

GESTIUNEA UNUI CINEMATOGRAF

Ioana Potlog

Proiect Sisteme de Gestiune a Bazelor de Date



1. Prezentare pe scurt a bazei de date:

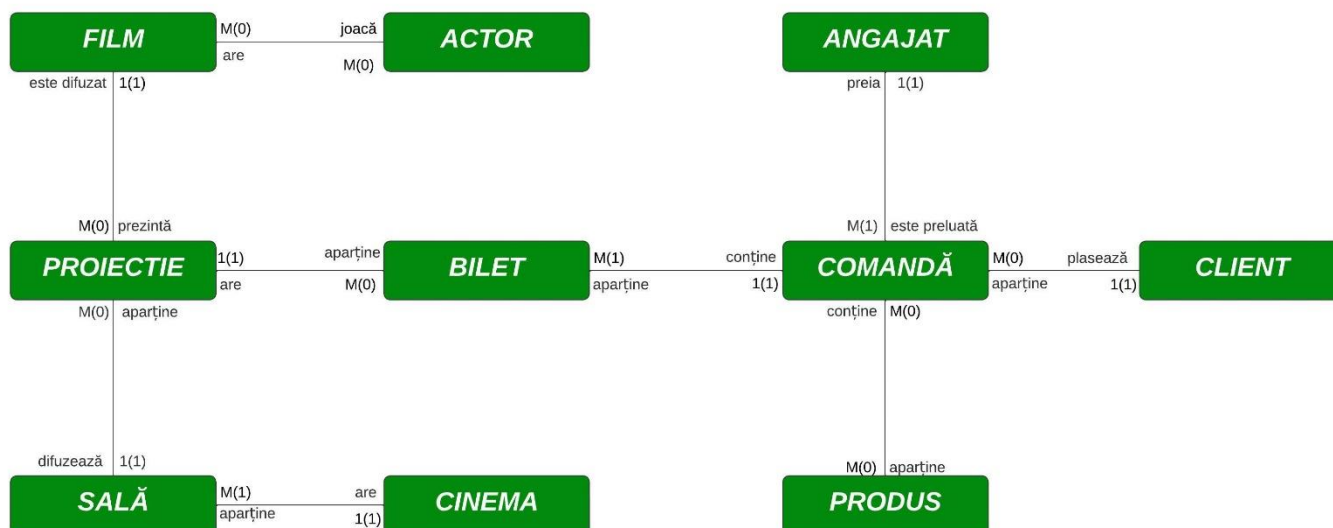
Descrierea modelului real: Tema pe care am ales-o pentru proiect este Cinematograful „The Retro Cinema”, un cinematograful unde se pot viziona filme din perioada anilor '80.

Prin intermediul bazei de date, cinematograful stochează, organizează și gestionează eficient date legate de filme, săli de cinema, comenzi, clienți, angajați și multe altele, astfel oferind o multitudine de avantaje. Printre acestea se regăsesc următoarele: planificarea eficientă a programului de proiecții a filmelor, gestionarea vânzărilor și stocului de bilete, produse alimentare și băuturi, analiza performanței cinematografului, ușurința accesării informațiilor etc.

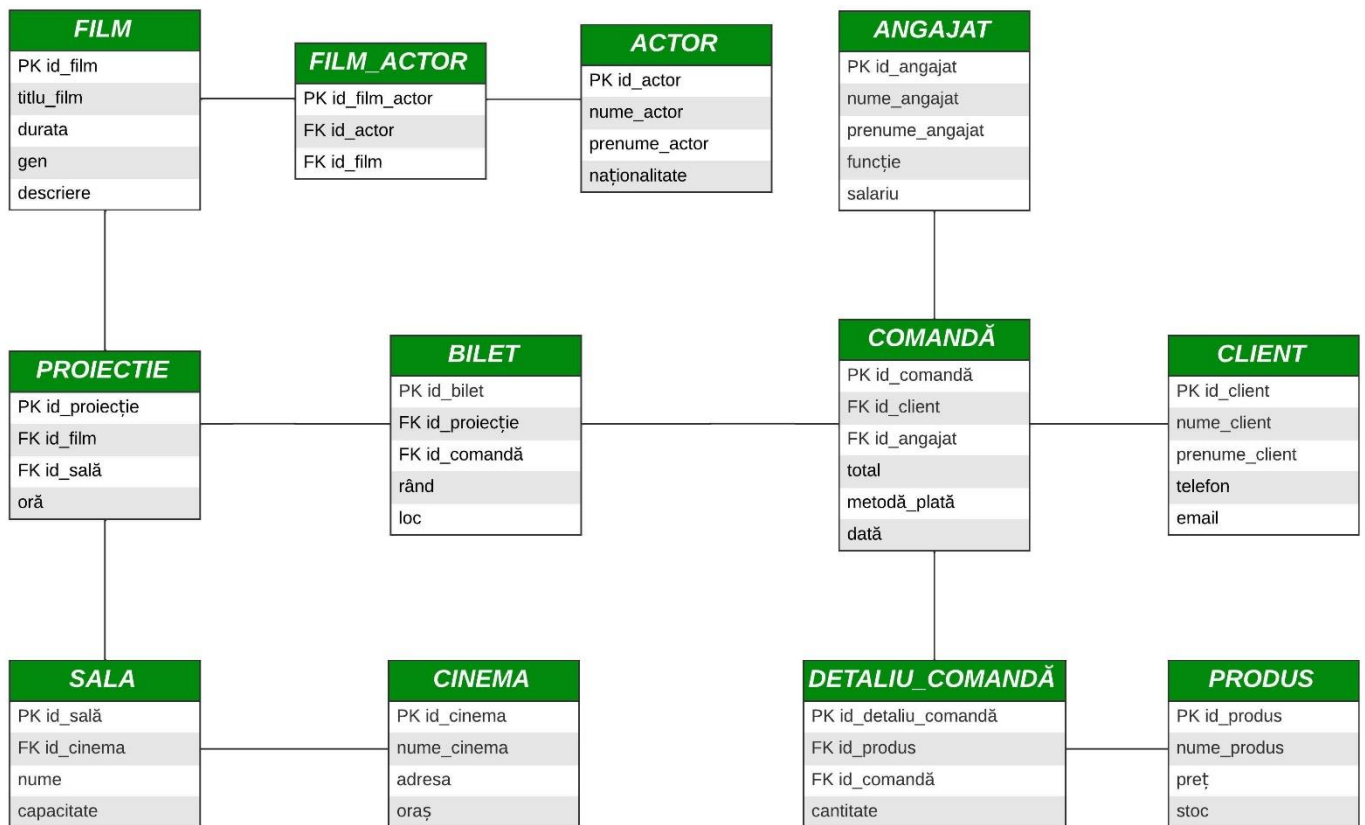
Reguli de funcționare:

- Un film poate fi difuzat în mai multe proiecții la cinema.
- O proiecție poate prezenta un singur film.
- O proiecție se desfășoară într-o anumită sală de cinema.
- O sală de cinema poate difuza mai multe proiecții.
- O sală de cinema este alocată unui cinema anume.
- Un cinema poate avea mai multe săli de cinema disponibile.
- O proiecție poate avea mai multe bilete vândute.
- Un bilet aparține unei singure proiecții.
- O comandă poate conține unul sau mai multe bilete și unul sau mai multe produse achiziționate de un client.
- Un bilet poate fi asociat cu o anumită comandă.
- Un produs poate fi asociat mai multor comenzi.
- Un angajat poate prelua mai multe comenzi.
- O comandă poate fi preluată de un singur angajat.
- O comandă aparține unui client.
- Un client poate să plaseze una sau multe comenzi.
- Un actor poate juca în mai multe filme.
- Un film poate avea mai mulți actori care joacă în acesta.

2. Diagrama Entitate-Relație:



3. Diagrama Conceptuală:



4. Implementarea diagramei în Oracle:

Tabela CINEMA:

```
CREATE TABLE CINEMA
( id_cinema NUMBER(5) CONSTRAINT PKEY_CINEMA PRIMARY KEY,
  nume_cinema VARCHAR(20) CONSTRAINT nume_cinema NOT NULL UNIQUE,
  adresa VARCHAR(50) CONSTRAINT adresa_cinema NOT NULL,
  oras VARCHAR(20) CONSTRAINT oras_cinema NOT NULL
);
```

Table CINEMA created.

	❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS
1	ID_CINEMA	NUMBER (5,0)	No	(null)	1	(null)
2	NUME_CINEMA	VARCHAR2 (20 BYTE)	No	(null)	2	(null)
3	ADRESA	VARCHAR2 (50 BYTE)	No	(null)	3	(null)
4	ORAS	VARCHAR2 (20 BYTE)	No	(null)	4	(null)

Tabela SALA:

```
CREATE TABLE SALA
(
    id_sala NUMBER(5) CONSTRAINT PKEY_SALA PRIMARY KEY,
    id_cinema NUMBER(5), CONSTRAINT fk_sala FOREIGN KEY(id_cinema)
REFERENCES CINEMA(id_cinema),
    nume VARCHAR(50) CONSTRAINT numar_sala NOT NULL,
    capacitate NUMBER(10) CONSTRAINT capacitate_sala NOT NULL
);
```

Table SALA created.

❖	COLUMN_NAME	❖	DATA_TYPE	❖	NULLABLE	DATA_DEFAULT	❖	COLUMN_ID	❖	COMMENTS
1	ID_SALA		NUMBER(5,0)		No	(null)		1		(null)
2	ID_CINEMA		NUMBER(5,0)		Yes	(null)		2		(null)
3	NUME		VARCHAR2(50 BYTE)		No	(null)		3		(null)
4	CAPACITATE		NUMBER(10,0)		No	(null)		4		(null)

Tabela FILM:

```
CREATE TABLE FILM
(
    id_film NUMBER(5) CONSTRAINT PKEY_FILM PRIMARY KEY,
    titlu_film VARCHAR(50) CONSTRAINT titlu_film NOT NULL,
    durata NUMBER(10) CONSTRAINT durata_film NOT NULL,
    gen VARCHAR(20) CONSTRAINT gen_film NOT NULL,
    descriere VARCHAR(100) CONSTRAINT descriere_film NULL
);
```

Table FILM created.

❖	COLUMN_NAME	❖	DATA_TYPE	❖	NULLABLE	DATA_DEFAULT	❖	COLUMN_ID	❖	COMMENTS
1	ID_FILM		NUMBER(5,0)		No	(null)		1		(null)
2	TITLU_FILM		VARCHAR2(50 BYTE)		No	(null)		2		(null)
3	DURATA		NUMBER(10,0)		No	(null)		3		(null)
4	GEN		VARCHAR2(20 BYTE)		No	(null)		4		(null)
5	DESCRIERE		VARCHAR2(100 BYTE)		Yes	(null)		5		(null)

Tabela PROIECTIE:

```
CREATE TABLE PROIECTIE
(
    id_proiectie NUMBER(5) CONSTRAINT PKEY_PROIECTIE PRIMARY KEY,
    id_film NUMBER(5), CONSTRAINT fk_proiectie_film FOREIGN KEY(id_film)
REFERENCES FILM(id_film),
    id_sala NUMBER(5), CONSTRAINT fk_proiectie_sala FOREIGN KEY(id_sala)
REFERENCES SALA(id_sala),
    ora VARCHAR(10) CONSTRAINT ora_proiectie NOT NULL
);
```

Table PROIECTIE created.

❖	COLUMN_NAME	❖	DATA_TYPE	❖	NULLABLE	DATA_DEFAULT	❖	COLUMN_ID	❖	COMMENTS
1	ID_PROIECTIE		NUMBER(5,0)		No	(null)		1		(null)
2	ID_FILM		NUMBER(5,0)		Yes	(null)		2		(null)
3	ID_SALA		NUMBER(5,0)		Yes	(null)		3		(null)
4	ORA		VARCHAR2(10 BYTE)		No	(null)		4		(null)

Tabela PRODUS:

```
CREATE TABLE PRODUS
(
    id_produs NUMBER(5) CONSTRAINT PKEY_PRODUS PRIMARY KEY,
    nume_produs VARCHAR(30) CONSTRAINT nume_produs NOT NULL,
    pret NUMBER(10) CONSTRAINT pret_produs NOT NULL,
    stoc NUMBER(10) CONSTRAINT stoc_produs NOT NULL
);
```

Table PRODUS created.

❖	COLUMN_NAME	❖	DATA_TYPE	❖	NULLABLE	DATA_DEFAULT	❖	COLUMN_ID	❖	COMMENTS
1	ID_PRODUS		NUMBER(5,0)		No	(null)		1		(null)
2	NUME_PRODUS		VARCHAR2(30 BYTE)		No	(null)		2		(null)
3	PRET		NUMBER(10,0)		No	(null)		3		(null)
4	STOC		NUMBER(10,0)		No	(null)		4		(null)

Tabela CLIENT:

```
CREATE TABLE CLIENT
```

```
( id_client NUMBER(5) CONSTRAINT PKEY_CLIENT PRIMARY KEY,
  nume_client VARCHAR(20) CONSTRAINT nume_client NOT NULL,
  prenume_client VARCHAR(20) CONSTRAINT prenume_client NOT NULL,
  telefon VARCHAR(10) CONSTRAINT telefon_client UNIQUE NOT NULL,
  email VARCHAR(30) CONSTRAINT email_client UNIQUE NOT NULL
);
```

Table CLIENT created.

❖	COLUMN_NAME	❖	DATA_TYPE	❖	NULLABLE	DATA_DEFAULT	❖	COLUMN_ID	❖	COMMENTS
1	ID_CLIENT		NUMBER(5,0)		No	(null)		1		(null)
2	NUME_CLIENT		VARCHAR2(20 BYTE)		No	(null)		2		(null)
3	PRENUME_CLIENT		VARCHAR2(20 BYTE)		No	(null)		3		(null)
4	TELEFON		VARCHAR2(10 BYTE)		No	(null)		4		(null)
5	EMAIL		VARCHAR2(30 BYTE)		No	(null)		5		(null)

Tabela ANGAJAT:

```
CREATE TABLE ANGAJAT
```

```
( id_angajat NUMBER(5) CONSTRAINT PKEY_ANGAJAT PRIMARY KEY,
  nume_angajat VARCHAR(20) CONSTRAINT nume_angajat NOT NULL,
  prenume_angajat VARCHAR(20) CONSTRAINT prenume_angajat NOT NULL,
  functie VARCHAR(20) CONSTRAINT functie_angajat NOT NULL,
  salariu NUMBER(10) CONSTRAINT salariu_angajat NOT NULL
);
```

Table ANGAJAT created.

❖	COLUMN_NAME	❖	DATA_TYPE	❖	NULLABLE	DATA_DEFAULT	❖	COLUMN_ID	❖	COMMENTS
1	ID_ANGAJAT		NUMBER(5,0)		No	(null)		1		(null)
2	NUME_ANGAJAT		VARCHAR2(20 BYTE)		No	(null)		2		(null)
3	PRENUME_ANGAJAT		VARCHAR2(20 BYTE)		No	(null)		3		(null)
4	FUNCTIE		VARCHAR2(20 BYTE)		No	(null)		4		(null)
5	SALARIU		NUMBER(10,0)		No	(null)		5		(null)

Tabela COMANDĂ:

```
CREATE TABLE COMANDA
```

```
( id_comanda NUMBER(5) CONSTRAINT PKEY_COMANDA PRIMARY KEY,
  id_client NUMBER(5), CONSTRAINT fk_comanda_client FOREIGN
KEY(id_client) REFERENCES CLIENT(id_client),
  id_angajat NUMBER(5), CONSTRAINT fk_comanda_angajat FOREIGN
KEY(id_angajat) REFERENCES ANGAJAT(id_angajat),
  total NUMBER(10) DEFAULT 0 CONSTRAINT total_comanda NOT NULL,
  metoda_plata VARCHAR(10) DEFAULT 'cash' CONSTRAINT
metoda_plata_comanda CHECK(metoda_plata IN ('cash', 'card')) NOT NULL,
  data DATE CONSTRAINT data_comanda NOT NULL );
```

Table COMANDA created.

↕	COLUMN_NAME	↕	DATA_TYPE	↕	NULLABLE	DATA_DEFAULT	↕	COLUMN_ID	↕	COMMENTS
1	ID_COMANDA		NUMBER(5,0)		No	(null)		1		(null)
2	ID_CLIENT		NUMBER(5,0)		Yes	(null)		2		(null)
3	ID_ANGAJAT		NUMBER(5,0)		Yes	(null)		3		(null)
4	TOTAL		NUMBER(10,0)		No	0		4		(null)
5	METODA_PLATA		VARCHAR2(10 BYTE)		No	'cash'		5		(null)
6	DATA		DATE		No	(null)		6		(null)

Tabela BILET:

```
CREATE TABLE BILET
```

```
( id_bilet NUMBER(5) CONSTRAINT PKEY_BILET PRIMARY KEY,
  id_proiectie NUMBER(5), CONSTRAINT fk_bilet_proiectie FOREIGN
KEY(id_proiectie) REFERENCES PROIECTIE(id_proiectie),
  id_comanda NUMBER(5), CONSTRAINT fk_bilet_comanda FOREIGN
KEY(id_comanda) REFERENCES COMANDA(id_comanda),
  rand VARCHAR(2) CONSTRAINT rand_bilet NOT NULL,
  loc NUMBER(10) CONSTRAINT loc_bilet NOT NULL
);
```

Table BILET created.

↕	COLUMN_NAME	↕	DATA_TYPE	↕	NULLABLE	DATA_DEFAULT	↕	COLUMN_ID	↕	COMMENTS
1	ID_BILET		NUMBER(5,0)		No	(null)		1		(null)
2	ID_PROIECTIE		NUMBER(5,0)		Yes	(null)		2		(null)
3	ID_COMANDA		NUMBER(5,0)		Yes	(null)		3		(null)
4	RAND		VARCHAR2(2 BYTE)		No	(null)		4		(null)
5	LOC		NUMBER(10,0)		No	(null)		5		(null)

Tabela DETALIU_COMANDĂ:

```
CREATE TABLE DETALIU_COMANDA
```

```
( id_detaliu_comanda NUMBER(5) CONSTRAINT PKEY_DETALIU_COMANDA PRIMARY
KEY,
  id_produs NUMBER(5), CONSTRAINT fk_detaliu_comanda_produs FOREIGN
KEY(id_produs) REFERENCES PRODUS(id_produs),
  id_comanda NUMBER(5), CONSTRAINT fk_detaliu_comanda_comanda FOREIGN
KEY(id_comanda) REFERENCES COMANDA(id_comanda),
  cantitate NUMBER(10) CONSTRAINT cantitate_comanda NOT NULL
);
```

Table DETALIU_COMANDA created.

↕	COLUMN_NAME	↕	DATA_TYPE	↕	NULLABLE	DATA_DEFAULT	↕	COLUMN_ID	↕	COMMENTS
1	ID_DETALIU_COMANDA		NUMBER(5,0)		No	(null)		1		(null)
2	ID_PRODUS		NUMBER(5,0)		Yes	(null)		2		(null)
3	ID_COMANDA		NUMBER(5,0)		Yes	(null)		3		(null)
4	CANTITATE		NUMBER(10,0)		No	(null)		4		(null)

Tabela ACTOR:

```
CREATE TABLE ACTOR
```

```
( id_actor NUMBER(5) CONSTRAINT PKEY_ACTOR PRIMARY KEY,
  nume_actor VARCHAR(20) CONSTRAINT nume_actor NOT NULL,
```

```

        prenume_actor VARCHAR(20) CONSTRAINT prenume_actor NOT NULL,
        nationalitate VARCHAR(20) CONSTRAINT nationalitate NOT NULL
    );

```

Table ACTOR created.

❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS
1 ID_ACTOR	NUMBER(5,0)	No	(null)	1 (null)	
2 NUME_ACTOR	VARCHAR2(20 BYTE)	No	(null)	2 (null)	
3 PRENUME_ACTOR	VARCHAR2(20 BYTE)	No	(null)	3 (null)	
4 NATIONALITATE	VARCHAR2(20 BYTE)	No	(null)	4 (null)	

Tabela FILM_ACTOR:

```

CREATE TABLE FILM_ACTOR
(
    id_film_actor NUMBER(5) CONSTRAINT PKEY_FILM_ACTOR PRIMARY KEY,
    id_actor      NUMBER(5),      CONSTRAINT fk_film_actor_actor FOREIGN
KEY(id_actor) REFERENCES ACTOR(id_actor),
    id_film       NUMBER(5), CONSTRAINT fk_film_actor_film FOREIGN KEY(id_film)
REFERENCES FILM(id_film)
);

```

Table FILM_ACTOR created.

❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS
1 ID_FILM_ACTOR	NUMBER(5,0)	No	(null)	1 (null)	
2 ID_ACTOR	NUMBER(5,0)	Yes	(null)	2 (null)	
3 ID_FILM	NUMBER(5,0)	Yes	(null)	3 (null)	

5. Implementarea diagramei în Oracle (adăugarea datelor):

Am creat o secvență pe care o voi utiliza la inserarea înregistrărilor în tabele.

```

CREATE SEQUENCE cinema_seq
START WITH      1
INCREMENT BY    1
NOCACHE
NOCYCLE;

```

Tabela CINEMA:

```

INSERT INTO CINEMA
VALUES (cinema_seq.nextval, 'Retroplex', 'Str. Victoriei nr. 15', 'Bucuresti');
INSERT INTO CINEMA
VALUES (cinema_seq.nextval, 'Starlight', 'Bd. Unirii nr. 20', 'Bucuresti');
INSERT INTO CINEMA
VALUES (cinema_seq.nextval, 'Pixelplex', 'Str. Palat nr. 5', 'Iasi');
INSERT INTO CINEMA
VALUES (cinema_seq.nextval, 'Cinematica', 'Str. Republicii nr. 10', 'Brasov');
INSERT INTO CINEMA
VALUES (cinema_seq.nextval, 'Flashback', 'Piata Unirii nr. 2', 'Cuj-Napoca');

```


ID_CINEMA	NUME_CINEMA	ADRESA	ORAS
1	Retroplex	Str. Victoriei nr. 15	Bucuresti
2	Starlight	Bd. Unirii nr. 20	Bucuresti
3	Pixelplex	Str. Palat nr. 5	Iasi
4	Cinematica	Str. Republicii nr. 10	Brasov
5	Flashback	Piata Unirii nr. 2	Cluj-Napoca

Tabela SALĂ:

```
INSERT INTO SALA
```

```
VALUES (cinema_seq.nextval, (SELECT id_cinema FROM CINEMA WHERE nume_cinema = 'Retroplex'), 'Ultra', 40);
```

```
INSERT INTO SALA
```

```
VALUES (cinema_seq.nextval, (SELECT id_cinema FROM CINEMA WHERE nume_cinema = 'Retroplex'), 'Galaxy', 30);
```

```
INSERT INTO SALA
```

```
VALUES (cinema_seq.nextval, (SELECT id_cinema FROM CINEMA WHERE nume_cinema = 'Starlight'), 'Epika', 25);
```

```
INSERT INTO SALA
```

```
VALUES (cinema_seq.nextval, (SELECT id_cinema FROM CINEMA WHERE nume_cinema = 'Starlight'), 'Infinity', 50);
```

```
INSERT INTO SALA
```

```
VALUES (cinema_seq.nextval, (SELECT id_cinema FROM CINEMA WHERE nume_cinema = 'Pixelplex'), 'Astral', 25);
```

```
INSERT INTO SALA
```

```
VALUES (cinema_seq.nextval, (SELECT id_cinema FROM CINEMA WHERE nume_cinema = 'Cinematica'), 'Orion', 15);
```

```
INSERT INTO SALA
```

```
VALUES (cinema_seq.nextval, (SELECT id_cinema FROM CINEMA WHERE nume_cinema = 'Flashback'), 'Mega', 30);
```

ID_SALA	ID_CINEMA	NUME	CAPACITATE
6	1	Ultra	40
7	1	Galaxy	30
8	2	Epika	25
9	2	Infinity	50
10	3	Astral	25
11	4	Orion	15
12	5	Mega	30

Tabela FILM:

```
INSERT INTO FILM
```

```
VALUES (cinema_seq.nextval, 'Dirty Dancing', 100, 'Romance', 'The movie showcases a forbidden romance, set against the backdrop of a dance competition.');
```

```
INSERT INTO FILM
```



```
VALUES (cinema_seq.nextval, 'Back to the Future', 116, 'Sci-Fi', '');

INSERT INTO FILM

VALUES (cinema_seq.nextval, 'GhostBusters', 107, 'Comedy', 'The movie features
a group of friends who start a ghost-catching business in New York City.');
```

```
INSERT INTO FILM

VALUES (cinema_seq.nextval, 'The Breakfast Club', 92, 'Drama', '');

INSERT INTO FILM

VALUES (cinema_seq.nextval, 'Dead Poets Society', 128, 'Drama', 'The movie
explores themes of conformity, self-discovery, and the power of literature.');
```

```
INSERT INTO FILM

VALUES (cinema_seq.nextval, 'The Terminator', 108, 'Action', '');

INSERT INTO FILM

VALUES (cinema_seq.nextval, 'The Shining', 142, 'Horror', ');
```

ID_FILM	TITLU_FILM	DURATA	GEN	DESCRIERE
13	Dirty Dancing	100	Romance	The movie showcases a forbidden romance, set against the backdrop of a dance competition.
14	Back to the Future	116	Sci-Fi	(null)
15	GhostBusters	107	Comedy	The movie features a group of friends who start a ghost-catching business in New York City.
16	The Breakfast Club	92	Drama	(null)
17	Dead Poets Society	128	Drama	The movie explores themes of conformity, self-discovery, and the power of literature.
18	The Terminator	108	Action	(null)
19	The Shining	142	Horror	(null)

Tabela PROIECTIE:

```
INSERT INTO PROIECTIE

VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film = 'Dirty
Dancing'), (SELECT id_sala FROM SALA WHERE nume = 'Ultra'), '10:00');
```

```
INSERT INTO PROIECTIE

VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film = 'Dead
Poets Society'), (SELECT id_sala FROM SALA WHERE nume = 'Galaxy'), '15:00');
```

```
INSERT INTO PROIECTIE

VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film = 'The
Breakfast Club'), (SELECT id_sala FROM SALA WHERE nume = 'Infinity'), '20:30');
```

```
INSERT INTO PROIECTIE

VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film = 'The
Terminator'), (SELECT id_sala FROM SALA WHERE nume = 'Mega'), '17:30');
```

```
INSERT INTO PROIECTIE

VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film = 'Dead
Poets Society'), (SELECT id_sala FROM SALA WHERE nume = 'Astral'), '18:00');
```

```
INSERT INTO PROIECTIE

VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film =
'GhostBusters'), (SELECT id_sala FROM SALA WHERE nume = 'Orion'), '18:00');
```

```
INSERT INTO PROIECTIE
```

```
VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film = 'Back to the Future'), (SELECT id_sala FROM SALA WHERE nume = 'Epika'), '12:30');
```

```
INSERT INTO PROIECTIE
```

```
VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film = 'The Shining'), (SELECT id_sala FROM SALA WHERE nume = 'Infinity'), '21:00');
```

```
INSERT INTO PROIECTIE
```

```
VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film = 'GhostBusters'), (SELECT id_sala FROM SALA WHERE nume = 'Ultra'), '12:00');
```

```
INSERT INTO PROIECTIE
```

```
VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film = 'Dirty Dancing'), (SELECT id_sala FROM SALA WHERE nume = 'Galaxy'), '11:30');
```

ID_PROIECTIE	ID_FILM	ID_SALA	ORA
20	13	6	10:00
21	17	7	15:00
22	16	9	20:30
23	18	12	17:30
24	17	10	18:00
25	15	11	18:00
26	14	8	12:30
27	19	9	21:00
28	15	6	12:00
29	13	7	11:30

Tabela PRODUS:

```
INSERT INTO PRODUS
```

```
VALUES (cinema_seq.nextval, 'popcorn', 20, 100);
```

```
INSERT INTO PRODUS
```

```
VALUES (cinema_seq.nextval, 'nachos', 25, 100);
```

```
INSERT INTO PRODUS
```

```
VALUES (cinema_seq.nextval, 'suc', 10, 150);
```

```
INSERT INTO PRODUS
```

```
VALUES (cinema_seq.nextval, 'apa', 5, 150);
```

```
INSERT INTO PRODUS
```

```
VALUES (cinema_seq.nextval, 'alune', 15, 100);
```

ID_PRODUS	NUME_PRODUS	PRET	STOC
30	popcorn	20	100
31	nachos	25	100
32	suc	10	150
33	apa	5	150
34	alune	15	100

Tabela CLIENT:

```
INSERT INTO CLIENT
```

```

VALUES (cinema_seq.nextval, 'Stan', 'Bianca', '0719472600',
'bianastan123@gmail.com');

INSERT INTO CLIENT

VALUES (cinema_seq.nextval, 'Dumitrescu', 'Eric', '0709012574',
'dumitrescueric@yahoo.com');

INSERT INTO CLIENT

VALUES (cinema_seq.nextval, 'Dobre', 'Stefan', '0799006584',
'stefan27@gmail.com');

INSERT INTO CLIENT

VALUES (cinema_seq.nextval, 'Paun', 'Raluca', '0771063900',
'ralucapaun@gmail.com');

INSERT INTO CLIENT

VALUES (cinema_seq.nextval, 'Munteanu', 'Laura', '0761352678',
'lalamunteanu@gmail.com');

INSERT INTO CLIENT

VALUES (cinema_seq.nextval, 'Popescu', 'Andrei', '0721123456',
'andrei.popescu@gmail.com');

INSERT INTO CLIENT

VALUES (cinema_seq.nextval, 'Ionescu', 'Maria', '0731987654',
'maria.ionescu@gmail.com');

INSERT INTO CLIENT

VALUES (cinema_seq.nextval, 'Stoica', 'Larisa', '0761890123',
'sstoica@gmail.com');

INSERT INTO CLIENT

VALUES (cinema_seq.nextval, 'Ciurnea', 'Ioan', '0761352875',
'ioan1234@yahoo.com');

INSERT INTO CLIENT

VALUES (cinema_seq.nextval, 'Pestritu', 'Monica', '0761000678',
'monicapestritu@gmail.com');

```

ID_CLIENT	NUME_CLIENT	PRENUME_CLIENT	TELEFON	EMAIL
35	Stan	Bianca	0719472600	bianastan123@gmail.com
36	Dumitrescu	Eric	0709012574	dumitrescueric@yahoo.com
37	Dobre	Stefan	0799006584	stefan27@gmail.com
38	Paun	Raluca	0771063900	ralucapaun@gmail.com
39	Munteanu	Laura	0761352678	lalamunteanu@gmail.com
40	Popescu	Andrei	0721123456	andrei.popescu@gmail.com
41	Ionescu	Maria	0731987654	maria.ionescu@gmail.com
42	Stoica	Larisa	0761890123	sstoica@gmail.com
43	Ciurnea	Ioan	0761352875	ioan1234@yahoo.com
44	Pestritu	Monica	0761000678	monicapestritu@gmail.com

Tabela ANGAJAT:

```
INSERT INTO ANGAJAT
VALUES (cinema_seq.nextval, 'Ionescu', 'Mihai', 'casier', 3000);
INSERT INTO ANGAJAT
VALUES (cinema_seq.nextval, 'Vasile', 'Iustina', 'casier', 3000);
INSERT INTO ANGAJAT
VALUES (cinema_seq.nextval, 'Popescu', 'Dragos', 'casier', 3000);
INSERT INTO ANGAJAT
VALUES (cinema_seq.nextval, 'Dascalu', 'Andrei', 'tehnician', 4000);
INSERT INTO ANGAJAT
VALUES (cinema_seq.nextval, 'Olteanu', 'Maria', 'curatenie', 1500);
INSERT INTO ANGAJAT
VALUES (cinema_seq.nextval, 'Dogareci', 'Alexandra', 'curatenie', 1500);
```

ID_ANGAJAT	NUME_ANGAJAT	PRENUME_ANGAJAT	FUNCTIE	SALARIU
45	Ionescu	Mihai	casier	3000
46	Vasile	Iustina	casier	3000
47	Popescu	Dragos	casier	3000
48	Dascalu	Andrei	tehnician	4000
49	Olteanu	Maria	curatenie	1500
50	Dogareci	Alexandra	curatenie	1500

Tabela COMANDĂ:

```
INSERT INTO COMANDA(id_comanda, id_client, id_angajat, metoda_plata, data)
VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0719472600'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Popescu'),
'card', TO_DATE('12-04-2023', 'DD-MM-YYYY'));
INSERT INTO COMANDA(id_comanda, id_client, id_angajat, data)
VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0709012574'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Vasile'),
TO_DATE('15-04-2023', 'DD-MM-YYYY'));
INSERT INTO COMANDA(id_comanda, id_client, id_angajat, metoda_plata, data)
VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0799006584'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Ionescu'),
'card', TO_DATE('23-04-2023', 'DD-MM-YYYY'));
INSERT INTO COMANDA(id_comanda, id_client, id_angajat, data)
VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0771063900'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Vasile'),
TO_DATE('10-05-2023', 'DD-MM-YYYY'));
INSERT INTO COMANDA(id_comanda, id_client, id_angajat, metoda_plata, data)
```

```

VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0761352678'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Popescu'),
'card', TO_DATE('24-05-2023', 'DD-MM-YYYY'));

INSERT INTO COMANDA(id_comanda, id_client, id_angajat, metoda_plata, data)
VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0721123456'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Ionescu'),
'card', TO_DATE('25-05-2023', 'DD-MM-YYYY'));

INSERT INTO COMANDA(id_comanda, id_client, id_angajat, data)
VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0731987654'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Popescu'),
TO_DATE('25-05-2023', 'DD-MM-YYYY'));

INSERT INTO COMANDA(id_comanda, id_client, id_angajat, data)
VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0761890123'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Vasile'),
TO_DATE('26-05-2023', 'DD-MM-YYYY'));

INSERT INTO COMANDA(id_comanda, id_client, id_angajat, metoda_plata, data)
VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0761352875'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Ionescu'),
'card', TO_DATE('29-05-2023', 'DD-MM-YYYY'));

INSERT INTO COMANDA(id_comanda, id_client, id_angajat, metoda_plata, data)
VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0761000678'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Vasile'),
'card', TO_DATE('01-06-2023', 'DD-MM-YYYY'));

```

Pentru tabela COMANDA, calcularea totalului comenzilor.

```

UPDATE COMANDA c

SET c.total = (

    SELECT NVL(SUM(p.pret * dc.cantitate), 0) as total

    FROM DETALIU_COMANDA dc

    JOIN PRODUS p ON dc.id_produs = p.id_produs

    WHERE dc.id_comanda = c.id_comanda

);

```

ID_COMA...	ID_CLIENT	ID_ANGA...	TOTAL	METODA_...	DATA
51	35	47	40	card	12-APR-23
52	36	46	0	cash	15-APR-23
53	37	45	35	card	23-APR-23
54	38	46	70	cash	10-MAY-23
55	39	47	80	card	24-MAY-23
56	40	45	15	card	25-MAY-23
57	41	47	50	cash	25-MAY-23
58	42	46	45	cash	26-MAY-23
59	43	45	20	card	29-MAY-23
60	44	46	30	card	01-JUN-23

Tabela BILET:

```
INSERT INTO BILET
VALUES (cinema_seq.nextval, 20, 51, 'E', 5);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 20, 51, 'E', 6);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 21, 52, 'C', 10);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 21, 52, 'C', 11);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 22, 53, 'D', 4);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 22, 53, 'E', 1);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 23, 54, 'C', 5);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 24, 55, 'A', 2);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 25, 56, 'C', 5);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 26, 57, 'A', 3);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 27, 58, 'B', 4);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 28, 59, 'A', 5);
INSERT INTO BILET
VALUES (cinema_seq.nextval, 29, 60, 'D', 3);
```

ID_BILET	ID_PROIECTIE	ID_COMANDA	RAND	LOC
61	20	51 E		5
62	20	51 E		6
63	21	52 C		10
64	21	52 C		11
65	22	53 D		4
66	22	53 E		1
67	23	54 C		5
68	24	55 A		2
69	25	56 C		5
70	26	57 A		3
71	27	58 B		4
72	28	59 A		5
73	29	60 D		3

TABELA DETALIU_COMANDĂ:

```
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'popcorn'), 51, 2);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'suc'), 53, 2);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'apa'), 53, 3);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'alune'), 54, 1);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'nachos'), 54, 1);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'alune'), 54, 2);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'popcorn'), 55, 4);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'alune'), 56, 1);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'nachos'), 57, 2);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'suc'), 58, 2);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'nachos'), 58, 1);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'popcorn'), 59, 1);
INSERT INTO DETALIU_COMANDA
```



```
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'apa'), 60, 2);
INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'popcorn'), 60, 1);
```

ID_DETALIU_COMANDA	ID_PRODUS	ID_COMANDA	CANTITATE
74	30	51	2
75	32	53	2
76	33	53	3
77	34	54	1
78	31	54	1
79	34	54	2
80	30	55	4
81	34	56	1
82	31	57	2
83	32	58	2
84	31	58	1
85	30	59	1
86	33	60	2
87	30	60	1

Tabela ACTOR:

```
INSERT INTO ACTOR
VALUES (cinema_seq.nextval, 'Murray', 'Bill', 'american');
INSERT INTO ACTOR
VALUES (cinema_seq.nextval, 'Swayze', 'Patrick', 'american');
INSERT INTO ACTOR
VALUES (cinema_seq.nextval, 'Williams', 'Robin', 'american');
INSERT INTO ACTOR
VALUES (cinema_seq.nextval, 'Schwarzenegger', 'Arnold', 'austrian');
INSERT INTO ACTOR
VALUES (cinema_seq.nextval, 'Duvall', 'Shelley', 'american');
INSERT INTO ACTOR
VALUES (cinema_seq.nextval, 'Hawke', 'Ethan', 'american');
INSERT INTO ACTOR
VALUES (cinema_seq.nextval, 'Grey', 'Jennifer', 'american');
INSERT INTO ACTOR
VALUES (cinema_seq.nextval, 'Nicholson', 'Jack', 'american');
INSERT INTO ACTOR
VALUES (cinema_seq.nextval, 'Hamilton', 'Linda', 'american');
```

ID_ACTOR	NUME_ACTOR	PRENUME_ACTOR	NATIONALITATE
88	Murray	Bill	american
89	Swayze	Patrick	american
90	Williams	Robin	american
91	Schwarzenegger	Arnold	austrian
92	Duvall	Shelley	american
93	Hawke	Ethan	american
94	Grey	Jennifer	american
95	Nicholson	Jack	american
96	Hamilton	Linda	american

TABELA FILM_ACTOR:

```

INSERT INTO FILM_ACTOR
VALUES (cinema_seq.nextval, (SELECT id_actor FROM ACTOR WHERE nume_actor =
'Swayze'), (SELECT id_film FROM FILM WHERE titlu_film = 'Dirty Dancing') );
INSERT INTO FILM_ACTOR
VALUES (cinema_seq.nextval, (SELECT id_actor FROM ACTOR WHERE nume_actor =
'Grey'), (SELECT id_film FROM FILM WHERE titlu_film = 'Dirty Dancing') );
INSERT INTO FILM_ACTOR
VALUES (cinema_seq.nextval, (SELECT id_actor FROM ACTOR WHERE nume_actor =
'Schwarzenegger'), (SELECT id_film FROM FILM WHERE titlu_film = 'The Terminator')
);
INSERT INTO FILM_ACTOR
VALUES (cinema_seq.nextval, (SELECT id_actor FROM ACTOR WHERE nume_actor =
'Hamilton'), (SELECT id_film FROM FILM WHERE titlu_film = 'The Terminator') );
INSERT INTO FILM_ACTOR
VALUES (cinema_seq.nextval, (SELECT id_actor FROM ACTOR WHERE nume_actor =
'Nicholson'), (SELECT id_film FROM FILM WHERE titlu_film = 'The Shining') );
INSERT INTO FILM_ACTOR
VALUES (cinema_seq.nextval, (SELECT id_actor FROM ACTOR WHERE nume_actor =
'Duvall'), (SELECT id_film FROM FILM WHERE titlu_film = 'The Shining') );
INSERT INTO FILM_ACTOR
VALUES (cinema_seq.nextval, (SELECT id_actor FROM ACTOR WHERE nume_actor =
'Murray'), (SELECT id_film FROM FILM WHERE titlu_film = 'GhostBusters') );
INSERT INTO FILM_ACTOR
VALUES (cinema_seq.nextval, (SELECT id_actor FROM ACTOR WHERE nume_actor =
'Williams'), (SELECT id_film FROM FILM WHERE titlu_film = 'Dead Poets Society')
);
INSERT INTO FILM_ACTOR
VALUES (cinema_seq.nextval, (SELECT id_actor FROM ACTOR WHERE nume_actor =
'Hawke'), (SELECT id_film FROM FILM WHERE titlu_film = 'Dead Poets Society') );
INSERT INTO FILM_ACTOR

```

```
VALUES (cinema_seq.nextval, (SELECT id_actor FROM ACTOR WHERE nume_actor =
'Hawke'), (SELECT id_film FROM FILM WHERE titlu_film = 'The Breakfast Club') );
INSERT INTO FILM_ACTOR
VALUES (cinema_seq.nextval, (SELECT id_actor FROM ACTOR WHERE nume_actor =
'Murray'), (SELECT id_film FROM FILM WHERE titlu_film = 'Back to the Future') );
```

ID_FILM_ACTOR	ID_ACTOR	ID_FILM
97	89	13
98	94	13
99	91	18
100	96	18
101	95	19
102	92	19
103	88	15
104	90	17
105	93	17
106	93	16
107	88	14

6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate (vector, tablou imbricat, tablou indexat).

Enunț: Definiți un subprogram stocat independent de tip procedură care primește numele și prenumele unui client si după ce verifică existența acestuia, calculează suma totală pe care a cheltuit-o la cinema și determină metoda de plată preferată.

Am folosit un vector (pentru a memora metodele de plată ale clientului), un tablou imbricat (pentru a memora toatalul comenzilor făcute de client) si un tablou indexat (pentru a memora numele clienților).

```
CREATE OR REPLACE PROCEDURE Gestionare_Client(
    var_nume_trimis IN VARCHAR2,
    var_prenume_trimis IN VARCHAR2
) IS
    TYPE tablou_indexat IS TABLE OF VARCHAR2(200) INDEX BY PLS_INTEGER;
    TYPE tablou_imbricat IS TABLE OF NUMBER;
    TYPE vector IS VARRAY(50) OF VARCHAR2(200);

    tablou_clienti_existenti tablou_indexat;
    tablou_preturi_comenzi tablou_imbricat := tablou_imbricat();
```

```

vector_metode_plata vector := vector();

var_exista_client BOOLEAN;
var_suma_totala NUMBER := 0;
var_metoda_plata_preferata VARCHAR2(10);

v_index PLS_INTEGER := 1;
v_prenume_db VARCHAR2(50);
v_id_client CLIENT.id_client%TYPE;

v_nr_plati_card NUMBER := 0;
v_nr_plati_cash NUMBER := 0;

BEGIN

    SELECT nume_client
    BULK COLLECT INTO tablou_clienti_existenti
    FROM CLIENT;

    var_exista_client := FALSE;

    FOR i IN tablou_clienti_existenti.FIRST..tablou_clienti_existenti.LAST LOOP
        IF tablou_clienti_existenti(i) = var_nume_trimis THEN
            var_exista_client := TRUE;

            SELECT prenume_client INTO v_prenume_db
            FROM CLIENT
            WHERE nume_client = var_nume_trimis;

            IF v_prenume_db <> var_prenume_trimis THEN
                var_exista_client := FALSE;
            END IF;
        END IF;
    END LOOP;

    IF var_exista_client = FALSE THEN
        DBMS_OUTPUT.PUT_LINE('Clientul nu exista.');
```

ELSE

```

        SELECT id_client INTO v_id_client
        FROM CLIENT
```

```
WHERE nume_client = var_nume_trimis AND prenume_client =  
var_prenume_trimis;
```

```
SELECT total  
BULK COLLECT INTO tablou_preturi_comenzi  
FROM COMANDA  
WHERE id_client = v_id_client;
```

```
FOR i IN tablou_preturi_comenzi.FIRST..tablou_preturi_comenzi.LAST LOOP  
    var_suma_totala := var_suma_totala + tablou_preturi_comenzi(i);  
END LOOP;
```

```
SELECT metoda_plata  
BULK COLLECT INTO vector_metode_plata  
FROM COMANDA  
WHERE id_client = v_id_client;
```

```
FOR i IN vector_metode_plata.FIRST..vector_metode_plata.LAST LOOP  
    IF vector_metode_plata(i) = 'card' THEN  
        v_nr_plati_card := v_nr_plati_card + 1;  
    END IF;  
    IF vector_metode_plata(i) = 'cash' THEN  
        v_nr_plati_cash := v_nr_plati_cash + 1;  
    END IF;  
END LOOP;
```

```
IF v_nr_plati_card > v_nr_plati_cash THEN  
    var_metoda_plata_preferata := 'Card';  
END IF;  
IF v_nr_plati_cash > v_nr_plati_card THEN  
    var_metoda_plata_preferata := 'Cash';  
END IF;  
IF v_nr_plati_cash = v_nr_plati_card THEN  
    var_metoda_plata_preferata := 'Card/Cash';  
END IF;
```

```
DBMS_OUTPUT.PUT_LINE('Suma totala a comenzilor: ' || var_suma_totala);  
DBMS_OUTPUT.PUT_LINE('Metoda de plata preferata: ' ||  
var_metoda_plata_preferata);
```

```

END IF;

EXCEPTION

    WHEN OTHERS THEN

        RAISE_APPLICATION_ERROR(-20002, 'A aparut o eroare!');

END Gestionare_Client;

/

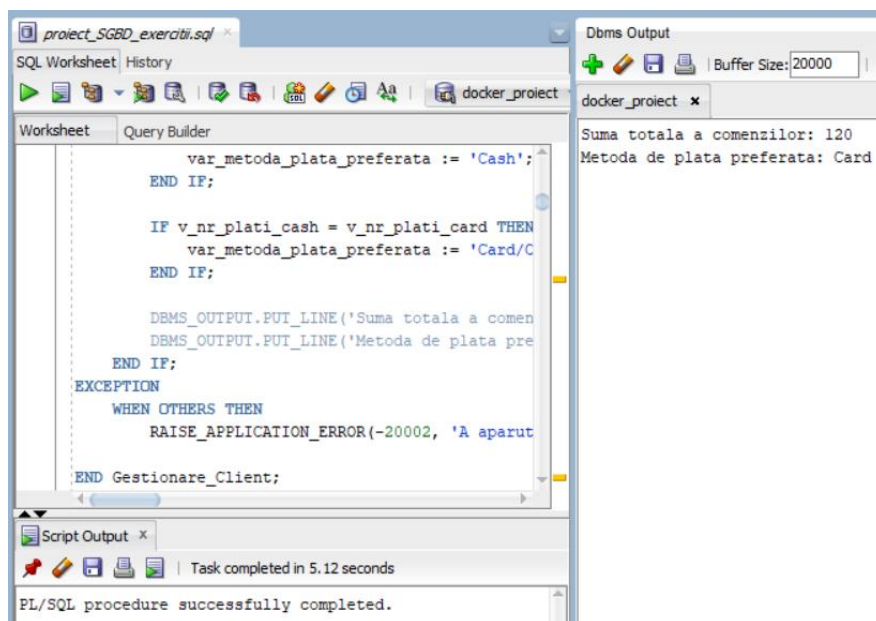
```

Procedure GESTIONARE_CLIENT compiled

Apelarea procedurii:

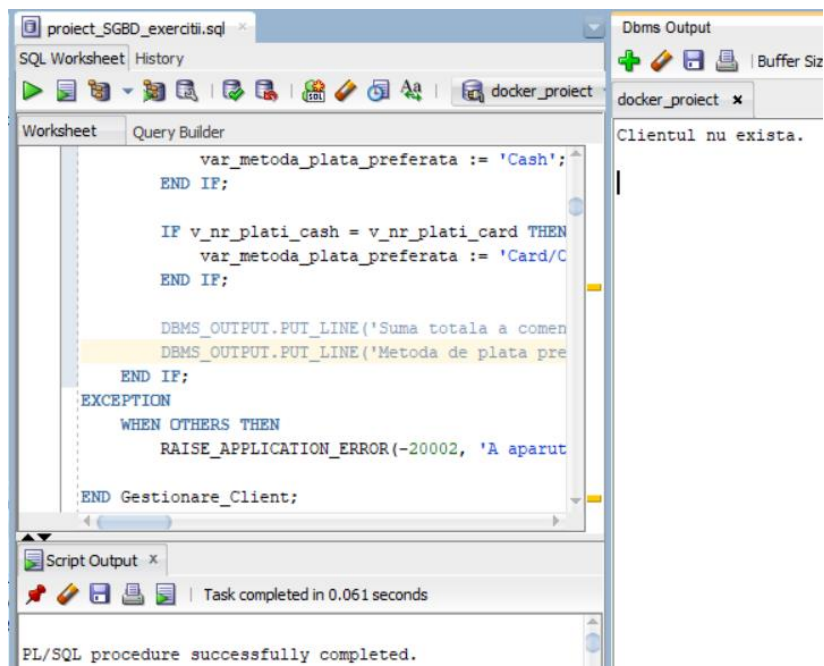
1. Cazul în care clientul există.

```
EXECUTE Gestionare_Client('Stan', 'Bianca');
```



2. Cazul în care clientul nu există.

```
EXECUTE Gestionare_Client('Toader', 'Raluca');
```



Pentru verificarea problemei am făcut următoarele adăugări în tabelele COMANDĂ și DETALIU_COMANDĂ:

```
INSERT INTO COMANDA(id_comanda, id_client, id_angajat, metoda_plata, data)
VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0719472600'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Vasile'),
'cash', TO_DATE('12-05-2023', 'DD-MM-YYYY'));

INSERT INTO COMANDA(id_comanda, id_client, id_angajat, metoda_plata, data)
VALUES (cinema_seq.nextval, (SELECT id_client FROM CLIENT WHERE telefon =
'0719472600'), (SELECT id_angajat FROM ANGAJAT WHERE nume_angajat = 'Popescu'),
'card', TO_DATE('12-06-2023', 'DD-MM-YYYY'));

INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'popcorn'), 112, 2);

INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'suc'), 112, 2);

INSERT INTO DETALIU_COMANDA
VALUES (cinema_seq.nextval, (SELECT id_produs FROM PRODUS WHERE nume_produs =
'apa'), 113, 4);
```

ID_COMA...	ID_CLIENT	ID_ANGA...	TOTAL	METODA_...	DATA
51	35	47	40	card	12-APR-23
52	36	46	0	cash	15-APR-23
53	37	45	35	card	23-APR-23
54	38	46	70	cash	10-MAY-23
55	39	47	80	card	24-MAY-23
56	40	45	15	card	25-MAY-23
57	41	47	50	cash	25-MAY-23
58	42	46	45	cash	26-MAY-23
59	43	45	20	card	29-MAY-23
60	44	46	30	card	01-JUN-23
112	35	45	60	cash	12-MAY-23
113	35	46	20	card	12-JUN-23

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

Enunț: Să se afișeze pentru fiecare cinema o listă numerotată cu toate sălile care aparțin acestuia și capacitatea acestora.


```

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE Afisare_Sali IS
    CURSOR CursorCinematografe IS
        SELECT id_cinema as id_cinematograf, nume_cinema as nume_cinematograf
        FROM CINEMA;

    CURSOR CursorSali (v_id_cinematograf CINEMA.id_cinema%type) IS
        SELECT nume as nume_sala, capacitate as capacitate_sala
        FROM SALA
        WHERE id_cinema = v_id_cinematograf;

    var_nume_sala SALA.nume%type;
    var_capacitate_sala SALA.capacitate%type;
    var_numar_sala NUMBER := 1;
    var_numar_cinematograf NUMBER := 1;
BEGIN
    FOR i IN CursorCinematografe LOOP
        DBMS_OUTPUT.PUT_LINE('-----');
        DBMS_OUTPUT.PUT_LINE(var_numar_cinematograf || '. Cinema: ' ||
i.nume_cinematograf);
        OPEN CursorSali(i.id_cinematograf);
        LOOP
            FETCH CursorSali INTO var_nume_sala, var_capacitate_sala;
            EXIT WHEN CursorSali%notfound;
            DBMS_OUTPUT.PUT_LINE(' ' || var_numar_sala || '. Sala: ' ||
var_nume_sala || ' -> capacitate = ' || var_capacitate_sala || ');');
            var_numar_sala := var_numar_sala + 1;
        END LOOP;
        var_numar_sala := 1;
        var_numar_cinematograf := var_numar_cinematograf + 1;
        DBMS_OUTPUT.PUT_LINE(' ');
        CLOSE CursorSali;
    END LOOP;
END Afisare_Sali;
/

```

```

| Procedure AFISARE_SALI compiled

```

Apelarea subprogramului:

1. SQL*PLUS

```
EXECUTE Afisare_Sali;
```

2. PLSQL

```
BEGIN
```

```
    Afisare_Sali;
```

```
END;
```

```
/
```

The screenshot displays the Oracle SQL Developer environment. The main window shows a SQL Worksheet with a PL/SQL procedure named `Afisare_Sali`. The procedure uses a cursor to iterate through cinema records, displaying the cinema name and the capacity of each hall. The output is shown in the 'Dbms Output' window, which lists the total number of movies (120) and the preferred payment method (Card). The output is formatted with dashed lines separating the cinema entries.

```
DBMS_OUTPUT.PUT_LINE(var_numar_cinematogr
OPEN CursorSali(i.id_cinematograf);
LOOP
    FETCH CursorSali INTO var_nume_sala,
    EXIT WHEN CursorSali%notfound;
    DBMS_OUTPUT.PUT_LINE('    ' || var_nume_sala || ' -> capacitate = ' || var_numar_sala);
    var_numar_sala := var_numar_sala + 1;
END LOOP;
var_numar_sala := 1;
var_numar_cinematograf := var_numar_cinematograf + 1;
CLOSE CursorSali;
END LOOP;
END Afisare_Sali;
```

Dbms Output

Suma totala a comenzilor: 120
Metoda de plata preferata: Card

1. Cinema: Cinematica
1. Sala: Orion -> capacitate = 15;

2. Cinema: Flashback
1. Sala: Mega -> capacitate = 30;

3. Cinema: Pixelplex
1. Sala: Astral -> capacitate = 25;

4. Cinema: Retroplex
1. Sala: Ultra -> capacitate = 40;
2. Sala: Galaxy -> capacitate = 30;

5. Cinema: Starlight
1. Sala: Epika -> capacitate = 25;
2. Sala: Infinity -> capacitate = 50;

Script Output x

Task completed in 0.191 seconds

PL/SQL procedure successfully completed.

8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții proprii. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.

Enunț: Definiți un subprogram prin care să obțineți numele actorilor care joacă într-un film al cărui titlu este introdus.

Cele trei tabele definite folosite în subprogram: ACTOR, FILM_ACTOR, FILM.

Cele doua excepții definite:

- Titlul filmului introdus nu are niciun actor in baza de date.
- Titlul filmului introdus nu corespunde cu titlul niciunui film din baza de date.

```

CREATE OR REPLACE FUNCTION Actori_din_Film
    (v_titlu_film FILM.titlu_film%TYPE DEFAULT 'GhostBusters')
RETURN VARCHAR2 IS
    v_rezultat VARCHAR2(20);
    v_cursor SYS_REFCURSOR;
    v_prenume_actor ACTOR.prenume_actor%TYPE;
    v_nume_actor ACTOR.nume_actor%TYPE;

    v_film_count NUMBER;
    v_actor_count NUMBER;

    titlul_filmului_nu_exista exception;
    titlul_filmului_nu_are_actori_introdusi exception;
BEGIN
    SELECT COUNT(*)
    INTO v_film_count
    FROM FILM
    WHERE titlu_film = v_titlu_film;

    IF v_film_count = 0 THEN
        RAISE titlul_filmului_nu_exista;
    END IF;

    OPEN v_cursor FOR
        SELECT a.prenume_actor, a.nume_actor
        FROM ACTOR a
        JOIN FILM_ACTOR fa ON a.id_actor = fa.id_actor
        JOIN FILM f ON fa.id_film = f.id_film
        WHERE f.titlu_film = v_titlu_film;

    SELECT COUNT(*)
    INTO v_actor_count
    FROM ACTOR a
    JOIN FILM_ACTOR fa ON a.id_actor = fa.id_actor
    JOIN FILM f ON fa.id_film = f.id_film
    WHERE f.titlu_film = v_titlu_film;

    IF v_actor_count = 0 THEN

```

```

        CLOSE v_cursor;

        RAISE titlul_filmului_nu_are_actori_introdusi;
    END IF;

    v_rezultat := '';

    LOOP

        FETCH v_cursor INTO v_prenume_actor, v_nume_actor;

        EXIT WHEN v_cursor%NOTFOUND;

        v_rezultat := v_rezultat || v_nume_actor || ' ' || v_prenume_actor || ',
';

    END LOOP;

    CLOSE v_cursor;

    v_rezultat := RTRIM(v_rezultat, ', ');

    RETURN v_rezultat;
EXCEPTION
    WHEN titlul_filmului_nu_are_actori_introdusi THEN
        DBMS_OUTPUT.PUT_LINE('Titlul filmului introdus nu are niciun actor in
baza de date.');
```

```

        RETURN 'exceptie';
    WHEN titlul_filmului_nu_exista THEN
        DBMS_OUTPUT.PUT_LINE('Titlul filmului introdus nu corespunde cu titlul
niciunui film din baza de date.');
```

```

        RETURN 'exceptie';
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Alta eroare' || SQLERRM);
        RETURN 'exceptie';
END Actori_din_Film;

/
Function ACTORI_DIN_FILM compiled

```

Apelarea funcției.

1. Cazul în care titlul filmului există.

```

DECLARE

    v_result VARCHAR2(100);

```

```

BEGIN

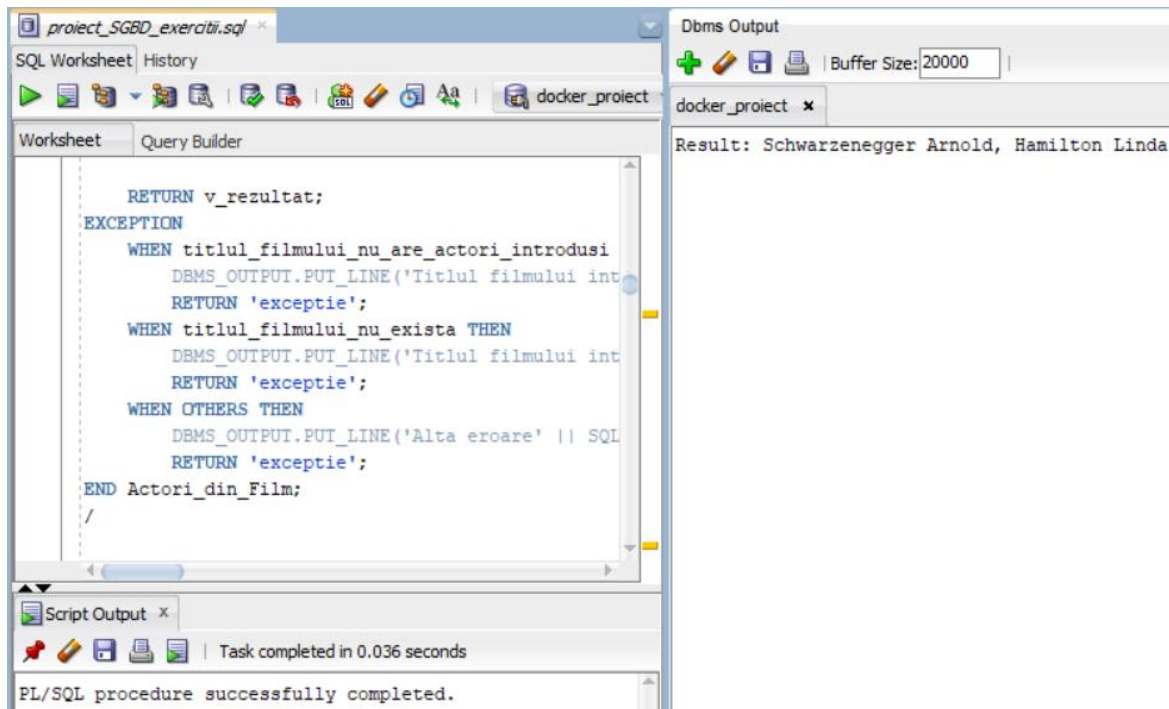
    v_result := Actori_din_Film('The Terminator');

    DBMS_OUTPUT.PUT_LINE('Actors: ' || v_result);

END;

/

```



2. Cazul în care titlul filmului nu există în baza de date.

```

DECLARE

    v_result VARCHAR2(100);

BEGIN

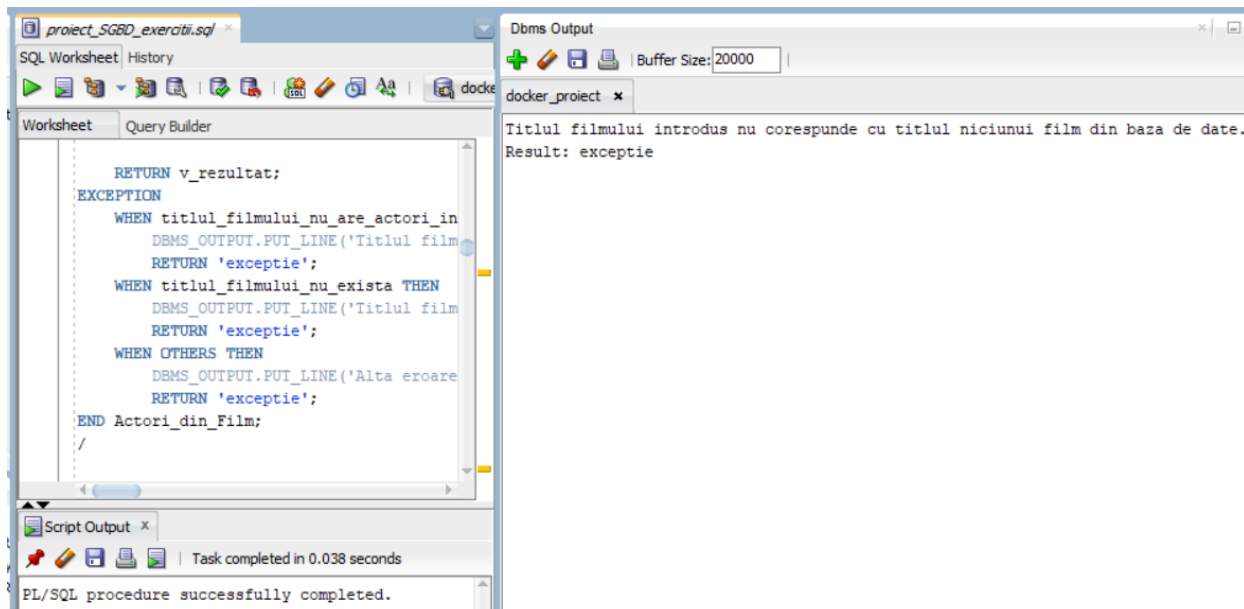
    v_result := Actori_din_Film('OverBoard');

    DBMS_OUTPUT.PUT_LINE('Result: ' || v_result);

END;

/

```



3. Cazul în care titlul filmului nu are actori introduși în baza de date.

DECLARE

```
v_result VARCHAR2(100);
```

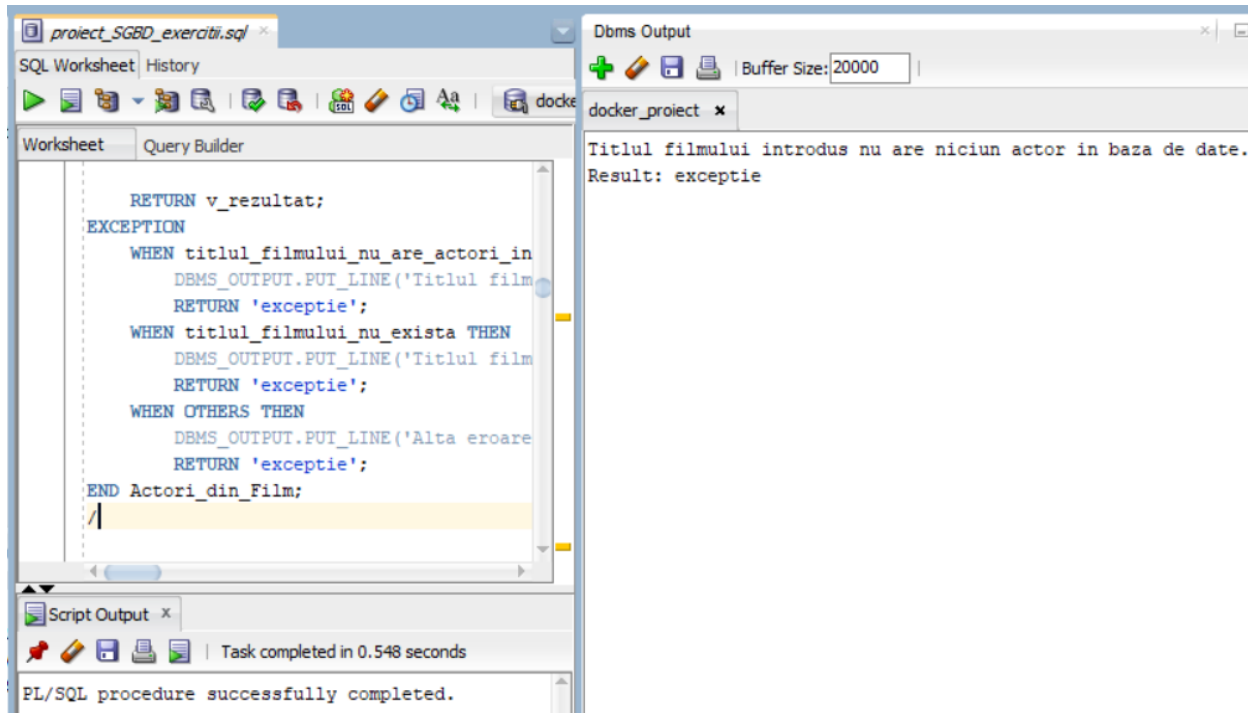
BEGIN

```
v_result := Actori_din_Film('E.T.');
```

```
DBMS_OUTPUT.PUT_LINE('Result: ' || v_result);
```

END;

/



9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile `NO_DATA_FOUND` și `TOO_MANY_ROWS`. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Enunț: Definiți un subprogram stocat independent de tip procedură care să obțină detaliile despre comanda plasată de un anumit client, identificat prin numele său.

Detaliile Comenzii -> id-ul comenzii, numele angajatului care s-a ocupat de comandă, prețul total, metoda de plată, numărul de produse și lista de produse comandate.

Cele cinci tabele definite folosite în subprogram: CLIENT, COMANDĂ, ANGAJAT, DETALIU_COMANDĂ, PRODUS.

Excepțiile tratate:

- Cazul în care numele clientului introdus nu se găsește în baza de date (NO_DATA_FOUND).
- Cazul în care numele clientului introdus se găsește de mai multe ori în baza de date (TOO_MANY_ROWS).
- Cazul în care clientul nu a plasat o comandă.

```

CREATE OR REPLACE PROCEDURE Detalii_Comanda
( p_nume IN CLIENT.nume_client%TYPE,
  p_id_comanda OUT COMANDA.id_comanda%TYPE,
  p_nume_angajat OUT ANGAJAT.nume_angajat%TYPE,
  p_total OUT COMANDA.total%TYPE,
  p_metoda_plata OUT COMANDA.metoda_plata%TYPE,
  p_nr_produce OUT NUMBER,
  p_produce OUT VARCHAR2 ) IS

  CLIENT_NOT_PLACED_ORDER EXCEPTION;

BEGIN
  p_id_comanda := NULL;
  p_nume_angajat := NULL;
  p_total := NULL;
  p_metoda_plata := NULL;
  p_nr_produce := NULL;
  p_produce := NULL;

  SELECT dc.id_comanda, a.nume_angajat, c.total, c.metoda_plata,
         NVL(SUM(dc.cantitate), 0) AS "Nr. produse",
         NVL(LISTAGG(DISTINCT p.nume_produș, ', ') WITHIN GROUP (ORDER BY
p.nume_produș), 'No products') AS "Numele produselor"
    INTO p_id_comanda, p_nume_angajat, p_total, p_metoda_plata, p_nr_produce,
p_produce
  FROM CLIENT cl
 LEFT JOIN COMANDA c ON cl.id_client = c.id_client
 LEFT JOIN ANGAJAT a ON c.id_angajat = a.id_angajat
 LEFT JOIN DETALIU_COMANDA dc ON dc.id_comanda = c.id_comanda
 LEFT JOIN PRODUS p ON p.id_produș = dc.id_produș
 WHERE cl.nume_client = p_nume
 GROUP BY cl.nume_client, dc.id_comanda, a.nume_angajat, c.total,
c.metoda_plata;

```



```

IF p_total = 0 AND p_id_comanda is NULL THEN
    RAISE CLIENT_NOT_PLACED_ORDER;
END IF;
EXCEPTION
    WHEN CLIENT_NOT_PLACED_ORDER THEN
        DBMS_OUTPUT.PUT_LINE('Clientul nu a plasat o comanda!');
        RAISE_APPLICATION_ERROR(-20000, 'Clientul nu a plasat o comanda!');
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista acest client!');
        RAISE_APPLICATION_ERROR(-20001, 'Nu exista acest client!');
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu acest nume!');
        RAISE_APPLICATION_ERROR(-20002, 'Exista mai multi clienti cu acest
nume!');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('A aparut o eroare!');
        RAISE_APPLICATION_ERROR(-20003, 'A aparut o eroare!');
END Detalii_Comanda;
/

```

Procedure DETALII_COMANDA compiled

Apelarea procedurii.

1. Cazul în care afișează corect.

```

DECLARE
    v_id_comanda COMANDA.id_comanda%TYPE;
    v_num_e_angajat ANGAJAT.num_e_angajat%TYPE;
    v_total COMANDA.total%TYPE;
    v_metoda_plata COMANDA.metoda_plata%TYPE;
    v_nr_produce NUMBER;
    v_produce VARCHAR2(4000);
    v_num_e_client CLIENT.num_e_client%TYPE := 'Paun';
BEGIN
    Detalii_Comanda(v_num_e_client, v_id_comanda, v_num_e_angajat, v_total,
v_metoda_plata, v_nr_produce, v_produce);

    DBMS_OUTPUT.PUT_LINE('ID Comanda: ' || v_id_comanda);
    DBMS_OUTPUT.PUT_LINE('Nume angajat: ' || v_num_e_angajat);
    DBMS_OUTPUT.PUT_LINE('Total: ' || v_total);

```

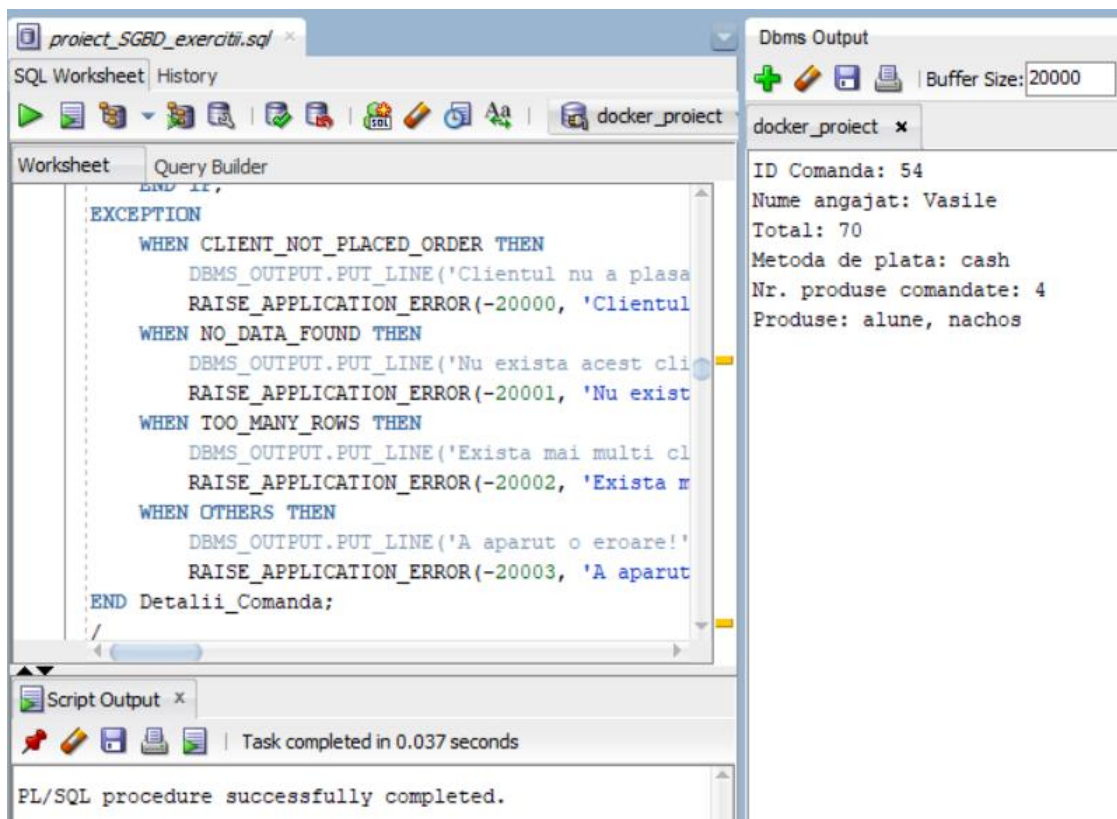
```

DBMS_OUTPUT.PUT_LINE('Metoda de plata: ' || v_metoda_plata);
DBMS_OUTPUT.PUT_LINE('Nr. produse comandate: ' || v_nr_produce);
DBMS_OUTPUT.PUT_LINE('Produse: ' || v_produce);

END;

/

```



2. Cazul în care numele clientului introdus nu există în baza de date.

DECLARE

```

v_id_comanda COMANDA.id_comanda%TYPE;
v_numa_angajat ANGAJAT.numa_angajat%TYPE;
v_total COMANDA.total%TYPE;
v_metoda_plata COMANDA.metoda_plata%TYPE;
v_nr_produce NUMBER;
v_produce VARCHAR2(4000);
v_numa_client CLIENT.numa_client%TYPE := 'Potlog';

```

BEGIN

```

    Detalii_Comanda(v_numa_client, v_id_comanda, v_numa_angajat, v_total,
v_metoda_plata, v_nr_produce, v_produce);

```

```

DBMS_OUTPUT.PUT_LINE('ID Comanda: ' || v_id_comanda);
DBMS_OUTPUT.PUT_LINE('Nume angajat: ' || v_numa_angajat);
DBMS_OUTPUT.PUT_LINE('Total: ' || v_total);
DBMS_OUTPUT.PUT_LINE('Metoda de plata: ' || v_metoda_plata);
DBMS_OUTPUT.PUT_LINE('Nr. produse comandate: ' || v_nr_produce);

```

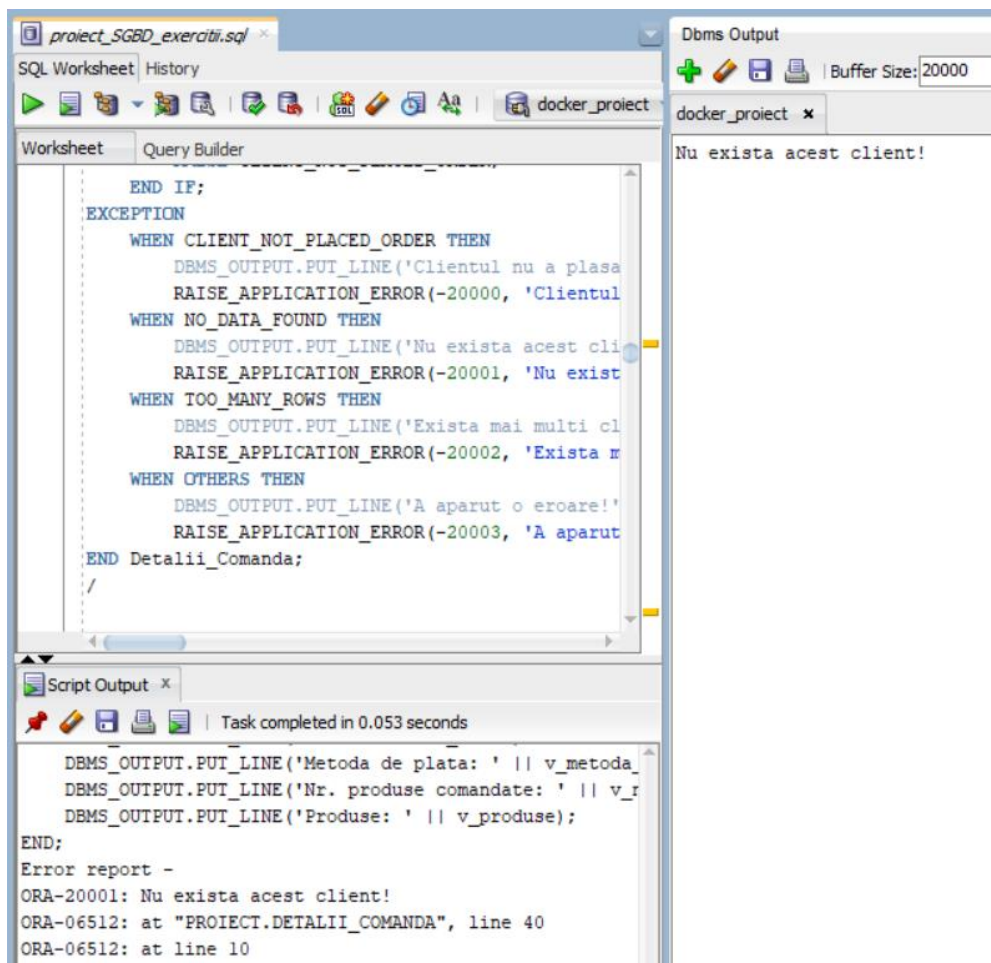
```

DBMS_OUTPUT.PUT_LINE('Produce: ' || v_produce);

END;

/

```



3. Cazul în care numele clientului introdus există de mai multe ori în baza de date.

```

DECLARE

```

```

    v_id_comanda COMANDA.id_comanda%TYPE;
    v_numa_angajat ANGAJAT.numa_angajat%TYPE;
    v_total COMANDA.total%TYPE;
    v_metoda_plata COMANDA.metoda_plata%TYPE;
    v_nr_produce NUMBER;
    v_produce VARCHAR2(4000);
    v_numa_client CLIENT.numa_client%TYPE := 'Pestritu';

```

```

BEGIN

```

```

    Detalii_Comanda(v_numa_client, v_id_comanda, v_numa_angajat, v_total,
v_metoda_plata, v_nr_produce, v_produce);

```

```

    DBMS_OUTPUT.PUT_LINE('ID Comanda: ' || v_id_comanda);
    DBMS_OUTPUT.PUT_LINE('Nume angajat: ' || v_numa_angajat);
    DBMS_OUTPUT.PUT_LINE('Total: ' || v_total);
    DBMS_OUTPUT.PUT_LINE('Metoda de plata: ' || v_metoda_plata);

```

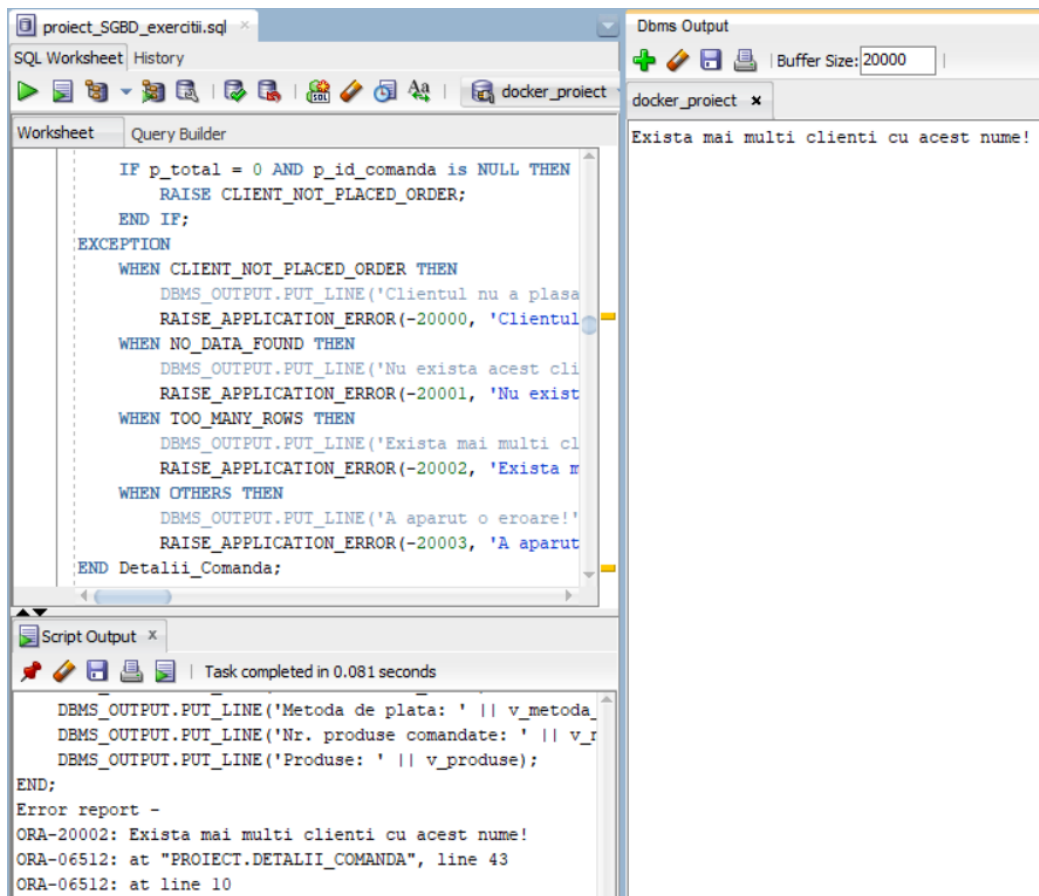
```

DBMS_OUTPUT.PUT_LINE('Nr. produse comandate: ' || v_nr_produce);
DBMS_OUTPUT.PUT_LINE('Produse: ' || v_produce);

END;

/

```



4. Cazul în care numele clientului introdus nu a plasat o comandă.

DECLARE

```

v_id_comanda COMANDA.id_comanda%TYPE;
v_numa_angajat ANGAJAT.numa_angajat%TYPE;
v_total COMANDA.total%TYPE;
v_metoda_plata COMANDA.metoda_plata%TYPE;
v_nr_produce NUMBER;
v_produce VARCHAR2(4000);
v_numa_client CLIENT.numa_client%TYPE := 'Dumitrescu';

```

BEGIN

```

    Detalii_Comanda(v_numa_client, v_id_comanda, v_numa_angajat, v_total,
v_metoda_plata, v_nr_produce, v_produce);

```

```

DBMS_OUTPUT.PUT_LINE('ID Comanda: ' || v_id_comanda);
DBMS_OUTPUT.PUT_LINE('Nume angajat: ' || v_numa_angajat);
DBMS_OUTPUT.PUT_LINE('Total: ' || v_total);
DBMS_OUTPUT.PUT_LINE('Metoda de plata: ' || v_metoda_plata);

```

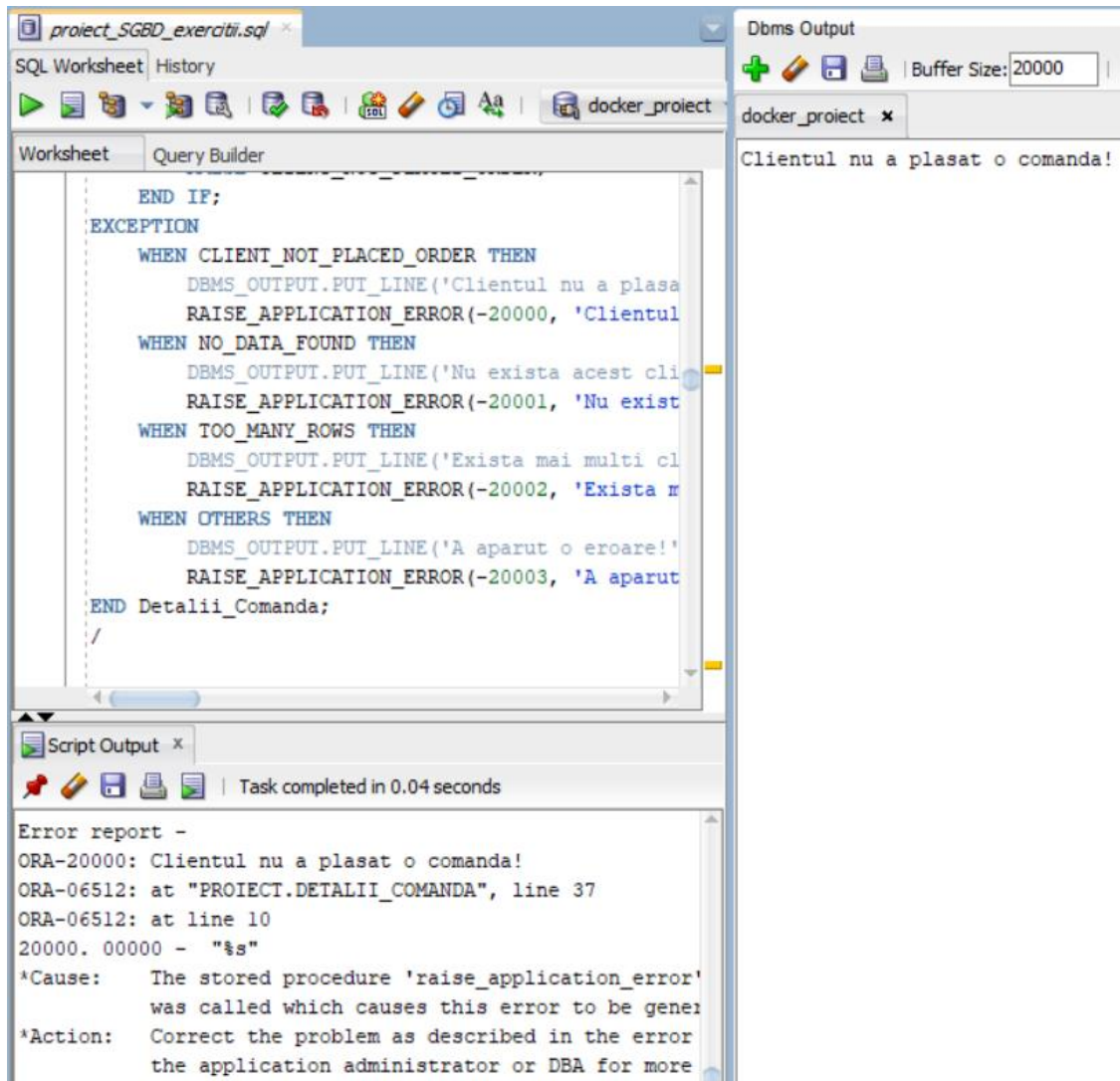
```

DBMS_OUTPUT.PUT_LINE('Nr. produse comandate: ' || v_nr_produce);
DBMS_OUTPUT.PUT_LINE('Produse: ' || v_produce);

END;

/

```



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

Enunț: Definiți un trigger care să permită lucrul asupra tabelului PROIECȚIE (INSERT, UPDATE, DELETE) în intervalul orar 10:00 - 20:00, de luni până vineri.

```

CREATE OR REPLACE TRIGGER trig_ex10
  BEFORE INSERT OR DELETE OR UPDATE on PROIECTIE
BEGIN
  IF (TO_CHAR(SYSDATE, 'D') IN (7, 1)) -- sambata si duminica
    OR (TO_CHAR(SYSDATE, 'HH24') NOT BETWEEN 10 AND 20) -- in afara
    intervalului orar 10:00 - 20:00
  THEN
    IF INSERTING THEN

```

```

        RAISE_APPLICATION_ERROR(-20001, 'Inserarea in tabel este permisa
doar in timpul programului de lucru!');

        ELSIF DELETING THEN

            RAISE_APPLICATION_ERROR(-20002, 'Stergerea din tabel este permisa
doar in timpul programului de lucru!');

        ELSE

            RAISE_APPLICATION_ERROR(-20003, 'Actualizarile in tabel sunt permise
doar in timpul programului de lucru!');

        END IF;

    END IF;

END;

/

```

Trigger TRIG_EX10 compiled

Declanșarea trigger-ului:

1. Pentru verificare (am modificat putin orele doar pentru rulare):

Comanda INSERT

```

SELECT TO_CHAR(SYSDATE, 'HH24:MI') AS current_hour, TO_CHAR(SYSDATE, 'DY') AS
current_day
FROM dual;

```

CURRENT_HOUR	CURRENT_DAY
09:59	6

```

INSERT INTO PROIECTIE
VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film =
'GhostBusters'), (SELECT id_sala FROM SALA WHERE nume = 'Galaxy'), '19:00');

```

The screenshot shows the SQL Developer interface. The top pane displays the SQL script for creating a trigger and inserting data. The bottom pane shows the execution results, including an error report.

```

CREATE OR REPLACE TRIGGER trig_ex10
BEFORE INSERT OR DELETE OR UPDATE on PROIECTIE
BEGIN
    IF (TO_CHAR(SYSDATE, 'D') IN (7, 1)) -- sambata si duminica
    OR (TO_CHAR(SYSDATE, 'HH24') NOT BETWEEN 10 AND 12) -- in afara intervalului orar 10:00 - 20:00
    THEN
        IF INSERTING THEN
            RAISE_APPLICATION_ERROR(-20001, 'Inserarea in tabel este permisa doar in timpul programului de lucru!');
        ELSIF DELETING THEN
            RAISE_APPLICATION_ERROR(-20002, 'Stergerea din tabel este permisa doar in timpul programului de lucru!');
        ELSE
            RAISE_APPLICATION_ERROR(-20000, 'Actualizarile in tabel sunt permise doar in timpul programului de lucru!');
        END IF;
    END IF;
END;

/

INSERT INTO PROIECTIE
VALUES (cinema_seq 5.nextval, (SELECT id_film FROM FILM WHERE titlu_film = 'GhostBusters'), (SELECT id_sala FROM SALA WHERE nume = 'Galaxy'), '19:00');

```

Task completed in 0.049 seconds

Error starting at line : 464 in command -

```

INSERT INTO PROIECTIE
VALUES (cinema_seq 5.nextval, (SELECT id_film FROM FILM WHERE titlu_film = 'GhostBusters'), (SELECT id_sala FROM SALA WHERE nume = 'Galaxy'), '19:00');

```

Error report -

```

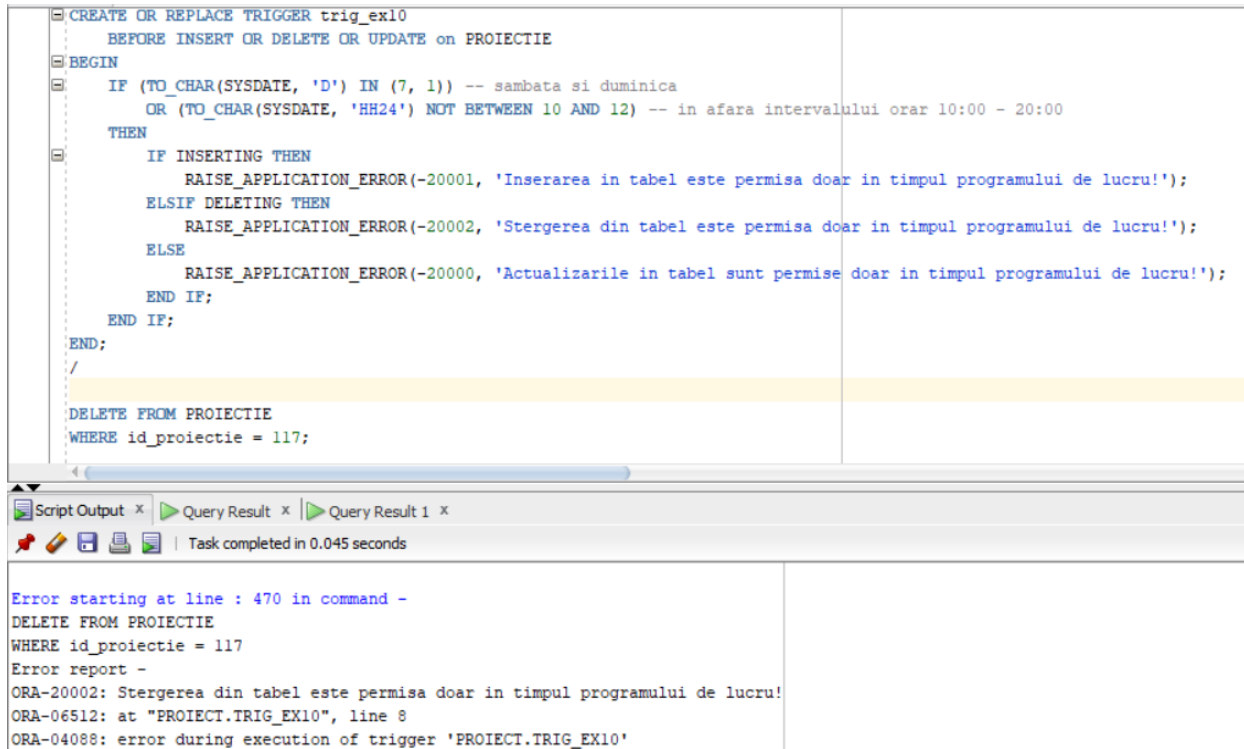
ORA-20001: Inserarea in tabel este permisa doar in timpul programului de lucru!
ORA-06512: at "PROIECT.TRIG_EX10", line 6
ORA-04088: error during execution of trigger 'PROIECT.TRIG_EX10'

```


Deși ziua curentă este Vineri (6) nu se poate face inserția în tabelul PROIECTIE, deoarece ora curentă (9:59) este în afara programului (10:00 – 20:00).

Comanda DELETE

```
DELETE FROM PROIECTIE
WHERE id_proiectie = 117;
```



The screenshot shows the SQL Developer interface. The top pane contains the following SQL code:

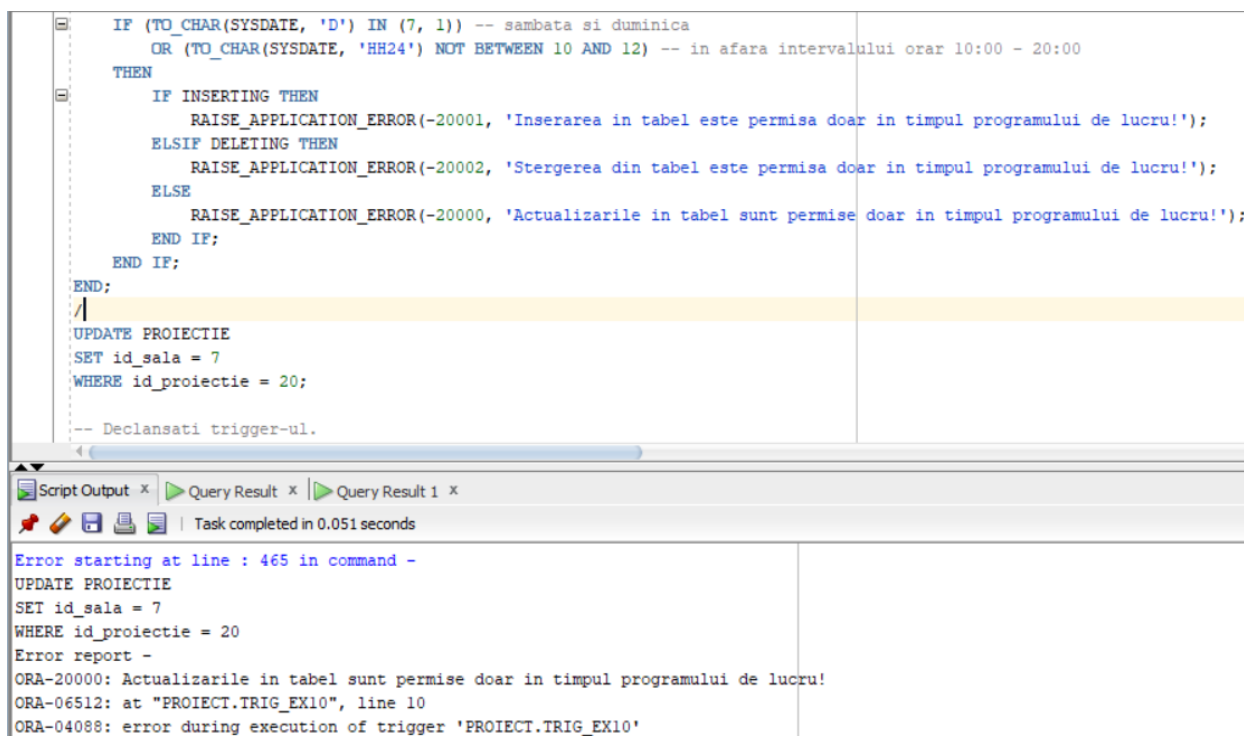
```
CREATE OR REPLACE TRIGGER trig_ex10
BEFORE INSERT OR DELETE OR UPDATE on PROIECTIE
BEGIN
    IF (TO_CHAR(SYSDATE, 'D') IN (7, 1)) -- sambata si duminica
        OR (TO_CHAR(SYSDATE, 'HH24') NOT BETWEEN 10 AND 12) -- in afara intervalului orar 10:00 - 20:00
    THEN
        IF INSERTING THEN
            RAISE_APPLICATION_ERROR(-20001, 'Inserarea in tabel este permisa doar in timpul programului de lucru!');
        ELSIF DELETING THEN
            RAISE_APPLICATION_ERROR(-20002, 'Stergerea din tabel este permisa doar in timpul programului de lucru!');
        ELSE
            RAISE_APPLICATION_ERROR(-20000, 'Actualizarile in tabel sunt permise doar in timpul programului de lucru!');
        END IF;
    END IF;
END;
```

The bottom pane shows the error report for the DELETE command:

```
Error starting at line : 470 in command -
DELETE FROM PROIECTIE
WHERE id_proiectie = 117
Error report -
ORA-20002: Stergerea din tabel este permisa doar in timpul programului de lucru!
ORA-06512: at "PROIECT.TRIG_EX10", line 8
ORA-04088: error during execution of trigger 'PROIECT.TRIG_EX10'
```

Comanda UPDATE

```
UPDATE PROIECTIE
SET id_sala = 7
WHERE id_proiectie = 20;
```



The screenshot shows the SQL Developer interface. The top pane contains the following SQL code:

```
IF (TO_CHAR(SYSDATE, 'D') IN (7, 1)) -- sambata si duminica
    OR (TO_CHAR(SYSDATE, 'HH24') NOT BETWEEN 10 AND 12) -- in afara intervalului orar 10:00 - 20:00
THEN
    IF INSERTING THEN
        RAISE_APPLICATION_ERROR(-20001, 'Inserarea in tabel este permisa doar in timpul programului de lucru!');
    ELSIF DELETING THEN
        RAISE_APPLICATION_ERROR(-20002, 'Stergerea din tabel este permisa doar in timpul programului de lucru!');
    ELSE
        RAISE_APPLICATION_ERROR(-20000, 'Actualizarile in tabel sunt permise doar in timpul programului de lucru!');
    END IF;
END IF;
END;
```

The bottom pane shows the error report for the UPDATE command:

```
Error starting at line : 465 in command -
UPDATE PROIECTIE
SET id_sala = 7
WHERE id_proiectie = 20
Error report -
ORA-20000: Actualizarile in tabel sunt permise doar in timpul programului de lucru!
ORA-06512: at "PROIECT.TRIG_EX10", line 10
ORA-04088: error during execution of trigger 'PROIECT.TRIG_EX10'
```


2. Pentru verificare:

```
SELECT TO_CHAR(SYSDATE, 'HH24:MI') AS current_hour, TO_CHAR(SYSDATE, 'DY') AS  
current_day  
FROM dual;
```

CURRENT_...	CURRENT_DAY
10:08	6

```
INSERT INTO PROIECTIE  
VALUES (cinema_seq.nextval, (SELECT id_film FROM FILM WHERE titlu_film =  
'GhostBusters'), (SELECT id_sala FROM SALA WHERE nume = 'Galaxy'), '19:00');  
UPDATE PROIECTIE  
SET id_sala = 7  
WHERE id_proiectie = 121;  
DELETE FROM PROIECTIE  
WHERE id_proiectie = 121;
```

The screenshot shows a SQL Developer window with a script editor and a results pane. The script defines a trigger named `trig_ex10` that fires before insert, delete, or update operations on the `PROIECTIE` table. The trigger logic checks if the operation is on a Saturday or Sunday, or if the time is outside the 10:00-20:00 interval. If the conditions are met, it raises an application error. The results pane shows the trigger was compiled successfully and that 1 row was inserted, 1 row was updated, and 1 row was deleted.

```
CREATE OR REPLACE TRIGGER trig_ex10  
BEFORE INSERT OR DELETE OR UPDATE on PROIECTIE  
BEGIN  
IF (TO_CHAR(SYSDATE, 'D') IN (7, 1)) -- sambata si duminica  
OR (TO_CHAR(SYSDATE, 'HH24') NOT BETWEEN 10 AND 20) -- in afara intervalului orar 10:00 - 20:00  
THEN  
IF INSERTING THEN  
RAISE_APPLICATION_ERROR(-20001, 'Inserarea in tabel este permisa doar in timpul programului de lucru!');  
ELSIF DELETING THEN  
RAISE_APPLICATION_ERROR(-20002, 'Stergerea din tabel este permisa doar in timpul programului de lucru!');  
ELSE  
RAISE_APPLICATION_ERROR(-20000, 'Actualizarile in tabel sunt permise doar in timpul programului de lucru!');  
END IF;  
END IF;
```

Trigger TRIG_EX10 compiled

1 row inserted.

1 row updated.

1 row deleted.

Insert

ID_PROIECTIE	ID_FILM	ID_SALA	ORA
20	13	6	10:00
21	17	7	15:00
22	16	9	20:30
23	18	12	17:30
24	17	10	18:00
25	15	11	18:00
26	14	8	12:30
27	19	9	21:00
28	15	6	12:00
29	13	7	11:30
108	15	7	19:00
121	15	8	19:00

Update

ID_PROIECTIE	ID_FILM	ID_SALA	ORA
20	13	6	10:00
21	17	7	15:00
22	16	9	20:30
23	18	12	17:30
24	17	10	18:00
25	15	11	18:00
26	14	8	12:30
27	19	9	21:00
28	15	6	12:00
29	13	7	11:30
108	15	7	19:00
121	15	7	19:00

Delete

ID_PROIECTIE	ID_FILM	ID_SALA	ORA
20	13	6	10:00
21	17	7	15:00
22	16	9	20:30
23	18	12	17:30
24	17	10	18:00
25	15	11	18:00
26	14	8	12:30
27	19	9	21:00
28	15	6	12:00
29	13	7	11:30
108	15	7	19:00

Insertia, actualizarea și ștergerea din tabelul PROIECȚIE au fost făcute corect.

Ștergerea trigger-ului:

```
DROP TRIGGER trig_ex10;
```

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

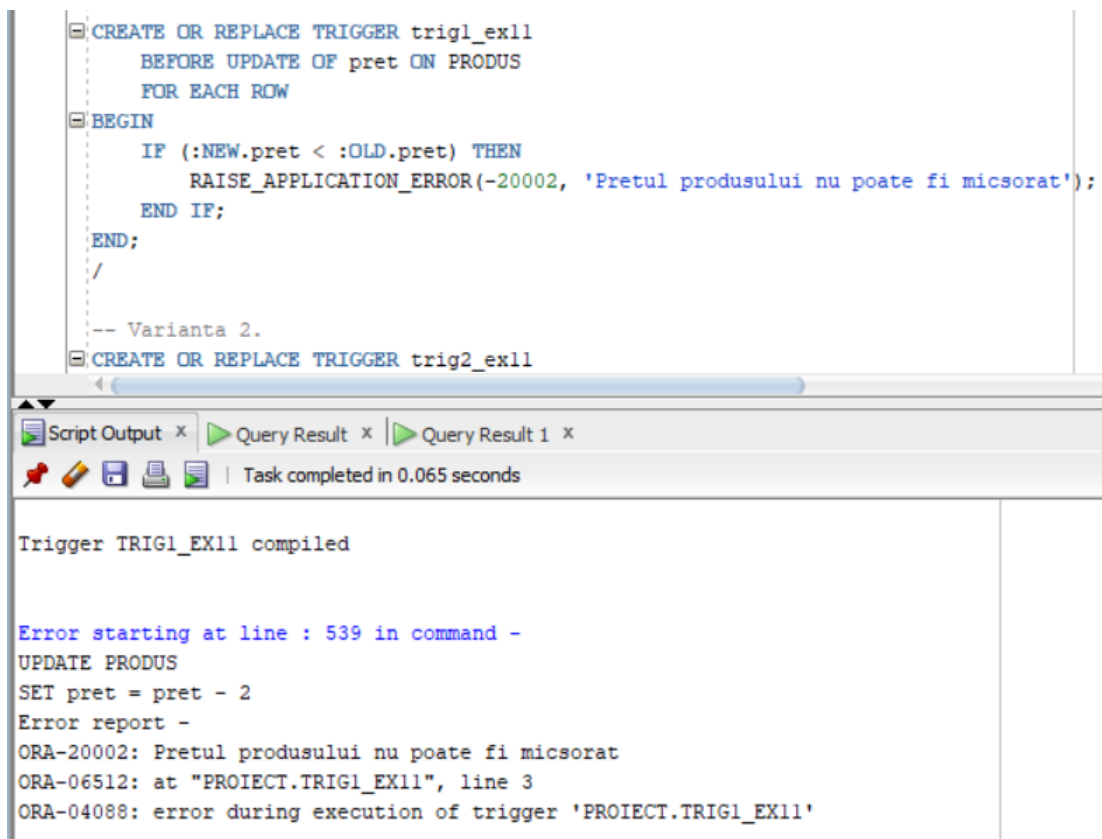
Enunț: Definiți un trigger prin care să nu se permită micșorarea prețurilor produselor din tabelul PRODUS.

Varianta 1:

```
CREATE OR REPLACE TRIGGER trig1_ex11
  BEFORE UPDATE OF pret ON PRODUS
  FOR EACH ROW
BEGIN
  IF (:NEW.pret < :OLD.pret) THEN
    RAISE_APPLICATION_ERROR(-20002, 'Pretul produsului nu poate fi
micsorat');
  END IF;
END;
/
```

Declanșarea trigger-ului:

```
UPDATE PRODUS
SET pret = pret - 2;
```



The screenshot displays the Oracle SQL Developer environment. The top pane shows the SQL script being executed, which includes the creation of a trigger and an update statement. The bottom pane shows the output of the execution, indicating that the trigger was compiled successfully but an error occurred during the execution of the update statement.

```
CREATE OR REPLACE TRIGGER trig1_ex11
  BEFORE UPDATE OF pret ON PRODUS
  FOR EACH ROW
BEGIN
  IF (:NEW.pret < :OLD.pret) THEN
    RAISE_APPLICATION_ERROR(-20002, 'Pretul produsului nu poate fi micsorat');
  END IF;
END;
/

-- Varianta 2.
CREATE OR REPLACE TRIGGER trig2_ex11
```

Script Output x Query Result x Query Result 1 x

Task completed in 0.065 seconds

Trigger TRIG1_EX11 compiled

Error starting at line : 539 in command -

```
UPDATE PRODUS
SET pret = pret - 2
```

Error report -

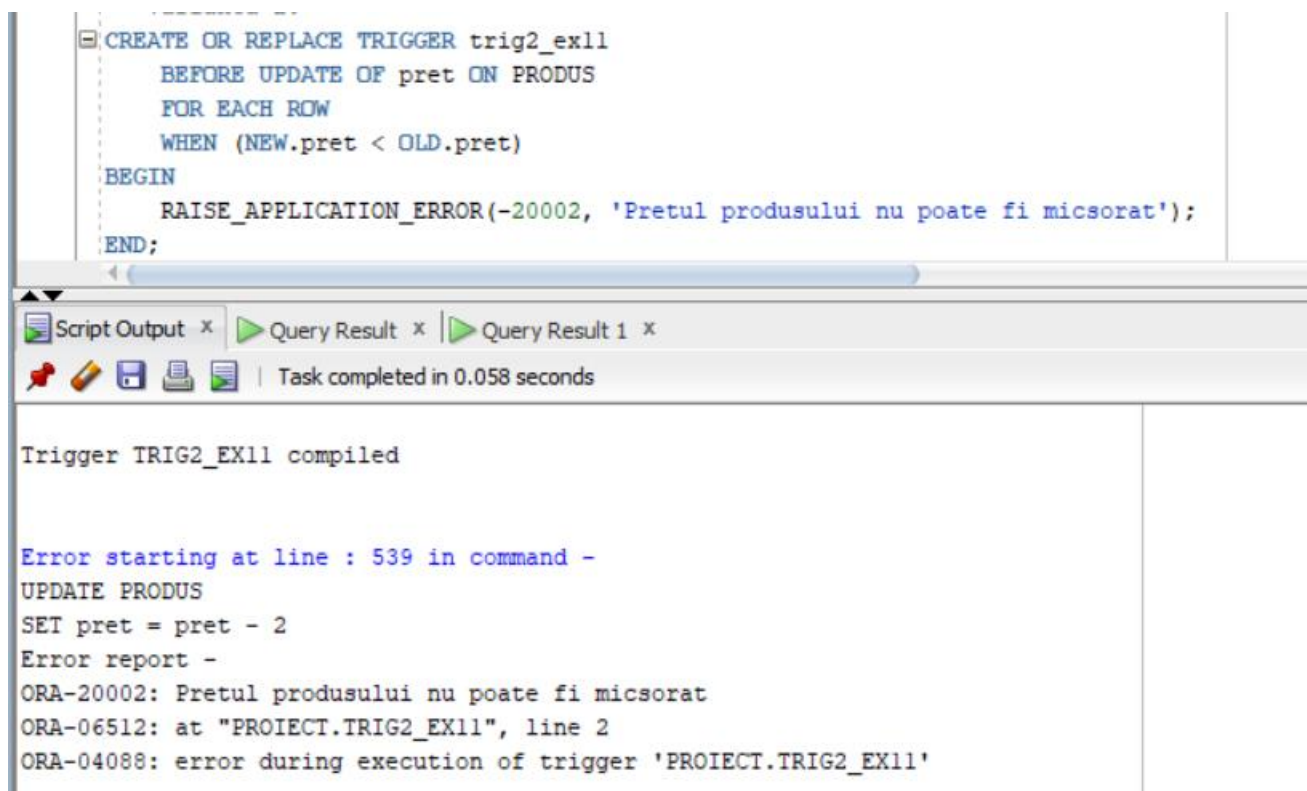
```
ORA-20002: Pretul produsului nu poate fi micsorat
ORA-06512: at "PROIECT.TRIG1_EX11", line 3
ORA-04088: error during execution of trigger 'PROIECT.TRIG1_EX11'
```

Varianta 2:

```
CREATE OR REPLACE TRIGGER trig2_ex11
    BEFORE UPDATE OF pret ON PRODUS
    FOR EACH ROW
    WHEN (NEW.pret < OLD.pret)
BEGIN
    RAISE_APPLICATION_ERROR(-20002, 'Pretul produsului nu poate fi micsorat');
END;
/
```

Declanșarea trigger-ului:

```
UPDATE PRODUS
SET pret = pret - 2;
```



Varianta 3 (cu procedură):

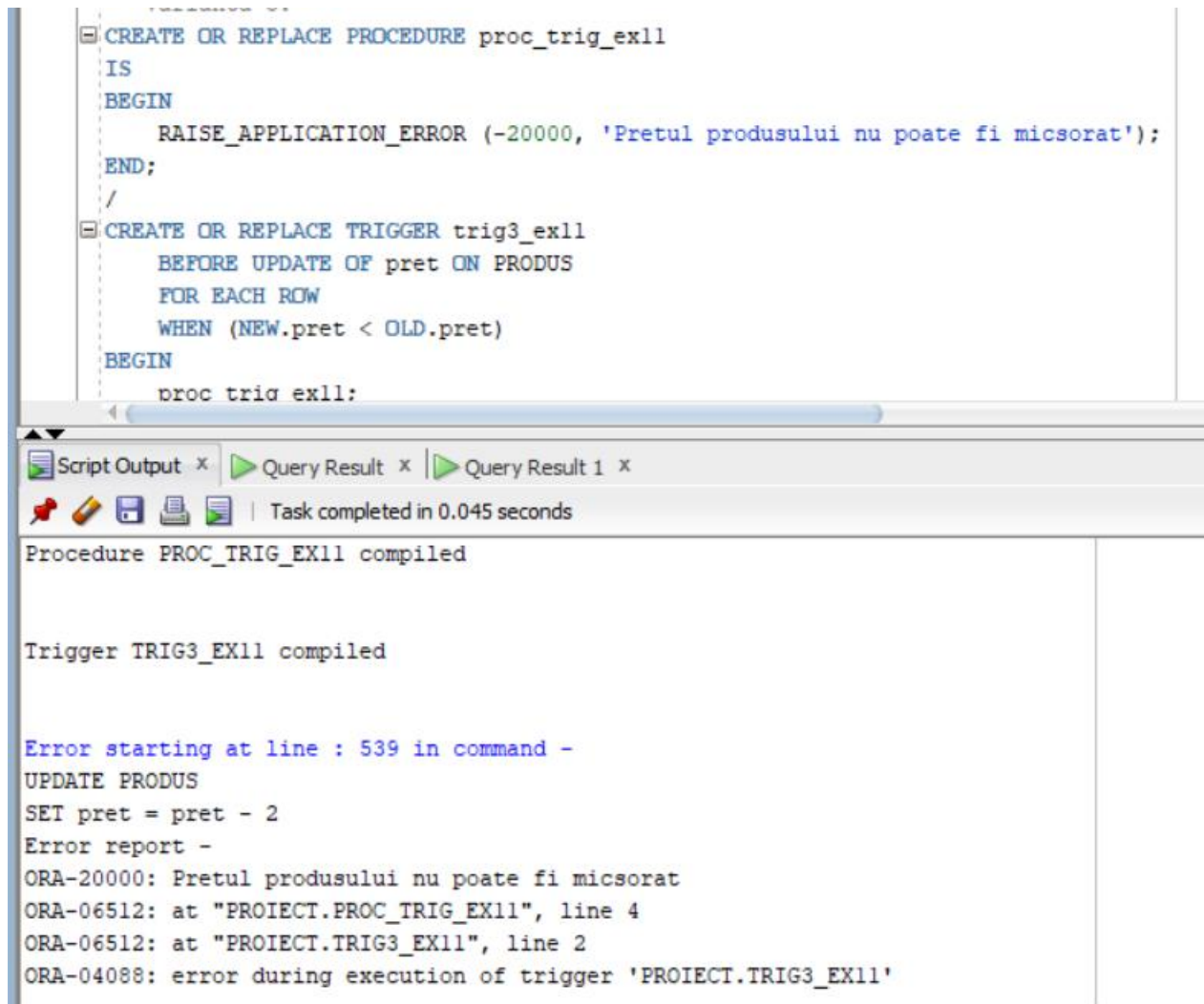
```
CREATE OR REPLACE PROCEDURE proc_trig_ex11
IS
BEGIN
    RAISE_APPLICATION_ERROR (-20000, 'Pretul produsului nu poate fi micsorat');
END;
/

CREATE OR REPLACE TRIGGER trig3_ex11
    BEFORE UPDATE OF pret ON PRODUS
    FOR EACH ROW
    WHEN (NEW.pret < OLD.pret)
```

```
BEGIN
    proc_trig_ex11;
END;
```

Declanșarea trigger-ului:

```
UPDATE PRODUS
SET pret = pret - 2;
```



```
CREATE OR REPLACE PROCEDURE proc_trig_ex11
IS
BEGIN
    RAISE_APPLICATION_ERROR (-20000, 'Pretul produsului nu poate fi micsorat');
END;
/
CREATE OR REPLACE TRIGGER trig3_ex11
BEFORE UPDATE OF pret ON PRODUS
FOR EACH ROW
WHEN (NEW.pret < OLD.pret)
BEGIN
    proc trig ex11;
END;
```

Script Output x | Query Result x | Query Result 1 x

Task completed in 0.045 seconds

Procedure PROC_TRIG_EX11 compiled

Trigger TRIG3_EX11 compiled

Error starting at line : 539 in command -
UPDATE PRODUS
SET pret = pret - 2
Error report -
ORA-20000: Pretul produsului nu poate fi micsorat
ORA-06512: at "PROIECT.PROC_TRIG_EX11", line 4
ORA-06512: at "PROIECT.TRIG3_EX11", line 2
ORA-04088: error during execution of trigger 'PROIECT.TRIG3_EX11'

Varianta 4 (cu procedură):

```
CREATE OR REPLACE TRIGGER trig4_ex11
BEFORE UPDATE OF pret ON PRODUS
FOR EACH ROW
WHEN (NEW.pret < OLD.pret)
CALL proc_trig_ex11
/
```

Declanșarea trigger-ului:

```
UPDATE PRODUS
SET pret = pret - 2;
```

```
CREATE OR REPLACE TRIGGER trig4_ex11
BEFORE UPDATE OF pret ON PRODUS
FOR EACH ROW
WHEN (NEW.pret < OLD.pret)
CALL proc_trig_ex11
/
```

Script Output x | Query Result x | Query Result 1 x

Task completed in 0.031 seconds

Trigger TRIG4_EX11 compiled

Error starting at line : 540 in command -
UPDATE PRODUS
SET pret = pret - 2
Error report -
ORA-20000: Pretul produsului nu poate fi micșorat
ORA-06512: at "PROIECT.PROC_TRIG_EX11", line 4
ORA-06512: at "PROIECT.TRIG3_EX11", line 2
ORA-04088: error during execution of trigger 'PROIECT.TRIG3_EX11'

Ștergerea triggerelor:

```
DROP TRIGGER trig1_ex11;
DROP TRIGGER trig2_ex11;
DROP TRIGGER trig3_ex11;
DROP TRIGGER trig4_ex11;
```

12. Definiți un trigger de tip LDD. Declanșați trigger-ul.

Enunț: Definiți un trigger care să introducă în tabelul nou definit EVENIMENT_EFECTUAT informații despre comanda LDD (CREATE, ALTER, DROP) folosită de utilizator, precum: numele bazei de date, numele utilizatorului, evenimentul, numele și tipul obiectului și data când s-a efectuat comanda.

```
CREATE TABLE EVENIMENTE_EFECTUATE
( baza_de_date VARCHAR2(50),
  nume_utilizator VARCHAR2(50),
  eveniment VARCHAR2(50),
  tip_obiect VARCHAR2(50),
  nume_obiect VARCHAR2(50),
  data DATE );
```

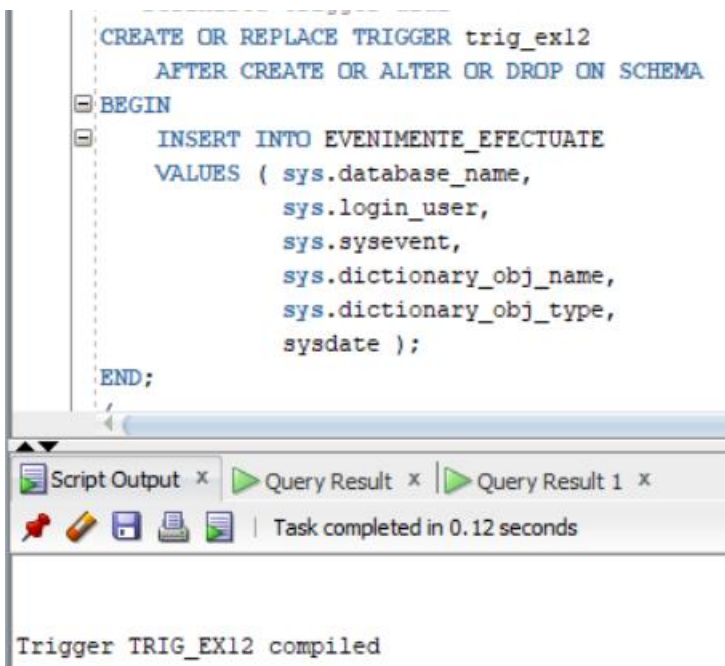
Table EVENIMENTE_EFECTUATE created.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
BAZA_DE_DATE	VARCHAR2(50 BYTE)	Yes	(null)	1	(null)
NUME_UTILIZATOR	VARCHAR2(50 BYTE)	Yes	(null)	2	(null)
EVENIMENT	VARCHAR2(50 BYTE)	Yes	(null)	3	(null)
TIP_OBIECT	VARCHAR2(50 BYTE)	Yes	(null)	4	(null)
NUME_OBIECT	VARCHAR2(50 BYTE)	Yes	(null)	5	(null)
DATA	DATE	Yes	(null)	6	(null)

```
CREATE OR REPLACE TRIGGER trig_ex12
  AFTER CREATE OR ALTER OR DROP ON SCHEMA
BEGIN
  INSERT INTO EVENIMENTE_EFECTUATE
VALUES ( sys.database_name,
        sys.login_user,
        sys.sysevent,
        sys.dictionary_obj_name,
        sys.dictionary_obj_type,
        sysdate );

END;

/
```



Declanșarea trigger-ului (comenzile efectuate):

```
CREATE TABLE TEST_TABEL(
  id_tabel NUMBER,
  nume_tabel VARCHAR2(50)
);
```

```
Table TEST_TABEL created.
```

```
ALTER TABLE TEST_TABEL
```

```
ADD (descriere varchar(50));
```

```
Table TEST_TABEL altered.
```

```
ALTER TABLE TEST_TABEL
```

```
DROP COLUMN descriere;
```

```
Table TEST_TABEL altered.
```

```
DROP TABLE TEST_TABEL;
```

```
Table TEST_TABEL dropped.
```

Informațiile din tabelul EVENIMENTE_EFECTUATE:

BAZA_DE_DATE	NUME_UTILIZATOR	EVENIMENT	TIP_OBIECT	NUME_OBIECT	DATA
ORCLPDB1	PROIECT	CREATE	TEST_TABEL	TABLE	08-JAN-24
ORCLPDB1	PROIECT	ALTER	TEST_TABEL	TABLE	08-JAN-24
ORCLPDB1	PROIECT	ALTER	TEST_TABEL	TABLE	08-JAN-24
ORCLPDB1	PROIECT	DROP	TEST_TABEL	TABLE	08-JAN-24

Ștergerea triggerelor:

```
DROP TRIGGER eveniment;
```

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

```
CREATE OR REPLACE PACKAGE pachet AS
```

```
    procedure Gestionare_Client
```

```
        ( var_num_e_trimis IN VARCHAR2,  
          var_prenume_trimis IN VARCHAR2 );
```

```
    procedure Afisare_Sali;
```

```
    function Actori_din_Film
```

```
        ( v_titlu_film FILM.titlu_film%TYPE DEFAULT 'GhostBusters' )  
    return VARCHAR2;
```

```
    procedure Detalii_Comanda
```

```
        ( p_num_e IN CLIENT.num_e_client%TYPE,  
          p_id_comanda OUT COMANDA.id_comanda%TYPE,  
          p_num_e_angajat OUT ANGAJAT.num_e_angajat%TYPE,  
          p_total OUT COMANDA.total%TYPE,  
          p_metoda_plata OUT COMANDA.metoda_plata%TYPE,  
          p_nr_produce OUT NUMBER,  
          p_produce OUT VARCHAR2 );
```

```
END pachet;
```

```
/
```

```
CREATE OR REPLACE PACKAGE BODY pachet AS
```

```
    PROCEDURE Gestionare_Client(
```

```
        var_nume_trimis IN VARCHAR2,
```

```
        var_prenume_trimis IN VARCHAR2
```

```
    ) IS
```

```
        TYPE tablou_indexat IS TABLE OF VARCHAR2(200) INDEX BY PLS_INTEGER;
```

```
        TYPE tablou_imbricat IS TABLE OF NUMBER;
```

```
        TYPE vector IS VARRAY(50) OF VARCHAR2(200);
```

```
        tablou_clienti_existenti tablou_indexat;
```

```
        tablou_preturi_comenzi tablou_imbricat := tablou_imbricat();
```

```
        vector_metode_plata vector := vector();
```

```
        var_exista_client BOOLEAN;
```

```
        var_suma_totala NUMBER := 0;
```

```
        var_metoda_plata_preferata VARCHAR2(10);
```

```
        v_index PLS_INTEGER := 1;
```

```
        v_prenume_db VARCHAR2(50);
```

```
        v_id_client CLIENT.id_client%TYPE;
```

```
        v_nr_plati_card NUMBER := 0;
```

```
        v_nr_plati_cash NUMBER := 0;
```

```
BEGIN
```

```
    SELECT nume_client
```

```
    BULK COLLECT INTO tablou_clienti_existenti
```

```
    FROM CLIENT;
```

```
    var_exista_client := FALSE;
```

```
    FOR i IN tablou_clienti_existenti.FIRST..tablou_clienti_existenti.LAST
```

```
LOOP
```

```
        IF tablou_clienti_existenti(i) = var_nume_trimis THEN
```

```
            var_exista_client := TRUE;
```



```

        SELECT prenume_client INTO v_prenume_db
        FROM CLIENT
        WHERE nume_client = var_nume_trimis;

        IF v_prenume_db <> var_prenume_trimis THEN
            var_exista_client := FALSE;
        END IF;
    END IF;
END LOOP;

IF var_exista_client = FALSE THEN
    DBMS_OUTPUT.PUT_LINE('Clientul nu exista.');
```

ELSE

```

    SELECT id_client INTO v_id_client
    FROM CLIENT
    WHERE nume_client = var_nume_trimis AND prenume_client =
var_prenume_trimis;

    SELECT total
    BULK COLLECT INTO tablou_preturi_comenzi
    FROM COMANDA
    WHERE id_client = v_id_client;

    FOR i IN tablou_preturi_comenzi.FIRST..tablou_preturi_comenzi.LAST
LOOP
        var_suma_totala := var_suma_totala + tablou_preturi_comenzi(i);
    END LOOP;

    SELECT metoda_plata
    BULK COLLECT INTO vector_metode_plata
    FROM COMANDA
    WHERE id_client = v_id_client;

    FOR i IN vector_metode_plata.FIRST..vector_metode_plata.LAST LOOP
        IF vector_metode_plata(i) = 'card' THEN
            v_nr_plati_card := v_nr_plati_card + 1;
        END IF;
        IF vector_metode_plata(i) = 'cash' THEN
```

```

        v_nr_plati_cash := v_nr_plati_cash + 1;
    END IF;
END LOOP;

IF v_nr_plati_card > v_nr_plati_cash THEN
    var_metoda_plata_preferata := 'Card';
END IF;

IF v_nr_plati_cash > v_nr_plati_card THEN
    var_metoda_plata_preferata := 'Cash';
END IF;

IF v_nr_plati_cash = v_nr_plati_card THEN
    var_metoda_plata_preferata := 'Card/Cash';
END IF;

    DBMS_OUTPUT.PUT_LINE('Suma      totala      a      comenzilor:      ' ||
var_suma_totala);
    DBMS_OUTPUT.PUT_LINE('Metoda      de      plata      preferata:      ' ||
var_metoda_plata_preferata);
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'A aparut o eroare!');

END Gestionare_Client;

PROCEDURE Afisare_Sali IS
    CURSOR CursorCinematografe IS
        SELECT      id_cinema      as      id_cinematograf,      nume_cinema      as
nume_cinematograf
        FROM CINEMA;

    CURSOR CursorSali (v_id_cinematograf CINEMA.id_cinema%type) IS
        SELECT nume as nume_sala, capacitate as capacitate_sala
        FROM SALA
        WHERE id_cinema = v_id_cinematograf;

    var_nume_sala SALA.nume%type;

```

```

var_capacitate_sala SALA.capacitate%type;
var_numar_sala NUMBER := 1;
var_numar_cinematograf NUMBER := 1;
BEGIN
    FOR i IN CursorCinematografe LOOP
        DBMS_OUTPUT.PUT_LINE('-----');
        DBMS_OUTPUT.PUT_LINE(var_numar_cinematograf || '. Cinema: ' ||
i.nume_cinematograf);
        OPEN CursorSali(i.id_cinematograf);
        LOOP
            FETCH CursorSali INTO var_nume_sala, var_capacitate_sala;
            EXIT WHEN CursorSali%notfound;
            DBMS_OUTPUT.PUT_LINE(' ' || var_numar_sala || '. Sala: ' ||
var_nume_sala || ' -> capacitate = ' || var_capacitate_sala || ');');
            var_numar_sala := var_numar_sala + 1;
        END LOOP;
        var_numar_sala := 1;
        var_numar_cinematograf := var_numar_cinematograf + 1;
        CLOSE CursorSali;
    END LOOP;
END Afisare_Sali;

FUNCTION Actori_din_Film
    (v_titlu_film FILM.titlu_film%TYPE DEFAULT 'GhostBusters')
RETURN VARCHAR2 IS
    v_rezultat VARCHAR2(100);
    v_cursor SYS_REFCURSOR;
    v_prenume_actor ACTOR.prenume_actor%TYPE;
    v_nume_actor ACTOR.nume_actor%TYPE;

    v_film_count NUMBER;
    v_actor_count NUMBER;

    titlul_filmului_nu_exista exception;
    titlul_filmului_nu_are_actori_introdusi exception;
BEGIN
    SELECT COUNT(*)
    INTO v_film_count
    FROM FILM

```

```
WHERE titlu_film = v_titlu_film;
```

```
IF v_film_count = 0 THEN
```

```
    RAISE titlul_filmului_nu_exista;
```

```
END IF;
```

```
OPEN v_cursor FOR
```

```
    SELECT a.prenume_actor, a.nume_actor
```

```
    FROM ACTOR a
```

```
    JOIN FILM_ACTOR fa ON a.id_actor = fa.id_actor
```

```
    JOIN FILM f ON fa.id_film = f.id_film
```

```
    WHERE f.titlu_film = v_titlu_film;
```

```
SELECT COUNT(*)
```

```
INTO v_actor_count
```

```
FROM ACTOR a
```

```
JOIN FILM_ACTOR fa ON a.id_actor = fa.id_actor
```

```
JOIN FILM f ON fa.id_film = f.id_film
```

```
WHERE f.titlu_film = v_titlu_film;
```

```
IF v_actor_count = 0 THEN
```

```
    CLOSE v_cursor;
```

```
    RAISE titlul_filmului_nu_are_actori_introdusi;
```

```
END IF;
```

```
v_rezultat := '';
```

```
LOOP
```

```
    FETCH v_cursor INTO v_prenume_actor, v_nume_actor;
```

```
    EXIT WHEN v_cursor%NOTFOUND;
```

```
    v_rezultat := v_rezultat || v_nume_actor || ' ' || v_prenume_actor  
|| ', ';
```

```
END LOOP;
```

```
CLOSE v_cursor;
```

```
v_rezultat := RTRIM(v_rezultat, ' ', '');
```

```

        RETURN v_rezultat;
EXCEPTION
    WHEN titlul_filmului_nu_are_actori_introdusi THEN
        DBMS_OUTPUT.PUT_LINE('Titlul filmului introdus nu are niciun actor
in baza de date.');
```

```

        RETURN 'exceptie';
    WHEN titlul_filmului_nu_exista THEN
        DBMS_OUTPUT.PUT_LINE('Titlul filmului introdus nu corespunde cu
titlul niciunui film din baza de date.');
```

```

        RETURN 'exceptie';
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Alta eroare' || SQLERRM);
        RETURN 'exceptie';
END Actori_din_Film;
```

```

PROCEDURE Detalii_Comanda
( p_nume IN CLIENT.nume_client%TYPE,
  p_id_comanda OUT COMANDA.id_comanda%TYPE,
  p_nume_angajat OUT ANGAJAT.nume_angajat%TYPE,
  p_total OUT COMANDA.total%TYPE,
  p_metoda_plata OUT COMANDA.metoda_plata%TYPE,
  p_nr_produce OUT NUMBER,
  p_produce OUT VARCHAR2 ) IS

    CLIENT_NOT_PLACED_ORDER EXCEPTION;
BEGIN
    p_id_comanda := NULL;
    p_nume_angajat := NULL;
    p_total := NULL;
    p_metoda_plata := NULL;
    p_nr_produce := NULL;
    p_produce := NULL;

    SELECT dc.id_comanda, a.nume_angajat, c.total, c.metoda_plata,
           NVL(SUM(dc.cantitate), 0) AS "Nr. produse",
           NVL(LISTAGG(DISTINCT p.nume_produ, ', ') WITHIN GROUP (ORDER BY
p.nume_produ), 'No products') AS "Numele produselor"
    INTO    p_id_comanda,    p_nume_angajat,    p_total,    p_metoda_plata,
p_nr_produce, p_produce
```

```

FROM CLIENT cl
LEFT JOIN COMANDA c ON cl.id_client = c.id_client
LEFT JOIN ANGAJAT a ON c.id_angajat = a.id_angajat
LEFT JOIN DETALIU_COMANDA dc ON dc.id_comanda = c.id_comanda
LEFT JOIN PRODUS p ON p.id_produs = dc.id_produs
WHERE cl.num_e_client = p_nume

GROUP BY cl.num_e_client, dc.id_comanda, a.num_e_angajat, c.total,
c.metoda_plata;

IF p_total = 0 AND p_id_comanda is NULL THEN
    RAISE CLIENT_NOT_PLACED_ORDER;
END IF;
EXCEPTION
    WHEN CLIENT_NOT_PLACED_ORDER THEN
        RAISE_APPLICATION_ERROR(-20003, 'Clientul nu a plasat o comanda!');
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista acest client!');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi clienti cu acest
nume!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');
END;
END pachet;
/

```

```
Package PACHET compiled
```

```
Package Body PACHET compiled
```

Apelarea procedurilor și funcțiilor din pachet și afișarea datelor rezultate:

```

DECLARE
    v_rezultat VARCHAR2(200);
    v_id_comanda COMANDA.id_comanda%TYPE;
    v_nume_angajat ANGAJAT.num_e_angajat%TYPE;
    v_total COMANDA.total%TYPE;
    v_metoda_plata COMANDA.metoda_plata%TYPE;
    v_nr_produce NUMBER;
    v_produce VARCHAR2(4000);

```

```
BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE('Exercitiul 6');  
    pachet.Gestionare_Client('Stan', 'Bianca');  
    DBMS_OUTPUT.PUT_LINE(null);
```

```
    DBMS_OUTPUT.PUT_LINE('Exercitiul 7');  
    pachet.Afisare_Sali;  
    DBMS_OUTPUT.PUT_LINE(null);
```

```
    DBMS_OUTPUT.PUT_LINE('Exercitiul 8');  
    v_rezultat := pachet.Actorii_din_Film('The Terminator');  
    DBMS_OUTPUT.PUT_LINE(v_rezultat);  
    DBMS_OUTPUT.PUT_LINE(null);
```

```
    DBMS_OUTPUT.PUT_LINE('Exercitiul 9');  
    pachet.Detalii_Comanda('Ionescu', v_id_comanda, v_numa_angajat, v_total,  
v_metoda_plata, v_nr_produce, v_produce);  
    DBMS_OUTPUT.PUT_LINE('ID Comanda: ' || v_id_comanda);  
    DBMS_OUTPUT.PUT_LINE('Nume Angajat: ' || v_numa_angajat);  
    DBMS_OUTPUT.PUT_LINE('Total: ' || v_total);  
    DBMS_OUTPUT.PUT_LINE('Metoda Plata: ' || v_metoda_plata);  
    DBMS_OUTPUT.PUT_LINE('Nr. Produce: ' || v_nr_produce);  
    DBMS_OUTPUT.PUT_LINE('Produce: ' || v_produce);
```

```
END;
```

```
/
```

proiect_SGBD_exercitii.sqlPROIECTIE

SQL WorksheetHistory

WorksheetQuery Builder

```
IF p_total = 0 AND p_id_comanda is NULL THEN
    RAISE CLIENT_NOT_PLACED_ORDER;
END IF;
EXCEPTION
    WHEN CLIENT_NOT_PLACED_ORDER THEN
        RAISE_APPLICATION_ERROR(-20003, 'Clientul nu a plasat o comanda!');
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista acest client!');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi clienti cu acest nume');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');
END;
END pachet;
/

-- Apelare pachet:
DECLARE
    v_resultat VARCHAR2(200);
    v_id_comanda COMANDA.id_comanda%TYPE;
    v_nume_angajat ANGAJATI.nume_angajat%TYPE;
```

Script OutputQuery ResultQuery Result 1

Task completed in 0.045 seconds

Package Body PACHET compiled

PL/SQL procedure successfully completed.

Dbms Output

docker_project x

Exercitiul 6

Suma totala a comenzilor: 120

Metoda de plata preferata: Card

Exercitiul 7

1. Cinema: Cinematica

1. Sala: Orion -> capacitate = 15;

2. Cinema: Flashback

1. Sala: Mega -> capacitate = 30;

3. Cinema: Pixelplex

1. Sala: Astral -> capacitate = 25;

4. Cinema: Retroplex

1. Sala: Ultra -> capacitate = 40;

2. Sala: Galaxy -> capacitate = 30;

5. Cinema: Starlight

1. Sala: Epika -> capacitate = 25;

2. Sala: Infinity -> capacitate = 50;

Exercitiul 8

Schwarzenegger Arnold, Hamilton Linda

Exercitiul 9

ID Comanda: 57

Nume Angajat: Popescu

Total: 50

Metoda Plata: cash

Nr. Produse: 2

Produse: nachos