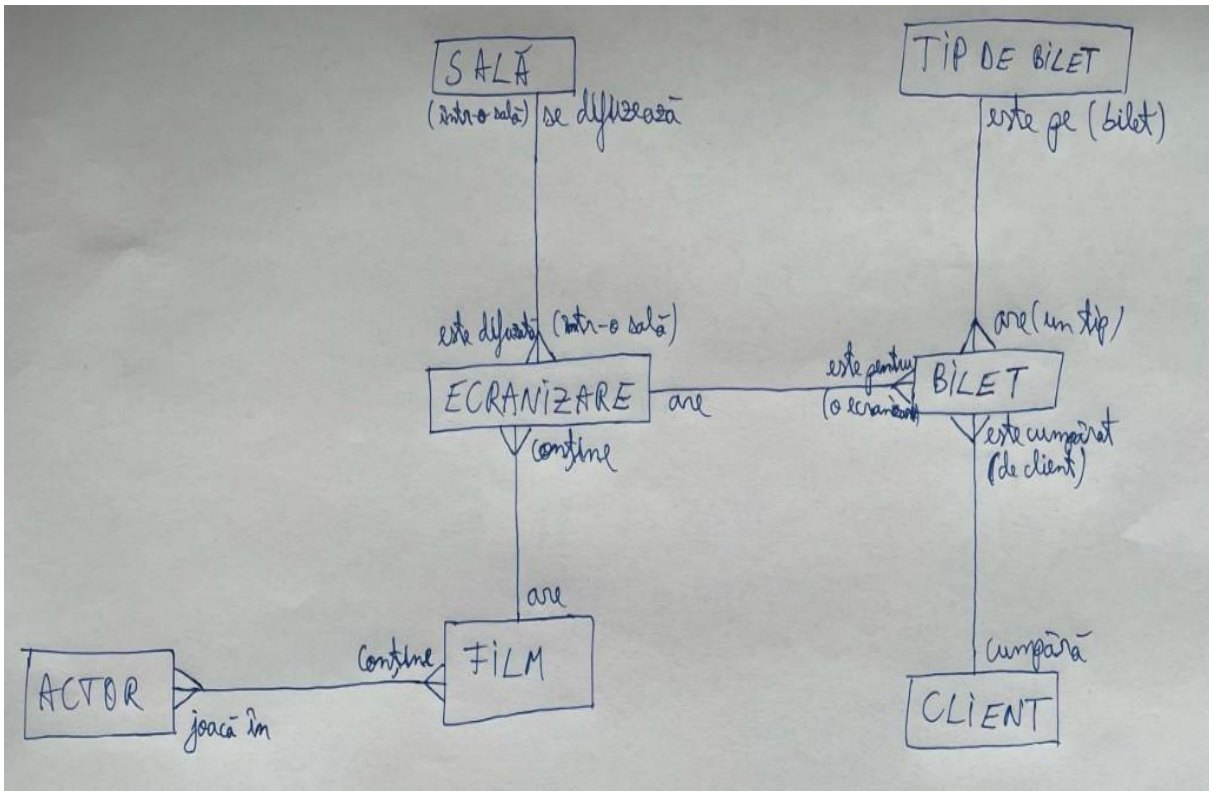
Tudorache Alexandru-Theodor
Grupa 242

Proiect SGBD

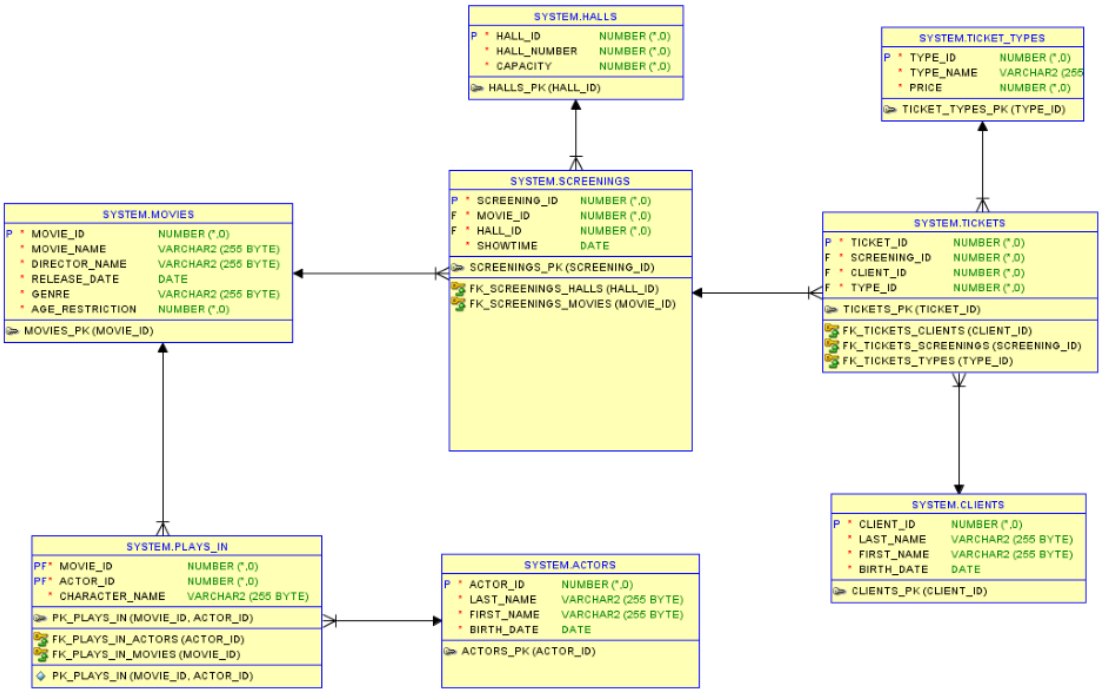
-Cinema Database-

1. Baza de date este a unui cinema cu o singură locație (nu un lanț de cinematografe precum Cinema City), dar cu săli multiple. Cu ajutorul acestei baze de date clienții își pot cumpăra bilete la anumite ecranizări ale filmelor pe care vor să le vizioneze, biletele având diferite tipuri pentru a putea determina prețul acestora. De asemenea, în baza de date se află și actorii care joacă în filmele care sunt ecranizate pentru a putea fi afișați alături de filmul/filmele în care joacă.

2.



3.



4.

```
CREATE TABLE ACTORS (
    actor_id INT PRIMARY KEY,
    last_name varchar(255) NOT NULL,
    first_name varchar(255) NOT NULL,
    birth_date DATE NOT NULL
);

CREATE TABLE MOVIES (
    movie_id INT PRIMARY KEY,
    movie_name varchar(255) NOT NULL,
    director_name varchar(255) NOT NULL,
    release_date DATE NOT NULL,
       genre varchar(255) NOT NULL,
    age_restriction int NOT NULL
);

CREATE TABLE PLAYS_IN (
    movie_id INT NOT NULL,
    actor_id INT NOT NULL,
    character_name varchar(255) NOT NULL,
       CONSTRAINT FK_PLAYS_IN_MOVIES FOREIGN KEY (movie_id) REFERENCES
MOVIES(movie_id),
       CONSTRAINT FK_PLAYS_IN_ACTORS FOREIGN KEY (actor_id) REFERENCES
ACTORS(actor_id),
       CONSTRAINT PK_PLAYS_IN PRIMARY KEY (movie_id, actor_id)
);

CREATE TABLE CLIENTS (
```

```sql
    client_id INT PRIMARY KEY,
    last_name varchar(255) NOT NULL,
    first_name varchar(255) NOT NULL,
    birth_date DATE NOT NULL
);

CREATE TABLE HALLS (
    hall_id INT PRIMARY KEY,
    hall_number INT NOT NULL,
      capacity INT NOT NULL
);

CREATE TABLE SCREENINGS (
    screening_id INT PRIMARY KEY,
    movie_id INT NOT NULL,
    hall_id INT NOT NULL,
      showtime DATE NOT NULL,
      CONSTRAINT FK_SCREENINGS_MOVIES FOREIGN KEY (movie_id) REFERENCES
MOVIES(movie_id),
      CONSTRAINT FK_SCREENINGS_HALLS FOREIGN KEY (hall_id) REFERENCES
HALLS(hall_id)
);

CREATE TABLE TICKET_TYPES (
    type_id INT PRIMARY KEY,
    type_name VARCHAR(255) NOT NULL,
      price INT NOT NULL
);

CREATE TABLE TICKETS (
    ticket_id INT PRIMARY KEY,
    screening_id INT NOT NULL,
    client_id INT NOT NULL,
    type_id INT NOT NULL,
      CONSTRAINT FK_TICKETS_SCREENINGS FOREIGN KEY (screening_id) REFERENCES
SCREENINGS(screening_id),
      CONSTRAINT FK_TICKETS_CLIENTS FOREIGN KEY (client_id) REFERENCES
CLIENTS(client_id),
      CONSTRAINT FK_TICKETS_TYPES FOREIGN KEY (type_id) REFERENCES
TICKET_TYPES(type_id)
);
```

```
Script Output   Query Result
Task completed in 0.266 seconds

Table ACTORS created.

Table MOVIES created.

Table PLAYS_IN created.

Table CLIENTS created.

Table HALLS created.

Table SCREENINGS created.

Table TICKET_TYPES created.

Table TICKETS created.
```

5.

```
-----ACTORS INSERTS-----

INSERT INTO ACTORS
VALUES (1, 'Leonardo', 'DiCaprio', TO_DATE('1974/11/11', 'yyyy/mm/dd'));

INSERT INTO ACTORS
VALUES (2, 'Jamie', 'Foxx', TO_DATE('1967/12/13', 'yyyy/mm/dd'));

INSERT INTO ACTORS
VALUES (3, 'Gary', 'Oldman', TO_DATE('1958/03/21', 'yyyy/mm/dd'));

INSERT INTO ACTORS
VALUES (4, 'Robert', 'Pattinson', TO_DATE('1986/05/13', 'yyyy/mm/dd'));

INSERT INTO ACTORS
VALUES (5, 'Matthew', 'McConaughey', TO_DATE('1969/11/04', 'yyyy/mm/dd'));

INSERT INTO ACTORS
VALUES (6, 'Joseph', 'Gordon-Levitt', TO_DATE('1981/02/17', 'yyyy/mm/dd'));

select * from actors;
```

Worksheet | Query Builder

```
-----ACTORS INSERTS-----

INSERT INTO ACTORS
VALUES (1, 'Leonardo', 'DiCaprio', TO_DATE('1974/11/11', 'yyyy/mm/dd'));

INSERT INTO ACTORS
VALUES (2, 'Jamie', 'Foxx', TO_DATE('1967/12/13', 'yyyy/mm/dd'));

INSERT INTO ACTORS
VALUES (3, 'Gary', 'Oldman', TO_DATE('1958/03/21', 'yyyy/mm/dd'));
```

Script Output | Query Result

Task completed in 0.124 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

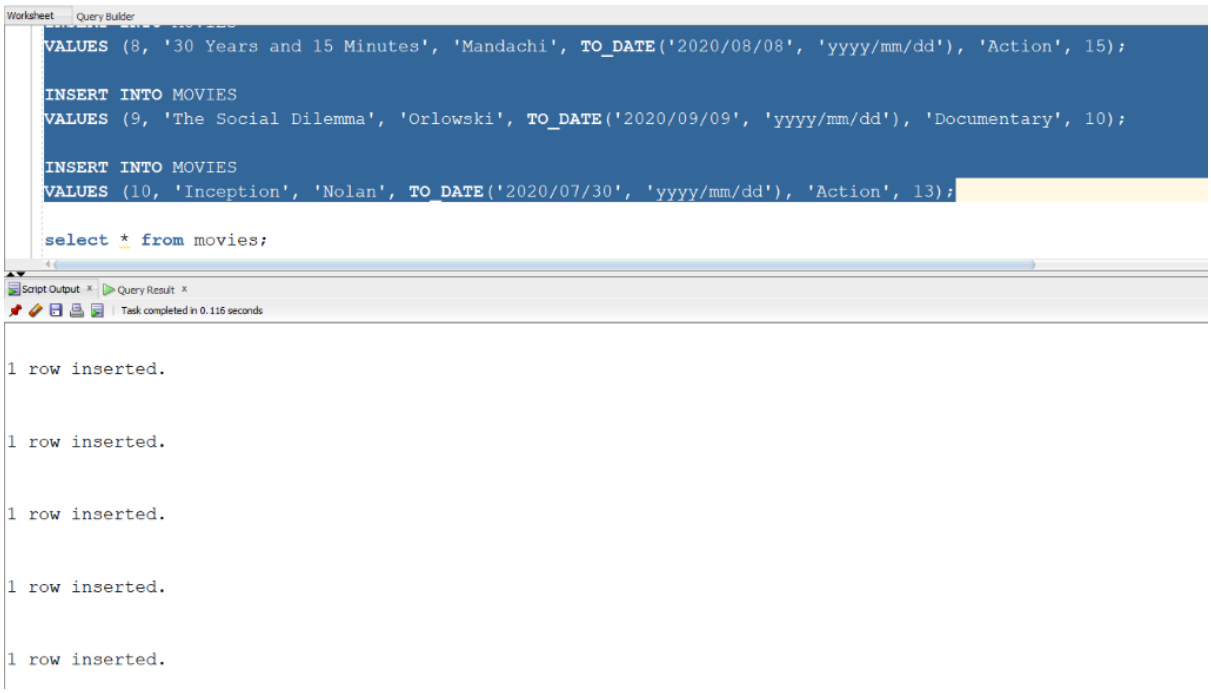
Script Output | Query Result

SQL | All Rows Fetched: 6 in 0.011 seconds

	ACTOR_ID	LAST_NAME	FIRST_NAME	BIRTH_DATE
1	1	Leonardo	DiCaprio	11-NOV-74
2	2	Jamie	Foxx	13-DEC-67
3	3	Gary	Oldman	21-MAR-58
4	4	Robert	Pattinson	13-MAY-86
5	5	Matthew	McConaughey	04-NOV-69
6	6	Joseph	Gordon-Levitt	17-FEB-81

```
-----MOVIES INSERTS-----

INSERT INTO MOVIES
VALUES (1, 'Soul', 'Docter', TO_DATE('2020/12/25', 'yyyy/mm/dd'), 'Drama', 10);

INSERT INTO MOVIES
VALUES (2, 'Mank', 'Fincher', TO_DATE('2020/11/13', 'yyyy/mm/dd'), 'Drama', 13);

INSERT INTO MOVIES
VALUES (3, 'Borat 2', 'Woliner', TO_DATE('2020/10/23', 'yyyy/mm/dd'), 'Comedy',
15);
```

```sql
INSERT INTO MOVIES
VALUES (4, 'Tenet', 'Nolan', TO_DATE('2020/09/18', 'yyyy/mm/dd'), 'Action', 13);

INSERT INTO MOVIES
VALUES (5, 'Interstellar', 'Nolan', TO_DATE('2014/11/07', 'yyyy/mm/dd'), 'Sci-
Fi', 13);

INSERT INTO MOVIES
VALUES (6, 'Dunkirk', 'Nolan', TO_DATE('2017/07/13', 'yyyy/mm/dd'), 'Action',
15);

INSERT INTO MOVIES
VALUES (7, 'Inglorious Basterds', 'Tarantino', TO_DATE('2020/09/04',
'yyyy/mm/dd'), 'Action', 13);

INSERT INTO MOVIES
VALUES (8, '30 Years and 15 Minutes', 'Mandachi', TO_DATE('2020/08/08',
'yyyy/mm/dd'), 'Action', 15);

INSERT INTO MOVIES
VALUES (9, 'The Social Dilemma', 'Orlowski', TO_DATE('2020/09/09',
'yyyy/mm/dd'), 'Documentary', 10);

INSERT INTO MOVIES
VALUES (10, 'Inception', 'Nolan', TO_DATE('2020/07/30', 'yyyy/mm/dd'), 'Action',
13);

select * from movies;
```



Script Output × | Query Result ×

SQL | All Rows Fetched: 10 in 0.002 seconds

	MOVIE_ID	MOVIE_NAME	DIRECTOR_NAME	RELEASE_DATE	GENRE	AGE_RESTRICTION
1	1	Soul	Docter	25-DEC-20	Drama	10
2	2	Mank	Fincher	13-NOV-20	Drama	13
3	3	Borat 2	Woliner	23-OCT-20	Comedy	15
4	4	Tenet	Nolan	18-SEP-20	Action	13
5	5	Interstellar	Nolan	07-NOV-14	Sci-Fi	13
6	6	Dunkirk	Nolan	13-JUL-17	Action	15
7	7	Inglorious Basterds	Tarantino	04-SEP-20	Action	13
8	8	30 Years and 15 ...	Mandachi	08-AUG-20	Action	15
9	9	The Social Dilemma	Orlowski	09-SEP-20	Doc...	10
10	10	Inception	Nolan	30-JUL-20	Action	13

```
-----PLAYS IN INSERTS-----

INSERT INTO PLAYS_IN
VALUES (1, 2, 'Joe');

INSERT INTO PLAYS_IN
VALUES (4, 4, 'Neil');

INSERT INTO PLAYS_IN
VALUES (4, 2, 'Protagonist');

INSERT INTO PLAYS_IN
VALUES (5, 5, 'Cooper');

INSERT INTO PLAYS_IN
VALUES (10, 1, 'Cobb');

INSERT INTO PLAYS_IN
VALUES (10, 6, 'Arthur');

INSERT INTO PLAYS_IN
VALUES (10, 5, 'Eames');

INSERT INTO PLAYS_IN
VALUES (7, 1, 'Aldo Raine');

INSERT INTO PLAYS_IN
VALUES (7, 6, 'Hans Landa');

INSERT INTO PLAYS_IN
VALUES (2, 5, 'Kane');

INSERT INTO PLAYS_IN
VALUES (2, 3, 'Mank');

INSERT INTO PLAYS_IN
VALUES (2, 4, 'Joe');

select * from plays_in;
```

Worksheet | Query Builder

```
VALUES (2, 5, 'Kane');

INSERT INTO PLAYS_IN
VALUES (2, 3, 'Mank');

INSERT INTO PLAYS_IN
VALUES (2, 4, 'Joe');

select * from plays_in;
```

Script Output | Query Result | Task completed in 0.133 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

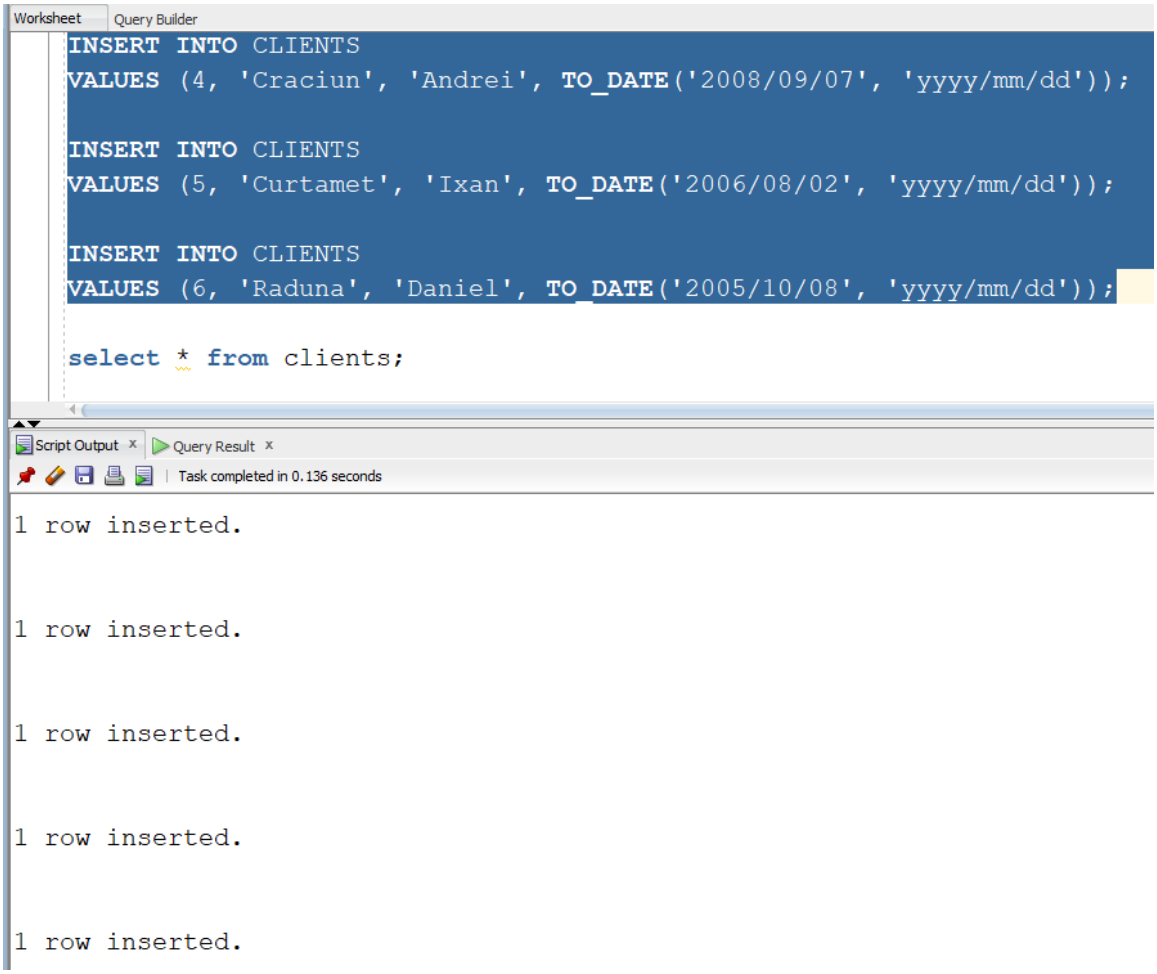
Script Output | Query Result | SQL | All Rows Fetched: 12 in 0.006 seconds

	MOVIE_ID	ACTOR_ID	CHARACTER_NAME
1	1	2	Joe
2	4	4	Neil
3	4	2	Protagonist
4	5	5	Cooper
5	10	1	Cobb
6	10	6	Arthur
7	10	5	Eames
8	7	1	Aldo Raine
9	7	6	Hans Landa
10	2	5	Kane
11	2	3	Mank
12	2	4	Joe

```
-----CLIENTS INSERTS-----

INSERT INTO CLIENTS
VALUES (1, 'Tudorache', 'Theodor', TO_DATE('2000/05/20', 'yyyy/mm/dd'));

INSERT INTO CLIENTS
```

```
VALUES (2, 'Bugheciu', 'Eduard', TO_DATE('2000/02/27', 'yyyy/mm/dd'));

INSERT INTO CLIENTS
VALUES (3, 'Constantin', 'Sorin', TO_DATE('1987/07/17', 'yyyy/mm/dd'));

INSERT INTO CLIENTS
VALUES (4, 'Craciun', 'Andrei', TO_DATE('2008/09/07', 'yyyy/mm/dd'));

INSERT INTO CLIENTS
VALUES (5, 'Curtamet', 'Ixan', TO_DATE('2006/08/02', 'yyyy/mm/dd'));

INSERT INTO CLIENTS
VALUES (6, 'Raduna', 'Daniel', TO_DATE('2005/10/08', 'yyyy/mm/dd'));
```

select * from clients;



Script Output x | Query Result x

SQL | All Rows Fetched: 6 in 0.006 seconds

	CLIENT_ID	LAST_NAME	FIRST_NAME	BIRTH_DATE
1	1	Tudorache	Theodor	20-MAY-00
2	2	Bugheciu	Eduard	27-FEB-00
3	3	Constantin	Sorin	17-JUL-87
4	4	Craciun	Andrei	07-SEP-08
5	5	Curtamet	Ixan	02-AUG-06
6	6	Raduna	Daniel	08-OCT-05

```
-----HALLS INSERTS-----

INSERT INTO HALLS
VALUES (1, 1, 100);

INSERT INTO HALLS
VALUES (2, 2, 50);

INSERT INTO HALLS
VALUES (3, 3, 250);

INSERT INTO HALLS
VALUES (4, 4, 100);

INSERT INTO HALLS
VALUES (5, 5, 150);

INSERT INTO HALLS
VALUES (6, 6, 5);

select * from halls;
```

Worksheet | Query Builder

```
INSERT INTO HALLS
VALUES (4, 4, 100);

INSERT INTO HALLS
VALUES (5, 5, 150);

INSERT INTO HALLS
VALUES (6, 6, 5);

select * from halls;
```

Script Output | Query Result | Task completed in 0.107 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

Script Output | Query Result | SQL | All Rows Fetched: 6 in 0.007 s

	HALL_ID	HALL_NUMBER	CAPACITY
1	1	1	100
2	2	2	50
3	3	3	250
4	4	4	100
5	5	5	150
6	6	6	5

```
-----SCREENINGS INSERTS-----

INSERT INTO SCREENINGS
VALUES (1, 1, 6, TO_DATE('2020-12-25 13:00', 'YYYY-MM-DD HH24:MI'));
```

```sql
INSERT INTO SCREENINGS
VALUES (2, 3, 6, TO_DATE('2020-12-25 17:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (3, 1, 1, TO_DATE('2020-12-30 21:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (4, 2, 2, TO_DATE('2021-01-02 20:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (5, 4, 3, TO_DATE('2020-01-03 17:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (6, 4, 3, TO_DATE('2021-01-03 17:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (7, 5, 4, TO_DATE('2021-01-03 13:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (8, 6, 5, TO_DATE('2021-01-05 11:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (9, 7, 6, TO_DATE('2021-01-05 20:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (10, 8, 6, TO_DATE('2021-01-01 21:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (11, 9, 5, TO_DATE('2020-12-29 17:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (12, 10, 4, TO_DATE('2020-12-31 10:00', 'YYYY-MM-DD HH24:MI'));

select * from screenings;
```

Worksheet | Query Builder

```sql
INSERT INTO SCREENINGS
VALUES (10, 8, 6, TO_DATE('2021-01-01 21:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (11, 9, 5, TO_DATE('2020-12-29 17:00', 'YYYY-MM-DD HH24:MI'));

INSERT INTO SCREENINGS
VALUES (12, 10, 4, TO_DATE('2020-12-31 10:00', 'YYYY-MM-DD HH24:MI'));

select * from screenings;
```

Script Output | Query Result

Task completed in 0.136 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

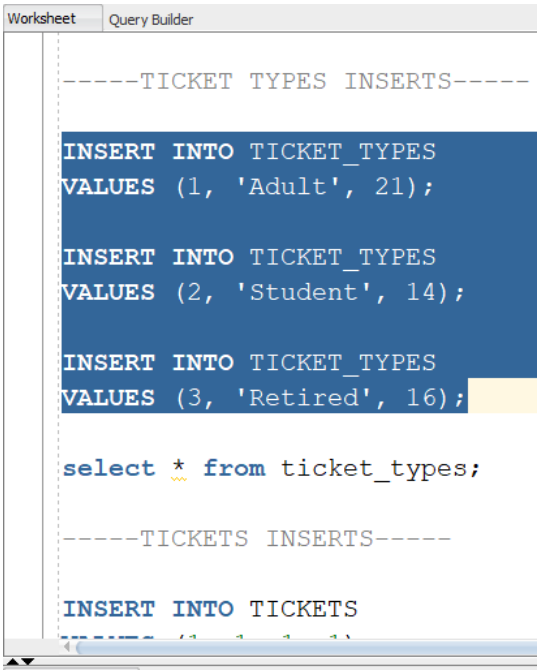
Script Output | Query Result

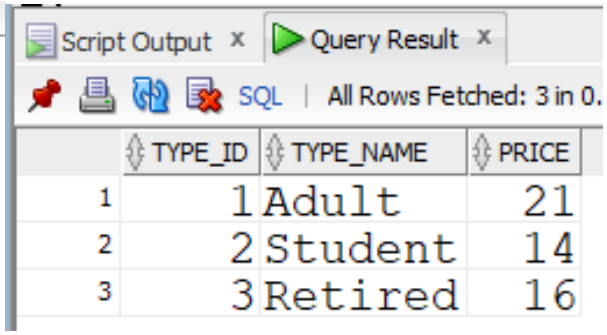
SQL | All Rows Fetched: 12 in 0.006 seconds

	SCREENING_ID	MOVIE_ID	HALL_ID	SHOWTIME
1	1	1	6	25-DEC-20
2	2	3	6	25-DEC-20
3	3	1	1	30-DEC-20
4	4	2	2	02-JAN-21
5	5	4	3	03-JAN-20
6	6	4	3	03-JAN-21
7	7	5	4	03-JAN-21
8	8	6	5	05-JAN-21
9	9	7	6	05-JAN-21
10	10	8	6	01-JAN-21
11	11	9	5	29-DEC-20
12	12	10	4	31-DEC-20

```sql
-----TICKET TYPES INSERTS-----

INSERT INTO TICKET_TYPES
VALUES (1, 'Adult', 21);

INSERT INTO TICKET_TYPES
```

```
VALUES (2, 'Student', 14);

INSERT INTO TICKET_TYPES
VALUES (3, 'Retired', 16);

select * from ticket_types;
```

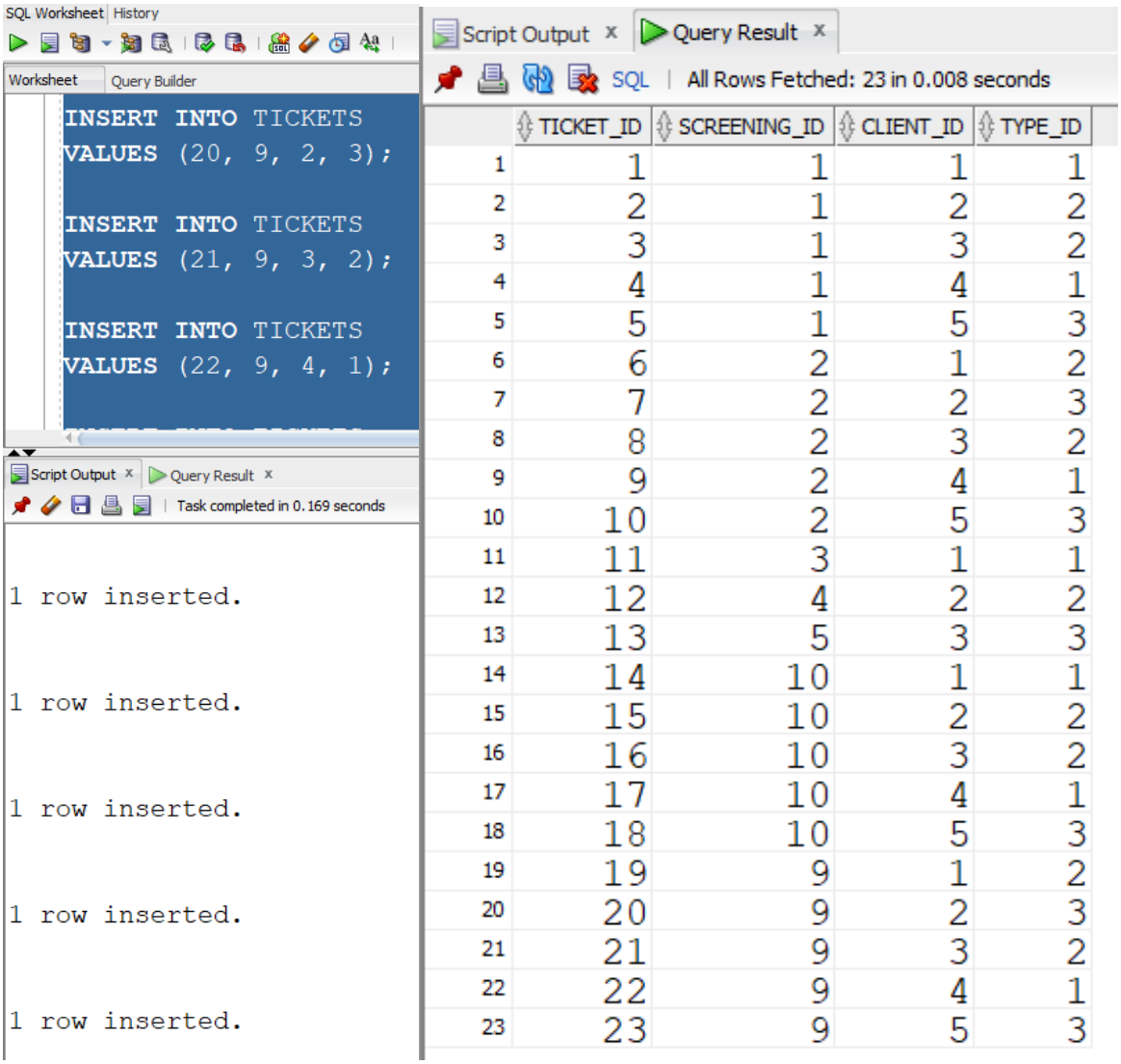





```
-----TICKETS INSERTS-----

INSERT INTO TICKETS
VALUES (1, 1, 1, 1);

INSERT INTO TICKETS
VALUES (2, 1, 2, 2);

INSERT INTO TICKETS
VALUES (3, 1, 3, 2);

INSERT INTO TICKETS
VALUES (4, 1, 4, 1);
```

```sql
INSERT INTO TICKETS
VALUES (5, 1, 5, 3);

INSERT INTO TICKETS
VALUES (6, 2, 1, 2);

INSERT INTO TICKETS
VALUES (7, 2, 2, 3);

INSERT INTO TICKETS
VALUES (8, 2, 3, 2);

INSERT INTO TICKETS
VALUES (9, 2, 4, 1);

INSERT INTO TICKETS
VALUES (10, 2, 5, 3);

INSERT INTO TICKETS
VALUES (11, 3, 1, 1);

INSERT INTO TICKETS
VALUES (12, 4, 2, 2);

INSERT INTO TICKETS
VALUES (13, 5, 3, 3);

INSERT INTO TICKETS
VALUES (14, 10, 1, 1);

INSERT INTO TICKETS
VALUES (15, 10, 2, 2);

INSERT INTO TICKETS
VALUES (16, 10, 3, 2);

INSERT INTO TICKETS
VALUES (17, 10, 4, 1);

INSERT INTO TICKETS
VALUES (18, 10, 5, 3);

INSERT INTO TICKETS
VALUES (19, 9, 1, 2);

INSERT INTO TICKETS
VALUES (20, 9, 2, 3);

INSERT INTO TICKETS
VALUES (21, 9, 3, 2);
```

```
INSERT INTO TICKETS
VALUES (22, 9, 4, 1);

INSERT INTO TICKETS
VALUES (23, 9, 5, 3);

select * from tickets;
```

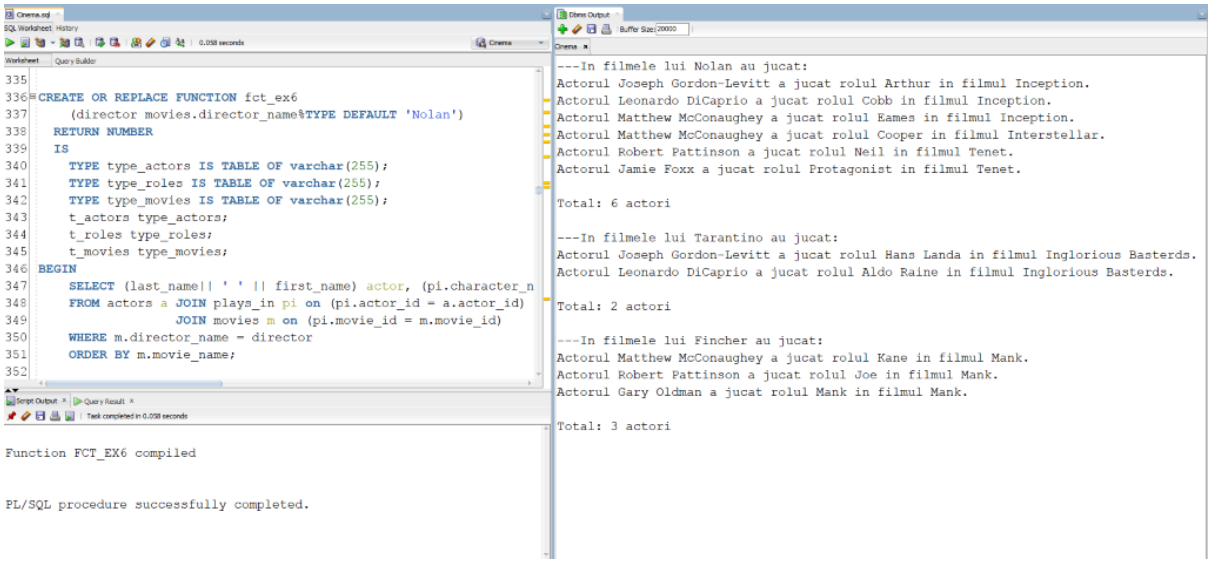


	TICKET_ID	SCREENING_ID	CLIENT_ID	TYPE_ID
1	1	1	1	1
2	2	1	2	2
3	3	1	3	2
4	4	1	4	1
5	5	1	5	3
6	6	2	1	2
7	7	2	2	3
8	8	2	3	2
9	9	2	4	1
10	10	2	5	3
11	11	3	1	1
12	12	4	2	2
13	13	5	3	3
14	14	10	1	1
15	15	10	2	2
16	16	10	3	2
17	17	10	4	1
18	18	10	5	3
19	19	9	1	2
20	20	9	2	3
21	21	9	3	2
22	22	9	4	1
23	23	9	5	3

6. Să se definească o funcție care să afișeze numele actorilor, dar și personajul lor, care au jucat în filmele regizate de un regizor cu nume dat că parametru(Parametrul default va fi 'Nolan'). Acest subprogram trebuie să returneze numărul acestor actori.

```
CREATE OR REPLACE FUNCTION fct_ex6
      (director movies.director_name%TYPE DEFAULT 'Nolan')
  RETURN NUMBER
```

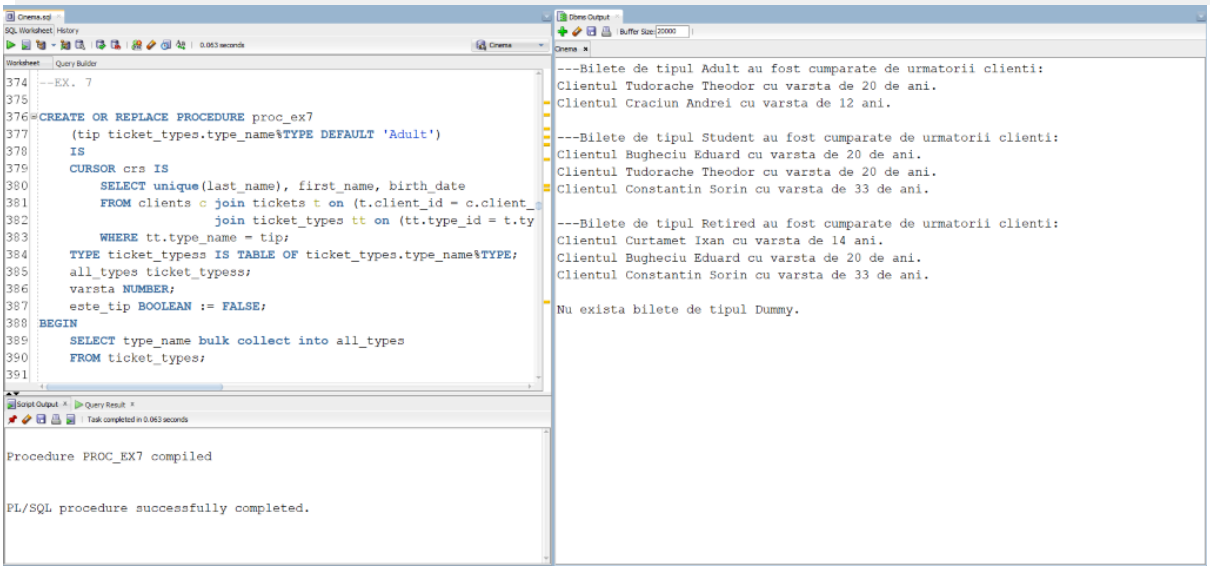
```
  IS
      TYPE type_actors IS TABLE OF varchar(255);
      TYPE type_roles IS TABLE OF varchar(255);
      TYPE type_movies IS TABLE OF varchar(255);
    t_actors type_actors;
    t_roles type_roles;
    t_movies type_movies;
BEGIN
      SELECT (last_name|| ' ' || first_name) actor,
(pi.character_name) personaj, (m.movie_name) BULK COLLECT INTO
t_actors, t_roles, t_movies
      FROM actors a JOIN plays_in pi on (pi.actor_id = a.actor_id)
                 JOIN movies m on (pi.movie_id = m.movie_id)
      WHERE m.director_name = director
      ORDER BY m.movie_name;

      DBMS_OUTPUT.PUT_LINE('---In filmele lui ' || director || ' au
jucat:');

      FOR I in t_actors.first..t_actors.last LOOP
      DBMS_OUTPUT.PUT_LINE('Actorul ' || t_actors(i) || ' a jucat
rolul ' || t_roles(i) || ' in filmul ' || t_movies(i) || '.');
      END LOOP;
      DBMS_OUTPUT.NEW_LINE;
      RETURN t_actors.count;
END fct_ex6;
/

--APELARE:

BEGIN
      DBMS_OUTPUT.PUT_LINE('Total: ' || fct_ex6() || ' actori');
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('Total: ' || fct_ex6('Tarantino') || '
actori');
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('Total: ' || fct_ex6('Fincher') || '
actori');
END;
/
```



7. Să se definească o procedură care să afișeze(o singură dată) numele și vârstă tuturor clienților care au cumpărat bilete de tipul dat ca parametru(Parametrul default va fi 'Adult').

```
CREATE OR REPLACE PROCEDURE proc_ex7
      (tip ticket_types.type_name%TYPE DEFAULT 'Adult')
      IS
      CURSOR crs IS
      SELECT unique(last_name), first_name, birth_date
      FROM clients c join tickets t on (t.client_id = c.client_id)
                        join ticket_types tt on (tt.type_id =
t.type_id)
      WHERE tt.type_name = tip;
      TYPE ticket_typess IS TABLE OF ticket_types.type_name%TYPE;
     all_types ticket_typess;
     varsta NUMBER;
     este_tip BOOLEAN := FALSE;
BEGIN
      SELECT type_name bulk collect into all_types
      FROM ticket_types;

      FOR i IN all_types.first..all_types.last LOOP
      IF (all_types(i) = tip) THEN
            este_tip := TRUE;
      END IF;
      END LOOP;
      IF (este_tip = TRUE) THEN
      DBMS_OUTPUT.PUT_LINE('---Bilete de tipul ' || tip || ' au
fost cumparate de urmatorii clienti: ');
      ELSE
```

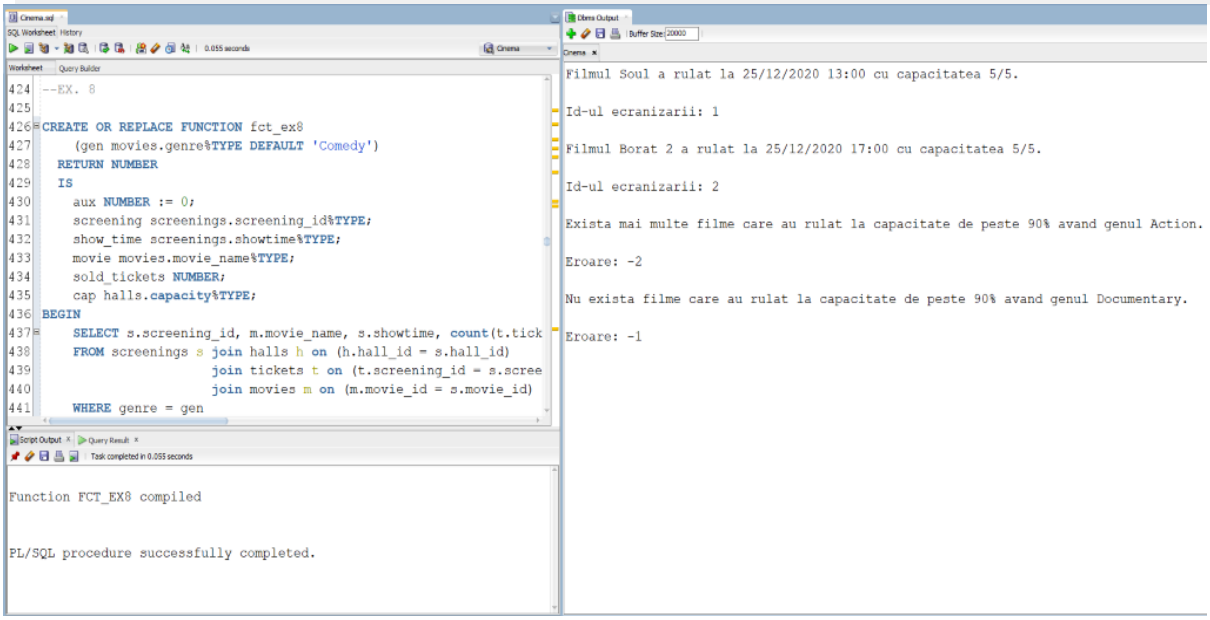
```
      DBMS_OUTPUT.PUT_LINE ('Nu exista bilete de tipul ' || tip ||
'.');
      END IF;
      FOR i in crs LOOP
         varsta := TRUNC(MONTHS_BETWEEN(sysdate, i.birth_date) /
12);
      IF (varsta >= 20) THEN
            DBMS_OUTPUT.PUT_LINE('Clientul ' || i.last_name || ' '
|| i.first_name ||  ' cu varsta de ' ||
TRUNC(MONTHS_BETWEEN(sysdate, i.birth_date) / 12) || ' de ani.');
      ELSE
            DBMS_OUTPUT.PUT_LINE('Clientul ' || i.last_name || ' '
|| i.first_name ||  ' cu varsta de ' ||
TRUNC(MONTHS_BETWEEN(sysdate, i.birth_date) / 12) || ' ani.');
      END IF;
      END LOOP;
      DBMS_OUTPUT.NEW_LINE;
END proc_ex7;
/

--APELARE:

BEGIN
      proc_ex7();
      proc_ex7('Student');
      proc_ex7('Retired');
      proc_ex7('Dummy'); -- nu se va afisa nimic
END;
/
```



8. Să se definească o funcție care să afișeze următoarele detalii: (numele filmului; dată și ora la care a fost ecranizat) despre filmul care a rulat la o capacitate de peste 90% din sala în care au fost ecranizate, având genul dat ca parametru. Subprogramul trebuie să returneze id-ul ecranizării respective.

```
CREATE OR REPLACE FUNCTION fct_ex8
      (gen movies.genre%TYPE DEFAULT 'Comedy')
  RETURN NUMBER
  IS
      aux NUMBER := 0;
      screening screenings.screening_id%TYPE;
    show_time screenings.showtime%TYPE;
      movie movies.movie_name%TYPE;
    sold_tickets NUMBER;
      cap halls.capacity%TYPE;
BEGIN
      SELECT s.screening_id, m.movie_name, s.showtime,
count(t.ticket_id) , h.capacity INTO screening, movie,
show_time,sold_tickets, cap
      FROM screenings s join halls h on (h.hall_id = s.hall_id)
                        join tickets t on (t.screening_id =
s.screening_id)
                        join movies m on (m.movie_id = s.movie_id)
      WHERE genre = gen
      GROUP BY s.screening_id, m.movie_name, s.showtime ,h.capacity
      HAVING COUNT(t.ticket_id) >= h.capacity * (9/10);
      DBMS_OUTPUT.PUT_LINE('Filmul ' || movie || ' a rulat la ' ||
TO_CHAR(show_time, 'dd/mm/yyyy HH24:MI') || ' cu capacitatea ' ||
sold_tickets || '/' || cap || '.');
      DBMS_OUTPUT.NEW_LINE;
      RETURN screening;
      EXCEPTION
      WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista filme care au rulat la
capacitate de peste 90% avand genul ' || gen || '.');
            DBMS_OUTPUT.NEW_LINE;
            return -1;
      WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Exista mai multe filme care au
rulat la capacitate de peste 90% avand genul ' || gen || '.');
            DBMS_OUTPUT.NEW_LINE;
            return -2;
```

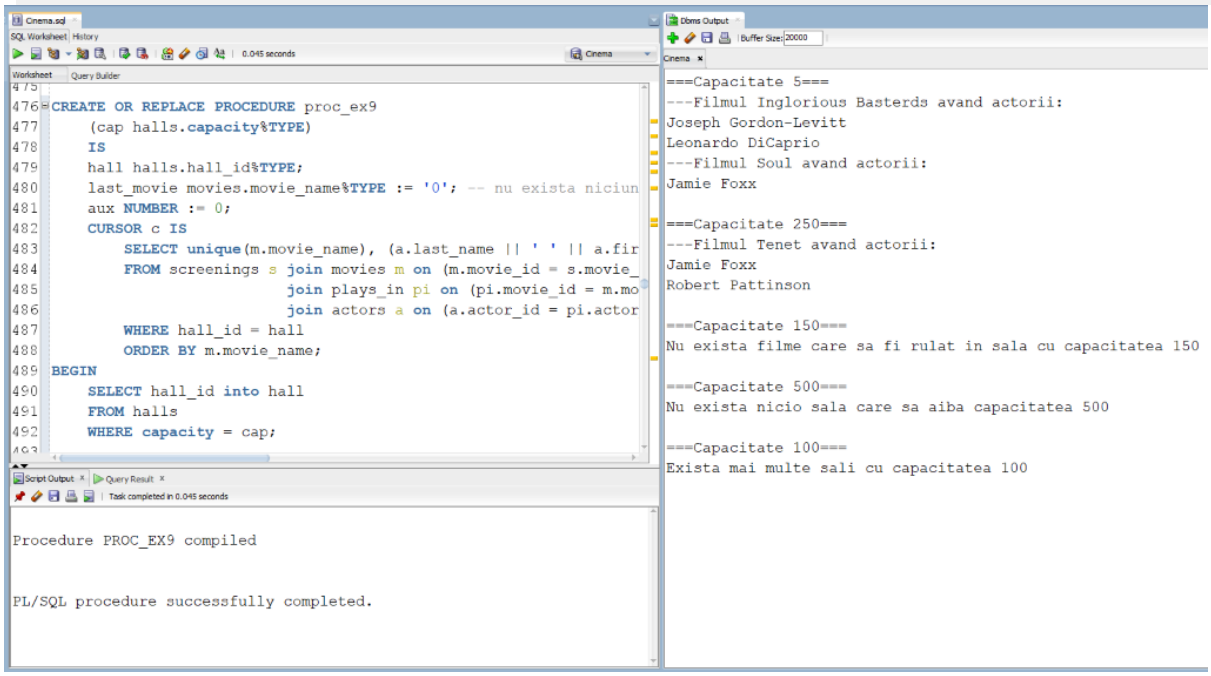
```
      WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Alta eroare!');
            return -3;
END fct_ex8;
/

--APELARE:

BEGIN
      DBMS_OUTPUT.PUT_LINE('Id-ul ecranizarii: ' ||
fct_ex8('Drama'));
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('Id-ul ecranizarii: ' ||
fct_ex8('Comedy'));
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('Eroare: ' || fct_ex8('Action')); -- mai
mult de 1 film
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('Eroare: ' || fct_ex8('Documentary')); -
- niciun film
      DBMS_OUTPUT.NEW_LINE;
END;
/
```



9. Să se definească o procedură care să afișeze numele filmelor(o singură dată) și numele actorilor care au jucat în ele, care au rulat în sala cu capacitate dată ca parametru.

```
CREATE OR REPLACE PROCEDURE proc_ex9
      (cap halls.capacity%TYPE)
      IS
      hall halls.hall_id%TYPE;
    last_movie movies.movie_name%TYPE := '0'; -- nu exista niciun
film in baza de date cu numele '0'
      aux NUMBER := 0;
      CURSOR c IS
      SELECT unique(m.movie_name), (a.last_name || ' ' ||
a.first_name) nume
      FROM screenings s join movies m on (m.movie_id = s.movie_id)
                              join plays_in pi on (pi.movie_id =
m.movie_id)
                              join actors a on (a.actor_id =
pi.actor_id)
      WHERE hall_id = hall
      ORDER BY m.movie_name;
BEGIN
      SELECT hall_id into hall
      FROM halls
      WHERE capacity = cap;

      FOR i in c LOOP
      IF (i.movie_name != last_movie) THEN
            DBMS_OUTPUT.PUT_LINE ('---Filmul ' || i.movie_name || '
avand actorii: ');
      END IF;
      DBMS_OUTPUT.PUT_LINE (i.nume);
         last_movie := i.movie_name;
         aux := aux + 1;
      END LOOP;
      IF (aux = 0) THEN
      DBMS_OUTPUT.PUT_LINE('Nu exista filme care sa fi rulat in
sala cu capacitatea ' || cap);
      END IF;
      EXCEPTION
      WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Exista mai multe sali cu
capacitatea ' || cap);
      WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista nicio sala care sa aiba
capacitatea ' || cap);
```

```
      WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Alta eroare');
END proc_ex9;
/

--APELARE:

BEGIN
      DBMS_OUTPUT.PUT_LINE('===Capacitate 5===');
      proc_ex9(5); -- mai multe filme
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('===Capacitate 250===');
      proc_ex9(250); -- un singur film
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('===Capacitate 150===');
      proc_ex9(150); -- niciun film
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('===Capacitate 500===');
      proc_ex9(500); -- nu exista capacitatea
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('===Capacitate 100===');
      proc_ex9(100); -- mai multe sali cu capacitatea respectiva
END;
/
```



10. Să se definească un declanșator care să nu permită nimănui să lucreze în ziua de Crăciun cu comenzile UPDATE, INSERT, DELETE pe tabela filmelor. (Declanșatorul se va testa cu ziua în care ne aflăm, în loc de 25/12.)

```
CREATE OR REPLACE TRIGGER trig_ex10
      BEFORE INSERT OR UPDATE OR DELETE ON movies
BEGIN
      IF (TO_CHAR(SYSDATE,'DD/MM') = '25/12') THEN
      RAISE_APPLICATION_ERROR(-20001,'Ia o pauza! E Craciunul.');
      END IF;
END;
/

--TESTARE:

INSERT INTO MOVIES
VALUES (15, 'Soul', 'Docter', TO_DATE('2020/12/25', 'yyyy/mm/dd'),
'Drama', 10);

DELETE FROM MOVIES
WHERE movie_id = 11;

UPDATE MOVIES
SET movie_name = 'Film'
WHERE movie_id = 11;
```

```
Trigger TRIG_EX10 compiled


Error starting at line : 546 in command -
INSERT INTO MOVIES
VALUES (15, 'Soul', 'Docter', TO_DATE('2020/12/25', 'yyyy/mm/dd'), 'Drama', 10)
Error report -
ORA-20001: Ia o pauza! E Craciunul.
ORA-06512: at "SYSTEM.TRIG_EX10", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRIG_EX10'


Error starting at line : 549 in command -
DELETE FROM MOVIES
WHERE movie_id = 11
Error report -
ORA-20001: Ia o pauza! E Craciunul.
ORA-06512: at "SYSTEM.TRIG_EX10", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRIG_EX10'
```

```
Error starting at line : 552 in command -
UPDATE MOVIES
SET movie_name = 'Film'
WHERE movie_id = 11
Error report -
ORA-20001: Ia o pauza! E Craciunul.
ORA-06512: at "SYSTEM.TRIG_EX10", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRIG_EX10'
```

11. Să se definească un declanșator care să nu permită(să afișeze o eroare) inserarea în tabela biletelor a unor valori corespunzătoare unui client care nu are vârsta necesară pentru a putea vedea o anumită ecranizare a unui film.

--functie auxiliara pentru declansator

```
CREATE OR REPLACE FUNCTION are_voie_ex11
      (screening tickets.screening_id%TYPE, id_client
clients.client_id%TYPE)
      RETURN BOOLEAN
      IS
      CURSOR crs IS
      SELECT m.age_restriction
      FROM screenings s join movies m on (m.movie_id = s.movie_id)
      WHERE s.screening_id = screening;
```

```
      CURSOR crs2 IS
      SELECT TRUNC(MONTHS_BETWEEN(sysdate, birth_date) / 12)
      FROM clients
      WHERE client_id = id_client;
      age number;
    age_r movies.age_restriction%TYPE;
BEGIN
      OPEN crs;
      FETCH crs INTO age_r;
      CLOSE crs;
      OPEN crs2;
      FETCH crs2 INTO age;
      CLOSE crs2;
      IF (age_r <= age) THEN
      RETURN TRUE;
      ELSE
      RETURN FALSE;
      END IF;
END are_voie_ex11;
/
CREATE OR REPLACE TRIGGER trig_ex11
      BEFORE INSERT ON tickets
      FOR EACH ROW
BEGIN
      IF (are_voie_ex11(:NEW.screening_id, :NEW.client_id) = FALSE)
THEN
      RAISE_APPLICATION_ERROR(-20001,'ATENTIE! Clientul nu are
varsta necesara pentru a vedea acest film!');
      END IF;
END;
/

--TESTARE:

INSERT INTO TICKETS
VALUES (28, 2, 4, 2); -- nu are voie

INSERT INTO TICKETS
VALUES (28, 2, 1, 2); -- are voie
```

```
/

--TESTARE:

INSERT INTO TICKETS
VALUES (28, 2, 4, 2); --nu are voie

INSERT INTO TICKETS
VALUES (28, 2, 1, 2); --are voie

--EX. 12
```

```
Trigger TRIG_EX11 compiled


Error starting at line : 619 in command -
INSERT INTO TICKETS
VALUES (28, 2, 4, 2)
Error report -
ORA-20001: ATENTIE! Clientul nu are varsta necesara pentru a vedea acest film!
ORA-06512: at "SYSTEM.TRIG_EX11", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRIG_EX11'


1 row inserted.
```

12. Să se definească un declanșator care să introducă date într-un tabel de tip audit după ce utilizatorul a folosit o comandă LDD.

```
CREATE TABLE audit_ex12 (
    user_name VARCHAR2(30),
    database_name VARCHAR2(50),
     event VARCHAR2(20),
    object_name VARCHAR2(30),
    date_triggered DATE
);

CREATE OR REPLACE TRIGGER trig_ex12
     AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
     INSERT INTO audit_ex12
     VALUES (SYS.LOGIN_USER,
          SYS.DATABASE_NAME,
```

```
            SYS.SYSEVENT,
            SYS.DICTIONARY_OBJ_NAME,
            SYSDATE);
END;
/

--TESTARE:

CREATE INDEX ind_ex12
ON movies(movie_name);

DROP INDEX ind_ex12;

SELECT * FROM audit_ex12;
```

Script Output | Query Result

Task completed in 0.074 seconds

```
Table AUDIT_EX12 created.

Trigger TRIG_EX12 compiled

Index IND_EX12 created.

Index IND_EX12 dropped.
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.011 seconds

	USER_NAME	DATABASE_NAME	EVENT	OBJECT_NAME	DATE_TRIGGERED
1	SYSTEM	XE	CREATE	IND_EX12	08-JAN-21
2	SYSTEM	XE	DROP	IND_EX12	08-JAN-21

13.

```
CREATE OR REPLACE PACKAGE package_ex13 AS
      FUNCTION pack_fct_ex6(director movies.director_name%TYPE
DEFAULT 'Nolan')
      RETURN NUMBER;
      PROCEDURE pack_proc_ex7(tip ticket_types.type_name%TYPE
DEFAULT 'Adult');
      FUNCTION pack_fct_ex8(gen movies.genre%TYPE DEFAULT 'Comedy')
      RETURN NUMBER;
      PROCEDURE pack_proc_ex9(cap halls.capacity%TYPE);
      FUNCTION pack_are_voie_ex11(screening
tickets.screening_id%TYPE, id_client clients.client_id%TYPE)
      RETURN BOOLEAN;
END package_ex13;
/

CREATE OR REPLACE PACKAGE BODY package_ex13 AS
      FUNCTION pack_fct_ex6
      (director movies.director_name%TYPE DEFAULT 'Nolan')
      RETURN NUMBER
      IS
      TYPE type_actors IS TABLE OF varchar(255);
      TYPE type_roles IS TABLE OF varchar(255);
      TYPE type_movies IS TABLE OF varchar(255);
         t_actors type_actors;
         t_roles type_roles;
         t_movies type_movies;
      BEGIN
      SELECT (last_name|| ' ' || first_name) actor,
(pi.character_name) personaj, (m.movie_name) BULK COLLECT INTO
t_actors, t_roles, t_movies
      FROM actors a JOIN plays_in pi on (pi.actor_id = a.actor_id)
                       JOIN movies m on (pi.movie_id = m.movie_id)
      WHERE m.director_name = director
      ORDER BY m.movie_name;

      DBMS_OUTPUT.PUT_LINE('---In filmele lui ' || director || ' au
jucat:');

      FOR I in t_actors.first..t_actors.last LOOP
            DBMS_OUTPUT.PUT_LINE('Actorul ' || t_actors(i) || ' a
```

```
jucat rolul ' || t_roles(i) || ' in filmul ' || t_movies(i) ||
'.');
      END LOOP;
      DBMS_OUTPUT.NEW_LINE;
      RETURN t_actors.count;
      END pack_fct_ex6;


      PROCEDURE pack_proc_ex7
      (tip ticket_types.type_name%TYPE DEFAULT 'Adult')
      IS
      CURSOR crs IS
            SELECT unique(last_name), first_name, birth_date
            FROM clients c join tickets t on (t.client_id =
c.client_id)
                              join ticket_types tt on (tt.type_id =
t.type_id)
            WHERE tt.type_name = tip;
      TYPE ticket_typess IS TABLE OF ticket_types.type_name%TYPE;
         all_types ticket_typess;
         varsta NUMBER;
         este_tip BOOLEAN := FALSE;
      BEGIN
      SELECT type_name bulk collect into all_types
      FROM ticket_types;

      FOR i IN all_types.first..all_types.last LOOP
            IF (all_types(i) = tip) THEN
                  este_tip := TRUE;
            END IF;
      END LOOP;
      IF (este_tip = TRUE) THEN
            DBMS_OUTPUT.PUT_LINE('---Bilete de tipul ' || tip || '
au fost cumparate de urmatorii clienti: ');
      ELSE
            DBMS_OUTPUT.PUT_LINE ('Nu exista bilete de tipul ' ||
tip || '.');
      END IF;
      FOR i in crs LOOP
             varsta := TRUNC(MONTHS_BETWEEN(sysdate, i.birth_date)
/ 12);
            IF (varsta >= 20) THEN
```

```
                DBMS_OUTPUT.PUT_LINE('Clientul ' || i.last_name ||
' ' || i.first_name ||  ' cu varsta de ' ||
TRUNC(MONTHS_BETWEEN(sysdate, i.birth_date) / 12) || ' de ani.');
           ELSE
                DBMS_OUTPUT.PUT_LINE('Clientul ' || i.last_name ||
' ' || i.first_name ||  ' cu varsta de ' ||
TRUNC(MONTHS_BETWEEN(sysdate, i.birth_date) / 12) || ' ani.');
           END IF;
      END LOOP;
      DBMS_OUTPUT.NEW_LINE;
      END pack_proc_ex7;


      FUNCTION pack_fct_ex8
      (gen movies.genre%TYPE DEFAULT 'Comedy')
      RETURN NUMBER
      IS
      aux NUMBER := 0;
      screening screenings.screening_id%TYPE;
         show_time screenings.showtime%TYPE;
      movie movies.movie_name%TYPE;
         sold_tickets NUMBER;
      cap halls.capacity%TYPE;
      BEGIN
      SELECT s.screening_id, m.movie_name, s.showtime,
count(t.ticket_id) , h.capacity INTO screening, movie,
show_time,sold_tickets, cap
      FROM screenings s join halls h on (h.hall_id = s.hall_id)
                              join tickets t on (t.screening_id =
s.screening_id)
                              join movies m on (m.movie_id =
s.movie_id)
      WHERE genre = gen
      GROUP BY s.screening_id, m.movie_name, s.showtime ,h.capacity
      HAVING COUNT(t.ticket_id) >= h.capacity * (9/10);
      DBMS_OUTPUT.PUT_LINE('Filmul ' || movie || ' a rulat la ' ||
TO_CHAR(show_time, 'dd/mm/yyyy HH24:MI') || ' cu capacitatea ' ||
sold_tickets || '/' || cap || '.');
      DBMS_OUTPUT.NEW_LINE;
      RETURN screening;
      EXCEPTION
           WHEN NO_DATA_FOUND THEN
```

```
                DBMS_OUTPUT.PUT_LINE('Nu exista filme care au
rulat la capacitate de peste 90% avand genul ' || gen);
                DBMS_OUTPUT.NEW_LINE;
                return -1;
           WHEN TOO_MANY_ROWS THEN
                DBMS_OUTPUT.PUT_LINE('Exista mai multe filme care
au rulat la capacitate de peste 90% avand genul ' || gen);
                DBMS_OUTPUT.NEW_LINE;
                return -2;
           WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('Alta eroare!');
                return -3;
      END pack_fct_ex8;

      PROCEDURE pack_proc_ex9
      (cap halls.capacity%TYPE)
      IS
      hall halls.hall_id%TYPE;
        last_movie movies.movie_name%TYPE := '0'; -- nu exista
niciun film in baza de date cu numele '0'
      aux NUMBER := 0;
      CURSOR c IS
           SELECT unique(m.movie_name), (a.last_name || ' ' ||
a.first_name) nume
           FROM screenings s join movies m on (m.movie_id =
s.movie_id)
                           join plays_in pi on (pi.movie_id =
m.movie_id)
                           join actors a on (a.actor_id =
pi.actor_id)
           WHERE hall_id = hall
           ORDER BY m.movie_name;
      BEGIN
      SELECT hall_id into hall
      FROM halls
      WHERE capacity = cap;

      FOR i in c LOOP
           IF (i.movie_name != last_movie) THEN
                DBMS_OUTPUT.PUT_LINE ('---Filmul ' || i.movie_name
|| ' avand actorii: ');
           END IF;
```

```
            DBMS_OUTPUT.PUT_LINE (i.nume);
             last_movie := i.movie_name;
             aux := aux + 1;
      END LOOP;
      IF (aux = 0) THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista filme care sa fi rulat
in sala cu capacitatea ' || cap);
      END IF;
      EXCEPTION
            WHEN TOO_MANY_ROWS THEN
                  DBMS_OUTPUT.PUT_LINE('Exista mai multe sali cu
capacitatea ' || cap);
            WHEN NO_DATA_FOUND THEN
                  DBMS_OUTPUT.PUT_LINE('Nu exista nicio sala care sa
aiba capacitatea ' || cap);
            WHEN OTHERS THEN
                  DBMS_OUTPUT.PUT_LINE('Alta eroare');
      END pack_proc_ex9;


      FUNCTION pack_are_voie_ex11
      (screening tickets.screening_id%TYPE, id_client
clients.client_id%TYPE)
      RETURN BOOLEAN
      IS
      CURSOR crs IS
            SELECT m.age_restriction
            FROM screenings s join movies m on (m.movie_id =
s.movie_id)
            WHERE s.screening_id = screening;
      CURSOR crs2 IS
            SELECT TRUNC(MONTHS_BETWEEN(sysdate, birth_date) / 12)
            FROM clients
            WHERE client_id = id_client;
      age number;
         age_r movies.age_restriction%TYPE;
      BEGIN
      OPEN crs;
            FETCH crs INTO age_r;
      CLOSE crs;
      OPEN crs2;
            FETCH crs2 INTO age;
```

```
      CLOSE crs2;
      IF (age_r <= age) THEN
            RETURN TRUE;
      ELSE
            RETURN FALSE;
      END IF;
      END pack_are_voie_ex11;
END package_ex13;
/

--APELARE:

BEGIN
      DBMS_OUTPUT.PUT_LINE('===6===');
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('Total: ' || package_ex13.pack_fct_ex6()
|| ' actori');
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('Total: ' ||
package_ex13.pack_fct_ex6('Tarantino') || ' actori');
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE('Total: ' ||
package_ex13.pack_fct_ex6('Fincher') || ' actori');
      DBMS_OUTPUT.NEW_LINE;

      DBMS_OUTPUT.PUT_LINE('===7===');
      DBMS_OUTPUT.NEW_LINE;
      package_ex13.pack_proc_ex7();
      package_ex13.pack_proc_ex7('Student');
      package_ex13.pack_proc_ex7('Retired');
      package_ex13.pack_proc_ex7('Dummy');

      DBMS_OUTPUT.PUT_LINE('===8===');
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE(package_ex13.pack_fct_ex8('Drama'));
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE(package_ex13.pack_fct_ex8('Comedy'));
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE(package_ex13.pack_fct_ex8('Action')); --
mai mult de 1 film
      DBMS_OUTPUT.NEW_LINE;
      DBMS_OUTPUT.PUT_LINE(package_ex13.pack_fct_ex8('Documentary')
```

```
); -- niciun film
      DBMS_OUTPUT.NEW_LINE;

      DBMS_OUTPUT.PUT_LINE('===9===');
      DBMS_OUTPUT.NEW_LINE;
      package_ex13.pack_proc_ex9(5); -- mai multe filme
      DBMS_OUTPUT.NEW_LINE;
      package_ex13.pack_proc_ex9(250); -- un singur film
      DBMS_OUTPUT.NEW_LINE;
      package_ex13.pack_proc_ex9(150); -- niciun film
      DBMS_OUTPUT.NEW_LINE;
      package_ex13.pack_proc_ex9(500); -- nu exista capacitatea
      DBMS_OUTPUT.NEW_LINE;
      package_ex13.pack_proc_ex9(100); -- mai multe sali cu
capacitatea respectiva
      DBMS_OUTPUT.NEW_LINE;

      DBMS_OUTPUT.PUT_LINE('===11===');
      DBMS_OUTPUT.NEW_LINE;
      IF (package_ex13.pack_are_voie_ex11(2, 4) = TRUE) THEN --
client cu varsta 12 vrea sa mearga la film cu restrictie de 15
      DBMS_OUTPUT.PUT_LINE('Allowed');
      ELSE
      DBMS_OUTPUT.PUT_LINE('Not allowed');
      END IF;
      IF (package_ex13.pack_are_voie_ex11(2, 1) = TRUE) THEN --
client cu varsta 20 vrea sa mearga la film cu restrictie de 15
      DBMS_OUTPUT.PUT_LINE('Allowed');
      ELSE
      DBMS_OUTPUT.PUT_LINE('Not allowed');
      END IF;
END;
/
```

Script Output x | Query Result x

Task completed in 0.095 seconds

```
Package PACKAGE_EX13 compiled

Package Body PACKAGE_EX13 compiled

PL/SQL procedure successfully completed.
```

```
===6===

---In filmele lui Nolan au jucat:
Actorul Joseph Gordon-Levitt a jucat rolul Arthur in filmul Inception.
Actorul Leonardo DiCaprio a jucat rolul Cobb in filmul Inception.
Actorul Matthew McConaughey a jucat rolul Eames in filmul Inception.
Actorul Matthew McConaughey a jucat rolul Cooper in filmul Interstellar.
Actorul Robert Pattinson a jucat rolul Neil in filmul Tenet.
Actorul Jamie Foxx a jucat rolul Protagonist in filmul Tenet.

Total: 6 actori

---In filmele lui Tarantino au jucat:
Actorul Joseph Gordon-Levitt a jucat rolul Hans Landa in filmul Inglorious Basterds.
Actorul Leonardo DiCaprio a jucat rolul Aldo Raine in filmul Inglorious Basterds.

Total: 2 actori

---In filmele lui Fincher au jucat:
Actorul Matthew McConaughey a jucat rolul Kane in filmul Mank.
Actorul Robert Pattinson a jucat rolul Joe in filmul Mank.
Actorul Gary Oldman a jucat rolul Mank in filmul Mank.

Total: 3 actori

===7===

---Bilete de tipul Adult au fost cumparate de urmatorii clienti:
Clientul Tudorache Theodor cu varsta de 20 de ani.
```

Cinema

```
===7===

---Bilete de tipul Adult au fost cumparate de urmatorii clienti:
Clientul Tudorache Theodor cu varsta de 20 de ani.
Clientul Craciun Andrei cu varsta de 12 ani.

---Bilete de tipul Student au fost cumparate de urmatorii clienti:
Clientul Bugheciu Eduard cu varsta de 20 de ani.
Clientul Tudorache Theodor cu varsta de 20 de ani.
Clientul Constantin Sorin cu varsta de 33 de ani.

---Bilete de tipul Retired au fost cumparate de urmatorii clienti:
Clientul Curtamet Ixan cu varsta de 14 ani.
Clientul Bugheciu Eduard cu varsta de 20 de ani.
Clientul Constantin Sorin cu varsta de 33 de ani.

Nu exista bilete de tipul Dummy.

===8===

Filmul Soul a rulat la 25/12/2020 13:00 cu capacitatea 5/5.

1

Filmul Borat 2 a rulat la 25/12/2020 17:00 cu capacitatea 6/5.

2

Exista mai multe filme care au rulat la capacitate de peste 90% avand genul Action
```

Cinema

```
-2

Nu exista filme care au rulat la capacitate de peste 90% avand genul Documentary

-1

===9===

---Filmul Inglorious Basterds avand actorii:
Joseph Gordon-Levitt
Leonardo DiCaprio
---Filmul Soul avand actorii:
Jamie Foxx

---Filmul Tenet avand actorii:
Jamie Foxx
Robert Pattinson

Nu exista filme care sa fi rulat in sala cu capacitatea 150

Nu exista nicio sala care sa aiba capacitatea 500

Exista mai multe sali cu capacitatea 100

===11===

Not allowed
Allowed
```