:

6. La teatru există zile foarte aglomerate, iar îngrijitorii au mult de lucru. De aceea managerii s-au gândit să ofere celor mai activi îngrijitori o mărire de salariu. Astfel, fiecare îngrijitor care a făcut curățenie în cel puțin x săli în data d, va primi o mărire de salariu de 5%. Scrieți un subprogram stocat care primește ca parametrii numărul de săli minim pentru care îngrijitorul este considerat activ și data în care teatrul a fost aglomerat și afișează cine a curățat fiecare sală din acea zi și modifică salariul îngrijitorilor activi.

```
CREATE OR REPLACE PROCEDURE ingrijitori_activi (x IN NUMBER,  
                                                d IN DATE DEFAULT SYSDATE)
```

```
AS
```

```
TYPE v_ingrijitori IS VARRAY(100) OF angajat%ROWTYPE;    --- varray
```

```
TYPE multime_sali IS TABLE OF sala.id_sala%TYPE;        --- nested table
```

```
v_ing      v_ingrijitori := v_ingrijitori();
```

```
sali      multime_sali := multime_sali();
```

```
sali_ing   multime_sali := multime_sali();
```

```
ing_activi v_ingrijitori := v_ingrijitori();
```

```
ct        NUMBER(3) := 1;
```

```
nr_max     NUMBER(3) := 100; --- la teatru nu vor fi angajati mai mult de 100 de ingrijitori o dat
```

```
NO_SHOW    EXCEPTION;
```

```
BEGIN
```

--- selectam toti ingrijitorii si ii salvam in array

SELECT *

BULK COLLECT INTO v_ing

FROM angajat a

WHERE id_angajat IN (SELECT id_angajat

FROM INGRIJITOR);

--- selectam toate salile ce au fost folosite in data d

SELECT DISTINCT id_sala

BULK COLLECT INTO sali

FROM reprezentatie

WHERE to_char(data_org, 'YYYY-MM-DD') = to_char(d, 'YYYY-MM-DD');

--- daca nu sunt spectacole in ziua respectiva ne oprim

IF sali.COUNT() = 0

THEN

RAISE NO_SHOW;

END IF;

--- pentru fiecare ingrijitor

FOR ing IN v_ing.FIRST..v_ing.LAST

LOOP

--- selectam salile in care a facut curat angajatul ing

SELECT DISTINCT id_sala

BULK COLLECT INTO sali_ing

FROM ingrijeste

WHERE id_angajat = v_ing(ing).id_angajat;

--- retinem in sali_ing doar salile in care au fost reprezentate specacole in data d

sali_ing := sali_ing MULTISSET INTERSECT sali;

--- daca ingrijitorul a curatat mai mult de x sali atunci el este foarte activ

IF sali_ing.COUNT() >= x

THEN

DBMS_OUTPUT.PUT('Ingrijitorul ' || v_ing(ing).nume || ' ' || v_ing(ing).prenume || ' a curatat salile: ');

FOR sala IN sali_ing.FIRST..sali_ing.LAST

LOOP

```

        DBMS_OUTPUT.PUT(sali_ing(sala) || ' ');
    END LOOP;
    DBMS_OUTPUT.PUT_LINE("");
    --- adaugam angajatul activ in vector
    ing_activi.EXTEND();
    ing_activi(ing_activi.LAST) := v_ing(ing);
    ct := ct + 1;
END IF;
END LOOP;
--- toti angajatii activi vor primi o marire de salariu
FORALL ing IN ing_activi.FIRST..ing_activi.LAST
    UPDATE angajat
    SET salariu = salariu + salariu * 0.05
    WHERE id_angajat = ing_activi(ing).id_angajat;
DBMS_OUTPUT.PUT_LINE("");
DBMS_OUTPUT.PUT_LINE('S-au actualizat salariile a ' || ing_activi.COUNT() || ' angajati');
EXCEPTION
WHEN NO_SHOW THEN
    RAISE_APPLICATION_ERROR(-20001, 'Nu exista spectacole in ziua respectiva');
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20000, 'Alta exceptie');
END ingrijitori_activi; /

```

Apelare:

```

DECLARE

    data_solicitata DATE := TO_DATE('25-MAR-20', 'DD-MON-YY');

BEGIN

    ingrijitori_activi(1, data_solicitata);

    --ingrijitori_activi(1);

END; /

```

The screenshot shows the Oracle SQL Developer interface. The top toolbar contains various icons for file operations, execution, and formatting. Below the toolbar, the 'Query Builder' tab is active, displaying a PL/SQL procedure named `ingrijitori_activi`. The procedure includes a `DBMS_OUTPUT.PUT_LINE` statement, an `EXCEPTION` block with `NO_SHOW` and `OTHERS` handlers, and a `DECLARE` section with a date variable. The `Script Output` window at the bottom shows the execution results, including the procedure compilation status and the output of the `ingrijitori_activi` procedure call.

```
DBMS_OUTPUT.PUT_LINE('S-au actualizat salariile a ' || ing_activi.COUNT() || ' angajati');

EXCEPTION
WHEN NO_SHOW THEN
    RAISE_APPLICATION_ERROR(-20001, 'Nu exista spectacole in ziua respectiva');
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20000, 'Alta exceptie');

END ingrijitori_activi;
/

DECLARE
    data_solicitata DATE := TO_DATE('25-MAR-20', 'DD-MON-YY');
BEGIN
    ingrijitori_activi(1, data_solicitata);
    --ingrijitori_activi(1);
END;
/
```

Script Output x

Task completed in 0.268 seconds

Procedure INGRIJITORI_ACTIVI compiled

Ingrijitorul Ramsey Sianna a curatat salile: 15 17
Ingrijitorul Hook Taya a curatat salile: 15
Ingrijitorul Cruz Jamelia a curatat salile: 17
Ingrijitorul Mair Christy a curatat salile: 15 17
Ingrijitorul Alford Bob a curatat salile: 15

S-au actualizat salariile a 5 angajati

PL/SQL procedure successfully completed.

```
DECLARE
    data_solicitata DATE := TO_DATE('25-MAR-20', 'DD-MON-YY');
BEGIN
    --ingrijitori_activi(1, data_solicitata);
    ingrijitori_activi(1);
END;
```

Script Output x

Task completed in 0.109 seconds

Error starting at line : 264 in command -

```
DECLARE
    data_solicitata DATE := TO_DATE('25-MAR-20', 'DD-MON-YY');
BEGIN
    --ingrijitori_activi(1, data_solicitata);
    ingrijitori_activi(1);
END;
```

Error report -

ORA-20001: Nu exista spectacole in ziua respectiva

ORA-06512: at "MATEI_DORIAN.INGRIJITORI_ACTIVI", line 77

ORA-06512: at line 5

Observații: Primul apel al procedurii ilustrează comportamentul acesteia pe cazul general, când în ziua furnizată ca parametru există reprezentații. Al doilea apel al funcției folosește valoarea default a parametrului d și ilustrează cazul particular în care nu avem nicio reprezentație în ziua respectivă, iar procedura va arunca o excepție. Tipurile de colecții folosite sunt VARRAY și NESTED TABLES.

SUBPROGRAM STOCAT CU UN TIP DE CURSOR:

7. Administratorul teatrului vrea să facă un afiș pentru teatrul său. Scrieți un subprogram stocat care să afișeze fiecare trupă ce este activă la teatru, iar pentru fiecare trupă să precizeze și fiecare actor membru. Pe afiș trupele de teatru vor fi trecute în ordine alfabetică, iar actorii vor fi trecuți în ordinea descrescătoare a numărului de piese în care joacă. Actorii independenți nu vor fi trecuți pe afiș.

```
CREATE OR REPLACE PROCEDURE generare_afis
```

```
IS
```

```
    TYPE ref_cursor IS REF CURSOR;
```

```
    CURSOR c_trupa IS
```

```
    SELECT nume,
```

```
    CURSOR (
```

```
        WITH nr_reprezentatii AS
```

```
        (
```

```
            SELECT CASE
```

```
                WHEN a.id_actor IS NOT NULL THEN
```

```
                    COUNT(*)
```

```
            ELSE
```

```
                0
```

```
            END nr, a.id_actor
```

```
        FROM actor a
```

```
        FULL OUTER JOIN joaca j ON a.id_actor = j.id_actor
```

```
        GROUP BY a.id_actor
```

```
    )
```

```
    SELECT nume, prenume
```

```
    FROM actor a
```

```
    JOIN nr_reprezentatii n ON a.id_actor = n.id_actor
```

```
    WHERE a.id_trupa = t.id_trupa
```

```

        ORDER BY nr DESC

    )

FROM trupa t

ORDER BY nume;

v_cursor    ref_cursor;

nume_trupa   trupa.nume%TYPE;

nume_actor   actor.nume%TYPE;

prenume_actor actor.prenume%TYPE;

BEGIN

OPEN c_trupa;

LOOP

    FETCH c_trupa INTO nume_trupa, v_cursor;

    EXIT WHEN c_trupa%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE(nume_trupa);

    LOOP

        FETCH v_cursor INTO nume_actor, prenume_actor;

        EXIT WHEN v_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(' ' || v_cursor%ROWCOUNT || ' ' || nume_actor || ' ' || prenume_actor);

    END LOOP;

    DBMS_OUTPUT.PUT_LINE("");

END LOOP;

CLOSE c_trupa;

END generare_afis;

/

```

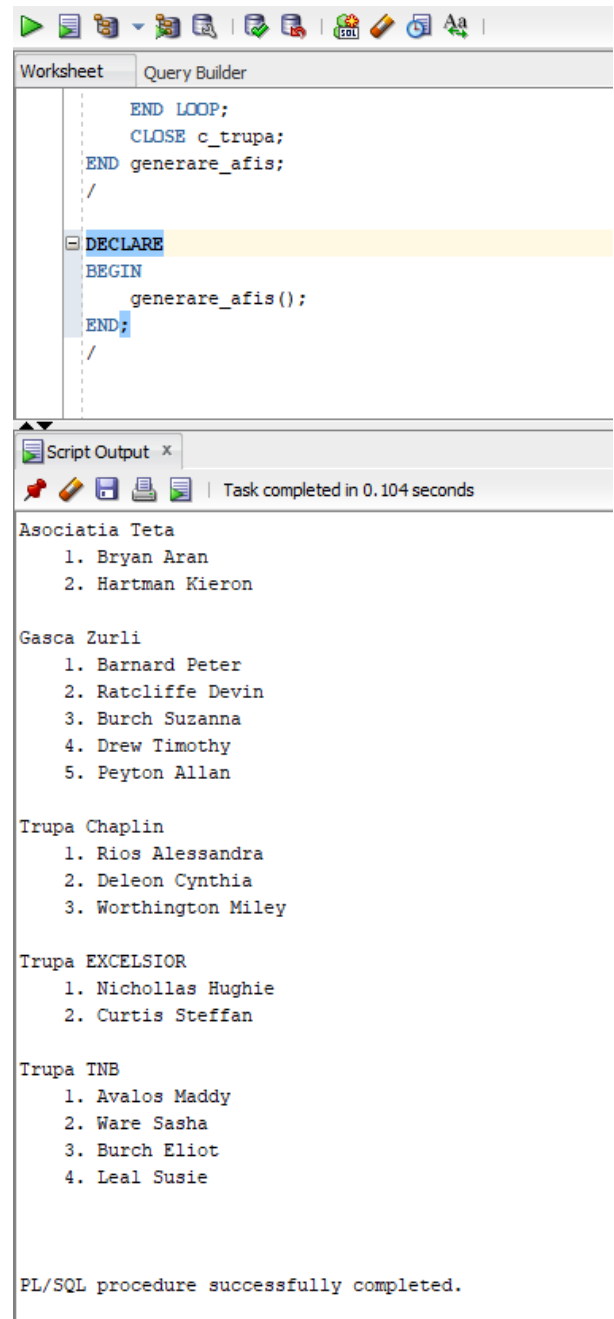
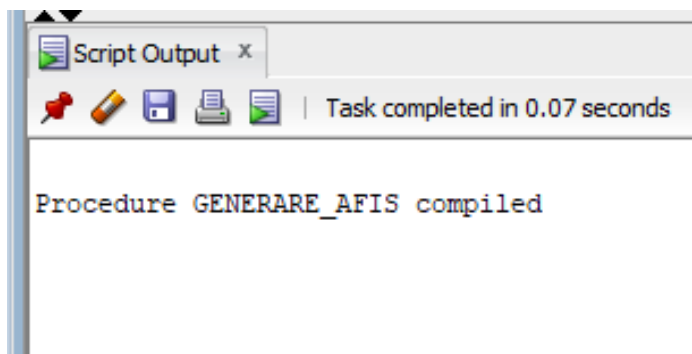

Apelare:

DECLARE

BEGIN

 generare_afis();

END; /



Observații: pentru rezolvarea problemei am folosit expresii cursor.

SUBPROGRAM STOCAT DE TIP FUNCȚIE:

8. Pentru a organiza o reprezentație a unei piese, fiecare actor are nevoie de cel puțin un costum. Scrieți o funcție care returnează numărul de costume pe care un actor poate să le folosească la o reprezentație (actorul poate folosi costumul dacă poartă aceeași mărime). În cazul în care actorului nu i se potrivește niciun costum se va arunca o eroare pentru a semnaliza această problemă.

```
CREATE OR REPLACE FUNCTION numar_costume(v_actor IN actor.id_actor%TYPE)
```

```
RETURN NUMBER IS
```

```
actor_gasit actor.id_actor%TYPE := 0;
```

```
raspuns      NUMBER;
```

```
BAD_ACTOR_ID EXCEPTION;
```

```
BEGIN
```

```
-- verificam daca actorul exista
```

```
SELECT COUNT(id_actor)
```

```
INTO actor_gasit
```

```
FROM actor
```

```
WHERE id_actor = v_actor;
```

```
IF(actor_gasit = 0) THEN
```

```
    RAISE BAD_ACTOR_ID;
```

```
END IF;
```

```
SELECT COUNT(*)
```

```
INTO raspuns
```

```
FROM (SELECT id_costum, id_actor
```

```
      FROM (SELECT id_actor, id_trupa, marime
```

```
            FROM actor
```

```
            WHERE id_actor = v_actor
```

```
            ) a
```

```
      JOIN trupa t ON a.id_trupa = t.id_trupa
```

```

        JOIN costum c ON c.id_trupa = t.id_trupa

        WHERE c.marime = a.marime

    )

    GROUP BY id_actor;

    RETURN raspuns;

EXCEPTION

    WHEN NO_DATA_FOUND THEN

        RAISE_APPLICATION_ERROR(-20003, 'Nu exista costume pentru acest actor');

    WHEN BAD_ACTOR_ID THEN

        RAISE_APPLICATION_ERROR(-20002, 'Nu exista actorul cu id-ul dat');

    WHEN OTHERS THEN

        RAISE_APPLICATION_ERROR(-20000, 'Eroare necunoscuta');

END numar_costume;

/

```

Apelare:

```

BEGIN

    DBMS_OUTPUT.PUT_LINE(numar_costume(9));

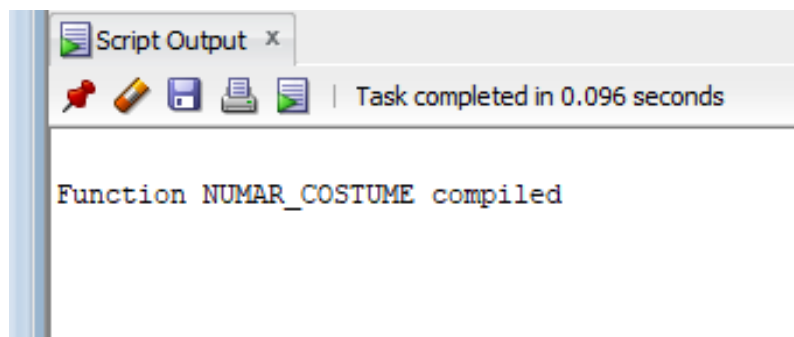
    --DBMS_OUTPUT.PUT_LINE(numar_costume(3));

    --DBMS_OUTPUT.PUT_LINE(numar_costume(10000));

END;

/

```



```
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20000, 'Eroare necunoscuta');
```

```
END numar_costume;
/
```

```
BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE(numar_costume(9));
    --DBMS_OUTPUT.PUT_LINE(numar_costume(3));
    --DBMS_OUTPUT.PUT_LINE(numar_costume(10000));
```

```
END;
/
```

Script Output x

Task completed in 0.1 seconds

Function NUMAR_COSTUME compiled

1

PL/SQL procedure successfully completed.

Worksheet Query Builder

```
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20000, 'Eroare necunoscuta');
```

```
END numar_costume;
/
```

```
BEGIN
```

```
    --DBMS_OUTPUT.PUT_LINE(numar_costume(9));
    DBMS_OUTPUT.PUT_LINE(numar_costume(3));
    --DBMS_OUTPUT.PUT_LINE(numar_costume(10000));
```

```
END;
/
```

Script Output x

Task completed in 0.079 seconds

Error starting at line : 388 in command -

BEGIN

```
    --DBMS_OUTPUT.PUT_LINE(numar_costume(9));
    DBMS_OUTPUT.PUT_LINE(numar_costume(3));
    --DBMS_OUTPUT.PUT_LINE(numar_costume(10000));
```

END;

Error report -

ORA-20003: Nu exista costume pentru acest actor

ORA-06512: at "MATEI_DORIAN.NUMAR_COSTUME", line 37

ORA-06512: at line 3

```
        RAISE_APPLICATION_ERROR(-20002, 'Nu exista actorul cu id-ul dat');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20000, 'Eroare necunoscuta');

    END numar_costume;
/

--BEGIN
--DBMS_OUTPUT.PUT_LINE(numar_costume(9));
--DBMS_OUTPUT.PUT_LINE(numar_costume(3));
DBMS_OUTPUT.PUT_LINE(numar_costume(10000));
END;
```

Script Output x

Task completed in 0.076 seconds

Error starting at line : 388 in command -
BEGIN
--DBMS_OUTPUT.PUT_LINE(numar_costume(9));
--DBMS_OUTPUT.PUT_LINE(numar_costume(3));
DBMS_OUTPUT.PUT_LINE(numar_costume(10000));
END;
Error report -
ORA-20002: Nu exista actorul cu id-ul dat
ORA-06512: at "MATEI_DORIAN.NUMAR_COSTUME", line 39
ORA-06512: at line 4

Observații: funcția folosește entitățile ACTOR, TRUPĂ și COSTUM. Primul apel al funcției ilustrează cazul general, când actorul are cel puțin un costum, iar funcția returnează numărul de costume. Al doilea apel ilustrează cazul în care actorul nu are niciun costum, iar în al treilea caz se încearcă apelarea funcției cu un id inexistent; în ambele cazuri funcția va returna o eroare sugestivă. Eroarea TOO MANY ROWS nu poate fi întâlnită în acest caz deoarece în ambele selecturi folosim COUNT() care returnează un singur număr.

SUBPROGRAM STOCAT DE TIP PROCEDURĂ:

9. La teatru există și aparatură mai veche care trebuie manevrată cu grijă. Aparatele vechi nu vor fi folosite mai mult de o dată pe zi. Scrieți o procedură care primește numele unui aparat și o dată și afișează numele și profesia tehnicianului care o folosește și spectacolul în care este folosit aparatul. Dacă există mai multe spectacole în aceeași zi care folosesc același dispozitiv se va afișa o eroare corespunzătoare.

```
CREATE OR REPLACE PROCEDURE verifica_aparatura_veche(v_ap IN aparatura.denumire%TYPE, v_data IN reprezentatie.data_org%TYPE)
```

```
IS
```

```
TYPE date_utilizare IS RECORD
```

```
(
```

```
    denumire    aparatura.denumire%TYPE,
```

```
    nume        angajat.nume%TYPE,
```

```
    prenume     angajat.prenume%TYPE,
```

```
    specializare tehnician.specializare%TYPE,
```

```
    titlu       piesa.titlu%TYPE
```

```
);
```

```
ap_gasit    NUMBER;
```

```
v_info      date_utilizare;
```

```
NO_DEVICE   EXCEPTION;
```

```
BEGIN
```

```
SELECT COUNT(id_aparatura)
```

```
INTO ap_gasit
```

```
FROM aparatura
```

```
WHERE denumire LIKE v_ap;
```

```
IF(ap_gasit = 0) THEN
```

```
    RAISE NO_DEVICE;
```

```
END IF;
```

```
SELECT a.denumire, ang.nume, ang.prenume, t.specializare, p.titlu
```

```

INTO v_info

FROM (SELECT id_aparatura, id_angajat, id_sala FROM foloseste) f

JOIN (SELECT id_aparatura, denumire FROM aparatura) a ON a.id_aparatura = f.id_aparatura

JOIN (SELECT id_angajat, nume, prenume FROM angajat) ang ON f.id_angajat = ang.id_angajat

JOIN (SELECT id_angajat, specializare FROM tehnician) t ON t.id_angajat = ang.id_angajat

JOIN (SELECT id_sala FROM sala) s ON s.id_sala = f.id_sala

JOIN (SELECT data_org, id_sala, id_piesa

      FROM reprezentatie

      WHERE to_char(data_org, 'DD-MM-YYYY') = to_char(v_data, 'DD-MM-YYYY')

      ) r ON r.id_sala = s.id_sala -- where

JOIN (SELECT id_piesa, titlu FROM piesa) p ON p.id_piesa = r.id_piesa

WHERE a.denumire LIKE v_ap;

DBMS_OUTPUT.PUT_LINE('OK: Tehnicianul ' || v_info.nume || ' ' || v_info.prenume || ' (' || v_info.specializare || ') '
||
      'foloseste ' || LOWER(v_info.denumire) || ' pentru piesa ' || v_info.titlu || ' in data de '
      || TO_CHAR(v_data, 'DD-MM-YYYY'));

EXCEPTION

  WHEN NO_DATA_FOUND THEN

    RAISE_APPLICATION_ERROR(-20004, 'Dispozitivul nu este folosit in aceasta zi');

  WHEN TOO_MANY_ROWS THEN

    RAISE_APPLICATION_ERROR(-20005, 'Dispozitivul este folosit de prea multe ori!');

  WHEN NO_DEVICE THEN

    RAISE_APPLICATION_ERROR(-20006, 'Dispozitiv inexistent');

  WHEN OTHERS THEN

    RAISE_APPLICATION_ERROR(-20000, 'Eroare necunoscuta');

END verifica_aparatura_veche; /

```

Apelare:

```
DECLARE

v_denumire aparatura.denumire%TYPE;

v_data    reprezentatie.data_org%TYPE;

BEGIN

--- TOO MANY ROWS

--v_denumire := 'Microfon';

--v_data    := TO_DATE('24-MAY-21');

--verifica_aparatura_veche(v_denumire, v_data);

--- NO DATA FOUND

--v_denumire := 'Microfon';

--v_data    := TO_DATE('24-MAY-23');

--verifica_aparatura_veche(v_denumire, v_data);

--- NO DEVICE

--v_denumire := 'Minge';

--v_data    := TO_DATE('24-MAY-23');

--verifica_aparatura_veche(v_denumire, v_data);

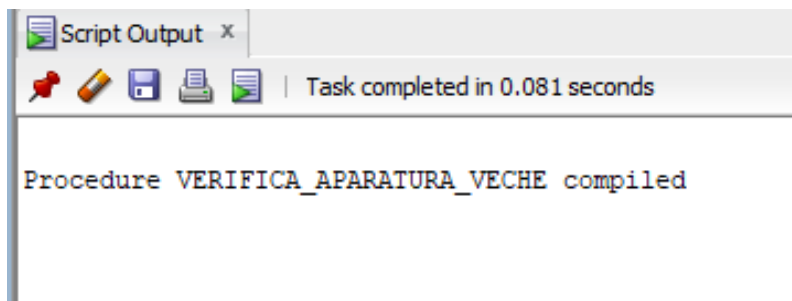
v_denumire := 'Laser';

v_data    := TO_DATE('25-APR-20');

verifica_aparatura_veche(v_denumire, v_data);

END;

/
```




```
--verifica_aparatura_veche(v_denumire, v_data);

--- NO DEVICE
--v_denumire := 'Minge';
--v_data      := TO_DATE('24-MAY-23');
--verifica_aparatura_veche(v_denumire, v_data);

v_denumire := 'Laser';
v_data      := TO_DATE('25-APR-20');
verifica_aparatura_veche(v_denumire, v_data);
END;
/
```

Script Output x

Task completed in 0.077 seconds

Procedure VERIFICA_APARATURA_VECHE compiled

OK: Tehnicianul Kemp Darien (programator) foloseste laser pentru piesa 0 noapte furtunoasa in data de 25-04-2020

PL/SQL procedure successfully completed.

```
DECLARE
v_denumire aparatura.denumire%TYPE;
v_data      reprezentatie.data_org%TYPE;
BEGIN

-- TOO MANY ROWS
v_denumire := 'Microfon';
v_data      := TO_DATE('24-MAY-21');
verifica_aparatura_veche(v_denumire, v_data);

--- NO DATA FOUND
--v_denumire := 'Microfon';
--v_data      := TO_DATE('24-MAY-23');
--verifica_aparatura_veche(v_denumire, v_data);
```

Script Output x

Task completed in 0.083 seconds

END;

Error report -

ORA-20005: Dispozitivul este folosit de prea multe ori!

ORA-06512: at "MATEI_DORIAN.VERIFICA_APARATURA_VECHE", line 50

ORA-06512: at line 9

```
--- TOO MANY ROWS
--v_denumire := 'Microfon';
--v_data      := TO_DATE('24-MAY-21');
--verifica_aparatura_veche(v_denumire, v_data);

--- NO DATA FOUND
v_denumire := 'Microfon';
v_data      := TO_DATE('24-MAY-23');
verifica_aparatura_veche(v_denumire, v_data);

--- NO DEVICE
--v_denumire := 'Minge';
--v_data      := TO_DATE('24-MAY-23');
```

Script Output x

Task completed in 0.096 seconds

END;

Error report -

ORA-20004: Dispozitivul nu este folosit in aceasta zi

ORA-06512: at "MATEI_DORIAN.VERIFICA_APARATURA_VECHE", line 48

ORA-06512: at line 14

```
--- NO DATA FOUND
--v_denumire := 'Microfon';
--v_data      := TO_DATE('24-MAY-23');
--verifica_aparatura_veche(v_denumire, v_data);

--- NO DEVICE
v_denumire := 'Minge';
v_data      := TO_DATE('24-MAY-23');
verifica_aparatura_veche(v_denumire, v_data);

--v_denumire := 'Laser';
--v_data      := TO_DATE('25-APR-20');
--verifica_aparatura_veche(v_denumire, v_data);
END;
```

Script Output x

Task completed in 0.075 seconds

END;

Error report -

ORA-20006: Dispozitiv inexistent

ORA-06512: at "MATEI_DORIAN.VERIFICA_APARATURA_VECHE", line 52

ORA-06512: at line 19

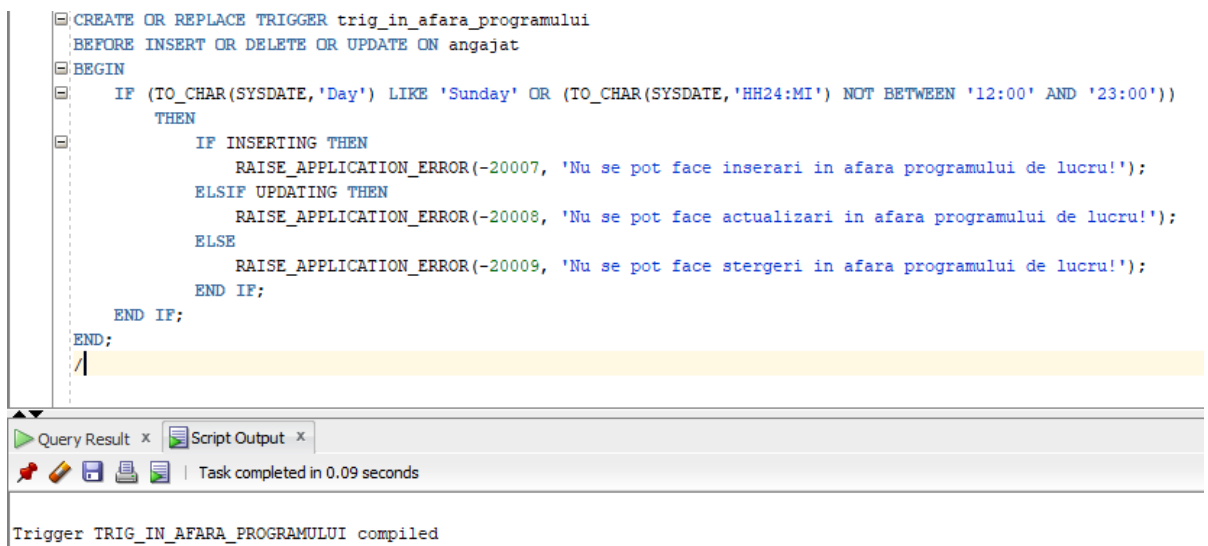
Observații: Procedura folosește entitățile FOLOSEȘTE, APARATURĂ, ANGAJAT, TEHNICIAN, SALA, REPREZENTAȚIE și PIESĂ. Am apelat procedura astfel încât să ilustrez cazul general, cazul cu TOO MANY ROWS, cu NO DATA FOUND și cu NO DEVICE (atunci când încercăm să căutăm un dispozitiv care nu există în baza de date).

TRIGGER LMD LA NIVEL DE COMANDĂ:

10. Datele personale ale unui angajat pot fi inserate/actualizate/șterse doar în timpul programului de lucru deoarece trebuie verificate de către un manager. Creați un trigger care să fie declanșat atunci când se încearcă modificarea tabelii în afara programului de lucru. Se știe că teatrul este deschis de luni până sâmbătă, între orele 12:00 și 23:00.

```
CREATE OR REPLACE TRIGGER trig_in_afara_programului
BEFORE INSERT OR DELETE OR UPDATE ON angajat
BEGIN
    IF (TO_CHAR(SYSDATE,'Day') LIKE 'Sunday' OR (TO_CHAR(SYSDATE,'HH24:MI') NOT BETWEEN
'12:00' AND '23:00'))
    THEN
        IF INSERTING THEN
            RAISE_APPLICATION_ERROR(-20007, 'Nu se pot face inserari in afara programului de lucru!');
        ELSIF UPDATING THEN
            RAISE_APPLICATION_ERROR(-20008, 'Nu se pot face actualizari in afara programului de lucru!');
        ELSE
            RAISE_APPLICATION_ERROR(-20009, 'Nu se pot face stingeri in afara programului de lucru!');
        END IF;
    END IF;
END;
```

/



```
CREATE OR REPLACE TRIGGER trig_in_afara_programului
BEFORE INSERT OR DELETE OR UPDATE ON angajat
BEGIN
    IF (TO_CHAR(SYSDATE,'Day') LIKE 'Sunday' OR (TO_CHAR(SYSDATE,'HH24:MI') NOT BETWEEN '12:00' AND '23:00'))
    THEN
        IF INSERTING THEN
            RAISE_APPLICATION_ERROR(-20007, 'Nu se pot face inserari in afara programului de lucru!');
        ELSIF UPDATING THEN
            RAISE_APPLICATION_ERROR(-20008, 'Nu se pot face actualizari in afara programului de lucru!');
        ELSE
            RAISE_APPLICATION_ERROR(-20009, 'Nu se pot face stingeri in afara programului de lucru!');
        END IF;
    END IF;
END;
```

Query Result x Script Output x

Task completed in 0.09 seconds

Trigger TRIG_IN_AFARA_PROGRAMULUI compiled

Declanșare:

```
UPDATE angajat
SET numar_telefon = '0700000000'
WHERE id_angajat = 7;
```

Script Output x

Task completed in 0.079 seconds

Trigger TRIG_IN_AFARA_PROGRAMULUI compiled

1 row updated.

10:18 PM
1/4/2022

```
UPDATE angajat
SET numar_telefon = '0700000000'
WHERE id_angajat = 7;
ROLLBACK;
```

Script Output x Query Result x

Task completed in 0.039 seconds

Error starting at line : 506 in command -
UPDATE angajat
SET numar_telefon = '0700000000'
WHERE id_angajat = 7
Error report -
ORA-20008: Nu se pot face actualizari in afara programului de lucru!
ORA-06512: at "MATEI_DORIAN.TRIG_IN_AFARA_PROGRAMULUI", line 7
ORA-04088: error during execution of trigger 'MATEI_DORIAN.TRIG_IN_AFARA_PROGRAMULUI'

11:16 PM
1/5/2022



TRIGGER LMD LA NIVEL DE LINIE:

11. La teatru, salariul fiecărui angajat este cuprins între salariul minim de 10000 de lei și salariul managerului angajatului respectiv. Scrieți un trigger care să se asigure că salariul unui angajat rămâne mereu între aceste limite. Administratorul teatrului nu este afectat de această regulă.

```
CREATE OR REPLACE TRIGGER trig_limita_salariu
BEFORE INSERT OR UPDATE ON angajat FOR EACH ROW
DECLARE
PRAGMA AUTONOMOUS_TRANSACTION;
limita_inf NUMBER := 10000;
limita_sup NUMBER;
v_manager angajat.id_manager%TYPE;
BEGIN
    v_manager := :NEW.id_manager;
    IF v_manager IS NOT NULL THEN --- pentru administratorul teatrului nu se aplica regula
        SELECT salariu
        INTO limita_sup
        FROM angajat
        WHERE id_angajat = v_manager;

        IF :NEW.salariu < limita_inf THEN
            RAISE_APPLICATION_ERROR(-20010, 'Salariul este prea mic!');
        ELSIF :NEW.salariu > limita_sup THEN
            RAISE_APPLICATION_ERROR(-20010, 'Salariul este prea mare!');
        ELSE DBMS_OUTPUT.PUT_LINE('Modificare efectuata cu succes!');
        END IF;
    ELSIF v_manager IS NULL OR v_manager = 1 THEN
        SELECT MAX(salariu)
        INTO limita_inf
        FROM ANGAJAT
```

```

WHERE id_manager = :NEW.id_angajat;

IF :NEW.salariu < limita_inf THEN

    RAISE_APPLICATION_ERROR(-20010, 'Salariul este prea mic!');

END IF;

ELSE DBMS_OUTPUT.PUT_LINE('Modificare efectuata cu succes!');

END IF;

END;

/

```

```

CREATE OR REPLACE TRIGGER trig_limita_salariu
BEFORE INSERT OR UPDATE ON angajat FOR EACH ROW
DECLARE
PRAGMA AUTONOMOUS_TRANSACTION;
limita_inf NUMBER := 10000;
limita_sup NUMBER;
v_manager    angajat.id_manager%TYPE;
BEGIN
    v_manager := :NEW.id_manager;
    IF v_manager IS NOT NULL THEN --- pentru administratorul teatrului nu se aplica regula
        SELECT salariu
        INTO limita_sup
        FROM angajat
        WHERE id_angajat = v_manager;

        IF :NEW.salariu < limita_inf THEN
            RAISE_APPLICATION_ERROR(-20010, 'Salariul este prea mic!');
        ELSIF :NEW.salariu > limita_sup THEN
            RAISE_APPLICATION_ERROR(-20010, 'Salariul este prea mare!');
        ELSE DBMS_OUTPUT.PUT_LINE('Modificare efectuata cu succes!');
        END IF;
    ELSIF v_manager IS NULL OR v_manager = 1 THEN
        SELECT MAX(salariu)
        INTO limita_inf
        FROM ANGAJAT
        WHERE id_manager = :NEW.id_angajat;
        IF :NEW.salariu < limita_inf THEN
            RAISE_APPLICATION_ERROR(-20010, 'Salariul este prea mic!');
        END IF;
        ELSE DBMS_OUTPUT.PUT_LINE('Modificare efectuata cu succes!');
        END IF;
END;

/

```

Query Result x Script Output x

Task completed in 0.081 seconds

Trigger TRIG_LIMITA_SALARIU compiled

Declanșare:

UPDATE angajat

SET salariu = 11000

WHERE id_angajat > 7 AND id_angajat < 12;

UPDATE angajat

SET salariu = 15;

```
UPDATE angajat
SET salariu = 11000
WHERE id_angajat > 7 AND id_angajat < 12;
```

Script Output x Query Result x Query Result 1 x

Task completed in 0.074 seconds

Trigger TRIG_LIMITA_SALARIU compiled

Rollback complete.

Modificare efectuata cu succes!
Modificare efectuata cu succes!
Modificare efectuata cu succes!

3 rows updated.

```
UPDATE angajat
SET salariu = 15;
```

Script Output x Query Result x Query Result 1 x

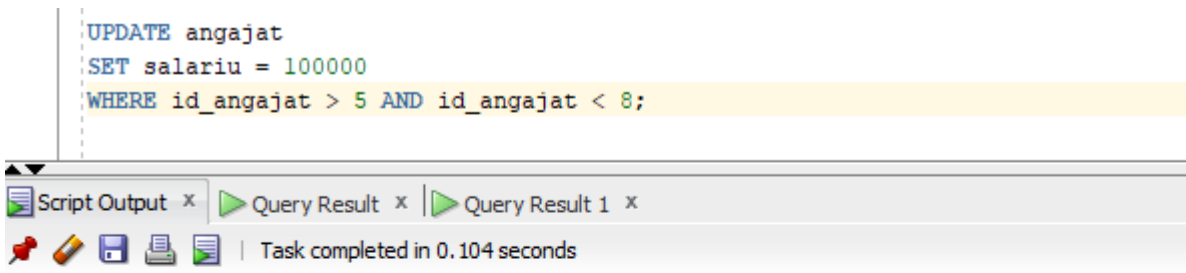
Task completed in 0.048 seconds

Error starting at line : 585 in command -
UPDATE angajat
SET salariu = 15
Error report -
ORA-20010: Salariul este prea mic!
ORA-06512: at "MATEI_DORIAN.TRIG_LIMITA_SALARIU", line 15
ORA-04088: error during execution of trigger 'MATEI_DORIAN.TRIG_LIMITA_SALARIU'

UPDATE angajat

SET salariu = 100000

WHERE id_angajat > 5 AND id_angajat < 8;



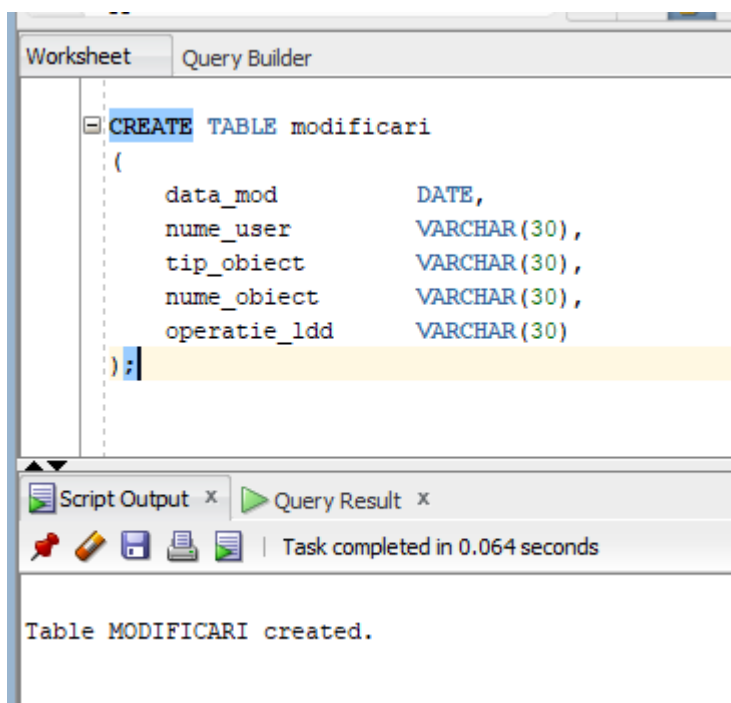
```
Error starting at line : 588 in command -
UPDATE angajat
SET salariu = 100000
WHERE id_angajat > 5 AND id_angajat < 8
Error report -
ORA-20010: Salariul este prea mare!
ORA-06512: at "MATEI_DORIAN.TRIG_LIMITA_SALARIU", line 17
ORA-04088: error during execution of trigger 'MATEI_DORIAN.TRIG_LIMITA_SALARIU'
```

TRIGGER de tip LDD:

12. Toate modificările aduse la baza de date trebuie înregistrate. Creați tabelul MODIFICĂRI care va înregistra detalii despre fiecare comandă LDD rulată. Creați un trigger care să insereze automat date în tabel după fiecare comandă LDD.

CREATE TABLE modificari

```
(
    data_mod      DATE,
    nume_user     VARCHAR(30),
    tip_obiect    VARCHAR(30),
    nume_obiect   VARCHAR(30),
    operatie_ldd  VARCHAR(30)
);
```

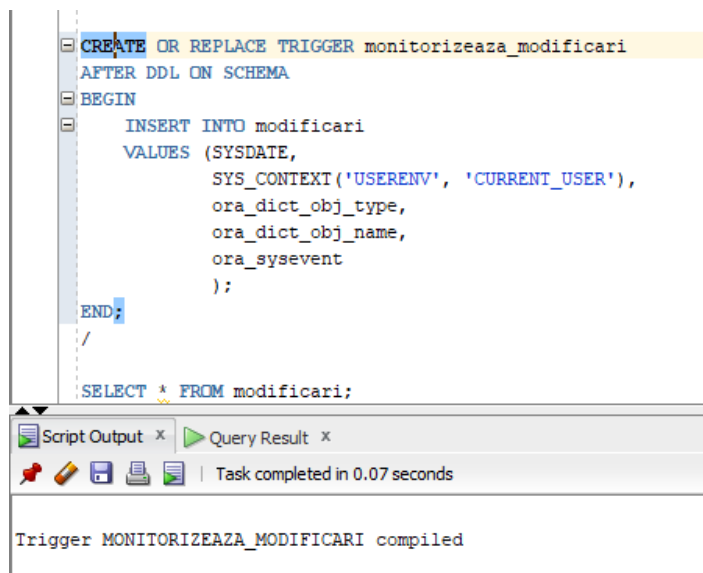



```

CREATE OR REPLACE TRIGGER monitorizeaza_modificari
AFTER DDL ON SCHEMA
BEGIN
    INSERT INTO modificari
    VALUES (SYSDATE,
            SYS_CONTEXT('USERENV', 'CURRENT_USER'),
            ora_dict_obj_type,
            ora_dict_obj_name,
            ora_sysevent
    );
END;
/

```

Triggerul se rulează când apelăm orice comandă de tip LDD. Pentru a vedea efectele triggerului în urma declanșării acestuia putem face un select pe tabela MODIFICĂRI.



The screenshot displays the Oracle SQL Developer environment. The main window shows the following SQL script:

```

CREATE OR REPLACE TRIGGER monitorizeaza_modificari
AFTER DDL ON SCHEMA
BEGIN
    INSERT INTO modificari
    VALUES (SYSDATE,
            SYS_CONTEXT('USERENV', 'CURRENT_USER'),
            ora_dict_obj_type,
            ora_dict_obj_name,
            ora_sysevent
    );
END;
/

SELECT * FROM modificari;

```

Below the script editor, the 'Script Output' tab is active, showing the message: 'Trigger MONITORIZEAZA_MODIFICARI compiled'. The 'Query Result' tab is also visible but empty. A status bar at the bottom indicates 'Task completed in 0.07 seconds'.

Script Output x Query Result x					
SQL All Rows Fetched: 6 in 0.003 seconds					
	DATA_MOD	NUME_USER	TIP_OBIECT	NUME_OBIECT	OPERATIE_LDD
1	04-JAN-22	MATEI_DORIAN	PROCEDURE	INGRIJITORI_ACTIVI	CREATE
2	04-JAN-22	MATEI_DORIAN	PROCEDURE	GENERARE_AFIS	CREATE
3	04-JAN-22	MATEI_DORIAN	FUNCTION	NUMAR_COSTUME	CREATE
4	04-JAN-22	MATEI_DORIAN	PROCEDURE	VERIFICA_APARATURA_VECHE	CREATE
5	04-JAN-22	MATEI_DORIAN	TRIGGER	TRIG_IN_AFARA_PROGRAMULUI	CREATE
6	04-JAN-22	MATEI_DORIAN	TRIGGER	TRIG_VERIFICA_NR_BILETE	CREATE

CREAREA UNUI PACHET DE BAZĂ:

13. Creați un pachet care să conțină toate obiectele create până acum.

```
CREATE OR REPLACE PACKAGE basic_pack
```

```
IS
```

```
    TYPE v_ingrijitori IS VARRAY(100) OF angajat%ROWTYPE;    --- varray
```

```
    TYPE multime_sali IS TABLE OF sala.id_sala%TYPE;        --- nested table
```

```
    TYPE date_utilizare IS RECORD
```

```
    (
```

```
        denumire    aparatura.denumire%TYPE,
```

```
        nume        angajat.nume%TYPE,
```

```
        prenume     angajat.prenume%TYPE,
```

```
        specializare tehnician.specializare%TYPE,
```

```
        titlu       piesa.titlu%TYPE
```

```
    );
```

```
    NO_SHOW    EXCEPTION;
```

```
    BAD_ACTOR_ID EXCEPTION;
```

```
    NO_DEVICE  EXCEPTION;
```

```
    PROCEDURE ingrijitori_activi (x IN NUMBER, d IN DATE DEFAULT SYSDATE);
```

```

PROCEDURE genereare_afis;

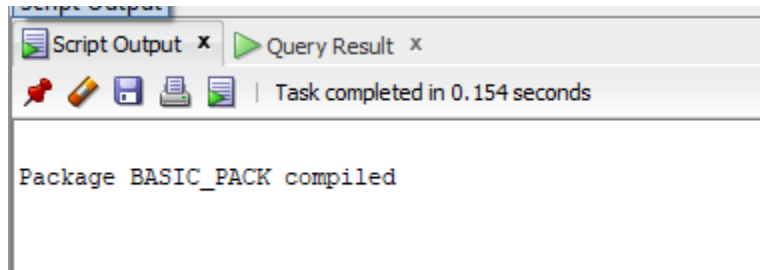
FUNCTION numar_costume(v_actor IN actor.id_actor%TYPE) RETURN NUMBER;

PROCEDURE verifica_aparatura_veche(v_ap IN aparatura.denumire%TYPE, v_data IN
reprezentatie.data_org%TYPE);

END basic_pack;

/

```



```

CREATE OR REPLACE PACKAGE BODY basic_pack
IS
PROCEDURE ingrijitori_activi (x IN NUMBER, d IN DATE DEFAULT SYSDATE)
AS
    v_ing    v_ingrijitori := v_ingrijitori();
    sali     multime_sali := multime_sali();
    sali_ing multime_sali := multime_sali();
    ing_activi v_ingrijitori := v_ingrijitori();
    ct        NUMBER(3) := 1;
    nr_max    NUMBER(3) := 100; --- la teatru nu vor fi angajati mai mult de 100 de ingrijitori o data
BEGIN
    --- selectam toti ingrijitorii si ii salvam in tabelul
    SELECT *
    BULK COLLECT INTO v_ing
    FROM angajat a
    WHERE id_angajat IN (SELECT id_angajat
                        FROM INGRIJITOR);

    --- selectam toate salile ce au fost folosite in data d
    SELECT DISTINCT id_sala
    BULK COLLECT INTO sali

```

```

FROM reprezentatie
WHERE to_char(data_org, 'YYYY-MM-DD') = to_char(d, 'YYYY-MM-DD');

--- daca nu sunt spectacole in ziua respectiva ne oprim
IF sali.COUNT() = 0
THEN
    RAISE NO_SHOW;
END IF;

--- pentru fiecare ingrijitor
FOR ing IN v_ing.FIRST..v_ing.LAST
LOOP
    --- selectam salile in care a facut curat angajatul ing
    SELECT DISTINCT id_sala
    BULK COLLECT INTO sali_ing
    FROM ingrijeste
    WHERE id_angajat = v_ing(ing).id_angajat;

    --- retinem in sali_ing doar salile in care au fost reprezentate specacole in data d
    sali_ing := sali_ing MULTISSET INTERSECT sali;

    --- daca ingrijitorul a curatat mai mult de x sali atunci el este foarte activ
    IF sali_ing.COUNT() >= x
    THEN
        DBMS_OUTPUT.PUT('Ingrijitorul ' || v_ing(ing).nume || ' ' || v_ing(ing).prenume || ' a curatat salile: ');
        FOR sala IN sali_ing.FIRST..sali_ing.LAST
        LOOP
            DBMS_OUTPUT.PUT(sali_ing(sala) || ' ');
        END LOOP;
        DBMS_OUTPUT.PUT_LINE("");

        --- adaugam angajatul activ in vector
        ing_activi.EXTEND();
        ing_activi(ing_activi.LAST) := v_ing(ing);
    END IF;
END LOOP;

```

```

        ct := ct + 1;
    END IF;
END LOOP;
--- toti angajatii activi vor primi o marire de salariu
FORALL ing IN ing_activi.FIRST..ing_activi.LAST
    UPDATE angajat
    SET salariu = salariu + salariu * 0.05
    WHERE id_angajat = ing_activi(ing).id_angajat;

DBMS_OUTPUT.PUT_LINE("");
DBMS_OUTPUT.PUT_LINE('S-au actualizat salariile a ' || ing_activi.COUNT() || ' angajati');
EXCEPTION
WHEN NO_SHOW THEN
    RAISE_APPLICATION_ERROR(-20001, 'Nu exista spectacole in ziua respectiva');
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20000, 'Alta exceptie');
END ingrijitori_activi;

```

```

PROCEDURE genereare_afis

```

```

IS

```

```

TYPE ref_cursor IS REF CURSOR;
CURSOR c_trupa IS
SELECT nume,
    CURSOR (
        WITH nr_reprezentatii AS
        (
            SELECT CASE
                WHEN a.id_actor IS NOT NULL THEN
                    COUNT(*)
                ELSE
                    0
            END nr, a.id_actor

```

```

        FROM actor a
        FULL OUTER JOIN joaca j ON a.id_actor = j.id_actor
        GROUP BY a.id_actor
    )
    SELECT nume, prenume
    FROM actor a
    JOIN nr_reprezentatii n ON a.id_actor = n.id_actor
    WHERE a.id_trupa = t.id_trupa
    ORDER BY nr DESC
)
FROM trupa t
ORDER BY nume;

```

```

v_cursor    ref_cursor;
nume_trupa   trupa.nume%TYPE;
nume_actor   actor.nume%TYPE;
prenume_actor actor.prenume%TYPE;
BEGIN
    OPEN c_trupa;
    LOOP
        FETCH c_trupa INTO nume_trupa, v_cursor;
        EXIT WHEN c_trupa%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(nume_trupa);
        LOOP
            FETCH v_cursor INTO nume_actor, prenume_actor;
            EXIT WHEN v_cursor%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(' ' || v_cursor%ROWCOUNT || ' ' || nume_actor || ' ' || prenume_actor);
        END LOOP;
        DBMS_OUTPUT.PUT_LINE("");
    END LOOP;
    CLOSE c_trupa;
END genereare_afis;

```

```
FUNCTION numar_costume(v_actor IN actor.id_actor%TYPE)
```

```
RETURN NUMBER IS
```

```
actor_gasit actor.id_actor%TYPE := 0;
```

```
raspuns NUMBER;
```

```
BEGIN
```

```
-- verificam daca actorul exista
```

```
SELECT COUNT(id_actor)
```

```
INTO actor_gasit
```

```
FROM actor
```

```
WHERE id_actor = v_actor;
```

```
IF(actor_gasit = 0) THEN
```

```
    RAISE BAD_ACTOR_ID;
```

```
END IF;
```

```
SELECT COUNT(*)
```

```
INTO raspuns
```

```
FROM (SELECT id_costum, id_actor
```

```
    FROM (SELECT id_actor, id_trupa, marime
```

```
        FROM actor
```

```
        WHERE id_actor = v_actor
```

```
    ) a
```

```
    JOIN trupa t ON a.id_trupa = t.id_trupa
```

```
    JOIN costum c ON c.id_trupa = t.id_trupa
```

```
    WHERE c.marime = a.marime
```

```
    )
```

```
GROUP BY id_actor;
```

```
RETURN raspuns;
```

```
EXCEPTION
```

```
WHEN NO_DATA_FOUND THEN
```

```

        RAISE_APPLICATION_ERROR(-20003, 'Nu exista costume pentru acest actor');
    WHEN BAD_ACTOR_ID THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nu exista actorul cu id-ul dat');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20000, 'Eroare necunoscuta');
END numar_costume;

```

```

-----
PROCEDURE verifica_aparatura_veche(v_ap IN aparatura.denumire%TYPE, v_data IN
reprezentatie.data_org%TYPE)

```

```

IS

```

```

ap_gasit    NUMBER;
v_info      date_utilizare;
NO_DEVICE    EXCEPTION;

```

```

BEGIN

```

```

    SELECT COUNT(id_aparatura)
    INTO ap_gasit
    FROM aparatura
    WHERE denumire LIKE v_ap;

```

```

    IF(ap_gasit = 0) THEN
        RAISE NO_DEVICE;
    END IF;

```

```

    SELECT a.denumire, ang.nume, ang.prenume, t.specializare, p.titlu
    INTO v_info
    FROM (SELECT id_aparatura, id_angajat, id_sala FROM foloseste) f
    JOIN (SELECT id_aparatura, denumire FROM aparatura) a ON a.id_aparatura = f.id_aparatura
    JOIN (SELECT id_angajat, nume, prenume FROM angajat) ang ON f.id_angajat = ang.id_angajat
    JOIN (SELECT id_angajat, specializare FROM tehnician) t ON t.id_angajat = ang.id_angajat
    JOIN (SELECT id_sala FROM sala) s ON s.id_sala = f.id_sala
    JOIN (SELECT data_org, id_sala, id_piesa
        FROM reprezentatie
        WHERE to_char(data_org, 'DD-MM-YYYY') = to_char(v_data, 'DD-MM-YYYY'))

```



```

    ) r ON r.id_sala = s.id_sala -- where
JOIN (SELECT id_piesa, titlu FROM piesa) p ON p.id_piesa = r.id_piesa
WHERE a.denumire LIKE v_ap;

DBMS_OUTPUT.PUT_LINE('OK: Tehnicianul ' || v_info.nume || ' ' || v_info.prenume || ' (' || v_info.specializare ||
') ' ||
    'foloseste ' || LOWER(v_info.denumire) || ' pentru piesa ' || v_info.titlu || ' in data de '
    || TO_CHAR(v_data, 'DD-MM-YYYY'));

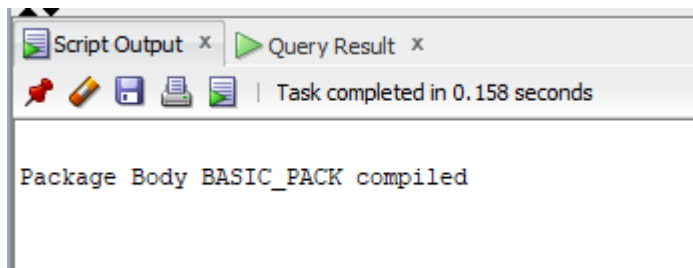
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Dispozitivul nu este folosit in aceasta zi');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20005, 'Dispozitivul este folosit de prea multe ori!');
    WHEN NO_DEVICE THEN
        RAISE_APPLICATION_ERROR(-20006, 'Dispozitiv inexistent');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20000, 'Eroare necunoscuta');

END verifica_aparatura_veche;

END basic_pack;

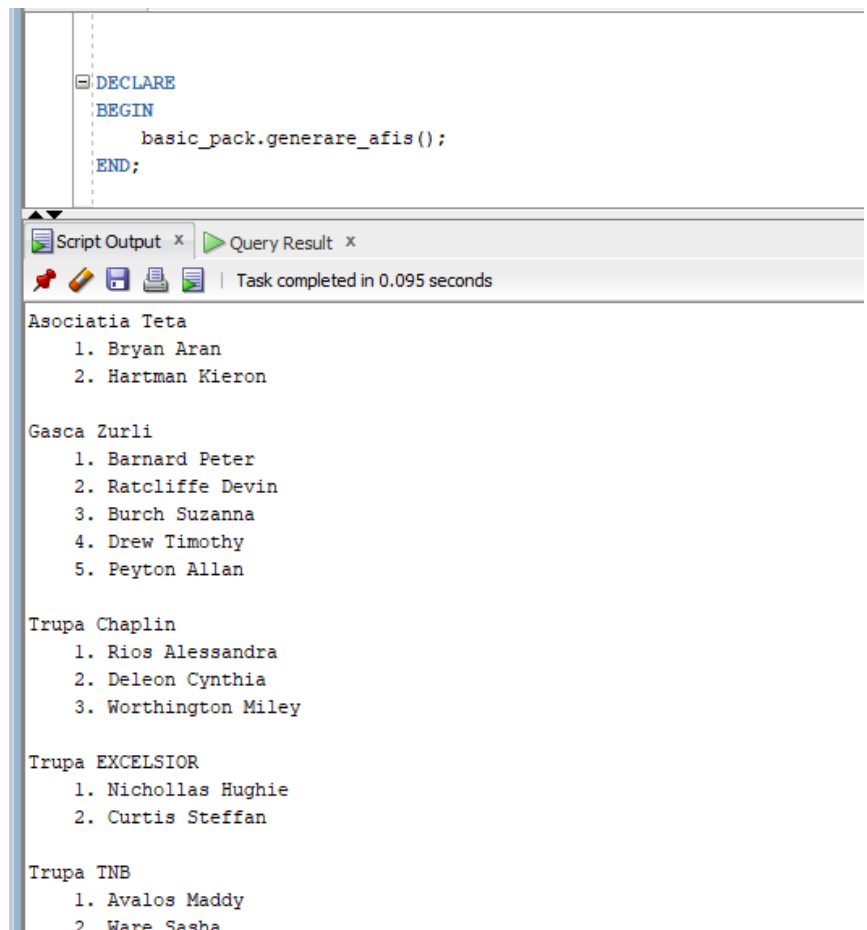
/

```



Folosirea pachetului:

```
DECLARE
BEGIN
    basic_pack.generare_afis();
END;
```



The screenshot shows a SQL IDE interface. The top pane displays a PL/SQL script: `DECLARE`, `BEGIN`, `basic_pack.generare_afis();`, and `END;`. The bottom pane shows the 'Query Result' tab with the output of the script. The output lists five categories, each with a list of names: 'Asociatia Teta' (Bryan Aran, Hartman Kieron), 'Gasca Zurli' (Barnard Peter, Ratcliffe Devin, Burch Suzanna, Drew Timothy, Peyton Allan), 'Trupa Chaplin' (Rios Alessandra, Deleon Cynthia, Worthington Miley), 'Trupa EXCELSIOR' (Nichollas Hughie, Curtis Steffan), and 'Trupa TNB' (Avalos Maddy, Ware Sasha). A status bar at the top of the results pane indicates 'Task completed in 0.095 seconds'.

```
DECLARE
BEGIN
    basic_pack.generare_afis();
END;
```

Script Output x Query Result x
Task completed in 0.095 seconds

Asociatia Teta

1. Bryan Aran
2. Hartman Kieron

Gasca Zurli

1. Barnard Peter
2. Ratcliffe Devin
3. Burch Suzanna
4. Drew Timothy
5. Peyton Allan

Trupa Chaplin

1. Rios Alessandra
2. Deleon Cynthia
3. Worthington Miley

Trupa EXCELSIOR

1. Nichollas Hughie
2. Curtis Steffan

Trupa TNB

1. Avalos Maddy
2. Ware Sasha

CREAREA UNUI PACHET PENTRU UN FLUX DE ACȚIUNI:

14. 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).

Vom defini pachetul `tickets_pack` care permite utilizatorului să facă anumite operații frecvent întâlnite în gestionarea vânzării biletelor precum: printarea unui bilet, rezervarea unui sau a mai multor locuri, câștigul pe care l-a adus o anumită reprezentație etc.

Pachetul conține:

1. Tipuri de date:

- tabel_casieri: tabel indexat folosit ca un hash map pentru a memora câte bilete a vândut fiecare casier;
- cap_sali: un record care reține id-ul unei săli și capacitatea acesteia;
- vector_sali: un vector de tip cap_sali folosit pentru a memora capacitatea fiecărei săli;
- date_bilet: un record care reține toate datele necesare printării unui bilet

2. Variabile:

- LOCURI_DEPASITE: excepție ce este aruncată atunci când se încearcă rezervarea unui număr prea mare de bilete.
- v_sali: vector de tip vector_sali;
- tab_casieri: tabel indexat de tip tabel_casieri.

3. Cursoare:

- c_casieri: cursor care returnează date despre fiecare casier
- c_spectacole: cursor care returnează date despre fiecare reprezentatie ale unei anumite piese;
- c_bilete: cursor care returnează date despre toate biletele vândute la o anumită reprezentatie.

4. Funcții și proceduri:

- printează_bilet: procedură care primește id-ul unui bilet și afișează toate informațiile ce trebuie să apară pe bilet;
- castig_rep: funcție care primește id-ul unei reprezentatii și returnează câți bani s-au strâns din vânzarea de bilete pentru reprezentatia respectivă;
- rezervare_bilet: procedură care inserează în tabelul BILET, biletele cumpărate de un spectator care dorește să facă o rezervare. Dacă sala devine plină, se va arunca o excepție. Locul și rândul vor fi calculate automat;
- initializare_sali: procedură care inițializează vectorul v_sali;
- initializare_tab_casieri: funcție folosită pentru a inițializa tab_casieri;

- bilete_vandute_la_rep: funcție ce returnează numărul de bilete vândute la o reprezentație;
- activitate_angajati: procedură ce afișează câte bilete a vândut fiecare casier;
- bilete_vandute_de_casier: funcție ce returnează numărul de bilete vândute de un casier;
- castig_zi: funcție ce returnează suma de bani câștigată într-o anumită zi.

```

CREATE OR REPLACE PACKAGE tickets_pack
IS

--- hash map cu cate bilete a vandut fiecare casier
TYPE tabel_casieri IS TABLE OF NUMBER INDEX BY PLS_INTEGER; --- key: id_casier / value: nr bilete
tab_casieri tabel_casieri;

--- vector cu toate capacitatea salilor
TYPE cap_sali IS RECORD
(
    id_sala sala.id_sala%TYPE,
    cap    sala.capacitate%TYPE
);
TYPE vector_sali IS VARRAY(5) OF cap_sali;
v_sali vector_sali := vector_sali();
LOCURI_DEPASITE EXCEPTION;

--- record bilet
TYPE date_bilet IS RECORD
(
    titlu_piesa piesa.titlu%TYPE,
    nume_trupa  trupa.nume%TYPE,
    nume_sala   sala.nume%TYPE,
    loc        bilet.loc%TYPE,
    rand       bilet.rand%TYPE,
    data_org   reprezentatie.data_org%TYPE,
    ora        reprezentatie.ora%TYPE

```

);

--- cursor cu casierii

CURSOR c_casieri RETURN angajat%ROWTYPE

IS

SELECT *

FROM angajat

WHERE id_angajat IN (SELECT id_angajat

FROM CASIER);

--- cursor cu spectacole

CURSOR c_spectacole(v_piesa piesa.id_piesa%TYPE) RETURN reprezentatie%ROWTYPE

IS

SELECT *

FROM reprezentatie

WHERE id_piesa = v_piesa;

--- cursor cu biletele

CURSOR c_bilete(v_rep reprezentatie.id_reprezentatie%TYPE) RETURN bilet%ROWTYPE

IS

SELECT *

FROM bilet

WHERE id_reprezentatie = v_rep;

--- printare bilet

PROCEDURE printeaza_bilet(v_bilet bilet.id_bilet%TYPE);

--- cati bani a facut o reprezentatie

FUNCTION castig_rep(v_rep reprezentatie.id_reprezentatie%TYPE) RETURN NUMBER;

--- rezervare bilet

PROCEDURE initializeaza_sali;

FUNCTION bilete_vandute_la_rep(v_rep reprezentatie.id_reprezentatie%TYPE) RETURN NUMBER;

PROCEDURE rezerva_bilet(v_casier angajat.id_angajat%TYPE,

v_rep reprezentatie.id_reprezentatie%TYPE,

```

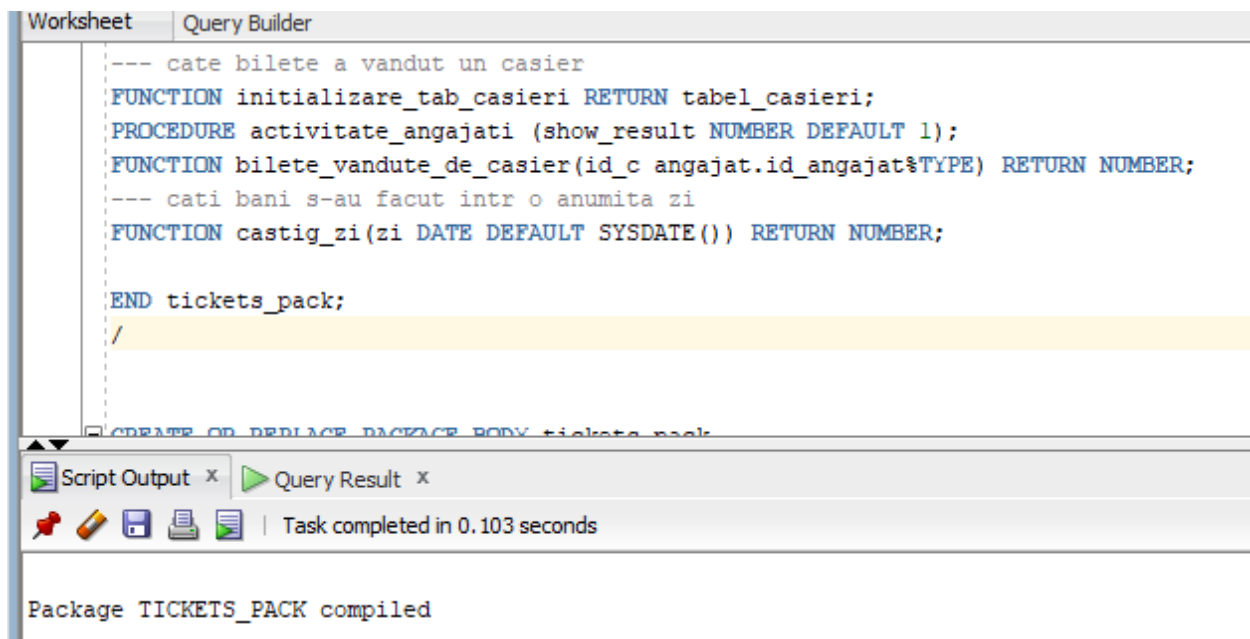
        v_spectator spectator.id_spectator%TYPE DEFAULT NULL,
        nr_locuri NUMBER DEFAULT 1);

--- cate bilete a vandut un casier
FUNCTION initializare_tab_casieri RETURN tabel_casieri;
PROCEDURE activitate_angajati (show_result NUMBER DEFAULT 1);
FUNCTION bilete_vandute_de_casier(id_c angajat.id_angajat%TYPE) RETURN NUMBER;
--- cati bani s-au facut intr o anumita zi
FUNCTION castig_zi(zi DATE DEFAULT SYSDATE()) RETURN NUMBER;

END tickets_pack;

/

```



```

CREATE OR REPLACE PACKAGE BODY tickets_pack
IS

```

```

PROCEDURE printeaza_bilet(v_bilet bilet.id_bilet%TYPE) IS
d_bilet date_bilet;
prescurtare VARCHAR(15);
BEGIN

```

```

SELECT p.titlu, t.num, s.num, b.loc, b.rand, r.data_org, r.ora
INTO d_bilet
FROM (SELECT * FROM bilet WHERE id_bilet = v_bilet) b
JOIN reprezentatie r ON b.id_reprezentatie = r.id_reprezentatie
JOIN piesa p ON p.id_piesa = r.id_piesa
JOIN joaca j ON j.id_piesa = p.id_piesa
JOIN actor a ON a.id_actor = j.id_actor
JOIN trupa t ON t.id_trupa = a.id_trupa
JOIN sala s ON r.id_sala = s.id_sala
WHERE ROWNUM = 1;

```

```

SELECT REGEXP_REPLACE(UPPER(d_bilet.num_trupa), '[A,E,I,O,U]', '')
INTO prescurtare
FROM dual;

```

```

DBMS_OUTPUT.PUT_LINE('Biletul dumneavoastra:');
DBMS_OUTPUT.PUT_LINE("");
DBMS_OUTPUT.PUT_LINE('Piesa: ' || d_bilet.titlu_piesa);
DBMS_OUTPUT.PUT_LINE('Jucata de: ' || prescurtare);
DBMS_OUTPUT.PUT_LINE('Sala: ' || d_bilet.num_sala);
DBMS_OUTPUT.PUT_LINE('Rand: ' || d_bilet.rand);
DBMS_OUTPUT.PUT_LINE('Loc: ' || d_bilet.loc);
DBMS_OUTPUT.PUT_LINE('Data: ' || TO_CHAR(d_bilet.data_org, 'DD-MM'));
DBMS_OUTPUT.PUT_LINE('Ora: ' || d_bilet.ora);
END printeaza_bilet;

```

```

FUNCTION castig_rep(v_rep reprezentatie.id_reprezentatie%TYPE) RETURN NUMBER IS
nr_bilete    NUMBER;
cost_piesa   piesa.pret%TYPE;
BEGIN

```

```

SELECT COUNT(*)
INTO nr_bilete
FROM bilet
WHERE id_reprezentatie = v_rep;

SELECT pret
INTO cost_piesa
FROM piesa p
JOIN reprezentatie r ON r.id_piesa = p.id_piesa
WHERE r.id_reprezentatie = v_rep;

RETURN nr_bilete * cost_piesa;
END castig_rep;

```

```

-----

PROCEDURE initializeaza_sali IS
BEGIN
    SELECT id_sala, capacitate
    BULK COLLECT INTO v_sali
    FROM sala;
END initializeaza_sali;

```

```

FUNCTION bilete_vandute_la_rep(v_rep reprezentatie.id_reprezentatie%TYPE) RETURN NUMBER IS
nr_bilete NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO nr_bilete
    FROM bilet
    WHERE id_reprezentatie = v_rep;

    RETURN NVL(nr_bilete, 0);
END bilete_vandute_la_rep;

```



```
PROCEDURE rezerva_bilet(v_casier angajat.id_angajat%TYPE,  
                        v_rep    reprezentatie.id_reprezentatie%TYPE,  
                        v_spectator spectator.id_spectator%TYPE DEFAULT NULL,  
                        nr_locuri NUMBER DEFAULT 1)
```

```
IS
```

```
v_rand    NUMBER := 1;  
v_loc     NUMBER := 1;  
nr_bilete NUMBER;  
v_sala    sala.id_sala%TYPE;  
cap       sala.capacitate%TYPE;
```

```
BEGIN
```

```
SELECT COUNT(*)  
INTO nr_bilete  
FROM bilet  
WHERE id_reprezentatie = v_rep;
```

```
IF nr_bilete != 0 THEN  
    SELECT loc, rand  
    INTO v_loc, v_rand  
    FROM bilet  
    WHERE ROWNUM = 1 AND id_reprezentatie = v_rep  
    ORDER BY rand DESC, loc DESC;  
END IF;
```

```
DBMS_OUTPUT.PUT_LINE(v_rand || ' ' || v_loc);
```

```
IF v_sali.COUNT = 0 THEN  
    initializeaza_sali();  
END IF;
```

```

SELECT s.id_sala
INTO v_sala
FROM sala s
JOIN reprezentatie r ON r.id_sala = s.id_sala
WHERE r.id_reprezentatie = v_rep;

FOR i IN v_sali.FIRST..v_sali.LAST
LOOP
    IF v_sala = v_sali(i).id_sala THEN
        cap := v_sali(i).cap;
    END IF;
END LOOP;

nr_bilete := bilete_vandute_la_rep(v_rep);
DBMS_OUTPUT.PUT_LINE(nr_bilete || ' ' || nr_locuri || ' ' || cap);
IF nr_bilete + nr_locuri > cap THEN
    RAISE LOCURI_DEPASITE;
END IF;

FOR i IN 1..nr_locuri
LOOP
    v_loc := v_loc + 1;
    IF v_loc > 30 THEN
        v_rand := v_rand + 1;
        v_loc := 1;
    END IF;

    INSERT INTO bilet
    VALUES(pk_bilet.NEXTVAL, v_casier, v_rep, v_spectator, v_rand, v_loc);
END LOOP;

DBMS_OUTPUT.PUT_LINE('REZERVARE FACCUA CU SUCCES');

```

EXCEPTION

WHEN LOCURI_DEPASITE THEN

RAISE_APPLICATION_ERROR(-20013, 'Nu exista locuri suficiente');

END rezerva_bilet;

--- procedura ce actualizeaza hash map ul tab_casieri si afiseaza cate bilete a vandut fiecare angajat

FUNCTION initializare_tab_casieri RETURN tabel_casieri IS

tab_c tabel_casieri;

BEGIN

FOR c in c_casieri

LOOP

tab_c(c.id_angajat) := 0;

END LOOP;

RETURN tab_c;

END initializare_tab_casieri;

PROCEDURE activitate_angajati(show_result NUMBER DEFAULT 1) IS

nr_bilete NUMBER := 0;

BEGIN

IF tab_casieri.COUNT() = 0 THEN

tab_casieri := initializare_tab_casieri();

END IF;

FOR c IN c_casieri

LOOP

SELECT COUNT(*)

INTO nr_bilete

FROM bilet

GROUP BY id_angajat

```

        HAVING id_angajat = c.id_angajat;

        tab_casieri(c.id_angajat) := nr_bilete;
        IF show_result != 0 THEN
            DBMS_OUTPUT.PUT_LINE('Casierul ' || c.nume || ' ' || c.prenume || ' a vandut ' || nr_bilete || ' bilete');
        END IF;
    END LOOP;

END activitate_angajati;

FUNCTION bilete_vandute_de_casier(id_c angajat.id_angajat%TYPE) RETURN NUMBER IS
BEGIN
    IF tab_casieri.COUNT() = 0 THEN
        activitate_angajati(0);
    END IF;

    RETURN tab_casieri(id_c);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20012, 'Nu exista angajatul cu id-ul dat');
END bilete_vandute_de_casier;

```

```

FUNCTION castig_zi(zi DATE DEFAULT SYSDATE()) RETURN NUMBER IS
suma NUMBER := 0;
BEGIN
    SELECT SUM(p.pret)
    INTO suma
    FROM (SELECT id_reprezentatie FROM BILET) b
    JOIN reprezentatie r ON r.id_reprezentatie = b.id_reprezentatie
    JOIN piesa p ON r.id_piesa = p.id_piesa

```

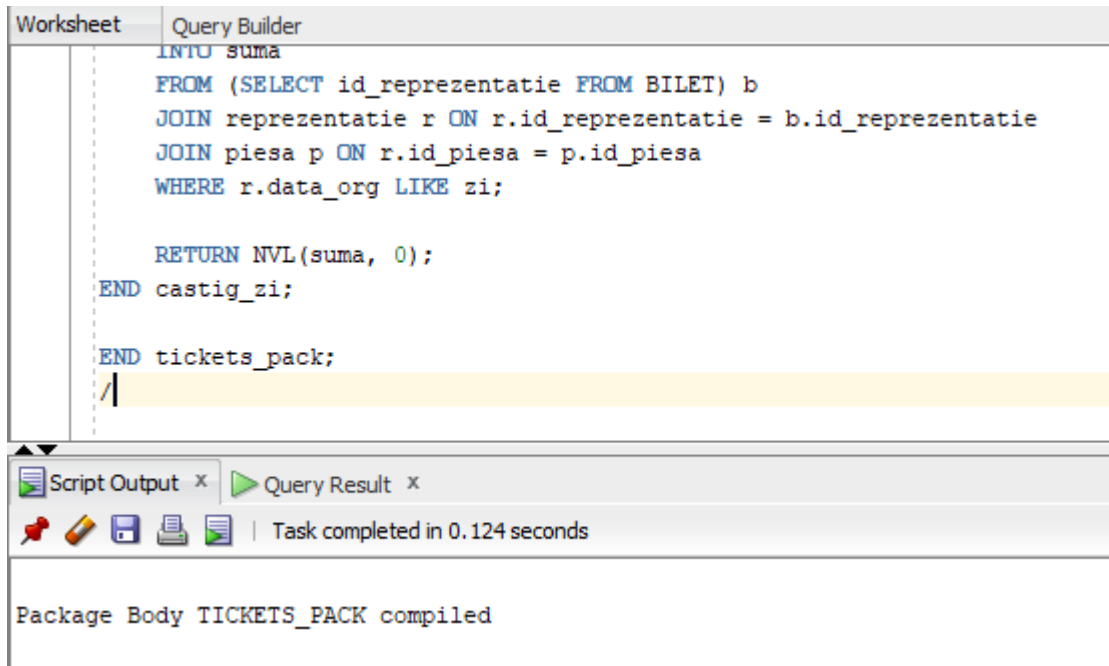
```
WHERE r.data_org LIKE zi;
```

```
RETURN NVL(suma, 0);
```

```
END castig_zi;
```

```
END tickets_pack;
```

```
/
```



Exemplu de apleare:

```
DECLARE
```

```
data_solicitata DATE := TO_DATE('25-MAR-20', 'DD-MON-YY');
```

```
BEGIN
```

```
tickets_pack.activitate_angajati();
```

```
DBMS_OUTPUT.PUT_LINE("");
```

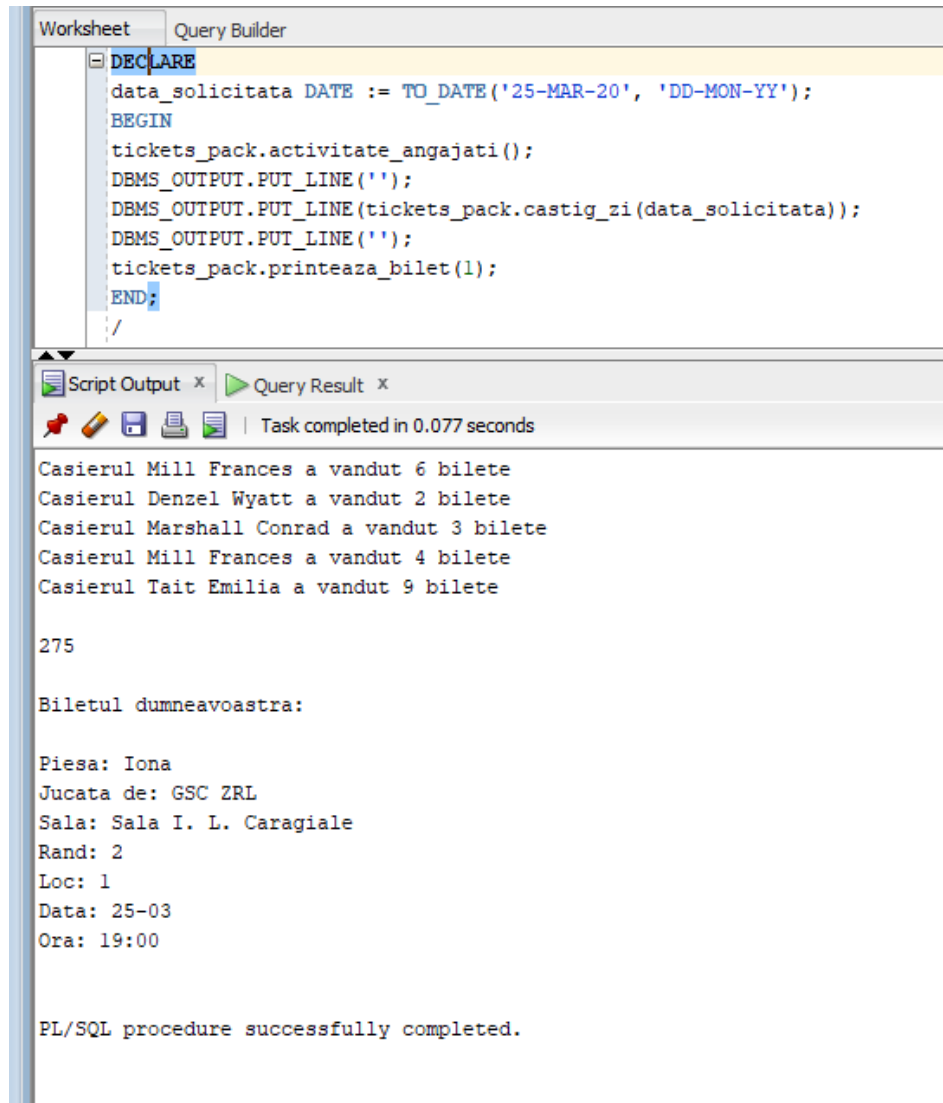
```
DBMS_OUTPUT.PUT_LINE(tickets_pack.castig_zi(data_solicitata));
```

```
DBMS_OUTPUT.PUT_LINE("");
```

```
tickets_pack.printeaza_bilet(1);
```

```
END;
```

```
/
```



The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Query Builder', contains a PL/SQL script. The script declares a date variable, calls a package procedure, and outputs several lines of text. The bottom pane, titled 'Script Output', shows the execution results, including a list of ticket purchases, a summary of the ticket, and a confirmation message.

```
Worksheet Query Builder
DECLARE
data_solicitata DATE := TO_DATE('25-MAR-20', 'DD-MON-YY');
BEGIN
tickets_pack.activitate_angajati();
DBMS_OUTPUT.PUT_LINE('');
DBMS_OUTPUT.PUT_LINE(tickets_pack.castig_zi(data_solicitata));
DBMS_OUTPUT.PUT_LINE('');
tickets_pack.printeaza_bilet(1);
END;
/
```

Script Output x Query Result x
Task completed in 0.077 seconds

```
Casierul Mill Frances a vandut 6 bilete
Casierul Denzel Wyatt a vandut 2 bilete
Casierul Marshall Conrad a vandut 3 bilete
Casierul Mill Frances a vandut 4 bilete
Casierul Tait Emilia a vandut 9 bilete

275

Biletul dumneavoastra:

Piesa: Iona
Jucata de: GSC ZRL
Sala: Sala I. L. Caragiale
Rand: 2
Loc: 1
Data: 25-03
Ora: 19:00

PL/SQL procedure successfully completed.
```