

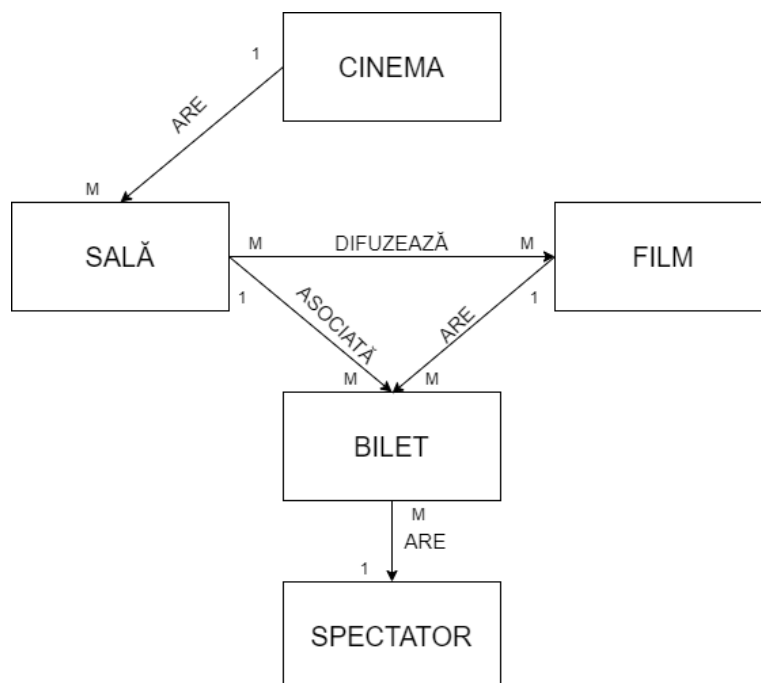
# **Sistem de gestiune a unui lanț de cinematografe**

realizat de Nimară Dan Gabriel, grupa 241

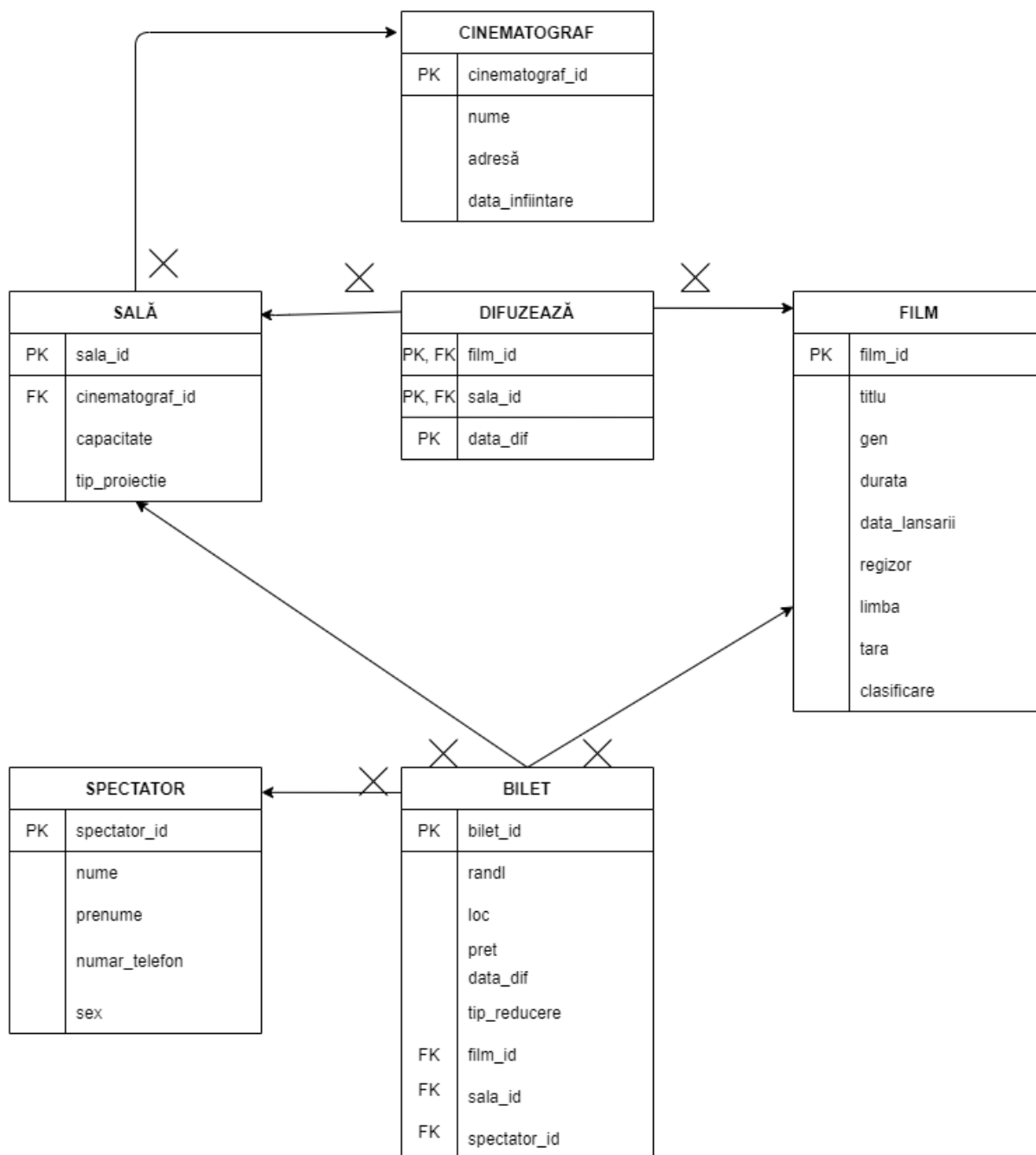
## 1. Prezențați pe scurt baza de date (utilitatea ei).

Datorită pasiunii mele pentru filme, am ales ca baza mea de date să fie despre gestiunea unui lanț de cinematografe. Am 5 entități independente(CINEMATOGRAF, SALA, FILM, BILET, SPECTATOR), dar și o tabelă asociativă între FILM și SALĂ(DIFUZEAZA). Tabelul cinematograf are cele mai puține date, iar de la el se formează întreaga rețea. Sălile aparțin fiecare de un anumit cinematograf, în ele pot fi difuzate unul sau mai multe filme, iar un film poate fi difuzat într-una sau mai multe săli, astfel a apărut necesitatea creării tabelii asociative pentru a modela relația M-M. În plus am definit un trigger care să nu permită introducerea unei difuzări a unui film în tabelul DIFUZEAZA la o dată anterioară datei lansării acestuia(reținută în FILM). Despre spectatori am reținut numele, prenumele, numărul de telefon și sexul. Ei dețin bilete la anumite filme, iar pe bilet reținem toate datele necesare precum locul, rândul, prețul, data difuzării, id-ul sălii, id-ul filmului și id-ul spectatorului. Pentru film am ales să am și un tabel imbricat ca atribut pentru a reține genurile acestuia(două-trei). Toate datele despre filme sunt consistente cu realitatea, cele despre cinematografe sunt tributuri la regizori importanți de filme plus o referință la serialul meu favorit Doctor Who. Prețurile билетelor le-am adaptat pentru fiecare sală și în funcție de reducerile aplicabile pe baza prețurilor lanțului Cinema City.

2.



3.



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

```
CREATE TABLE cinematograf
```

```
(cinematograf_id number(4) constraint pk_cin primary key,  
  nume varchar2(50) constraint nume_cin not null,  
  adresa varchar2(100) not null,  
  data_infiintare date default sysdate  
);
```

```
CREATE TABLE sala
```

```
(sala_id number(4) constraint pk_sala primary key,  
  capacitate number(3) constraint cap_sala not null,  
  tip_proiectie varchar(10) default '2D',  
  cinematograf_id number(4) constraint fk_sala references cinematograf(cinematograf_id) ON DELETE CASCADE  
);
```

```
CREATE OR REPLACE TYPE tab_imb IS TABLE OF varchar2(20);
```

```
/
```

```
CREATE TABLE film
```

```
(film_id number(4) constraint pk_film primary key,  
  titlu varchar2(225) constraint titlu_film not null,  
  gen tab_imb default tab_imb(),  
  durata number(3),  
  data_lansarii date default sysdate,  
  regizor varchar2(50),  
  limba varchar2(3),  
  tara varchar2(20),  
  clasificare varchar2(9)  
)
```

```
NESTED TABLE gen STORE AS tab_imb_gen;
```

```
CREATE TABLE difuzeaza
```

```
(film_id number(4) constraint fk_dif_film references film(film_id) ON DELETE CASCADE,  
  sala_id number(4) constraint fk_dif_sala references sala(sala_id) ON DELETE CASCADE,  
  data_dif date default sysdate,  
  primary key(film_id, sala_id, data_dif)  
);
```

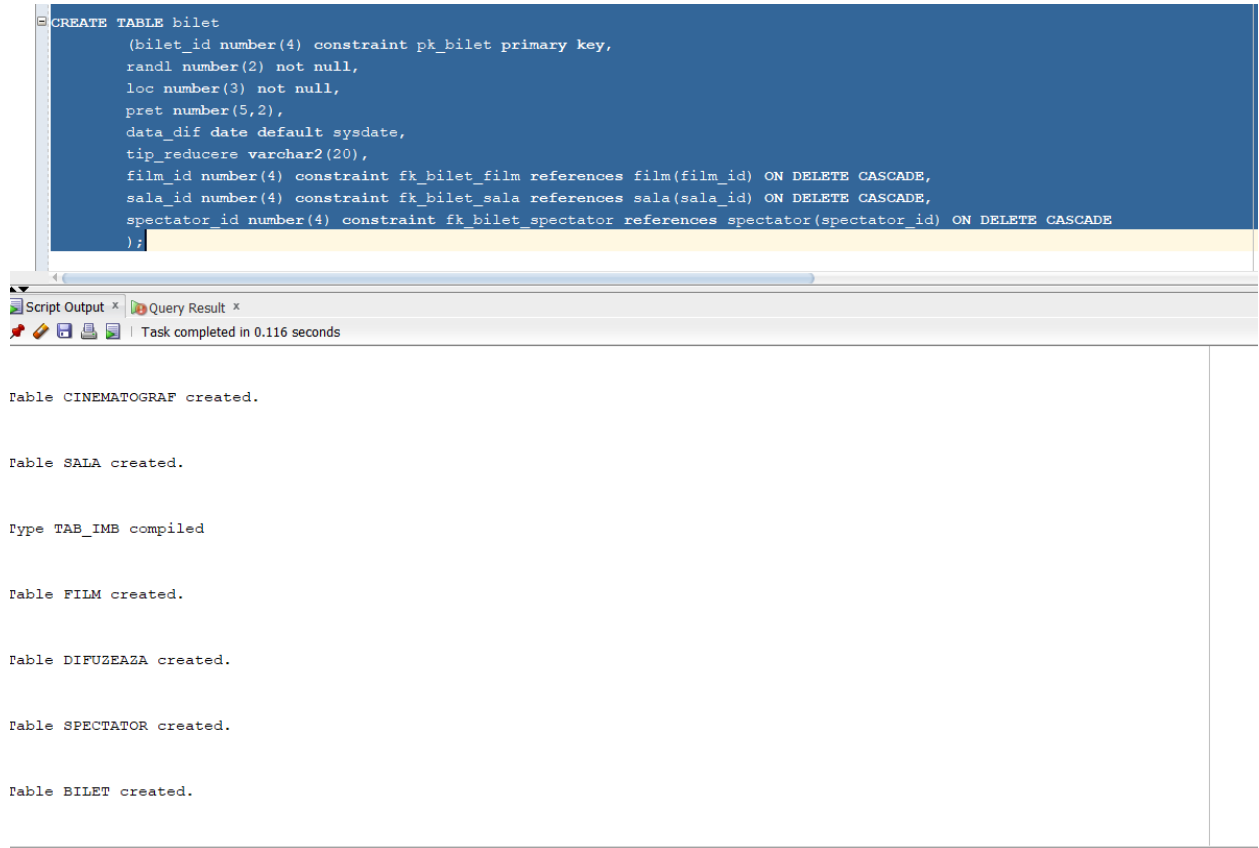
```
CREATE TABLE spectator
```

```
(spectator_id number(4) constraint pk_spectator primary key,  
  nume varchar2(20) constraint nume_spectator not null,  
  prenume varchar2(20),  
  numar_telefon char(10) unique,  
  sex char(1)  
);
```

```
CREATE TABLE bilet
```

```
(bilet_id number(4) constraint pk_bilet primary key,  
  randl number(2) not null,  
  loc number(3) not null,  
  pret number(5,2),  
  data_dif date default sysdate,
```

```
tip_reducere varchar2(20),
film_id number(4) constraint fk_bilet_film references film(film_id) ON DELETE CASCADE,
sala_id number(4) constraint fk_bilet_sala references sala(sala_id) ON DELETE CASCADE,
spectator_id number(4) constraint fk_bilet_spectator references spectator(spectator_id) ON DELETE CASCADE
);
```



The screenshot shows a SQL IDE window with a script editor and a results pane. The script editor contains the following SQL code:

```
CREATE TABLE bilet
(
    bilet_id number(4) constraint pk_bilet primary key,
    randl number(2) not null,
    loc number(3) not null,
    pret number(5,2),
    data_dif date default sysdate,
    tip_reducere varchar2(20),
    film_id number(4) constraint fk_bilet_film references film(film_id) ON DELETE CASCADE,
    sala_id number(4) constraint fk_bilet_sala references sala(sala_id) ON DELETE CASCADE,
    spectator_id number(4) constraint fk_bilet_spectator references spectator(spectator_id) ON DELETE CASCADE
);
```

The results pane shows the output of the script execution:

```
Table CINEMATOGRAF created.

Table SALA created.

Type TAB_IMB compiled

Table FILM created.

Table DIFUZEAZA created.

Table SPECTATOR created.

Table BILET created.
```

## 5. Adăugați informații coerente în tabelele create (minim 3-5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

--Atunci cand se introduce o noua difuzare, trebuie sa ne asiguram ca data difuzarii nu preceda data lansarii filmului.

```
CREATE OR REPLACE TRIGGER difuzare_permisa
BEFORE INSERT OR UPDATE ON difuzeaza
FOR EACH ROW
DECLARE
    lansare film.data_lansarii%TYPE;
BEGIN
    SELECT data_lansarii
    INTO lansare
    FROM film
    WHERE film_id = :NEW.film_id;
    IF :NEW.data_dif < lansare THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu se poate difuza un film inainte de data lansarii acestuia!');
    END IF;
```

```
END;  
/  
--ALTER TRIGGER difuzare_permisa DISABLE;  
--ALTER TRIGGER difuzare_permisa ENABLE;
```

```
--Atunci cand se introduce o noua difuzare, trebuie sa ne asiguram ca data difuzarii nu preceda data lansarii filmului.  
  
CREATE OR REPLACE TRIGGER difuzare_permisa  
BEFORE INSERT OR UPDATE ON difuzeaza  
FOR EACH ROW  
DECLARE  
    lansare film.data_lansarii%TYPE;  
BEGIN  
    SELECT data_lansarii  
    INTO lansare  
    FROM film  
    WHERE film_id = :NEW.film_id;  
    IF :NEW.data_dif < lansare THEN  
        RAISE_APPLICATION_ERROR(-20000, 'Nu se poate difuza un film inainte de data lansarii acestuia!');  
    END IF;  
END;  
/  
  
Script Output x  
Task completed in 0.342 seconds  
  
Trigger DIFUZARE_PERMISA compiled
```

```
INSERT INTO cinematograf  
VALUES(1, 'Jay and Silent Bob Multiplex', '58 Leonard Avenue, Leonardo', '19-OCT-1994');
```

```
INSERT INTO cinematograf  
VALUES(2, 'Hitchcock Multiplex', '517 High Road, Leytonstone, London', '13-AUG-1899');
```

```
INSERT INTO cinematograf  
VALUES(3, 'The Supercalifragilistic Cinema Experience', '500 S. Buena Vista St., Burbank, CA', '27-AUG-1964');
```

```
INSERT INTO cinematograf  
VALUES(4, 'Gisaengchung Cinema', 'Bongdeok-dong, Daegu, South Korea', '19-FEB-2000');
```

```
INSERT INTO cinematograf  
VALUES(5, 'Inglorious, Fictitious and Unchained Feet Cinema', '1822 Sepulveda Blvd., Manhattan Beach, CA, USA', '09-OCT-1992');
```

```
INSERT INTO cinematograf  
VALUES(6, 'Hitchcock Multiplex', 'Bel Air, Los Angeles, CA, USA', '29-APR-1980');
```

```
INSERT INTO cinematograf  
VALUES(7, 'Time and Relative Dimensions in Cinema', 'White City, W12 7RJ, London, UK', '23-NOV-1963');
```

```
SELECT * FROM cinematograf;
```

```
INSERT INTO sala  
VALUES(1, 120, '2D', 1);
```

```
INSERT INTO sala  
VALUES(2, 160, '3D', 1);
```

```
INSERT INTO sala  
VALUES(3, 200, '3D', 2);
```

## Nimară Dan Gabriel, grupa 241

```
INSERT INTO sala
VALUES(4, 300, 'IMAX 3D', 2);
```

```
INSERT INTO sala
VALUES(5, 100, '4DX 2D', 5);
```

```
INSERT INTO sala
VALUES(6, 25, 'VIP 2D', 3);
```

```
INSERT INTO sala
VALUES(7, 25, 'VIP 3D', 3);
```

```
INSERT INTO sala
VALUES(8, 160, '3D', 4);
```

```
INSERT INTO sala
VALUES(9, 150, '4DX 3D', 1);
```

```
INSERT INTO sala
VALUES(10, 100, 'IMAX 2D', 2);
```

```
INSERT INTO sala
VALUES(11, 250, '2D', 6);
```

```
INSERT INTO sala
VALUES(12, 110, 'IMAX 2D', 7);
```

```
INSERT INTO sala
VALUES(13, 215, '3D', 6);
```

```
SELECT * FROM sala;
```

```
INSERT INTO film
VALUES(1, 'Back to the Future', tab_imb('Adventure', 'Comedy', 'Sci-Fi'), 116, '19-JUL-1985', 'Robert Zemeckis', 'EN', 'USA', 'PG-13');
```

```
INSERT INTO film
VALUES(2, 'It's A Wonderful Life', tab_imb('Drama', 'Family', 'Fantasy'), 131, '20-DEC-1946', 'Frank Capra', 'EN', 'USA', 'PG');
```

```
INSERT INTO film
VALUES(3, 'Gone Girl', tab_imb('Drama', 'Mystery', 'Thriller'), 149, '03-OCT-2014', 'David Fincher', 'EN', 'USA', 'R');
```

```
INSERT INTO film
VALUES(4, 'Psycho', tab_imb('Horror', 'Mystery', 'Thriller'), 109, '08-SEP-1960', 'Alfred Hitchcock', 'EN', 'USA', 'R');
```

```
INSERT INTO film
VALUES(5, 'Baby Driver', tab_imb('Action', 'Crime', 'Music'), 113, '28-JUN-2017', 'Edgar Wright', 'EN', 'USA', 'R');
```

```
INSERT INTO film
VALUES(6, 'Knives Out', tab_imb('Comedy', 'Crime', 'Drama'), 130, '27-NOV-2019', 'Rian Johnson', 'EN', 'USA', 'PG-13');
```

```
INSERT INTO film
VALUES(7, 'Parasite', tab_imb('Comedy', 'Drama', 'Thriller'), 132, '30-MAY-2019', 'Bong Joon-ho', 'KO', 'South Korea', 'R');
```

```
INSERT INTO film
VALUES(8, 'Forrest Gump', tab_imb('Drama', 'Romance'), 142, '06-JUL-1994', 'Robert Zemeckis', 'EN', 'USA', 'PG-13');
```

## Nimară Dan Gabriel, grupa 241

```
INSERT INTO film
VALUES(9, 'Metropolis', tab_imb('Drama', 'Sci-Fi'), 153, '10-JAN-1927', 'Fritz Lang', 'DE', 'Germany', 'PG-13');

INSERT INTO film
VALUES(10, '12 Angry Men', tab_imb('Crime', 'Drama'), 96, '10-APR-1957', 'Sidney Lumet', 'EN', 'USA', 'PG-13');

INSERT INTO film
VALUES(11, 'A Trip to the Moon', tab_imb('Short', 'Adventure', 'Comedy'), 9, '01-SEP-1902', 'Georges Méliès', 'SIL', 'France', 'G');

INSERT INTO film
VALUES(12, 'City Lights', tab_imb('Comedy', 'Drama', 'Romance'), 87, '07-MAR-1931', 'Charles Chaplin', 'EN', 'USA', 'G');

SELECT * FROM film;

INSERT INTO difuzeaza
VALUES(1, 2, sysdate-30);

INSERT INTO difuzeaza
VALUES(2, 1, sysdate-2.3);

INSERT INTO difuzeaza
VALUES(2, 2, sysdate-5.6);

INSERT INTO difuzeaza
VALUES(3, 1, sysdate-7);

INSERT INTO difuzeaza
VALUES(3, 4, sysdate-10.1);

INSERT INTO difuzeaza
VALUES(4, 5, sysdate-1.2);

INSERT INTO difuzeaza
VALUES(5, 5, sysdate-15);

INSERT INTO difuzeaza
VALUES(5, 3, sysdate-13.71);

INSERT INTO difuzeaza
VALUES(1, 3, sysdate-4.3);

INSERT INTO difuzeaza
VALUES(5, 1, sysdate-0.3);

INSERT INTO difuzeaza
VALUES(3, 2, sysdate-132.3);
--
INSERT INTO difuzeaza
VALUES(10, 8, sysdate-451.8);

INSERT INTO difuzeaza
VALUES(7, 8, sysdate-512.6);

INSERT INTO difuzeaza
VALUES(9, 1, sysdate-645.04);

INSERT INTO difuzeaza
```



## Nimară Dan Gabriel, grupa 241

```
VALUES(2, 10, sysdate-99.7);

INSERT INTO difuzeaza
VALUES(8, 4, sysdate-222.1);

INSERT INTO difuzeaza
VALUES(5, 3, sysdate-1.13);

-- Nu se poate difuza un film inainte de data lansarii acestuia!
--INSERT INTO difuzeaza
--    VALUES(6, 6, sysdate-873.6);

INSERT INTO difuzeaza
VALUES(9, 5, sysdate-451);

INSERT INTO difuzeaza
VALUES(7, 9, sysdate-51);

INSERT INTO difuzeaza
VALUES(4, 2, sysdate-11);

INSERT INTO difuzeaza
VALUES(7, 1, sysdate-332.8);
----
INSERT INTO difuzeaza
VALUES(2, 13, sysdate-123.2);

INSERT INTO difuzeaza
VALUES(9, 12, sysdate-570);

INSERT INTO difuzeaza
VALUES(10, 11, sysdate-3.5);

INSERT INTO difuzeaza
VALUES(1, 7, sysdate-100);

SELECT film_id, sala_id, TO_CHAR(data_dif,'DD-MM-YYYY HH:MM:SS') FROM difuzeaza;

INSERT INTO spectator
VALUES(1, 'Nimara', 'Dan-Gabriel', '0762669402', 'M');

INSERT INTO spectator
VALUES(2, 'Alexandrescu', 'Paula', '0799168286', 'F');

INSERT INTO spectator
VALUES(3, 'Barbu', 'Mariana-Lenuta', '0748984069', 'F');

INSERT INTO spectator
VALUES(4, 'Oprian', 'George-Adrian', '0743441168', 'M');

INSERT INTO spectator
VALUES(5, 'Pavalasc', 'Irina', '0756161429', 'F');

INSERT INTO spectator
VALUES(6, 'Negulescu', 'Radu', '0755004495', 'M');

INSERT INTO spectator
```

## Nimară Dan Gabriel, grupa 241

```
VALUES(7, 'Lapadus', 'Raluca', '0723161118', 'F');

INSERT INTO spectator
VALUES(8, 'Simion', 'Roberto-Florian', '0720617882', 'M');

INSERT INTO spectator
VALUES(9, 'Apostoiu', 'Antonio-Ciprian', '0727761387', 'M');

INSERT INTO spectator
VALUES(10, 'Varga', 'Robert', '0767251023', 'M');

INSERT INTO spectator
VALUES(11, 'Dima', 'Oana-Teodora', '0720309259', 'F');

INSERT INTO spectator
VALUES(12, 'Bigan', 'Marian', '0720078066', 'M');

INSERT INTO spectator
VALUES(13, 'Moanga', 'Natalia', '07XXXXXXX', 'F');

SELECT * FROM spectator;

INSERT INTO bilet
VALUES(1, 12, 14, 41, sysdate-1.2, NULL, 4, 5, 1);

INSERT INTO bilet
VALUES(2, 6, 12, 19.5, sysdate-2.3, 'Pensionar', 2, 1, 12);

INSERT INTO bilet
VALUES(3, 5, 12, 17.5, sysdate-15, 'Student', 5, 5, 10);

INSERT INTO bilet
VALUES(4, 14, 10, 41, sysdate-4.3, NULL, 1, 3, 3);

INSERT INTO bilet
VALUES(5, 8, 9, 16.5, sysdate-2.3, 'Copil', 2, 1, 11);

INSERT INTO bilet
VALUES(6, 11, 2, 25.5, sysdate-30, NULL, 1, 2, 6);

INSERT INTO bilet
VALUES(7, 6, 8, 46, sysdate-51, NULL, 7, 9, 8);
--nou
INSERT INTO bilet
VALUES(8, 16, 8, 23.5, sysdate-7, NULL, 3, 1, 6);

INSERT INTO bilet
VALUES(9, 12, 14, 21.5, sysdate-1.13, 'Pensionar', 5, 3, 4);

INSERT INTO bilet
VALUES(10, 5, 7, 42.5, sysdate-451, 'Avanpremiera', 9, 5, 1);

INSERT INTO bilet
VALUES(11, 3, 10, 66, sysdate-873.6, 'Pensionar', 6, 6, 12);

INSERT INTO bilet
VALUES(12, 10, 4, 20.5, sysdate-99.7, 'Student', 2, 10, 5);
```

## Nimară Dan Gabriel, grupa 241

```
INSERT INTO bileț
VALUES(13, 2, 6, 70, sysdate-100, NULL, 1, 7, 3);

INSERT INTO bileț
VALUES(14, 13, 13, 25.5, sysdate-1.13, NULL, 5, 3, 13);

INSERT INTO bileț
VALUES(15, 10, 8, 41, sysdate-15, NULL, 5, 5, 2);

INSERT INTO bileț
VALUES(16, 9, 11, 18.5, sysdate-99.7, NULL, 2, 10, 11);

INSERT INTO bileț
VALUES(17, 15, 12, 25.5, sysdate-5.6, NULL, 2, 2, 3);

INSERT INTO bileț
VALUES(18, 6, 16, 17.5, sysdate-645.04, 'Student', 9, 1, 3);

INSERT INTO bileț
VALUES(19, 8, 10, 21.5, sysdate-512.6, 'Student', 7, 8, 9);

INSERT INTO bileț
VALUES(20, 10, 18, 29.5, sysdate-222.1, NULL, 8, 4, 13);

INSERT INTO bileț
VALUES(21, 6, 17, 23.5, sysdate-645.04, NULL, 9, 1, 1);

INSERT INTO bileț
VALUES(22, 7, 9, 19.5, sysdate-3.5, 'Pensionar', 10, 11, 12);

INSERT INTO bileț
VALUES(23, 9, 10, 21.5, sysdate-570, 'Pensionar', 9, 12, 4);

SELECT * FROM bileț;
```



The screenshot shows a SQL query editor with a 'Query Builder' tab. The query contains several INSERT statements into a table named 'cinematograf'. The table has columns: CINEMATOGRAF\_ID, NUME, ADRESA, and DATA\_INFINTARE. The results are displayed in a table below the query.

CINEMATOGRAF_ID	NUME	ADRESA	DATA_INFINTARE
1	Jay and Silent Bob Multiplex	58 Leonard Avenue, Leonardo	19-OCT-94
2	Hitchcock Multiplex	517 High Road, Leytonstone, London	13-AUG-99
3	The Supercalifragilistic Cinema Experience	500 S. Buena Vista St., Burbank, CA	27-AUG-64
4	Gisaengchung Cinema	Bongdeok-dong, Daegu, South Korea	19-FEB-00
5	Inglorious, Fictitious and Unchained Feet Cinema	1822 Sepulveda Blvd., Manhattan Beach, CA, USA	09-OCT-92
6	Hitchcock Multiplex	Bel Air, Los Angeles, CA, USA	29-APR-80
7	Time and Relative Dimensions in Cinema	White City, W12 7RJ, London, UK	23-NOV-63

Worksheet Query Builder

```

INSERT INTO sala
VALUES(9, 150, '4DX 3D', 1);

INSERT INTO sala
VALUES(10, 100, 'IMAX 2D', 2);

INSERT INTO sala
VALUES(11, 250, '2D', 6);

INSERT INTO sala
VALUES(12, 110, 'IMAX 2D', 7);

INSERT INTO sala
VALUES(13, 215, '3D', 6);

SELECT * FROM sala;

```

Script Output x Query Result 1 x Query Result 2 x Query Result 3 x Query Result 4 x Query Result 5 x

SQL All Rows Fetched: 13 in 0.005 seconds

	SALA_ID	CAPACITATE	TIP_PROIECTIE	CINEMATOGRAF_ID
1	1	120 2D		1
2	2	160 3D		1
3	3	200 3D		2
4	4	300 IMAX 3D		2
5	5	100 4DX 2D		5
6	6	25 VIP 2D		3
7	7	25 VIP 3D		3
8	8	160 3D		4
9	9	150 4DX 3D		1
10	10	100 IMAX 2D		2
11	11	250 2D		6
12	12	110 IMAX 2D		7
13	13	215 3D		6

SQL Worksheet History

Worksheet Query Builder

```

INSERT INTO film
VALUES(8, 'Forrest Gump', tab_imb('Drama', 'Romance'), 142, '06-JUL-1994', 'Robert Zemeckis', 'EN', 'USA', 'PG-13');

INSERT INTO film
VALUES(9, 'Metropolis', tab_imb('Drama', 'Sci-Fi'), 153, '10-JAN-1927', 'Fritz Lang', 'DE', 'Germany', 'PG-13');

INSERT INTO film
VALUES(10, '12 Angry Men', tab_imb('Crime', 'Drama'), 96, '10-APR-1957', 'Sidney Lumet', 'EN', 'USA', 'PG-13');

INSERT INTO film
VALUES(11, 'A Trip to the Moon', tab_imb('Short', 'Adventure', 'Comedy'), 9, '01-SEP-1902', 'Georges Méliès', 'SIL', 'France', 'G');

INSERT INTO film
VALUES(12, 'City Lights', tab_imb('Comedy', 'Drama', 'Romance'), 87, '07-MAR-1931', 'Charles Chaplin', 'EN', 'USA', 'G');

SELECT * FROM film;

```

Script Output x Query Result x Query Result 1 x Query Result 2 x Query Result 3 x Query Result 4 x Query Result 5 x

SQL All Rows Fetched: 12 in 0.124 seconds

FILM_ID	TITLU	GEN	DURATA	DATA_LANSARII	REGIZOR	LIMBA	TARA	CLASIFICA
1	Back to the Future	DAN.TAB_IMB('Adventure', 'Comedy', 'Sci-Fi')	116	19-JUL-85	Robert Zemeckis	EN	USA	PG-13
2	It's A Wonderful Life	DAN.TAB_IMB('Drama', 'Family', 'Fantasy')	131	20-DEC-46	Frank Capra	EN	USA	PG
3	Gone Girl	DAN.TAB_IMB('Drama', 'Mystery', 'Thriller')	149	03-OCT-14	David Fincher	EN	USA	R
4	Psycho	DAN.TAB_IMB('Horror', 'Mystery', 'Thriller')	109	08-SEP-60	Alfred Hitchcock	EN	USA	R
5	Baby Driver	DAN.TAB_IMB('Action', 'Crime', 'Music')	113	28-JUN-17	Edgar Wright	EN	USA	R
6	Knives Out	DAN.TAB_IMB('Comedy', 'Crime', 'Drama')	130	27-NOV-19	Rian Johnson	EN	USA	PG-13
7	Parasite	DAN.TAB_IMB('Comedy', 'Drama', 'Thriller')	132	30-MAY-19	Bong Joon-ho	KO	South Korea	R
8	Forrest Gump	DAN.TAB_IMB('Drama', 'Romance')	142	06-JUL-94	Robert Zemeckis	EN	USA	PG-13
9	Metropolis	DAN.TAB_IMB('Drama', 'Sci-Fi')	153	10-JAN-27	Fritz Lang	DE	Germany	PG-13
10	12 Angry Men	DAN.TAB_IMB('Crime', 'Drama')	96	10-APR-57	Sidney Lumet	EN	USA	PG-13
11	A Trip to the Moon	DAN.TAB_IMB('Short', 'Adventure', 'Comedy')	9	01-SEP-02	Georges Méliès	SIL	France	G
12	City Lights	DAN.TAB_IMB('Comedy', 'Drama', 'Romance')	87	07-MAR-31	Charles Chaplin	EN	USA	G

## Nimară Dan Gabriel, grupa 241

SQL Worksheet | History

Worksheet | Query Builder

```
VALUES(8, 4, sysdate-222.1);

INSERT INTO difuzeaza
VALUES(5, 3, sysdate-1.13);

-- Nu se poate difuza un film inainte de data lansarii acestuia!
--INSERT INTO difuzeaza
--VALUES(6, 6, sysdate-873.6);

INSERT INTO difuzeaza
VALUES(9, 5, sysdate-451);

INSERT INTO difuzeaza
VALUES(7, 9, sysdate-51);

INSERT INTO difuzeaza
VALUES(4, 2, sysdate-11);
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4 | Query Result 5

SQL | All Rows Fetched: 25 in 0.007 seconds

FILM_ID	SALA_ID	TO_CHAR(DATA_DIF,'DD-MM-YYYYHH:MM:SS')
1	1	2 29-11-2020 04:11:44
2	1	3 25-12-2020 09:12:44
3	1	7 20-09-2020 04:09:44
4	2	1 27-12-2020 09:12:44
5	2	2 24-12-2020 02:12:44
6	2	10 20-09-2020 11:09:44
7	2	13 28-08-2020 11:08:44
8	3	1 22-12-2020 04:12:44
9	3	2 19-08-2020 09:08:44
10	3	4 19-12-2020 02:12:44
11	4	2 18-12-2020 04:12:44
12	4	5 28-12-2020 11:12:44
13	5	1 29-12-2020 09:12:44
14	5	3 15-12-2020 11:12:20
15	5	3 28-12-2020 01:12:32
16	5	5 14-12-2020 04:12:44
17	7	1 31-01-2020 09:01:44
18	7	8 05-08-2019 02:08:44

SQL Worksheet | History

Worksheet | Query Builder

```
INSERT INTO spectator
VALUES(1, 'Nimara', 'Dan-Gabriel', '0762669402', 'M');

INSERT INTO spectator
VALUES(2, 'Alexandrescu', 'Paula', '0799168286', 'F');

INSERT INTO spectator
VALUES(3, 'Barbu', 'Mariana-Lenuta', '0748984069', 'F');

INSERT INTO spectator
VALUES(4, 'Oprian', 'George-Adrian', '0743441168', 'M');

INSERT INTO spectator
VALUES(5, 'Pavalasc', 'Irina', '0756161429', 'F');

INSERT INTO spectator
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4 | Query Result 5

SQL | All Rows Fetched: 13 in 0.005 seconds

SPECTATOR_ID	NUME	PRENUME	NUMAR_TELEFON	SEX
1	Nimara	Dan-Gabriel	0762669402	M
2	Alexandrescu	Paula	0799168286	F
3	Barbu	Mariana-Lenuta	0748984069	F
4	Oprian	George-Adrian	0743441168	M
5	Pavalasc	Irina	0756161429	F
6	Negulescu	Radu	0755004495	M
7	Lapadus	Raluca	0723161118	F
8	Simion	Roberto-Florian	0720617882	M
9	Apostoiu	Antonio-Ciprian	0727761387	M
10	Varga	Robert	0767251023	M
11	Dima	Oana-Teodora	0720309259	F
12	Bigan	Marian	0720078066	M
13	Moanga	Natalia	07XXXXXXX	F

SQL Worksheet History

Worksheet Query Builder

```

INSERT INTO bilet
VALUES(19, 8, 10, 21.5, sysdate-512.6, 'Student', 7, 8, 9);

INSERT INTO bilet
VALUES(20, 10, 18, 29.5, sysdate-222.1, NULL, 8, 4, 13);

INSERT INTO bilet
VALUES(21, 6, 17, 23.5, sysdate-645.04, NULL, 9, 1, 1);

INSERT INTO bilet
VALUES(22, 7, 9, 19.5, sysdate-3.5, 'Pensionar', 10, 11, 12);

INSERT INTO bilet
VALUES(23, 9, 10, 21.5, sysdate-570, 'Pensionar', 9, 12, 4);

SELECT * FROM bilet;

```

Script Output x Query Result x Query Result 1 x Query Result 2 x Query Result 3 x Query Result 4 x Query Result 5 x

SQL All Rows Fetched: 23 in 0.005 seconds

	BILET_ID	RANDL	LOC	PRET	DATA_DIF	TIP_REDUCERE	FILM_ID	SALA_ID	SPECTATOR_ID
1	1	12	14	41.28	DEC-20	(null)	4	5	1
2	2	6	12	19.5	27-DEC-20	Pensionar	2	1	12
3	3	5	12	17.5	14-DEC-20	Student	5	5	10
4	4	14	10	41.25	DEC-20	(null)	1	3	3
5	5	8	9	16.5	27-DEC-20	Copil	2	1	11
6	6	11	2	25.5	29-NOV-20	(null)	1	2	6
7	7	6	8	46.08	NOV-20	(null)	7	9	8
8	8	16	8	23.5	22-DEC-20	(null)	3	1	6
9	9	12	14	21.5	28-DEC-20	Pensionar	5	3	4
10	10	5	7	42.5	05-OCT-19	Avanpremiera	9	5	1
11	11	3	10	66.09	AUG-18	Pensionar	6	6	12
12	12	10	4	20.5	20-SEP-20	Student	2	10	5
13	13	2	6	70.20	SEP-20	(null)	1	7	3
14	14	13	13	25.5	28-DEC-20	(null)	5	3	13
15	15	10	8	41.14	DEC-20	(null)	5	5	2
16	16	9	11	18.5	20-SEP-20	(null)	2	10	11
17	17	15	12	25.5	24-DEC-20	(null)	2	2	3

## 6. Definiți un subprogram stocat care să utilizeze un tip de colecție studiat. Apelați subprogramul.

Scrieți un subprogram stocat care primește ca parametru de intrare un an și afișează detalii despre filmele apărute până în acel an inclusive printr-un parametru de ieșire de tip colecție.

```

CREATE OR REPLACE PACKAGE p6_pkg IS
TYPE tablou_indexat IS TABLE OF film%ROWTYPE INDEX BY BINARY_INTEGER;
END;
/

CREATE OR REPLACE PROCEDURE
p6(an IN VARCHAR2,
    detalii_filme OUT p6_pkg.tablou_indexat) IS
    dummy NUMBER;
BEGIN
    SELECT null INTO dummy

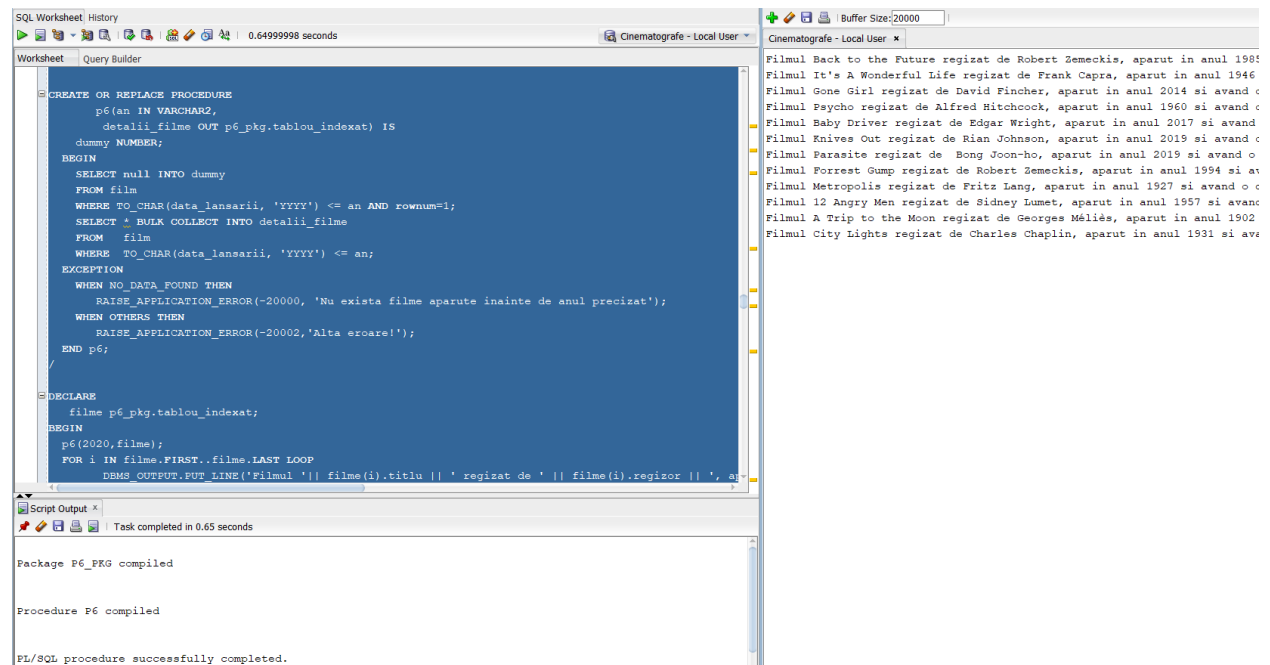
```

```

FROM film
WHERE TO_CHAR(data_lansarii, 'YYYY') <= an AND rownum=1;
SELECT * BULK COLLECT INTO detalii_filme
FROM film
WHERE TO_CHAR(data_lansarii, 'YYYY') <= an;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20000, 'Nu exista filme aparute inainte de anul precizat');
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');
END p6;
/

DECLARE
  filme p6_pkg.tablou_indexat;
BEGIN
  p6(2020,filme);
  FOR i IN filme.FIRST..filme.LAST LOOP
    DBMS_OUTPUT.PUT_LINE('Filmul ' || filme(i).titlu || ' regizat de ' || filme(i).regizor || ', aparut
in anul ' || TO_CHAR(filme(i).data_lansarii, 'YYYY') || ' si avand o durata de ' || filme(i).durata || '
minute');
  END LOOP;
END;
/

```



## 7. Definiți un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

Afișează pentru fiecare film numele cinematografelor care l-au rulat de cele mai multe ori într-un an dat ca parametru unui subprogram stocat. În cazul în care un anumit film nu a rulat în anul respectiv la niciun cinematograf, se va preciza acest lucru.

```
CREATE OR REPLACE PROCEDURE
    p7(an VARCHAR2) IS
    CURSOR c1 IS
        SELECT film_id id1, titlu titlu, (SELECT count(film_id)
                                         FROM difuzeaza d
                                         WHERE TO_CHAR(data_dif,'YYYY') = an and film_id=f.film_id) nr1
    FROM film f;
    CURSOR c2 IS
        SELECT nume nume, film_id id2, count(film_id) nr2
        FROM cinematograf c JOIN sala s ON (c.cinematograf_id=s.cinematograf_id)
            JOIN difuzeaza d ON (s.sala_id=d.sala_id)
        WHERE TO_CHAR(d.data_dif,'YYYY') = an
        GROUP BY nume, film_id
        ORDER BY 3 DESC;
    nr3 number(4);
BEGIN
    FOR i in c1 LOOP
        nr3 := 0;
        IF i.nr1=0 THEN
            DBMS_OUTPUT.PUT_LINE('Filmul ' || i.titlu || ' nu a fost difuzat in anul ' || an || ' la niciu
n cinematograf.');
```

```
        ELSE
            DBMS_OUTPUT.PUT('Filmul ' || i.titlu || ' a fost difuzat ');
            FOR j in c2 LOOP
                IF i.id1=j.id2 THEN
                    IF nr3=0 THEN
                        IF j.nr2=1 THEN
                            DBMS_OUTPUT.PUT_LINE('o data la: ');
                        ELSE DBMS_OUTPUT.PUT_LINE('de ' || j.nr2 || ' ori la: ');
                        END IF;
                    END IF;
                    EXIT WHEN nr3!=j.nr2 AND nr3!=0;
                    DBMS_OUTPUT.PUT_LINE('Cinematograful ' || j.nume);
                    nr3 := j.nr2;
                END IF;
            END LOOP;
        END IF;
    END LOOP;
```



```

        END LOOP;

    END IF;

    DBMS_OUTPUT.NEW_LINE();

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

    DBMS_OUTPUT.NEW_LINE();

END LOOP;

END p7;

/

BEGIN

    p7(2020);

END;

/

```

The screenshot shows an SQL Worksheet application with a query builder and a script output window. The query builder contains a PL/SQL procedure named p7, which is designed to calculate the profit for a given year (2020). The procedure uses two cursors to retrieve film and cinema data, and then calculates the profit for each film based on the cinema's profit margin (45%). The script output window shows the results of the procedure, listing the film titles and their corresponding profits for the year 2020.

```

CREATE OR REPLACE PROCEDURE
p7(an VARCHAR2) IS
    CURSOR c1 IS
        SELECT film_id id1, titlu titlu, (SELECT count(film_id)
        FROM difuzeaza d
        WHERE TO_CHAR(data_dif,'YYYY') = an and film_id=f.film_id) nr1
    FROM film f;
    CURSOR c2 IS
        SELECT nume nume, film_id id2, count(film_id) nr2
    FROM cinematograf c JOIN sala s ON (c.cinematograf_id=s.cinematograf_id)
    JOIN difuzeaza d ON (s.sala_id=d.sala_id)
    WHERE TO_CHAR(d.data_dif,'YYYY') = an
    GROUP BY nume, film_id
    ORDER BY 3 DESC;
    nr3 number(4);
BEGIN
    FOR i in c1 LOOP
        nr3 := 0;
        IF i.nr1=0 THEN
            DBMS_OUTPUT.PUT_LINE('Filmul ' || i.titlu || ' nu a fost difuzat in anul ' || an || ' la ');
        ELSE
            DBMS_OUTPUT.PUT('Filmul ' || i.titlu || ' a fost difuzat ');
            FOR j in c2 LOOP
                IF i.id1=j.id2 THEN
                    IF nr3=0 THEN
                        DBMS_OUTPUT.PUT_LINE('Filmul ' || i.titlu || ' a fost difuzat de 2 ori la:');
                    ELSE
                        DBMS_OUTPUT.PUT_LINE('Cinematograful ' || j.nume || '');
                    END IF;
                    nr3 := nr3 + 1;
                END IF;
            END LOOP;
        END IF;
    END LOOP;
END p7;

```

Script Output:

```

Task completed in 0.049 seconds
PL/SQL procedure successfully completed.

Procedure P7 compiled

PL/SQL procedure successfully completed.

Filmul Back to the Future a fost difuzat o data la:
Cinematograful Jay and Silent Bob Multiplex
Cinematograful The Supercalifragilistic Cinema Experience
Cinematograful Hitchcock Multiplex

Filmul It's A Wonderful Life a fost difuzat de 2 ori la:
Cinematograful Hitchcock Multiplex
Cinematograful Jay and Silent Bob Multiplex

Filmul Gone Girl a fost difuzat de 2 ori la:
Cinematograful Jay and Silent Bob Multiplex

Filmul Psycho a fost difuzat o data la:
Cinematograful Inglorious, Fictitious and Unchained Feet Cinema
Cinematograful Jay and Silent Bob Multiplex

Filmul Baby Driver a fost difuzat de 2 ori la:
Cinematograful Hitchcock Multiplex

Filmul Knives Out nu a fost difuzat in anul 2020 la niciun cinematograf

Filmul Parasite a fost difuzat de 2 ori la:
Cinematograful Jay and Silent Bob Multiplex

```

**8. Definiți un subprogram stocat de tip funcție care să utilizeze 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.**

Dat fiind numele unui cinematograf, determinați profitul obținut de acesta în anul 2020. Se știe că un cinematograf păstrează 45% din costul biletului.

```
SELECT * FROM cinematograf;
CREATE OR REPLACE FUNCTION f8
  (v_nume cinematograf.nume%TYPE DEFAULT 'Jay and Silent Bob Multiplex')
RETURN NUMBER IS
  profit NUMBER(10,2);
  idul cinematograf.cinematograf_id%TYPE;
BEGIN
  SELECT c.cinematograf_id INTO idul
  FROM cinematograf c
  WHERE nume=v_nume;
  SELECT SUM(pret-0.55*pret) INTO profit
  FROM cinematograf c JOIN sala s ON (c.cinematograf_id=s.cinematograf_id)
    JOIN bilet b ON (s.sala_id=b.sala_id)
  WHERE nume=v_nume AND TO_CHAR(data_dif,'YYYY')='2020';
  RETURN profit;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20000, 'Cinematograful respectiv nu exista. ');
  WHEN TOO_MANY_ROWS THEN
    RAISE_APPLICATION_ERROR(-20001, 'Exista mai multe cinematografe cu numele dat!');
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END f8;
/

-- SQL
SELECT f8 FROM DUAL; -- functioneaza
--Cinematograful respectiv nu exista
SELECT f8('Cinema City') FROM DUAL;

-- PL/SQL
-- Exista mai multe cinematografe cu numele dat!
BEGIN
  DBMS_OUTPUT.PUT_LINE('Profitul este '|| f8('Hitchcock Multiplex'));
END;
/
```

```
Worksheet Query Builder

SELECT * FROM cinematograf;
CREATE OR REPLACE FUNCTION f8
(v_nume cinematograf.nume%TYPE DEFAULT 'Jay and Silent Bob Multiplex')
RETURN NUMBER IS
profit NUMBER(10,2);
idul cinematograf.cinematograf_id%TYPE;
BEGIN
SELECT c.cinematograf_id INTO idul
FROM cinematograf c
WHERE nume=v_nume;
SELECT SUM(pret-0.55*pret) INTO profit
FROM cinematograf c JOIN sala s ON (c.cinematograf_id=s.cinematograf_id)
JOIN bilet b ON (s.sala_id=b.sala_id)
WHERE nume=v_nume AND TO_CHAR(data_dif,'YYYY')='2020';
RETURN profit;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20000, 'Cinematograful respectiv nu exista. ');
WHEN TOO_MANY_ROWS THEN
RAISE_APPLICATION_ERROR(-20001, 'Exista mai multe cinematografe cu numele dat!');
WHEN OTHERS THEN
RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END f8;
/
-- SQL
SELECT f8 FROM DUAL; -- functioneaza
```

Script Output x Query Result 1 x Query Result 2 x

SQL | All Rows Fetched: 1 in 0.004 seconds

F8
1 70.43

```
----Cinematograful respectiv nu exista
SELECT f8('Cinema City') FROM DUAL;
--
```

Script Output x Query Result 1 x Query Result 2 x

SQL | Executing:SELECT f8('Cinema City') FROM DUAL in 0 seconds

ORA-20000: Cinematograful respectiv nu exista.  
ORA-06512: at "DAN.F8", line 17  
ORA-06512: at line 1  
20000.00000 - "%s"  
\*Cause: The stored procedure 'raise\_application\_error'  
was called which causes this error to be generated.  
\*Action: Correct the problem as described in the error message or contact  
the application administrator or DBA for more information.

```
---- PL/SQL
---- Exista mai multe cinematografe cu numele dat!
BEGIN
DBMS_OUTPUT.PUT_LINE('Profitul este '|| f8('Hitchcock Multiplex'));
END;
```

Script Output x Query Result 1 x Query Result 2 x

Task completed in 0.806 seconds

```
BEGIN
DBMS_OUTPUT.PUT_LINE('Profitul este '|| f8('Hitchcock Multiplex'));
END;
Error report -
ORA-20001: Exista mai multe cinematografe cu numele dat!
ORA-06512: at "DAN.F8", line 19
ORA-06512: at line 2
```

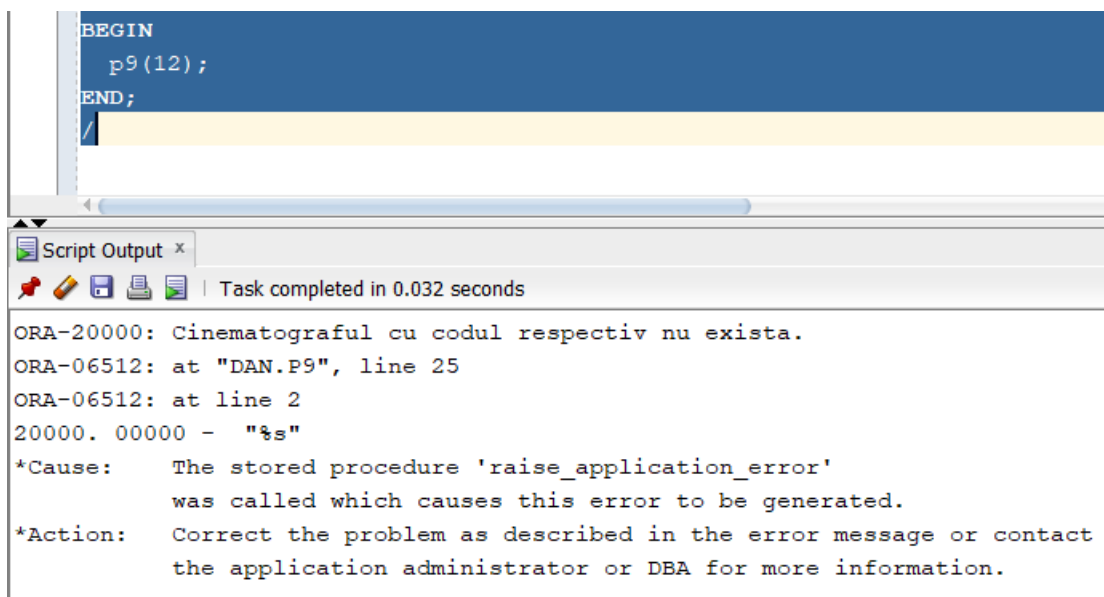
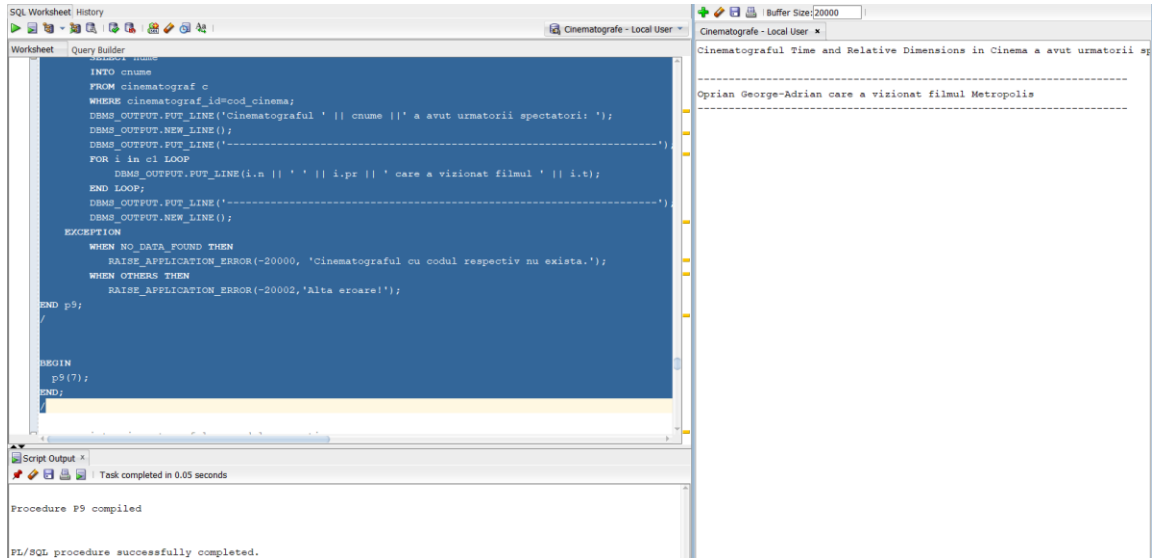
**9. Definiți un subprogram stocat de tip procedură care să utilizeze 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.**

Pentru codul unui cinematograful transmis ca parametru unui proceduri afișați numele acestuia, precum și spectatorii și filmele la care au mers aceștia.

```
CREATE OR REPLACE PROCEDURE
    p9(cod_cinema cinematograful.cinematograful_id%TYPE) IS
    cnume cinematograful.nume%TYPE;
    CURSOR c1 IS
        SELECT nume n, prenume pr, titlu t, cinematograful_id cid2
        FROM spectator sp JOIN bilet b ON (sp.spectator_id=b.spectator_id)
            JOIN film f ON (b.film_id=f.film_id)
            JOIN sala s ON (b.sala_id=s.sala_id)
        WHERE cinematograful_id=cod_cinema;
BEGIN
    SELECT nume
    INTO cnume
    FROM cinematograful c
    WHERE cinematograful_id=cod_cinema;
    DBMS_OUTPUT.PUT_LINE('Cinematograful ' || cnume || ' a avut urmatorii spectatori: ');
    DBMS_OUTPUT.NEW_LINE();
    DBMS_OUTPUT.PUT_LINE('-----');
    FOR i IN c1 LOOP
        DBMS_OUTPUT.PUT_LINE(i.n || ' ' || i.pr || ' care a vizionat filmul ' || i.t);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('-----');
    DBMS_OUTPUT.NEW_LINE();
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Cinematograful cu codul respectiv nu exista. ');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare! ');
END p9;
/

BEGIN
    p9(7);
END;
```

```
/
-- nu exista cinematograful cu codul respectiv
BEGIN
  p9(12);
END;
/
```



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

La cinematografele din sistem filmele pentru weekend-ul următor ajung duminica. Atunci angajații le pot insera în baza de date, însă difuzările nu pot fi programate decât luni, cel târziu marți, în intervalul 8-14, atunci când volumul de muncă al angajaților este mai mic. Același lucru se întâmplă și în cazul reprogramărilor unor filme difuzate anterior sau ștergerea difuzărilor anterioare.

```
CREATE OR REPLACE TRIGGER trig_10
BEFORE INSERT OR DELETE OR UPDATE on difuzeaza
BEGIN
    IF (TO_CHAR(SYSDATE,'D') NOT IN (2,3))
        OR (TO_CHAR(SYSDATE,'D') IN (2,3) AND TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 and 14) THEN
        IF INSERTING THEN
            RAISE_APPLICATION_ERROR(-20001,'Inserarea unei noi difuzari
            nu se poate face in acest interval!');
        ELSIF DELETING THEN
            RAISE_APPLICATION_ERROR(-20002,'Stergerea unei vechi difuzari
            nu se poate face in acest interval!');
        ELSE
            RAISE_APPLICATION_ERROR(-20003,'Actualizarea unei vechi difuzari
            nu se poate face in acest interval!');
        END IF;
    END IF;
END;
/
ALTER TRIGGER trig_10 DISABLE;
--ALTER TRIGGER trig_10 ENABLE;

--Exemplu
INSERT INTO difuzeaza
VALUES(1,1,sysdate-150);
```

```
CREATE OR REPLACE TRIGGER trig_10
BEFORE INSERT OR DELETE OR UPDATE on difuzeaza
BEGIN
    IF (TO_CHAR(SYSDATE,'D') NOT IN (2,3))
    OR (TO_CHAR(SYSDATE,'D') IN (2,3) AND TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 and 14) THEN
    IF INSERTING THEN
        RAISE_APPLICATION_ERROR(-20001,'Inserarea unei noi difuzari
        nu se poate face in acest interval!');
    ELSIF DELETING THEN
        RAISE_APPLICATION_ERROR(-20002,'Stergerea unei vechi difuzari
        nu se poate face in acest interval!');
    ELSE
        RAISE_APPLICATION_ERROR(-20003,'Actualizarea unei vechi difuzari
        nu se poate face in acest interval!');
    END IF;
END IF;
END;
```

```
ALTER TRIGGER trig_10 DISABLE;
--ALTER TRIGGER trig_10 ENABLE;

--Exemplu
INSERT INTO difuzeaza
VALUES(1,1,sysdate-150);
```

Script Output x

Task completed in 0.041 seconds

Trigger TRIG\_10 compiled

```
--Exemplu
INSERT INTO difuzeaza
VALUES(1,1,sysdate-150);
```

```
--11. Avand doi triggeri, unul care verifica ca data di:
```

Script Output x

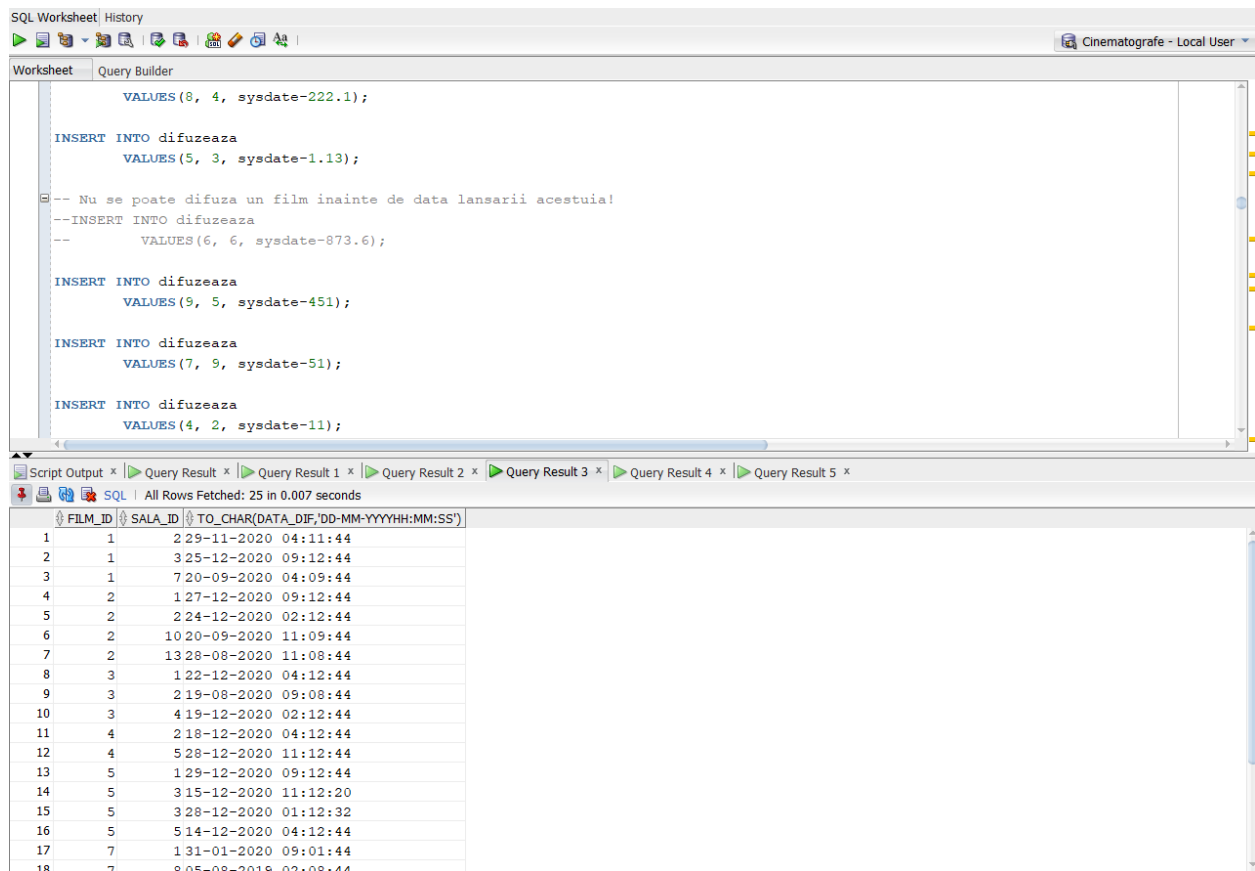
Task completed in 0.047 seconds

```
INSERT INTO difuzeaza
VALUES(1,1,sysdate-150)
Error report -
ORA-20001: Inserarea unei noi difuzari
        nu se poate face in acest interval!
ORA-06512: at "DAN.TRIG_10", line 5
ORA-04088: error during execution of trigger 'DAN.TRIG_10'
```

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

Atunci când introduc o nouă difuzare în tabelul difuzează, triggerul ne verifică ca nu cumva data difuzării să precede data lansării filmului.

```
CREATE OR REPLACE TRIGGER difuzare_permisa
BEFORE INSERT OR UPDATE ON difuzeaza
FOR EACH ROW
DECLARE
    lansare film.data_lansarii%TYPE;
BEGIN
    SELECT data_lansarii
    INTO lansare
    FROM film
    WHERE film_id = :NEW.film_id;
    IF :NEW.data_dif < lansare THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu se poate difuza un film inainte de data lansarii acestuia!');
    END IF;
END;
/
```



The screenshot shows a SQL Worksheet interface with a script editor and a query result table. The script contains several INSERT statements into the 'difuzeaza' table, some of which are commented out. The query result table displays 18 rows of data with columns: FILM\_ID, SALA\_ID, and TO\_CHAR(DATA\_DIF, 'DD-MM-YYYYHH:MM:SS').

FILM_ID	SALA_ID	TO_CHAR(DATA_DIF, 'DD-MM-YYYYHH:MM:SS')
1	1	2 29-11-2020 04:11:44
2	1	3 25-12-2020 09:12:44
3	1	7 20-09-2020 04:09:44
4	2	1 27-12-2020 09:12:44
5	2	2 24-12-2020 02:12:44
6	2	10 20-09-2020 11:09:44
7	2	13 28-08-2020 11:08:44
8	3	1 22-12-2020 04:12:44
9	3	2 19-08-2020 09:08:44
10	3	4 19-12-2020 02:12:44
11	4	2 18-12-2020 04:12:44
12	4	5 28-12-2020 11:12:44
13	5	1 29-12-2020 09:12:44
14	5	3 15-12-2020 11:12:20
15	5	3 28-12-2020 01:12:32
16	5	5 14-12-2020 04:12:44
17	7	1 31-01-2020 09:01:44
18	7	8 05-08-2019 02:08:44



```
--Atunci cand se introduce o noua difuzare, trebuie sa ne asiguram ca data difuzarii nu preceda data lansarii filmului.

CREATE OR REPLACE TRIGGER difuzare_permisa
BEFORE INSERT OR UPDATE ON difuzeaza
FOR EACH ROW
DECLARE
    lansare film.data_lansarii%TYPE;
BEGIN
    SELECT data_lansarii
    INTO lansare
    FROM film
    WHERE film_id = :NEW.film_id;
    IF :NEW.data_dif < lansare THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu se poate difuza un film inainte de data lansarii acestuia!');
    END IF;
END;
/
```

Script Output x

Task completed in 0.342 seconds

Trigger DIFUZARE\_PERMISA compiled

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

```
CREATE TABLE actiuni_user
(
    nume_baza_de_date VARCHAR2(50),
    user_curent VARCHAR2(30),
    actiune VARCHAR2(20),
    tip_obiect_referit VARCHAR2(30),
    nume_obiect_referit VARCHAR2(30),
    data TIMESTAMP(3));

----

Definiti un declansator care sa introduca date in acest tabel dupa ce utilizatorul a folosit o comanda LDD
.

CREATE OR REPLACE TRIGGER trig_12
AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
    INSERT INTO actiuni_user
    VALUES (SYS.DATABASE_NAME, SYS.LOGIN_USER,
            SYS.SYSEVENT, SYS.DICTIONARY_OBJ_TYPE,
            SYS.DICTIONARY_OBJ_NAME, SYSTIMESTAMP(3));
END;
/
```

```
SELECT * from actiuni_user;
```

```
CREATE OR REPLACE TRIGGER trig_12
AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
    INSERT INTO actiuni_user
    VALUES (SYS.DATABASE_NAME, SYS.LOGIN_USER,
            SYS.SYSEVENT, SYS.DICTIONARY_OBJ_TYPE,
            SYS.DICTIONARY_OBJ_NAME, SYSTIMESTAMP(3));
END;
/

SELECT * from actiuni_user;

--DROP TRIGGER audit_schema;
--DROP TABLE audit user;
```

Script Output x Query Result x

Task completed in 0.039 seconds

Trigger TRIG\_12 compiled

```
SELECT * from actiuni_user;
```

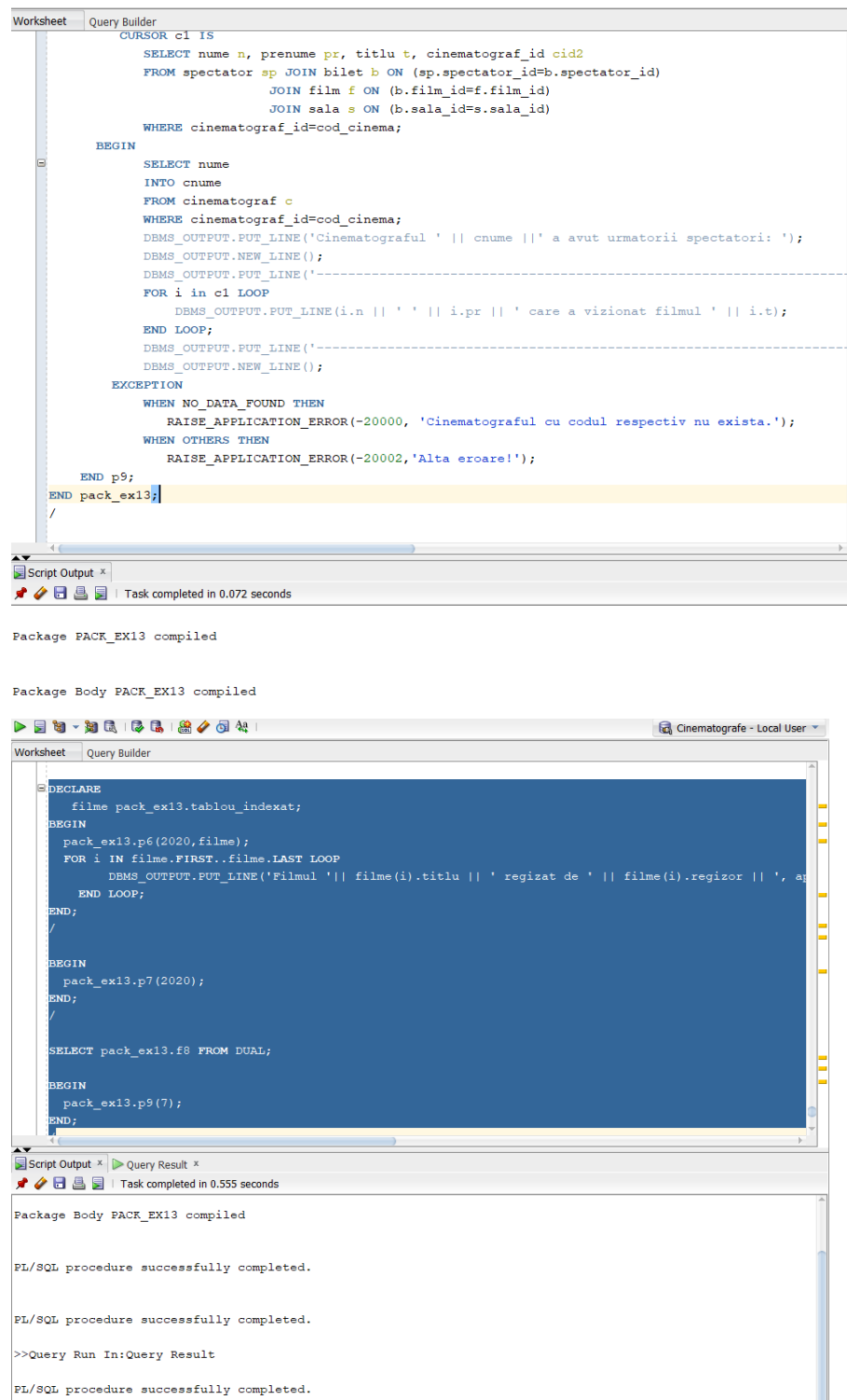
Script Output x Query Result x

SQL | All Rows Fetched: 27 in 0.003 seconds

NUME_BAZA_DE_DATE	USER_CURENT	ACTIUNE	TIP_OBJECT_REFERIT	NUME_OBJECT_REFERIT	DATA
1 XE	DAN	CREATE INDEX	PK_CIN		29-DEC-20 06.06.55.68800000
2 XE	DAN	CREATE TABLE	CINEMATOGRAF		29-DEC-20 06.06.55.69200000
3 XE	DAN	CREATE INDEX	PK_SALA		29-DEC-20 06.06.55.70100000
4 XE	DAN	CREATE TABLE	SALA		29-DEC-20 06.06.55.70300000
5 XE	DAN	CREATE TYPE	TAB_IMB		29-DEC-20 06.06.55.71700000
6 XE	DAN	CREATE TABLE	TAB_IMB_GEN		29-DEC-20 06.06.55.73000000
7 XE	DAN	CREATE INDEX	SYS_FK0000020335N00003\$		29-DEC-20 06.06.55.73200000
8 XE	DAN	CREATE INDEX	PK_FILM		29-DEC-20 06.06.55.73400000
9 XE	DAN	CREATE INDEX	SYS_C007305		29-DEC-20 06.06.55.73500000
10 XE	DAN	CREATE TABLE	FILM		29-DEC-20 06.06.55.73800000
11 XE	DAN	CREATE INDEX	SYS_C007306		29-DEC-20 06.06.55.74500000
12 XE	DAN	CREATE TABLE	DIFUZEAZA		29-DEC-20 06.06.55.74700000
13 XE	DAN	CREATE INDEX	PK_SPECTATOR		29-DEC-20 06.06.55.75400000
14 XE	DAN	CREATE INDEX	SYS_C007311		29-DEC-20 06.06.55.75600000
15 XE	DAN	CREATE TABLE	SPECTATOR		29-DEC-20 06.06.55.75800000
16 XE	DAN	CREATE INDEX	PK_BILET		29-DEC-20 06.06.55.76600000
17 XE	DAN	CREATE TABLE	BILET		29-DEC-20 06.06.55.77000000
18 XE	DAN	CREATE TRIGGER	DIFUZARE_PERMISA		29-DEC-20 06.06.55.77800000
19 XE	DAN	CREATE PACKAGE	P6_PKG		29-DEC-20 06.06.57.99200000
20 XE	DAN	CREATE PROCEDURE	P6		29-DEC-20 06.06.58.00100000
21 XE	DAN	CREATE PROCEDURE	P7		29-DEC-20 06.06.58.04500000
22 XE	DAN	CREATE FUNCTION	F8		29-DEC-20 06.06.58.08800000
23 XE	DAN	CREATE PROCEDURE	P9		29-DEC-20 06.06.58.28400000
24 XE	DAN	CREATE TRIGGER	TRIG_10		29-DEC-20 06.06.58.33300000
25 XE	DAN	ALTER TRIGGER	TRIG_10		29-DEC-20 06.06.58.34100000
26 XE	DAN	CREATE TRIGGER	TRIG_11_PRIM		29-DEC-20 06.06.58.38300000
27 XE	DAN	ALTER TRIGGER	TRIG_11_PRIM		29-DEC-20 06.06.58.39100000

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

Codul se găsește în fișierul .sql atașat.



```
Worksheet Query Builder
CURSOR c1 IS
    SELECT nume n, prenume pr, titlu t, cinematograf_id cid2
    FROM spectator sp JOIN bilet b ON (sp.spectator_id=b.spectator_id)
        JOIN film f ON (b.film_id=f.film_id)
        JOIN sala s ON (b.sala_id=s.sala_id)
    WHERE cinematograf_id=cod_cinema;

BEGIN
    SELECT nume
    INTO cnume
    FROM cinematograf c
    WHERE cinematograf_id=cod_cinema;
    DBMS_OUTPUT.PUT_LINE('Cinematograful ' || cnume || ' a avut urmatoorii spectatori: ');
    DBMS_OUTPUT.NEW_LINE();
    DBMS_OUTPUT.PUT_LINE('-----');
    FOR i in c1 LOOP
        DBMS_OUTPUT.PUT_LINE(i.n || ' ' || i.pr || ' care a vizionat filmul ' || i.t);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('-----');
    DBMS_OUTPUT.NEW_LINE();

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Cinematograful cu codul respectiv nu exista.');
```

Script Output x

Task completed in 0.072 seconds

Package PACK\_EX13 compiled

Package Body PACK\_EX13 compiled

```
Worksheet Query Builder
DECLARE
    filme pack_ex13.tablou_indexat;
BEGIN
    pack_ex13.p6(2020,filme);
    FOR i IN filme.FIRST..filme.LAST LOOP
        DBMS_OUTPUT.PUT_LINE('Filmul ' || filme(i).titlu || ' regizat de ' || filme(i).regizor || ', a');
    END LOOP;
END;
/

BEGIN
    pack_ex13.p7(2020);
END;
/

SELECT pack_ex13.f8 FROM DUAL;

BEGIN
    pack_ex13.p9(7);
END;
```

Script Output x Query Result x

Task completed in 0.555 seconds

Package Body PACK\_EX13 compiled

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

>>Query Run In:Query Result

PL/SQL procedure successfully completed.

**14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare pentru acțiuni integrate.**

***ACHIZIȚIE BILET ȘI ISTORIC VIZIONĂRI***

Pentru un anumit spectator afișez istoricul filmelor vizionate de acesta. De asemenea, dacă el nu se află în baza de date îl inserez, iar dacă dorește să achiziționeze un bilet la un film fac actualizările necesare.

Utilizez **un tablou imbricat de record** pentru a afișa filmele vizionate de spectatori și regizorul acestora.

Pachetul conține patru proceduri:

- o procedură `achizitie_bilet_coresp` care verifică dacă un anumit titlu de film este difuzat într-o anumită zi;
- o procedură `actualizare_spectator` care inserează în baza de date un spectator în cazul în care id-ul acestuia nu se găsește în tabel;
- o procedură `actualizare_bilet` care inserează un bilet în tabel atunci când spectatorul dorește să facă o achiziție
- o procedură `istoric_spectator` care afișează filmele vizionate de un anumit spectator al cărui id este dat ca parametru. În cazul în care id-ul nu se regăsește în tabelul spectator, se introduce spectatorul în tabel prin apelul procedurii `actualizare_spectator`. De asemenea, spectatorul poate opta să facă o achiziție(al doilea parametru este 0 – pentru nicio achiziție sau 1 – pentru achiziție de bilet), caz în care se actualizează corespunzător tabelul prin apelul procedurii de `actualizare_bilet`. Înainte de actualizarea tabelului bilet se apelează procedura `achizitie_bilet_coresp` care verifică dacă există o difuzare a filmului respective în data respectivă. În caz contrar, afișez un mesaj corespunzător.

```

CREATE SEQUENCE pack_dni
START WITH 100;

CREATE OR REPLACE PACKAGE pack_ex14
IS
    FUNCTION achizitie_bilet_coresp(titlu_film film.titlu%TYPE, data_diff difuzeaza.data_diff%TYPE)
    RETURN NUMBER;
    PROCEDURE actualizare_spectator(idul spectator.spectator_id%TYPE, nm spectator.nume%TYPE, pren spectat
or.prenume%TYPE,
                                tel spectator.numar_telefon%TYPE, sex spectator.sex%TYPE);
    PROCEDURE actualizare_bilet(idul bilet.bilet_id%TYPE, rand bilet.rand1%TYPE, loc bilet.loc%TYPE,
                                pret bilet.pret%TYPE, data_dif bilet.data_dif%TYPE, red bilet.tip_reducere
%TYPE,
                                id_film film.film_id%TYPE, id_sala sala.sala_id%TYPE, id_spectator spectat
or.spectator_id%TYPE);
    PROCEDURE istoric_spectator(id_spectator spectator.spectator_id%TYPE, achizitie NUMBER, nm spectator.n
ume%TYPE,
                                pren spectator.prenume%TYPE, tel spectator.numar_telefon%TYPE, sex spectator.sex%T
YPE,
                                id_bilet bilet.bilet_id%TYPE, rand bilet.rand1%TYPE, loc bilet.loc%TYPE,
                                pret bilet.pret%TYPE, data_diff bilet.data_dif%TYPE, red bilet.tip_reducere%TYPE,
                                titlu film.titlu%TYPE);
END pack_ex14;
/

CREATE OR REPLACE PACKAGE BODY pack_ex14
IS
    FUNCTION achizitie_bilet_coresp(titlu_film film.titlu%TYPE, data_diff difuzeaza.data_diff%TYPE)
    RETURN NUMBER
    IS
        exista NUMBER;
        nr NUMBER;
        idul film.film_id%TYPE;
    BEGIN
        SELECT film_id
        INTO idul
        FROM film
        WHERE upper(titlu)=upper(titlu_film);
        SELECT count(*)
        INTO nr
        FROM difuzeaza
        WHERE film_id=idul and to_char(data_dif,'DD/MM/YYYY')=to_char(data_diff,'DD/MM/YYYY');
        IF nr=0 THEN
            RETURN 0;
        ELSE RETURN 1;
    END;

```

```

        END IF;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20000, 'Filmul cu titlul respectiv nu exista.');
```

WHEN OTHERS THEN
 RAISE\_APPLICATION\_ERROR(-20002, 'Alta eroare!');
 END achizitie\_bilet\_coresp;
 PROCEDURE actualizare\_spectator(idul spectator.spectator\_id%TYPE, nm spectator.nume%TYPE, pren spectat
 or.prenume%TYPE,
 tel spectator.numar\_telefon%TYPE, sex spectator.sex%TYPE)
 IS
 BEGIN
 INSERT INTO spectator
 VALUES(idul, nm, pren, tel, sex);
 END actualizare\_spectator;
 PROCEDURE actualizare\_bilet(idul bilet.bilet\_id%TYPE, rand bilet.rand1%TYPE, loc bilet.loc%TYPE,
 pret bilet.pret%TYPE, data\_dif bilet.data\_dif%TYPE, red bilet.tip\_reducere
 %TYPE,
 id\_film film.film\_id%TYPE, id\_sala sala.sala\_id%TYPE, id\_spectator spectat
 or.spectator\_id%TYPE)
 IS
 BEGIN
 INSERT INTO bilet
 VALUES(idul, rand, loc, pret, data\_dif, red, id\_film, id\_sala, id\_spectator);
 END actualizare\_bilet;
 PROCEDURE istoric\_spectator(id\_spectator spectator.spectator\_id%TYPE, achizitie NUMBER, nm spectator.nume%
 TYPE,
 pren spectator.prenume%TYPE, tel spectator.numar\_telefon%TYPE, sex spectator.sex%T
 YPE,
 id\_bilet bilet.bilet\_id%TYPE, rand bilet.rand1%TYPE, loc bilet.loc%TYPE,
 pret bilet.pret%TYPE, data\_diff bilet.data\_dif%TYPE, red bilet.tip\_reducere%TYPE,
 titlu film.titlu%TYPE)
 IS
 nr NUMBER;
 find\_spec NUMBER;
 id\_film difuzeaza.film\_id%TYPE;
 id\_sala difuzeaza.sala\_id%TYPE;
 TYPE record\_filme IS RECORD (
 titlu\_film film.titlu%TYPE,
 regizor film.regizor%TYPE
 );
 TYPE matrice IS TABLE OF record\_filme;
 istoric matrice := matrice();
 BEGIN
 SELECT COUNT(\*)

```

        INTO find_spec
    FROM spectator
    WHERE spectator_id=id_spectator;
    IF find_spec=0 THEN
        pack_ex14.actualizare_spectator(id_spectator, nm, pren, tel, sex);
    END IF;
    IF achizitie=1 THEN
        SELECT pack_ex14.achizitie_bilet_coresp(titlu, data_diff) INTO nr FROM DUAL;
        IF nr=1 THEN
            SELECT film_id, sala_id
            INTO id_film, id_sala
            FROM difuzeaza
            WHERE to_char(data_dif, 'DD/MM/YYYY')=to_char(data_diff, 'DD/MM/YYYY');
            pack_ex14.actualizare_bilet(id_bilet, rand, loc, pret, data_diff, red, id_film, id_sala, id_spectator);
        ELSE
            DBMS_OUTPUT.PUT_LINE('Nu exista o difuzare a filmului respectiv la acea data');
        END IF;
    END IF;
    SELECT titlu, regizor BULK COLLECT INTO istoric
    FROM film f JOIN bilet b ON (f.film_id=b.film_id)
    WHERE spectator_id = id_spectator;
    DBMS_OUTPUT.PUT_LINE('Istoricul filmelor vizionate de '||nm||' '||pren);
    DBMS_OUTPUT.PUT_LINE('-----');
    DBMS_OUTPUT.NEW_LINE;
    IF istoric.count=0 THEN
        DBMS_OUTPUT.PUT_LINE('Spectatorul nu a vizionat niciun film');
    ELSE
        FOR i IN istoric.FIRST..istoric.LAST LOOP
            DBMS_OUTPUT.PUT_LINE(istoric(i).titlu_film ||' '||' regizat de '||istoric(i).regizor);
        END LOOP;
    END IF;
END istoric_spectator;
END pack_ex14;
/

DECLARE
    nr NUMBER;
BEGIN
    pack_ex14.istoric_spectator(21, 0, 'Bertalan', 'Victor', '0722855111', 'M', pack_dni.NEXTVAL,
        10, 12, 19.5, TO_DATE('19/11/2020', 'DD/MM/YYYY'), NULL, 'Parasite');
END;
/

```

```
SELECT * FROM spectator;
SELECT * FROM bilet;
```

Tabelele înainte de rularea unui apel:

```

975 SELECT * FROM spectator;
976 SELECT * FROM bilet;
977 SELECT b.spectator_id, nume, prenume, titlu, to_char(data_dif,'DD/MM/YYYY')
978 FROM spectator s JOIN bilet b on (s.spectator_id=b.spectator_id)
979 JOIN film f on (b.film_id=f.film_id);
980

```

	BILET_ID	RANDL	LOC	PRET	DATA_DIF	TIP_REDUCTERE	FILM_ID	SALA_ID	SPECTATOR_ID
19	19	8	10	21.5	15-AUG-19	Student	7	8	9
20	20	10	18	29.5	31-MAY-20	(null)	8	4	13
21	21	6	17	23.5	04-APR-19	(null)	9	1	1
22	22	7	9	19.5	05-JAN-21	Pensionar	10	11	12
23	23	9	10	21.5	19-JUN-19	Pensionar	9	12	4
24	112	12	14	19.5	19-NOV-20	(null)	7	9	6

Compiler - Log

```

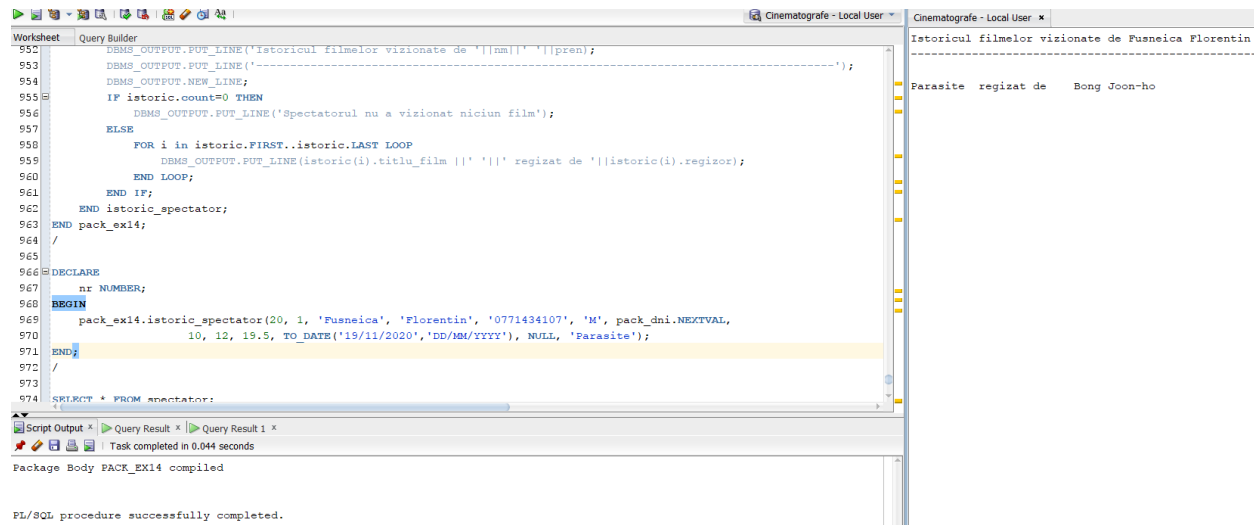
975 SELECT * FROM spectator;
976 SELECT * FROM bilet;
977 SELECT b.spectator_id, nume, prenume, titlu, to_char(data_dif,'DD/MM/YYYY')
978 FROM spectator s JOIN bilet b on (s.spectator_id=b.spectator_id)
979 JOIN film f on (b.film_id=f.film_id);
980

```

	SPECTATOR_ID	NUME	PRENUME	NUMAR_TELEFON	SEX
8	8	Simion	Roberto-Florian	0720617882	M
9	9	Apostoiu	Antonio-Ciprian	0727761387	M
10	10	Varga	Robert	0767251023	M
11	11	Dima	Oana-Teodora	0720309259	F
12	12	Bigan	Marian	0720078066	M
13	13	Moanga	Natalia	07XXXXXXX	F

Tabelele după ce am rulat codul: al doilea parametru este 1, adică spectatorul dorește să facă o achiziție. De asemenea, spectatorul nu se regăsea în baza de date.





```

952 DBMS_OUTPUT.PUT_LINE('Istoricul filmelor vizionate de '||nm||' '||pre||');
953 DBMS_OUTPUT.PUT_LINE('-----');
954 DBMS_OUTPUT.NEW_LINE;
955 IF istoric.count=0 THEN
956   DBMS_OUTPUT.PUT_LINE('Spectatorul nu a vizionat niciun film');
957 ELSE
958   FOR i in istoric.FIRST..istoric.LAST LOOP
959     DBMS_OUTPUT.PUT_LINE(istoric(i).titlu_film ||' '||' regizat de '||istoric(i).regizor);
960   END LOOP;
961 END IF;
962 END istoric_spectator;
963 END pack_ex14;
964 /
965
966 DECLARE
967   nr NUMBER;
968 BEGIN
969   pack_ex14.istoric_spectator(20, 1, 'Fusneica', 'Florentin', '0771434107', 'M', pack_dni.NEXTVAL,
970     10, 12, 19.5, TO_DATE('19/11/2020','DD/MM/YYYY'), NULL, 'Parasite');
971 END;
972 /
973
974 SELECT * FROM spectator;

```

Package Body PACK\_EX14 compiled

PL/SQL procedure successfully completed.

Istoricul filmelor vizionate de Fusneica Florentin

Parasite regizat de Bong Joon-ho

Script Output x Query Result 2 x

SQL | All Rows Fetched: 14 in 0.003 seconds

SPECTATOR_ID	NUME	PRENUME	NUMAR_TELEFON	SEX
9	9 Apostoiu	Antonio-Ciprian	0727761387	M
10	10 Varga	Robert	0767251023	M
11	11 Dima	Oana-Teodora	0720309259	F
12	12 Bigan	Marian	0720078066	M
13	13 Moanga	Natalia	07XXXXXXX	F
14	20 Fusneica	Florentin	0771434107	M

Spectatorul cu id-ul 21 nu există, așa că este introdus în tabel. El nu a optat pentru o achiziție de bilet, așa că nu va figura cu niciun film în tabelul bilet(cum se poate observa în DBMS\_OUTPUT).

Script Output x Query Result x

SQL | All Rows Fetched: 25 in 0.002 seconds

BILET_ID	RANDL	LOC	PRET	DATA_DIF	TIP_REDUCERE	FILM_ID	SALA_ID	SPECTATOR_ID
20	20	10	18	29.5 31-MAY-20	(null)	8	4	13
21	21	6	17	23.5 04-APR-19	(null)	9	1	1
22	22	7	9	19.5 05-JAN-21	Pensionar	10	11	12
23	23	9	10	21.5 19-JUN-19	Pensionar	9	12	4
24	112	12	14	19.5 19-NOV-20	(null)	7	9	6
25	115	10	12	19.5 19-NOV-20	(null)	7	9	20

## Nimară Dan Gabriel, grupa 241

```
955 IF istoric.count=0 THEN
956     DBMS_OUTPUT.PUT_LINE('Spectatorul nu a vizionat niciun film');
957 ELSE
958     FOR i in istoric.FIRST..istoric.LAST LOOP
959         DBMS_OUTPUT.PUT_LINE(istoric(i).titlu_film || ' ' || regizat de '||istoric(i).regizor);
960     END LOOP;
961 END IF;
962 END istoric_spectator;
963 END pack_ex14;
964 /
965
966 DECLARE
967     nr NUMBER;
968 BEGIN
969     pack_ex14.istoric_spectator(21, 0, 'Bertalan', 'Victor', '0722855111', 'M', pack_dni.NEXTVAL,
970     10, 12, 19.5, TO_DATE('19/11/2020','DD/MM/YYYY'), NULL, 'Parasite');
971 END;
972 /
973
```

Istoricul filmelor vizionate de Bertalan Victor

Spectatorul nu a vizionat niciun film