PROIECT - Baza de date a cinematografelor din Romania

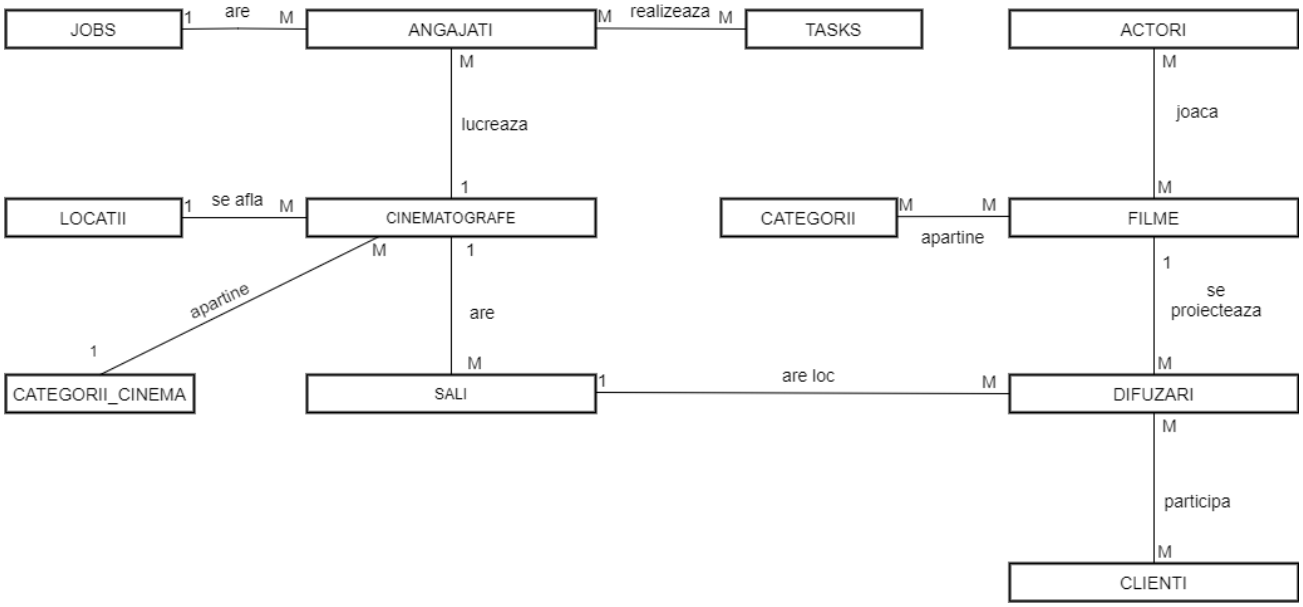
Borza Maria - Cristina

Grupa 244

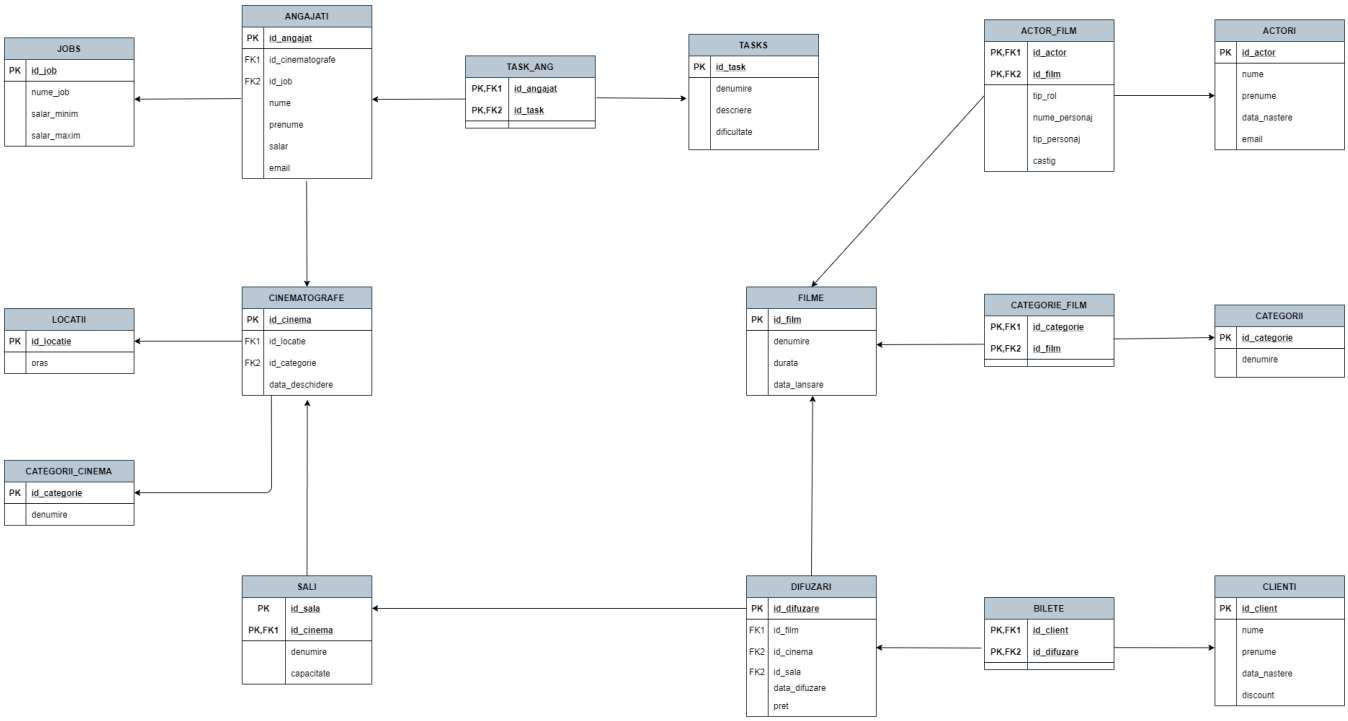
1 Prezentați pe scurt baza de date (utilitatea ei)

Eu am ales sa realizez baza de date a cinematografelor din Romania. Un cinematograf face parte dintr-o singura categorie (aparetine unei firme), dar unei categorii pot apartine mai multe cinematografe. Intr-o locatie se pot afla mai multe cinematografe, iar un cinematograf se poate afla intr-o singura locatie. De asemenea, intr-un cinematograf lucreaza mai multi angajati, fiecare avand un anumit job si mai multe task-uri de facut.
Un cinematograf are mai multe sali. O difuzare are loc intr-o singura sala, si se proiecteaza un sigur film, dar intr-o sala au loc mai multe difuzari. Un client isi poate cumpara bilet la mai multe difuzari, iar la o difuzare pot participa mai multi clienti.
Un film poate apartine mai multor categorii, iar din fiecare categorie pot face parte mai multe filme. De asemnea, intr-un film pot juca mai multi actori, iau un actor poate juca in mai multe filme.
Aceasta baza de date este utila pentru ca ne permite sa aflam si sa modificam rapid informatii despre cinematografele din Romania.

2 Realizați diagrama entitate - relație



3 Pornind de la diagrama entitate - relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare



4 Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare

```
CREATE TABLE locatii(id_locatie NUMBER PRIMARY KEY,
                     oras VARCHAR2(20) NOT NULL);

CREATE TABLE categorii_cinema(id_categorie NUMBER PRIMARY KEY,
                              denumire VARCHAR2(20) NOT NULL);

CREATE TABLE cinematografe(id_cinema NUMBER PRIMARY KEY,
                           id_locatie NUMBER NOT NULL,
                           id_categorie NUMBER NOT NULL,
                           data_deschidere DATE,

            FOREIGN KEY (id_locatie) REFERENCES locatii(id_locatie)
            ON DELETE CASCADE,
            FOREIGN KEY (id_categorie) REFERENCES categorii_cinema(id_categorie)
            ON DELETE CASCADE);

CREATE TABLE jobs(id_job NUMBER PRIMARY KEY,
                  nume_job VARCHAR2(20) NOT NULL,
                  salar_minim NUMBER,
                  salar_maxim NUMBER);

CREATE TABLE angajati(id_angajat NUMBER PRIMARY KEY,
                      id_cinematografe NUMBER NOT NULL,
```

```
                    nume VARCHAR2(20) NOT NULL,
                    prenume VARCHAR2(20) NOT NULL,
                    salar NUMBER,
                    id_job NUMBER NOT NULL,
                    email VARCHAR2(50),

                    FOREIGN KEY (id_cinematografe) REFERENCES cinematografe(id_cinema)
                    ON DELETE CASCADE,
                    FOREIGN KEY (id_job) REFERENCES jobs(id_job)
                    ON DELETE CASCADE);

CREATE TABLE tasks(id_task NUMBER PRIMARY KEY,
                   denumire VARCHAR2(50) NOT NULL,
                   descriere VARCHAR2(256),
                   dificultate NUMBER);


CREATE TABLE task_ang(id_angajat NUMBER,
                    id_task NUMBER,

                    PRIMARY KEY(id_angajat, id_task),
                    FOREIGN KEY (id_angajat) REFERENCES angajati(id_angajat)
                    ON DELETE CASCADE,
                    FOREIGN KEY(id_task) REFERENCES tasks(id_task)
                    ON DELETE CASCADE);


CREATE TABLE sali(id_cinematograf NUMBER,
                  id_sala NUMBER,
                  denumire VARCHAR2(20) NOT NULL,
                  capacitate NUMBER,

                  PRIMARY KEY(id_cinematograf, id_sala),
                  FOREIGN KEY(id_cinematograf) REFERENCES cinematografe(id_cinema)
                  ON DELETE CASCADE);

CREATE TABLE filme(id_film NUMBER PRIMARY KEY,
                   denumire VARCHAR2(50) NOT NULL,
                   durata NUMBER NOT NULL,
                   data_lansare DATE);

CREATE TABLE difuzari(id_difuzare NUMBER PRIMARY KEY,
                    data_difuzare DATE NOT NULL,
                    id_film NUMBER NOT NULL,
                    id_cinema NUMBER NOT NULL,
                    id_sala NUMBER NOT NULL,
                    pret NUMBER NOT NULL,

            FOREIGN KEY (id_cinema, id_sala) REFERENCES sali(id_cinematograf, id_sala)
            ON DELETE CASCADE,
            FOREIGN KEY (id_film) REFERENCES filme(id_film)
            ON DELETE CASCADE);
```

```
CREATE TABLE clienti(id_client NUMBER PRIMARY KEY,
                     nume VARCHAR2(20) NOT NULL,
                     prenume VARCHAR2(20) NOT NULL,
                     discount NUMBER,
                     data_nastere DATE NOT NULL);

CREATE TABLE bilete(id_client NUMBER,
                    id_difuzare NUMBER,

                    PRIMARY KEY (id_client, id_difuzare),
                    FOREIGN KEY (id_client) REFERENCES clienti(id_client)
                    ON DELETE CASCADE,
                    FOREIGN KEY (id_difuzare) REFERENCES difuzari(id_difuzare)
                    ON DELETE CASCADE);

CREATE TABLE categorii(id_categorie NUMBER PRIMARY KEY,
                       denumire VARCHAR2(20) NOT NULL);

CREATE TABLE categorie_film(id_categorie NUMBER,
                            id_film NUMBER,

                            PRIMARY KEY (id_categorie, id_film),
                            FOREIGN KEY (id_categorie) REFERENCES categorii(id_categorie)
                            ON DELETE CASCADE,
                            FOREIGN KEY (id_film) REFERENCES filme(id_film)
                            ON DELETE CASCADE);

CREATE TABLE actori(id_actor NUMBER PRIMARY KEY,
                    nume VARCHAR2(20) NOT NULL,
                    prenume VARCHAR2(20) NOT NULL,
                    email VARCHAR2(50),
                    data_nastere DATE NOT NULL);

CREATE TABLE actor_film(id_film NUMBER,
                        id_actor NUMBER,
                        tip_rol VARCHAR2(10) NOT NULL,
                        nume_personaj VARCHAR2(20),
                        tip_personaj VARCHAR2(10),
                        castig NUMBER,

                        PRIMARY KEY (id_film, id_actor),
                        FOREIGN KEY (id_actor) REFERENCES actori(id_actor)
                        ON DELETE CASCADE,
                        FOREIGN KEY (id_film) REFERENCES filme(id_film)
                        ON DELETE CASCADE);
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```oracle_db_schema
--Creare tabele + constrangeri
CREATE TABLE locatii(id_locatie NUMBER PRIMARY KEY,
                     oras VARCHAR2(20) NOT NULL);
CREATE TABLE categorii_cinema(id_categorie NUMBER PRIMARY KEY,
                              denumire VARCHAR2(20) NOT NULL);
CREATE TABLE cinematografe(id_cinema NUMBER PRIMARY KEY,
                           id_locatie NUMBER NOT NULL,
                           id_categorie NUMBER NOT NULL,
                           data_deschidere DATE,

                           FOREIGN KEY (id_locatie) REFERENCES locatii(id_locatie)
                           ON DELETE CASCADE,
                           FOREIGN KEY (id_categorie) REFERENCES categorii_cinema(id_categorie)
                           ON DELETE CASCADE);
CREATE TABLE jobs(id_job NUMBER PRIMARY KEY,
                  nume_job VARCHAR2(20) NOT NULL,
                  salar_minim NUMBER,
                  salar_maxim NUMBER);

CREATE TABLE angajati(id_angajat NUMBER PRIMARY KEY,
                      id_cinematografe NUMBER NOT NULL,
                      nume VARCHAR2(20) NOT NULL,
                      prenume VARCHAR2(20) NOT NULL,
                      salar NUMBER,
                      id_job NUMBER NOT NULL,
                      email VARCHAR2(50),

                      FOREIGN KEY (id_cinematografe) REFERENCES cinematografe(id_cinema)
                      ON DELETE CASCADE,
                      FOREIGN KEY (id_job) REFERENCES jobs(id_job)
                      ON DELETE CASCADE);
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```oracle_db_schema
CREATE TABLE tasks(id_task NUMBER PRIMARY KEY,
                   denumire VARCHAR2(50) NOT NULL,
                   descriere VARCHAR2(256),
                   dificultate NUMBER);


CREATE TABLE task_ang(id_angajat NUMBER,
                      id_task NUMBER,

                      PRIMARY KEY(id_angajat, id_task),
                      FOREIGN KEY (id_angajat) REFERENCES angajati(id_angajat)
                      ON DELETE CASCADE,
                      FOREIGN KEY(id_task) REFERENCES tasks(id_task)
                      ON DELETE CASCADE);


CREATE TABLE sali(id_cinematograf NUMBER,
                  id_sala NUMBER,
                  denumire VARCHAR2(20) NOT NULL,
                  capacitate NUMBER,

                  PRIMARY KEY(id_cinematograf, id_sala),
                  FOREIGN KEY(id_cinematograf) REFERENCES cinematografe(id_cinema)
                  ON DELETE CASCADE);

CREATE TABLE filme(id_film NUMBER PRIMARY KEY,
                   denumire VARCHAR2(50) NOT NULL,
                   durata NUMBER NOT NULL,
                   data_lansare DATE);
```

db_proiect | Relational_1 (Untitled_1)

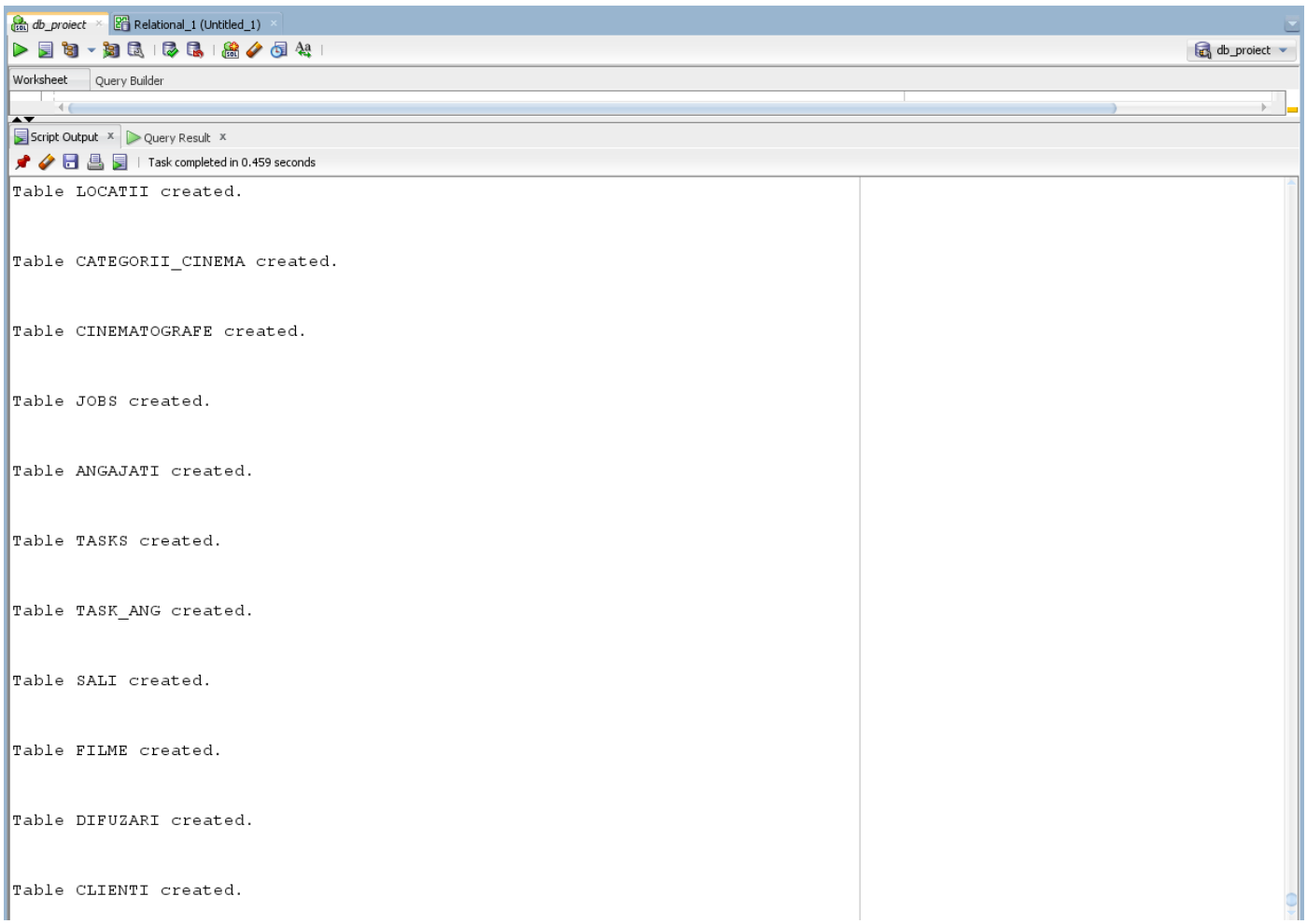
Worksheet | Query Builder

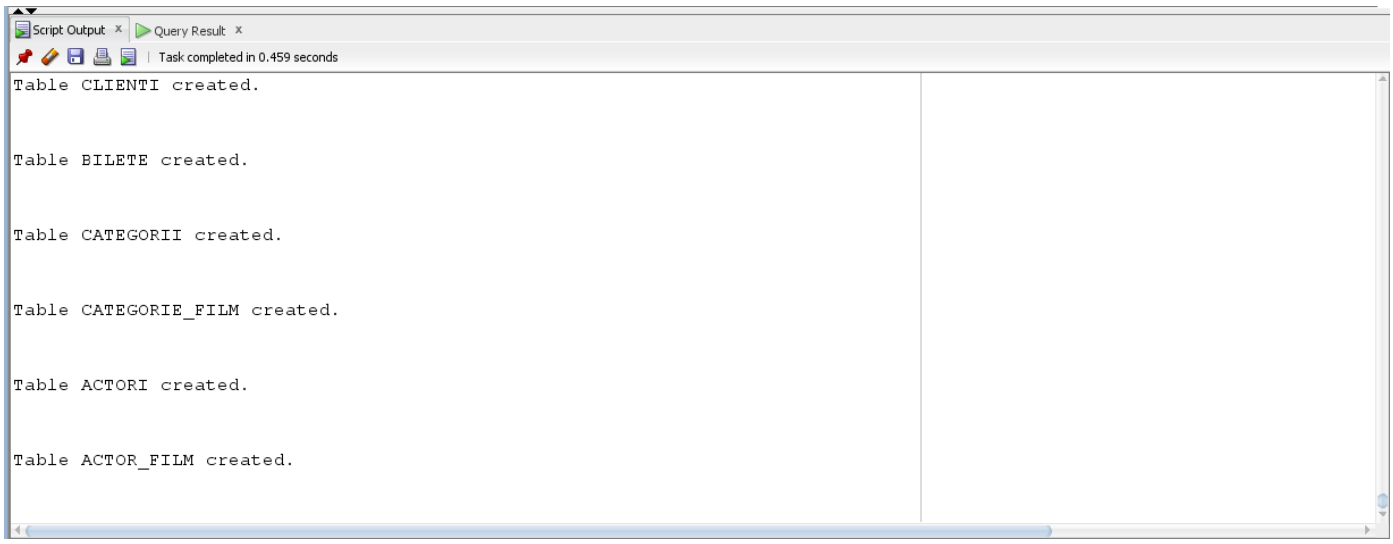
```
CREATE TABLE difuzari(id_difuzare NUMBER PRIMARY KEY,
                      data_difuzare DATE NOT NULL,
                      id_film NUMBER NOT NULL,
                      id_cinema NUMBER NOT NULL,
                      id_sala NUMBER NOT NULL,
                      pret NUMBER NOT NULL,

                      FOREIGN KEY (id_cinema, id_sala) REFERENCES sali(id_cinematograf, id_sala)
                      ON DELETE CASCADE,
                      FOREIGN KEY (id_film) REFERENCES filme(id_film)
                      ON DELETE CASCADE);

CREATE TABLE clienti(id_client NUMBER PRIMARY KEY,
                     nume VARCHAR2(20) NOT NULL,
                     prenume VARCHAR2(20) NOT NULL,
                     discount NUMBER,
                     data_nastere DATE NOT NULL);

CREATE TABLE bilete(id_client NUMBER,
                    id_difuzare NUMBER,

                    PRIMARY KEY (id_client, id_difuzare),
                    FOREIGN KEY (id_client) REFERENCES clienti(id_client)
                    ON DELETE CASCADE,
                    FOREIGN KEY (id_difuzare) REFERENCES difuzari(id_difuzare)
                    ON DELETE CASCADE);

CREATE TABLE categorii(id_categorie NUMBER PRIMARY KEY,
                       denumire VARCHAR2(20) NOT NULL);
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```
CREATE TABLE categorie_film(id_categorie NUMBER,
                            id_film NUMBER,

                            PRIMARY KEY (id_categorie, id_film),
                            FOREIGN KEY (id_categorie) REFERENCES categorii(id_categorie)
                            ON DELETE CASCADE,
                            FOREIGN KEY (id_film) REFERENCES filme(id_film)
                            ON DELETE CASCADE);

CREATE TABLE actori(id_actor NUMBER PRIMARY KEY,
                    nume VARCHAR2(20) NOT NULL,
                    prenume VARCHAR2(20) NOT NULL,
                    email VARCHAR2(50),
                    data_nastere DATE NOT NULL);

CREATE TABLE actor_film(id_film NUMBER,
                        id_actor NUMBER,
                        tip_rol VARCHAR2(10) NOT NULL,
                        nume_personaj VARCHAR2(20),
                        tip_personaj VARCHAR2(10),
                        castig NUMBER,

                        PRIMARY KEY (id_film, id_actor),
                        FOREIGN KEY (id_actor) REFERENCES actori(id_actor)
                        ON DELETE CASCADE,
                        FOREIGN KEY (id_film) REFERENCES filme(id_film)
                        ON DELETE CASCADE);
```





5 Adăugați informații coerente în tabelele create

```
INSERT INTO locatii VALUES (1, 'Bucuresti');
INSERT INTO locatii VALUES (2, 'Iasi');
INSERT INTO locatii VALUES (3, 'Cluj');
INSERT INTO locatii VALUES (4, 'Timisoara');
```

```
INSERT INTO locatii VALUES (5, 'Brasov');
INSERT INTO locatii VALUES (6, 'Constata');
INSERT INTO locatii VALUES (7, 'Ploiest');
INSERT INTO locatii VALUES (8, 'Sibiu');
INSERT INTO locatii VALUES (9, 'Oradea');
INSERT INTO locatii VALUES (10, 'Pitesti');
INSERT INTO categorii_cinema VALUES (1, 'Cinema City');
INSERT INTO categorii_cinema VALUES (2, 'Cinema Victoria');
INSERT INTO categorii_cinema VALUES (3, 'Movieplex');
INSERT INTO categorii_cinema VALUES (4, 'Cinema Europa');
INSERT INTO categorii_cinema VALUES (5, 'Cinema Dacia');
INSERT INTO categorii_cinema VALUES (6, 'Cinema One Laserplex');
INSERT INTO cinematografe VALUES (1, 1, 1, '10-OCT-2008');
INSERT INTO cinematografe VALUES (2, 1, 2, '25-DEC-2015');
INSERT INTO cinematografe VALUES (3, 1, 3, sysdate);
INSERT INTO cinematografe VALUES (4, 2, 2, '04-SEP-2013');
INSERT INTO cinematografe VALUES (5, 2, 6, '07-JUN-2005');
INSERT INTO cinematografe VALUES (6, 2, 3, '10-JAN-2011');
INSERT INTO cinematografe VALUES (7, 1, 6, '10-FEB-2009');
INSERT INTO cinematografe VALUES (8, 1, 5, '13-MAR-2003');
INSERT INTO cinematografe VALUES (9, 3, 1, '28-JUL-2009');
INSERT INTO cinematografe VALUES (10, 3, 2, '19-NOV-2010');
INSERT INTO cinematografe VALUES (11, 3, 4, '07-JUN-2020');
INSERT INTO cinematografe VALUES (12, 3, 6, '10-MAY-2011');
INSERT INTO cinematografe VALUES (13, 4, 1, '01-APR-2012');
INSERT INTO cinematografe VALUES (14, 4, 4, '12-DEC-2012');
INSERT INTO cinematografe VALUES (15, 4, 5, '04-FEB-2007');
INSERT INTO cinematografe VALUES (16, 5, 2, '19-MAY-2016');
INSERT INTO cinematografe VALUES (17, 5, 6, '07-JUN-2005');
INSERT INTO cinematografe VALUES (18, 5, 5, '10-MAY-2011');
INSERT INTO cinematografe VALUES (19, 5, 1, '10-APR-2005');
INSERT INTO cinematografe VALUES (20, 6, 1, '28-FEB-2014');
INSERT INTO cinematografe VALUES (21, 7, 3, sysdate);
INSERT INTO cinematografe VALUES (22, 8, 1, '04-SEP-2013');
INSERT INTO cinematografe VALUES (23, 9, 4, '07-JUN-2005');
INSERT INTO cinematografe VALUES (24, 9, 1, '10-JAN-2011');
INSERT INTO cinematografe VALUES (25, 10, 1, '10-AUG-2019');
INSERT INTO cinematografe VALUES (26, 10, 3, '10-AUG-2019');

INSERT INTO jobs VALUES (1, 'Casier', 2000, 3000);
INSERT INTO jobs VALUES (2, 'Paznic', 2100, 3200);
INSERT INTO jobs VALUES (3, 'Manager', 3000, 5700);
INSERT INTO jobs VALUES (4, 'Femeie de servici', 1900, 2700);
INSERT INTO jobs VALUES (5, 'Inginer', NULL, NULL);

INSERT INTO angajati VALUES (1, 1, 'Popescu', 'Ion', 2500, 1, 'popescu.ion@gmail.com');
INSERT INTO angajati VALUES (2, 1, 'Georgescu', 'Ana', 3500, 3, NULL);
INSERT INTO angajati VALUES (3, 1, 'Popescu', 'Vasile', 3000, 5, NULL);
INSERT INTO angajati VALUES (4, 2, 'Ionel', 'Ion', 2400, 2, 'ion@yahoo.ro');
INSERT INTO angajati VALUES (5, 2, 'Vasilica', 'Gigel', 5500, 3, 'gigel@gmail.com');
INSERT INTO angajati VALUES (6, 3, 'Ionescu', 'Maria', 5700, 5, NULL);
INSERT INTO angajati VALUES (7, 4, 'Steven', 'King', 2500, 5, NULL);
INSERT INTO angajati VALUES (8, 5, 'Ionescu', 'Ana', 2300, 4, 'ionescu@yahoo.com');
```

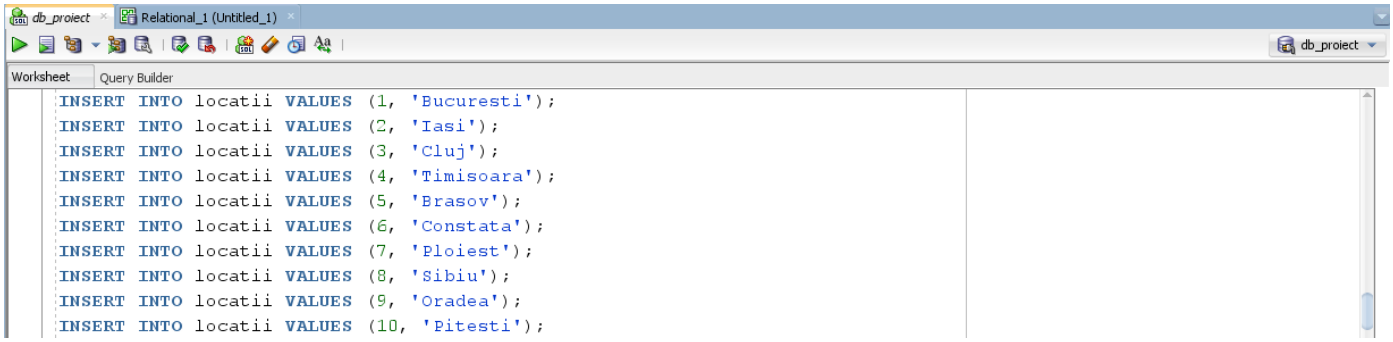
```
INSERT INTO angajati VALUES (9, 6, 'Popescu', 'Vasile', 3000, 3, 'vasile@mail.com');
INSERT INTO angajati VALUES (10, 7, 'Ionel', 'Ion', 3200, 3, NULL);
INSERT INTO angajati VALUES (11, 7, 'Petrescu', 'Alexandru', 4000, 5, NULL);
INSERT INTO angajati VALUES (12, 8, 'Vasilica', 'Gigel', 5500, 3, 'gigel@mail.ro');
INSERT INTO angajati VALUES (13, 9, 'Ionescu', 'Andreea', 4000, 3, NULL);
INSERT INTO angajati VALUES (14, 9, 'Popescu', 'Ana', 2300, 4, NULL);
INSERT INTO angajati VALUES (15, 10, 'Popescu', 'Gigel', 4400, 3, NULL);
INSERT INTO angajati VALUES (16, 11, 'Lorentz', 'Diana', 4700, 3, 'Diana@gmail.com');
INSERT INTO angajati VALUES (17, 12, 'Vasilescu', 'Paul', 4000, 3, NULL);
INSERT INTO angajati VALUES (18, 12, 'Pop', 'Ionut', 2600, 5, NULL);
INSERT INTO angajati VALUES (19, 13, 'Andrei', 'Alexandra', 4500, 3, 'alex@yahoo.com');
INSERT INTO angajati VALUES (20, 14, 'Ionel', 'Gigel', 5700, 3, 'ionel.gigel@gmail.com');
INSERT INTO angajati VALUES (21, 15, 'Popa', 'Dorian', 4800, 3, NULL);
INSERT INTO angajati VALUES (22, 15, 'Maria', 'Andreea', 1900, 4, NULL);
INSERT INTO angajati VALUES (23, 16, 'Popescu', 'Ion', 4900, 3, NULL);
INSERT INTO angajati VALUES (24, 17, 'Georgescu', 'Ana',4500, 3, NULL);
INSERT INTO angajati VALUES (25, 18, 'Petrescu', 'Marius', 2100, 1, 'marius@yahoo.com');
INSERT INTO angajati VALUES (26, 18, 'Vasilescu', 'Vasile', 4500, 3, 'vasile@mail.ro');
INSERT INTO angajati VALUES (27, 19, 'Marinescu', 'Ion', 5000, 3, NULL);
INSERT INTO angajati VALUES (28, 20, 'Vasilica', 'Gigel', 3500, 3, NULL);
INSERT INTO angajati VALUES (29, 21, 'Ionescu', 'Maria', 3000, 5, NULL);
INSERT INTO angajati VALUES (30, 22, 'Pop', 'Vasile', 3500, 3, 'pop@yahoo.com');
INSERT INTO angajati VALUES (31, 23, 'Georgescu', 'Serban', 3000, 3, 'serban@gmail.com');
INSERT INTO angajati VALUES (32, 23, 'Vasile', 'Ana', 4200, 3, NULL);
INSERT INTO angajati VALUES (33, 24, 'Popescu', 'Vasile', 3100, 3, NULL);
INSERT INTO angajati VALUES (34, 25, 'Ionel', 'Ion', 4500, 3, NULL);
INSERT INTO angajati VALUES (35, 26, 'Vasilica', 'Gigel', 5500, 3, NULL);
INSERT INTO angajati VALUES (36, 26, 'Ionescu', 'Maria', 2600, 4, NULL);

INSERT INTO tasks VALUES(1, 'Curata sala', NULL, 1);
INSERT INTO tasks VALUES(2, 'Vinde bilete', NULL, 2);
INSERT INTO tasks VALUES(3, 'Verifica bilete la intrare', NULL, 1);
INSERT INTO tasks VALUES(4, 'Proiecteza filmul', NULL, 3);
INSERT INTO tasks VALUES(5, 'Actualizeaza site-ul', NULL, 2);
INSERT INTO tasks VALUES(6, 'Imparte task-uri', NULL, 4);
INSERT INTO tasks VALUES(7, 'Intalniri cu colaboratorii', NULL, 5);

INSERT INTO task_ang VALUES(1, 2);
INSERT INTO task_ang VALUES(2, 6);
INSERT INTO task_ang VALUES(3, 4);
INSERT INTO task_ang VALUES(3, 5);
INSERT INTO task_ang VALUES(4, 3);
INSERT INTO task_ang VALUES(4, 1);
INSERT INTO task_ang VALUES(5, 6);
INSERT INTO task_ang VALUES(6, 5);
INSERT INTO task_ang VALUES(7, 4);
INSERT INTO task_ang VALUES(7, 5);
INSERT INTO task_ang VALUES(8, 1);
INSERT INTO task_ang VALUES(9, 7);
INSERT INTO task_ang VALUES(9, 6);
INSERT INTO task_ang VALUES(10, 7);
INSERT INTO task_ang VALUES(11, 4);
INSERT INTO task_ang VALUES(12, 6);
```

```
INSERT INTO task_ang VALUES(12, 7);
INSERT INTO task_ang VALUES(13, 7);
INSERT INTO task_ang VALUES(14, 1);
INSERT INTO task_ang VALUES(15, 6);
INSERT INTO task_ang VALUES(16, 6);
INSERT INTO task_ang VALUES(17, 7);
INSERT INTO task_ang VALUES(18, 5);
INSERT INTO task_ang VALUES(18, 4);
INSERT INTO task_ang VALUES(19, 7);
INSERT INTO task_ang VALUES(20, 6);
INSERT INTO task_ang VALUES(21, 7);
INSERT INTO task_ang VALUES(22, 1);
INSERT INTO task_ang VALUES(23, 6);
INSERT INTO task_ang VALUES(24, 6);
INSERT INTO task_ang VALUES(25, 2);
INSERT INTO task_ang VALUES(26, 6);
INSERT INTO task_ang VALUES(26, 7);
INSERT INTO task_ang VALUES(27, 7);
INSERT INTO task_ang VALUES(28, 6);
INSERT INTO task_ang VALUES(29, 4);
INSERT INTO task_ang VALUES(29, 5);
INSERT INTO task_ang VALUES(30, 7);
INSERT INTO task_ang VALUES(31, 7);
INSERT INTO task_ang VALUES(32, 6);
INSERT INTO task_ang VALUES(33, 6);
INSERT INTO task_ang VALUES(34, 7);
INSERT INTO task_ang VALUES(35, 6);
INSERT INTO task_ang VALUES(36, 5);

INSERT INTO sali VALUES(1, 1, 'Sala 1', 120);
INSERT INTO sali VALUES(1, 2, 'Sala 2', 100);
INSERT INTO sali VALUES(1, 3, 'Sala 3', 210);
INSERT INTO sali VALUES(2, 1, 'Sala 1', 130);
INSERT INTO sali VALUES(2, 2, 'Sala 2', 110);
INSERT INTO sali VALUES(3, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(3, 2, 'Sala 2', 140);
INSERT INTO sali VALUES(3, 3, 'Sala 3', 105);
INSERT INTO sali VALUES(4, 1, 'Sala 1', 125);
INSERT INTO sali VALUES(5, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(6, 1, 'Sala 1', 210);
INSERT INTO sali VALUES(7, 1, 'Sala 1', 145);
INSERT INTO sali VALUES(7, 2, 'Sala 2', 90);
INSERT INTO sali VALUES(8, 1, 'Sala 1', 125);
INSERT INTO sali VALUES(9, 1, 'Sala 1', 130);
INSERT INTO sali VALUES(10, 1, 'Sala 1', 50);
INSERT INTO sali VALUES(11, 2, 'Sala 2', 100);
INSERT INTO sali VALUES(11, 3, 'Sala 3', 210);
INSERT INTO sali VALUES(12, 1, 'Sala 1', 130);
INSERT INTO sali VALUES(12, 2, 'Sala 2', 110);
INSERT INTO sali VALUES(13, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(13, 2, 'Sala 2', 140);
INSERT INTO sali VALUES(13, 3, 'Sala 3', 105);
INSERT INTO sali VALUES(14, 1, 'Sala 1', 125);
```

```
INSERT INTO sali VALUES(15, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(16, 1, 'Sala 1', 210);
INSERT INTO sali VALUES(17, 1, 'Sala 1', 145);
INSERT INTO sali VALUES(17, 2, 'Sala 2', 90);
INSERT INTO sali VALUES(18, 1, 'Sala 1', 125);
INSERT INTO sali VALUES(19, 1, 'Sala 1', 130);
INSERT INTO sali VALUES(20, 1, 'Sala 1', 50);
INSERT INTO sali VALUES(21, 2, 'Sala 2', 100);
INSERT INTO sali VALUES(21, 3, 'Sala 3', 210);
INSERT INTO sali VALUES(22, 1, 'Sala 1', 130);
INSERT INTO sali VALUES(22, 2, 'Sala 2', 110);
INSERT INTO sali VALUES(23, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(23, 2, 'Sala 2', 140);
INSERT INTO sali VALUES(23, 3, 'Sala 3', 105);
INSERT INTO sali VALUES(24, 1, 'Sala 1', 125);
INSERT INTO sali VALUES(25, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(26, 1, 'Sala 1', 210);

INSERT INTO filme VALUES(1, 'Pirates of the Caribbean', 140, '9-JUL-2016');
INSERT INTO filme VALUES(2, 'Murder on the Orient Express', 120, '10-NOV-2017');
INSERT INTO filme VALUES(3, 'The Lord of the Rings', 200, '17-DEC-2003');
INSERT INTO filme VALUES(4, 'The Godfather', 175, '24-MAR-1972');
INSERT INTO filme VALUES(5, 'Schindler List', 200, '4-FEB-1993');
INSERT INTO filme VALUES(6, 'Inception', 148, '16-JUL-2010');
INSERT INTO filme VALUES(7, 'The Wolf of Wall Street', 180, '25-DEC-2013');

INSERT INTO difuzari VALUES(1, sysdate, 1, 1, 3, 15);
INSERT INTO difuzari VALUES(2, '01-DEC-2020', 2, 1, 2, 20);
INSERT INTO difuzari VALUES(3, '20-NOV-2020', 3, 1, 1, 18);
INSERT INTO difuzari VALUES(4, sysdate, 1, 2, 1, 20);
INSERT INTO difuzari VALUES(5, '12-MAY-2019', 7, 3, 1, 16);
INSERT INTO difuzari VALUES(6, '13-FEB-2020', 5, 4, 1, 12);
INSERT INTO difuzari VALUES(7, sysdate, 5, 5, 1, 15);
INSERT INTO difuzari VALUES(8, '22-OCT-2020', 3, 6, 1, 10);
INSERT INTO difuzari VALUES(9, '02-JAN-2019', 7, 7, 2, 15);
INSERT INTO difuzari VALUES(10, sysdate, 3, 8, 1, 20);
INSERT INTO difuzari VALUES(11, '13-APR-2018', 4, 9, 1, 12);
INSERT INTO difuzari VALUES(12, '20-MAY-2019', 6, 10, 1, 22);
INSERT INTO difuzari VALUES(13, sysdate, 6, 11, 2, 10);
INSERT INTO difuzari VALUES(14, sysdate, 5, 12, 1, 17);
INSERT INTO difuzari VALUES(15, '08-JUN-2020', 6, 12, 1, 24);
INSERT INTO difuzari VALUES(16, '10-JUL-2020', 1, 12, 2, 20);
INSERT INTO difuzari VALUES(17, '01-MAY-2019', 4, 13, 1, 13);
INSERT INTO difuzari VALUES(18, '20-FEB-2020', 6, 14, 1, 18);
INSERT INTO difuzari VALUES(19, '19-AUG-2018', 5, 15, 1, 20);
INSERT INTO difuzari VALUES(20, '09-MAR-2020', 3, 16, 1, 17);
INSERT INTO difuzari VALUES(21, '03-SEP-2019', 4, 17, 1, 22);
INSERT INTO difuzari VALUES(22, sysdate, 2, 17, 2, 16);
INSERT INTO difuzari VALUES(23, sysdate, 4, 18, 1, 24);
INSERT INTO difuzari VALUES(24, sysdate, 2, 19, 1, 20);
INSERT INTO difuzari VALUES(25, '13-APR-2018', 1, 20, 1, 12);
INSERT INTO difuzari VALUES(26, '20-MAY-2019', 6, 21, 3, 22);
INSERT INTO difuzari VALUES(27, sysdate, 2, 22, 2, 10);
```

```
INSERT INTO difuzari VALUES(28, sysdate, 5, 23, 1, 17);
INSERT INTO difuzari VALUES(29, '08-JUN-2020', 1, 24, 1, 24);
INSERT INTO difuzari VALUES(30, '08-JUN-2020', 5, 25, 1, 24);
INSERT INTO difuzari VALUES(31, sysdate, 3, 26, 1, 24);

INSERT INTO clienti VALUES(1, 'Maria', 'Cristina', 0.25, '10-OCT-2000');
INSERT INTO clienti VALUES(2, 'Gigel', 'Gigel', NULL, '09-MAR-1993');
INSERT INTO clienti VALUES(3, 'Ionescu', 'Andrei', 0.5, '04-FEB-1950');
INSERT INTO clienti VALUES(4, 'Georgescu', 'Maria', NULL, '10-JUL-1978');
INSERT INTO clienti VALUES(5, 'Vasilescu', 'Ion', 1, '03-AUG-2015');

INSERT INTO bilete VALUES(1, 1);
INSERT INTO bilete VALUES(1, 5);
INSERT INTO bilete VALUES(1, 7);
INSERT INTO bilete VALUES(1, 21);
INSERT INTO bilete VALUES(1, 30);
INSERT INTO bilete VALUES(2, 4);
INSERT INTO bilete VALUES(2, 10);
INSERT INTO bilete VALUES(2, 17);
INSERT INTO bilete VALUES(2, 23);
INSERT INTO bilete VALUES(3, 1);
INSERT INTO bilete VALUES(3, 2);
INSERT INTO bilete VALUES(4, 8);
INSERT INTO bilete VALUES(4, 15);
INSERT INTO bilete VALUES(4, 29);
INSERT INTO bilete VALUES(4, 31);
INSERT INTO bilete VALUES(4, 7);
INSERT INTO bilete VALUES(4, 13);
INSERT INTO bilete VALUES(5, 1);
INSERT INTO bilete VALUES(5, 5);
INSERT INTO bilete VALUES(5, 6);
INSERT INTO bilete VALUES(5, 11);
INSERT INTO bilete VALUES(5, 14);
INSERT INTO bilete VALUES(5, 28);
INSERT INTO bilete VALUES(5, 29);
INSERT INTO bilete VALUES(5, 30);

INSERT INTO categorii VALUES(1, 'Comedie');
INSERT INTO categorii VALUES(2, 'Horror');
INSERT INTO categorii VALUES(3, 'Romanic');
INSERT INTO categorii VALUES(4, 'Actiune');
INSERT INTO categorii VALUES(5, 'Drama');

INSERT INTO categorie_film VALUES(2, 1);
INSERT INTO categorie_film VALUES(3, 1);
INSERT INTO categorie_film VALUES(5, 1);
INSERT INTO categorie_film VALUES(1, 2);
INSERT INTO categorie_film VALUES(2, 2);
INSERT INTO categorie_film VALUES(4, 2);
INSERT INTO categorie_film VALUES(1, 3);
INSERT INTO categorie_film VALUES(3, 3);
INSERT INTO categorie_film VALUES(4, 3);
INSERT INTO categorie_film VALUES(5, 3);
```

```
INSERT INTO categorie_film VALUES(1, 4);
INSERT INTO categorie_film VALUES(2, 4);
INSERT INTO categorie_film VALUES(3, 4);
INSERT INTO categorie_film VALUES(3, 5);
INSERT INTO categorie_film VALUES(4, 5);
INSERT INTO categorie_film VALUES(5, 5);
INSERT INTO categorie_film VALUES(1, 6);
INSERT INTO categorie_film VALUES(3, 6);
INSERT INTO categorie_film VALUES(1, 7);
INSERT INTO categorie_film VALUES(2, 7);
INSERT INTO categorie_film VALUES(3, 7);
INSERT INTO categorie_film VALUES(4, 7);

INSERT INTO actori VALUES(1, 'Depp', 'Johnny', NULL, '09-JUN-1963');
INSERT INTO actori VALUES(2, 'Cruz', 'Penelope', 'Penelope@yahoo.com', '28-APR-1974');
INSERT INTO actori VALUES(3, 'Claflin', 'Sam', 'sam@gmail.com', '27-JUN-1986');
INSERT INTO actori VALUES(4, 'Brando', 'Marlon', NULL, '03-APR-1924');
INSERT INTO actori VALUES(5, 'Pacino', 'Al', NULL, '25-APR-1940');
INSERT INTO actori VALUES(6, 'DiCaprio', 'Leonardo', NULL, '11-NOV-1974');
INSERT INTO actori VALUES(7, 'Robbie', 'Margot', 'rm@gmail.com', '02-JUL-1990');
INSERT INTO actori VALUES(8, 'Bloom', 'Orlando', 'orlando@gmail.com', '13-JAN-1977');
INSERT INTO actori VALUES(9, 'Howard', 'Alan', 'alan@mail.com', '05-AUG-1937');
INSERT INTO actori VALUES(10, 'Neeson', 'Liam', NULL, '07-JUN-1952');
INSERT INTO actori VALUES(11, 'Goodall', 'Caroline', NULL, '13-NOV-1959');

INSERT INTO actor_film VALUES(1, 1, 'Principal', 'Jack Sparrow', 'Pozitiv', 5000000);
INSERT INTO actor_film VALUES(1, 2, 'Principal', 'Angelica', 'Negativ', 1000000);
INSERT INTO actor_film VALUES(1, 3, 'Secundar', 'Philip', 'Pozitiv', 200000);
INSERT INTO actor_film VALUES(1, 7, 'Episodic', 'Gillette', 'Negativ', 20000);
INSERT INTO actor_film VALUES(2, 6, 'Principal', 'Young Policeman', 'Pozitiv', 70000);
INSERT INTO actor_film VALUES(2, 3, 'Secundar', 'Hercule Poirot', 'Negativ', 10000);
INSERT INTO actor_film VALUES(2, 2, 'Figuratie', NULL, NULL, 10000);
INSERT INTO actor_film VALUES(3, 9, 'Principal', 'Voice of the Ring', 'Pozitiv', 900000);
INSERT INTO actor_film VALUES(3, 8, 'Secundar', 'Legolas', NULL, 300000);
INSERT INTO actor_film VALUES(4, 4, 'Principal', 'Don Vito Corleone', 'Negativ', 7000000);
INSERT INTO actor_film VALUES(4, 5, 'Principal', 'Michael Corleone', 'Negativ', 6000000);
INSERT INTO actor_film VALUES(5, 10, 'Principal', 'Oskar Schindler', 'Pozitiv', 3000000);
INSERT INTO actor_film VALUES(5, 11, 'Principal', 'Emilie Schindler', NULL, 1000000);
INSERT INTO actor_film VALUES(6, 6, 'Principal', 'Cobb', 'Pozitiv', 400000);
INSERT INTO actor_film VALUES(6, 8, 'Figuratie', NULL, NULL, 5000);
INSERT INTO actor_film VALUES(7, 6, 'Principal', 'Jordan Belfort', 'Negativ', 4000000);
INSERT INTO actor_film VALUES(7, 7, 'Secundar', 'Naomi Lapaglia', NULL, 800000);
```



db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```
INSERT INTO categorii_cinema VALUES (1, 'Cinema City');
INSERT INTO categorii_cinema VALUES (2, 'Cinema Victoria');
INSERT INTO categorii_cinema VALUES (3, 'Movieplex');
INSERT INTO categorii_cinema VALUES (4, 'Cinema Europa');
INSERT INTO categorii_cinema VALUES (5, 'Cinema Dacia');
INSERT INTO categorii_cinema VALUES (6, 'Cinema One Laserplex');
INSERT INTO cinematografe VALUES (1, 1, 1, '10-OCT-2008');
INSERT INTO cinematografe VALUES (2, 1, 2, '25-DEC-2015');
INSERT INTO cinematografe VALUES (3, 1, 3, sysdate);
INSERT INTO cinematografe VALUES (4, 2, 2, '04-SEP-2013');
INSERT INTO cinematografe VALUES (5, 2, 6, '07-JUN-2005');
INSERT INTO cinematografe VALUES (6, 2, 3, '10-JAN-2011');
INSERT INTO cinematografe VALUES (7, 1, 6, '10-FEB-2009');
INSERT INTO cinematografe VALUES (8, 1, 5, '13-MAR-2003');
INSERT INTO cinematografe VALUES (9, 3, 1, '28-JUL-2009');
INSERT INTO cinematografe VALUES (10, 3, 2, '19-NOV-2010');
INSERT INTO cinematografe VALUES (11, 3, 4, '07-JUN-2020');
INSERT INTO cinematografe VALUES (12, 3, 6, '10-MAY-2011');
INSERT INTO cinematografe VALUES (13, 4, 1, '01-APR-2012');
INSERT INTO cinematografe VALUES (14, 4, 4, '12-DEC-2012');
INSERT INTO cinematografe VALUES (15, 4, 5, '04-FEB-2007');
INSERT INTO cinematografe VALUES (16, 5, 2, '19-MAY-2016');
INSERT INTO cinematografe VALUES (17, 5, 6, '07-JUN-2005');
INSERT INTO cinematografe VALUES (18, 5, 5, '10-MAY-2011');
INSERT INTO cinematografe VALUES (19, 5, 1, '10-APR-2005');
INSERT INTO cinematografe VALUES (20, 6, 1, '28-FEB-2014');
INSERT INTO cinematografe VALUES (21, 7, 3, sysdate);
INSERT INTO cinematografe VALUES (22, 8, 1, '04-SEP-2013');
INSERT INTO cinematografe VALUES (23, 9, 4, '07-JUN-2005');
INSERT INTO cinematografe VALUES (24, 9, 1, '10-JAN-2011');
INSERT INTO cinematografe VALUES (25, 10, 1, '10-AUG-2019');
INSERT INTO cinematografe VALUES (26, 10, 3, '10-AUG-2019');
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```
INSERT INTO jobs VALUES (1, 'Casier', 2000, 3000);
INSERT INTO jobs VALUES (2, 'Paznic', 2100, 3200);
INSERT INTO jobs VALUES (3, 'Manager', 3000, 5700);
INSERT INTO jobs VALUES (4, 'Femeie de servici', 1900, 2700);
INSERT INTO jobs VALUES (5, 'Inginer', NULL, NULL);

INSERT INTO angajati VALUES (1, 1, 'Popescu', 'Ion', 2500, 1, 'popescu.ion@gmail.com');
INSERT INTO angajati VALUES (2, 1, 'Georgescu', 'Ana', 3500, 3, NULL);
INSERT INTO angajati VALUES (3, 1, 'Popescu', 'Vasile', 3000, 5, NULL);
INSERT INTO angajati VALUES (4, 2, 'Ionel', 'Ion', 2400, 2, 'ion@yahoo.ro');
INSERT INTO angajati VALUES (5, 2, 'Vasilica', 'Gigel', 5500, 3, 'gigel@gmail.com');
INSERT INTO angajati VALUES (6, 3, 'Ionescu', 'Maria', 5700, 5, NULL);
INSERT INTO angajati VALUES (7, 4, 'Steven', 'King', 2500, 5, NULL);
INSERT INTO angajati VALUES (8, 5, 'Ionescu', 'Ana', 2300, 4, 'ionescu@yahoo.com');
INSERT INTO angajati VALUES (9, 6, 'Popescu', 'Vasile', 3000, 3, 'vasile@mail.com');
INSERT INTO angajati VALUES (10, 7, 'Ionel', 'Ion', 3200, 3, NULL);
INSERT INTO angajati VALUES (11, 7, 'Petrescu', 'Alexandru', 4000, 5, NULL);
INSERT INTO angajati VALUES (12, 8, 'Vasilica', 'Gigel', 5500, 3, 'gigel@mail.ro');
INSERT INTO angajati VALUES (13, 9, 'Ionescu', 'Andreea', 4000, 3, NULL);
INSERT INTO angajati VALUES (14, 9, 'Popescu', 'Ana', 2300, 4, NULL);
INSERT INTO angajati VALUES (15, 10, 'Popescu', 'Gigel', 4400, 3, NULL);
INSERT INTO angajati VALUES (16, 11, 'Lorentz', 'Diana', 4700, 3, 'Diana@gmail.com');
INSERT INTO angajati VALUES (17, 12, 'Vasilescu', 'Paul', 4000, 3, NULL);
INSERT INTO angajati VALUES (18, 12, 'Pop', 'Ionut', 2600, 5, NULL);
INSERT INTO angajati VALUES (19, 13, 'Andrei', 'Alexandra', 4500, 3, 'alex@yahoo.com');
INSERT INTO angajati VALUES (20, 14, 'Ionel', 'Gigel', 5700, 3, 'ionel.gigel@gmail.com');
INSERT INTO angajati VALUES (21, 15, 'Popa', 'Dorian', 4800, 3, NULL);
INSERT INTO angajati VALUES (22, 15, 'Maria', 'Andreea', 1900, 4, NULL);
INSERT INTO angajati VALUES (23, 16, 'Popescu', 'Ion', 4900, 3, NULL);
INSERT INTO angajati VALUES (24, 17, 'Georgescu', 'Ana',4500, 3, NULL);
INSERT INTO angajati VALUES (25, 18, 'Petrescu', 'Marius', 2100, 1, 'marius@yahoo.com');
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```
INSERT INTO angajati VALUES (26, 18, 'Vasilescu', 'Vasile', 4500, 3, 'vasile@mail.ro');
INSERT INTO angajati VALUES (27, 19, 'Marinescu', 'Ion', 5000, 3, NULL);
INSERT INTO angajati VALUES (28, 20, 'Vasilica', 'Gigel', 3500, 3, NULL);
INSERT INTO angajati VALUES (29, 21, 'Ionescu', 'Maria', 3000, 5, NULL);
INSERT INTO angajati VALUES (30, 22, 'Pop', 'Vasile', 3500, 3, 'pop@yahoo.com');
INSERT INTO angajati VALUES (31, 23, 'Georgescu', 'Serban', 3000, 3, 'serban@gmail.com');
INSERT INTO angajati VALUES (32, 23, 'Vasile', 'Ana', 4200, 3, NULL);
INSERT INTO angajati VALUES (33, 24, 'Popescu', 'Vasile', 3100, 3, NULL);
INSERT INTO angajati VALUES (34, 25, 'Ionel', 'Ion', 4500, 3, NULL);
INSERT INTO angajati VALUES (35, 26, 'Vasilica', 'Gigel', 5500, 3, NULL);
INSERT INTO angajati VALUES (36, 26, 'Ionescu', 'Maria', 2600, 4, NULL);

INSERT INTO tasks VALUES(1, 'Curata sala', NULL, 1);
INSERT INTO tasks VALUES(2, 'Vinde bilete', NULL, 2);
INSERT INTO tasks VALUES(3, 'Verifica bilete la intrare', NULL, 1);
INSERT INTO tasks VALUES(4, 'Proiecteza filmul', NULL, 3);
INSERT INTO tasks VALUES(5, 'Actualizeaza site-ul', NULL, 2);
INSERT INTO tasks VALUES(6, 'Imparte task-uri', NULL, 4);
INSERT INTO tasks VALUES(7, 'Intalniri cu colaboratorii', NULL, 5);

INSERT INTO task_ang VALUES(1, 2);
INSERT INTO task_ang VALUES(2, 6);
INSERT INTO task_ang VALUES(3, 4);
INSERT INTO task_ang VALUES(3, 5);
INSERT INTO task_ang VALUES(4, 3);
INSERT INTO task_ang VALUES(4, 1);
INSERT INTO task_ang VALUES(5, 6);
INSERT INTO task_ang VALUES(6, 5);
INSERT INTO task_ang VALUES(7, 4);
INSERT INTO task_ang VALUES(7, 5);
INSERT INTO task_ang VALUES(8, 1);
INSERT INTO task_ang VALUES(9, 7);
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```
INSERT INTO task_ang VALUES(8, 1);
INSERT INTO task_ang VALUES(9, 7);
INSERT INTO task_ang VALUES(9, 6);
INSERT INTO task_ang VALUES(10, 7);
INSERT INTO task_ang VALUES(11, 4);
INSERT INTO task_ang VALUES(12, 6);
INSERT INTO task_ang VALUES(12, 7);
INSERT INTO task_ang VALUES(13, 7);
INSERT INTO task_ang VALUES(14, 1);
INSERT INTO task_ang VALUES(15, 6);
INSERT INTO task_ang VALUES(16, 6);
INSERT INTO task_ang VALUES(17, 7);
INSERT INTO task_ang VALUES(18, 5);
INSERT INTO task_ang VALUES(18, 4);
INSERT INTO task_ang VALUES(19, 7);
INSERT INTO task_ang VALUES(20, 6);
INSERT INTO task_ang VALUES(21, 7);
INSERT INTO task_ang VALUES(22, 1);
INSERT INTO task_ang VALUES(23, 6);
INSERT INTO task_ang VALUES(24, 6);
INSERT INTO task_ang VALUES(25, 2);
INSERT INTO task_ang VALUES(26, 6);
INSERT INTO task_ang VALUES(26, 7);
INSERT INTO task_ang VALUES(27, 7);
INSERT INTO task_ang VALUES(28, 6);
INSERT INTO task_ang VALUES(29, 4);
INSERT INTO task_ang VALUES(29, 5);
INSERT INTO task_ang VALUES(30, 7);
INSERT INTO task_ang VALUES(31, 7);
INSERT INTO task_ang VALUES(32, 6);
INSERT INTO task_ang VALUES(33, 6);
INSERT INTO task_ang VALUES(34, 7);
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```
INSERT INTO sali VALUES(1, 1, 'Sala 1', 120);
INSERT INTO sali VALUES(1, 2, 'Sala 2', 100);
INSERT INTO sali VALUES(1, 3, 'Sala 3', 210);
INSERT INTO sali VALUES(2, 1, 'Sala 1', 130);
INSERT INTO sali VALUES(2, 2, 'Sala 2', 110);
INSERT INTO sali VALUES(3, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(3, 2, 'Sala 2', 140);
INSERT INTO sali VALUES(3, 3, 'Sala 3', 105);
INSERT INTO sali VALUES(4, 1, 'Sala 1', 125);
INSERT INTO sali VALUES(5, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(6, 1, 'Sala 1', 210);
INSERT INTO sali VALUES(7, 1, 'Sala 1', 145);
INSERT INTO sali VALUES(7, 2, 'Sala 2', 90);
INSERT INTO sali VALUES(8, 1, 'Sala 1', 125);
INSERT INTO sali VALUES(9, 1, 'Sala 1', 130);
INSERT INTO sali VALUES(10, 1, 'Sala 1', 50);
INSERT INTO sali VALUES(11, 2, 'Sala 2', 100);
INSERT INTO sali VALUES(11, 3, 'Sala 3', 210);
INSERT INTO sali VALUES(12, 1, 'Sala 1', 130);
INSERT INTO sali VALUES(12, 2, 'Sala 2', 110);
INSERT INTO sali VALUES(13, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(13, 2, 'Sala 2', 140);
INSERT INTO sali VALUES(13, 3, 'Sala 3', 105);
INSERT INTO sali VALUES(14, 1, 'Sala 1', 125);
INSERT INTO sali VALUES(15, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(16, 1, 'Sala 1', 210);
INSERT INTO sali VALUES(17, 1, 'Sala 1', 145);
INSERT INTO sali VALUES(17, 2, 'Sala 2', 90);
INSERT INTO sali VALUES(18, 1, 'Sala 1', 125);
INSERT INTO sali VALUES(19, 1, 'Sala 1', 130);
INSERT INTO sali VALUES(20, 1, 'Sala 1', 50);
INSERT INTO sali VALUES(21, 2, 'Sala 2', 100);
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```
INSERT INTO sali VALUES(21, 3, 'Sala 3', 210);
INSERT INTO sali VALUES(22, 1, 'Sala 1', 130);
INSERT INTO sali VALUES(22, 2, 'Sala 2', 110);
INSERT INTO sali VALUES(23, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(23, 2, 'Sala 2', 140);
INSERT INTO sali VALUES(23, 3, 'Sala 3', 105);
INSERT INTO sali VALUES(24, 1, 'Sala 1', 125);
INSERT INTO sali VALUES(25, 1, 'Sala 1', 150);
INSERT INTO sali VALUES(26, 1, 'Sala 1', 210);

INSERT INTO filme VALUES(1, 'Pirates of the Caribbean', 140, '9-JUL-2016');
INSERT INTO filme VALUES(2, 'Murder on the Orient Express', 120, '10-NOV-2017');
INSERT INTO filme VALUES(3, 'The Lord of the Rings', 200, '17-DEC-2003');
INSERT INTO filme VALUES(4, 'The Godfather', 175, '24-MAR-1972');
INSERT INTO filme VALUES(5, 'Schindler List', 200, '4-FEB-1993');
INSERT INTO filme VALUES(6, 'Inception', 148, '16-JUL-2010');
INSERT INTO filme VALUES(7, 'The Wolf of Wall Street', 180, '25-DEC-2013');

INSERT INTO difuzari VALUES(1, sysdate, 1, 1, 3, 15);
INSERT INTO difuzari VALUES(2, '01-DEC-2020', 2, 1, 2, 20);
INSERT INTO difuzari VALUES(3, '20-NOV-2020', 3, 1, 1, 18);
INSERT INTO difuzari VALUES(4, sysdate, 1, 2, 1, 20);
INSERT INTO difuzari VALUES(5, '12-MAY-2019', 7, 3, 1, 16);
INSERT INTO difuzari VALUES(6, '13-FEB-2020', 5, 4, 1, 12);
INSERT INTO difuzari VALUES(7, sysdate, 5, 5, 1, 15);
INSERT INTO difuzari VALUES(8, '22-OCT-2020', 3, 6, 1, 10);
INSERT INTO difuzari VALUES(9, '02-JAN-2019', 7, 7, 2, 15);
INSERT INTO difuzari VALUES(10, sysdate, 3, 8, 1, 20);
INSERT INTO difuzari VALUES(11, '13-APR-2018', 4, 9, 1, 12);
INSERT INTO difuzari VALUES(12, '20-MAY-2019', 6, 10, 1, 22);
INSERT INTO difuzari VALUES(13, sysdate, 6, 11, 2, 10);
INSERT INTO difuzari VALUES(14, sysdate, 5, 12, 1, 17);
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```
INSERT INTO difuzari VALUES(13, sysdate, 6, 11, 2, 10);
INSERT INTO difuzari VALUES(14, sysdate, 5, 12, 1, 17);
INSERT INTO difuzari VALUES(15, '08-JUN-2020', 6, 12, 1, 24);
INSERT INTO difuzari VALUES(16, '10-JUL-2020', 1, 12, 2, 20);
INSERT INTO difuzari VALUES(17, '01-MAY-2019', 4, 13, 1, 13);
INSERT INTO difuzari VALUES(18, '20-FEB-2020', 6, 14, 1, 18);
INSERT INTO difuzari VALUES(19, '19-AUG-2018', 5, 15, 1, 20);
INSERT INTO difuzari VALUES(20, '09-MAR-2020', 3, 16, 1, 17);
INSERT INTO difuzari VALUES(21, '03-SEP-2019', 4, 17, 1, 22);
INSERT INTO difuzari VALUES(22, sysdate, 2, 17, 2, 16);
INSERT INTO difuzari VALUES(23, sysdate, 4, 18, 1, 24);
INSERT INTO difuzari VALUES(24, sysdate, 2, 19, 1, 20);
INSERT INTO difuzari VALUES(25, '13-APR-2018', 1, 20, 1, 12);
INSERT INTO difuzari VALUES(26, '20-MAY-2019', 6, 21, 3, 22);
INSERT INTO difuzari VALUES(27, sysdate, 2, 22, 2, 10);
INSERT INTO difuzari VALUES(28, sysdate, 5, 23, 1, 17);
INSERT INTO difuzari VALUES(29, '08-JUN-2020', 1, 24, 1, 24);
INSERT INTO difuzari VALUES(30, '08-JUN-2020', 5, 25, 1, 24);
INSERT INTO difuzari VALUES(31, sysdate, 3, 26, 1, 24);

INSERT INTO clienti VALUES(1, 'Maria', 'Cristina', 0.25, '10-OCT-2000');
INSERT INTO clienti VALUES(2, 'Gigel', 'Gigel', NULL, '09-MAR-1993');
INSERT INTO clienti VALUES(3, 'Ionescu', 'Andrei', 0.5, '04-FEB-1950');
INSERT INTO clienti VALUES(4, 'Georgescu', 'Maria', NULL, '10-JUL-1978');
INSERT INTO clienti VALUES(5, 'Vasilescu', 'Ion', 1, '03-AUG-2015');
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```
INSERT INTO bilete VALUES(1, 1);
INSERT INTO bilete VALUES(1, 5);
INSERT INTO bilete VALUES(1, 7);
INSERT INTO bilete VALUES(1, 21);
INSERT INTO bilete VALUES(1, 30);
INSERT INTO bilete VALUES(2, 4);
INSERT INTO bilete VALUES(2, 10);
INSERT INTO bilete VALUES(2, 17);
INSERT INTO bilete VALUES(2, 23);
INSERT INTO bilete VALUES(3, 1);
INSERT INTO bilete VALUES(3, 2);
INSERT INTO bilete VALUES(4, 8);
INSERT INTO bilete VALUES(4, 15);
INSERT INTO bilete VALUES(4, 29);
INSERT INTO bilete VALUES(4, 31);
INSERT INTO bilete VALUES(4, 7);
INSERT INTO bilete VALUES(4, 13);
INSERT INTO bilete VALUES(5, 1);
INSERT INTO bilete VALUES(5, 5);
INSERT INTO bilete VALUES(5, 6);
INSERT INTO bilete VALUES(5, 11);
INSERT INTO bilete VALUES(5, 14);
INSERT INTO bilete VALUES(5, 28);
INSERT INTO bilete VALUES(5, 29);
INSERT INTO bilete VALUES(5, 30);

INSERT INTO categorii VALUES(1, 'Comedie');
INSERT INTO categorii VALUES(2, 'Horror');
INSERT INTO categorii VALUES(3, 'Romanic');
INSERT INTO categorii VALUES(4, 'Actiune');
INSERT INTO categorii VALUES(5, 'Drama');
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```
INSERT INTO categorie_film VALUES(2, 1);
INSERT INTO categorie_film VALUES(3, 1);
INSERT INTO categorie_film VALUES(5, 1);
INSERT INTO categorie_film VALUES(1, 2);
INSERT INTO categorie_film VALUES(2, 2);
INSERT INTO categorie_film VALUES(4, 2);
INSERT INTO categorie_film VALUES(1, 3);
INSERT INTO categorie_film VALUES(3, 3);
INSERT INTO categorie_film VALUES(4, 3);
INSERT INTO categorie_film VALUES(5, 3);
INSERT INTO categorie_film VALUES(1, 4);
INSERT INTO categorie_film VALUES(2, 4);
INSERT INTO categorie_film VALUES(3, 4);
INSERT INTO categorie_film VALUES(3, 5);
INSERT INTO categorie_film VALUES(4, 5);
INSERT INTO categorie_film VALUES(5, 5);
INSERT INTO categorie_film VALUES(1, 6);
INSERT INTO categorie_film VALUES(3, 6);
INSERT INTO categorie_film VALUES(1, 7);
INSERT INTO categorie_film VALUES(2, 7);
INSERT INTO categorie_film VALUES(3, 7);
INSERT INTO categorie_film VALUES(4, 7);

INSERT INTO actori VALUES(1, 'Depp', 'Johnny', NULL, '09-JUN-1963');
INSERT INTO actori VALUES(2, 'Cruz', 'Penelope', 'Penelope@yahoo.com', '28-APR-1974');
INSERT INTO actori VALUES(3, 'Claflin', 'Sam', 'sam@gmail.com', '27-JUN-1986');
INSERT INTO actori VALUES(4, 'Brando', 'Marlon', NULL, '03-APR-1924');
INSERT INTO actori VALUES(5, 'Pacino', 'Al', NULL, '25-APR-1940');
INSERT INTO actori VALUES(6, 'DiCaprio', 'Leonardo', NULL, '11-NOV-1974');
INSERT INTO actori VALUES(7, 'Robbie', 'Margot', 'rm@gmail.com', '02-JUL-1990');
INSERT INTO actori VALUES(8, 'Bloom', 'Orlando', 'orlando@gmail.com', '13-JAN-1977');
```

db_proiect | Relational_1 (Untitled_1)

Worksheet | Query Builder

```
INSERT INTO actori VALUES(1, 'Depp', 'Johnny', NULL, '09-JUN-1963');
INSERT INTO actori VALUES(2, 'Cruz', 'Penelope', 'Penelope@yahoo.com', '28-APR-1974');
INSERT INTO actori VALUES(3, 'Claflin', 'Sam', 'sam@gmail.com', '27-JUN-1986');
INSERT INTO actori VALUES(4, 'Brando', 'Marlon', NULL, '03-APR-1924');
INSERT INTO actori VALUES(5, 'Pacino', 'Al', NULL, '25-APR-1940');
INSERT INTO actori VALUES(6, 'DiCaprio', 'Leonardo', NULL, '11-NOV-1974');
INSERT INTO actori VALUES(7, 'Robbie', 'Margot', 'rm@gmail.com', '02-JUL-1990');
INSERT INTO actori VALUES(8, 'Bloom', 'Orlando', 'orlando@gmail.com', '13-JAN-1977');
INSERT INTO actori VALUES(9, 'Howard', 'Alan', 'alan@mail.com', '05-AUG-1937');
INSERT INTO actori VALUES(10, 'Neeson', 'Liam', NULL, '07-JUN-1952');
INSERT INTO actori VALUES(11, 'Goodall', 'Caroline', NULL, '13-NOV-1959');

INSERT INTO actor_film VALUES(1, 1, 'Principal', 'Jack Sparrow', 'Pozitiv', 5000000);
INSERT INTO actor_film VALUES(1, 2, 'Principal', 'Angelica', 'Negativ', 1000000);
INSERT INTO actor_film VALUES(1, 3, 'Secundar', 'Philip', 'Pozitiv', 200000);
INSERT INTO actor_film VALUES(1, 7, 'Episodic', 'Gillette', 'Negativ', 20000);
INSERT INTO actor_film VALUES(2, 6, 'Principal', 'Young Policeman', 'Pozitiv', 70000);
INSERT INTO actor_film VALUES(2, 3, 'Secundar', 'Hercule Poirot', 'Negativ', 10000);
INSERT INTO actor_film VALUES(2, 2, 'Figuratie', NULL, NULL, 10000);
INSERT INTO actor_film VALUES(3, 9, 'Principal', 'Voice of the Ring', 'Pozitiv', 900000);
INSERT INTO actor_film VALUES(3, 8, 'Secundar', 'Legolas', NULL, 300000);
INSERT INTO actor_film VALUES(4, 4, 'Principal', 'Don Vito Corleone', 'Negativ', 7000000);
INSERT INTO actor_film VALUES(4, 5, 'Principal', 'Michael Corleone', 'Negativ', 6000000);
INSERT INTO actor_film VALUES(5, 10, 'Principal', 'Oskar Schindler', 'Pozitiv', 3000000);
INSERT INTO actor_film VALUES(5, 11, 'Principal', 'Emilie Schindler', NULL, 1000000);
INSERT INTO actor_film VALUES(6, 6, 'Principal', 'Cobb', 'Pozitiv', 400000);
INSERT INTO actor_film VALUES(6, 8, 'Figuratie', NULL, NULL, 5000);
INSERT INTO actor_film VALUES(7, 6, 'Principal', 'Jordan Belfort', 'Negativ', 4000000);
INSERT INTO actor_film VALUES(7, 7, 'Secundar', 'Naomi Lapaglia', NULL, 800000);
```

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

Am creat o procedura care foloseste tablori indexate pentru a mari cu 10% salariile tuturor angajatilor care lucreaza in top 3 cele mai profitabile cinematografe. Daca dupa marirea de salar, noul salar ar fi mai mare decat salariul maxim permis pentru job-ul respectiv, atunci angajatul va avea salariul maxim permis. (Daca sunt mai multe cinematografe la egalitate pe locul 3, atunci se vor considera toate).

```
CREATE OR REPLACE PROCEDURE mareste_salar AS
    TYPE tabel IS TABLE OF cinematografe.id_cinema%TYPE
                            INDEX BY BINARY_INTEGER;
    cinemas tabel;
BEGIN
    SELECT DISTINCT id_cinema BULK COLLECT INTO cinemas
    FROM difuzari dif JOIN bilete bil ON (dif.id_difuzare = bil.id_difuzare)
    WHERE (SELECT COUNT(id_cinema)
           FROM cinematografe cin
           WHERE (SELECT SUM(pret * (1 - NVL(discount, 0)))
                  FROM difuzari d JOIN bilete b ON (d.id_difuzare = b.id_difuzare)
                                  JOIN clienti c ON (b.id_client = c.id_client)
                  WHERE d.id_cinema = cin.id_cinema
                  GROUP BY d.id_cinema) >
                   (SELECT SUM(pret * (1 - NVL(discount, 0)))
                         FROM difuzari d JOIN bilete b ON (d.id_difuzare = b.id_difuzare)
                         JOIN clienti c ON (b.id_client = c.id_client)
                         WHERE d.id_cinema = dif.id_cinema
                         GROUP BY d.id_cinema)) <= 2;
    FOR i IN cinemas.FIRST..cinemas.LAST LOOP
        UPDATE angajati
        SET salar = (SELECT
                        CASE
                            WHEN salar * 1.1 > salar_maxim THEN salar_maxim
                            ELSE salar * 1.1
                        END
                     FROM jobs
                     WHERE jobs.id_job = angajati.id_job)
        WHERE id_cinematografe = cinemas(i);
    END LOOP;
END;
/

BEGIN
    mareste_salar();
END;
/
```

```
CREATE OR REPLACE PROCEDURE mareste_salar AS
    TYPE tabel IS TABLE OF cinematografe.id_cinema%TYPE
                            INDEX BY BINARY_INTEGER;
    cinemas tabel;
BEGIN
    SELECT DISTINCT id_cinema BULK COLLECT INTO cinemas
    FROM difuzari dif JOIN bilete bil ON (dif.id_difuzare = bil.id_difuzare)
    WHERE (SELECT COUNT(id_cinema)
           FROM cinematografe cin
           WHERE (SELECT SUM(pret * (1 - NVL(discount, 0)))
                  FROM difuzari d JOIN bilete b ON (d.id_difuzare = b.id_difuzare)
                                  JOIN clienti c ON (b.id_client = c.id_client)
                  WHERE d.id_cinema = cin.id_cinema
                  GROUP BY d.id_cinema) > (SELECT SUM(pret * (1 - NVL(discount, 0)))
                                           FROM difuzari d JOIN bilete b ON (d.id_difuzare = b.id_difuzare)
                                                           JOIN clienti c ON (b.id_client = c.id_client)
                                           WHERE d.id_cinema = dif.id_cinema
                                           GROUP BY d.id_cinema)) <= 2;
    FOR i IN cinemas.FIRST..cinemas.LAST LOOP
        UPDATE angajati
        SET salar = (SELECT
                         CASE
                             WHEN salar * 1.1 > salar_maxim THEN salar_maxim
                             ELSE salar * 1.1
                         END
                     FROM jobs
                     WHERE jobs.id_job = angajati.id_job)
        WHERE id_cinematografe = cinemas(i);
    END LOOP;
END;
/
```

```
/

BEGIN
    mareste_salar();
END;
/
```

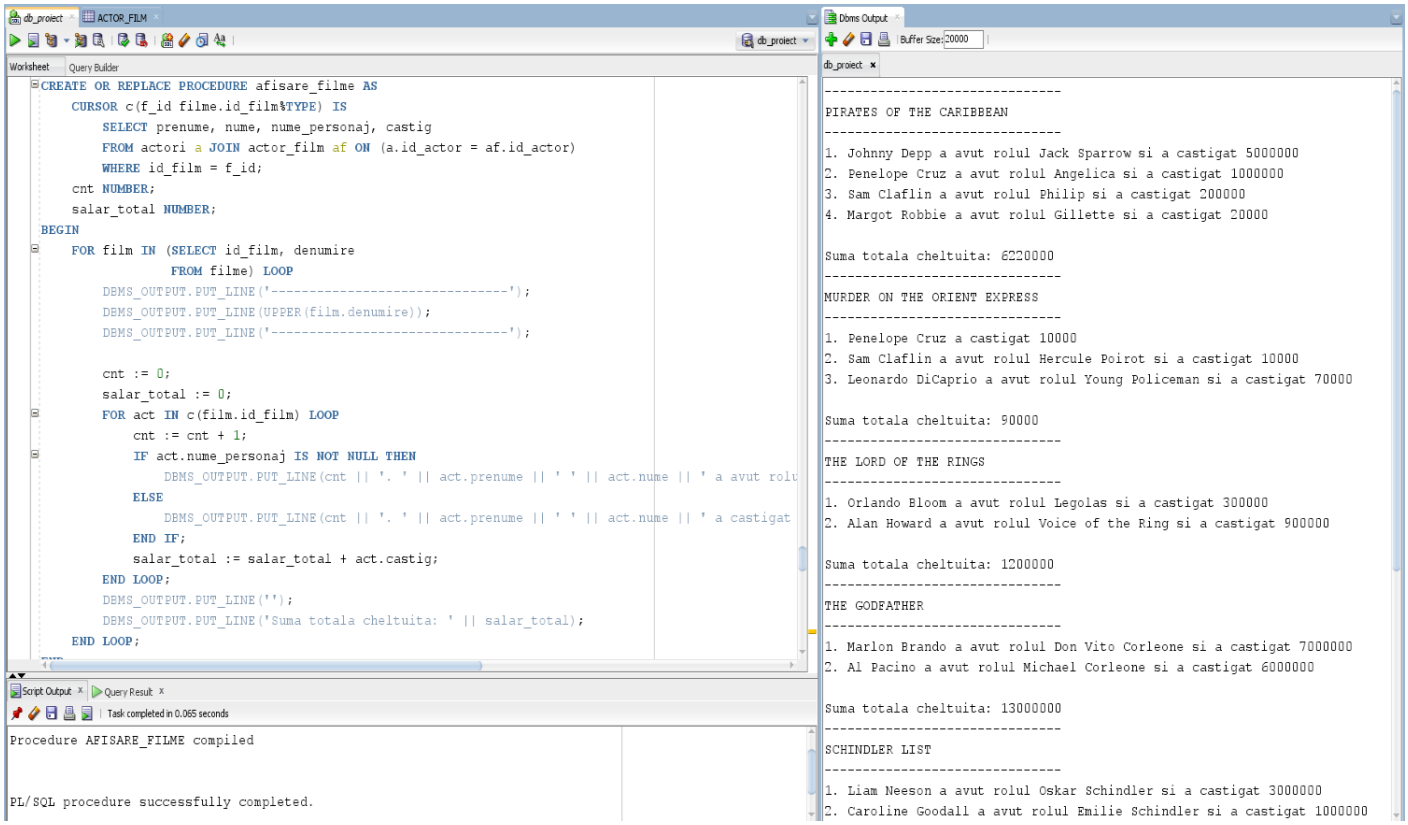
```
Procedure MARESTE_SALAR compiled


PL/SQL procedure successfully completed.
```

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

Am creat o procedura care pentru fiecare film afiseaza numele filmul, toti actorii care au jucat in filmul respectiv si suma totala care s-a cheltuit pentru salariile actorilor.

```
CREATE OR REPLACE PROCEDURE afisare_filme AS
    CURSOR c(f_id filme.id_film%TYPE) IS
        SELECT prenume, nume, nume_personaj, castig
        FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
        WHERE id_film = f_id;
    cnt NUMBER;
    salar_total NUMBER;
```

```
BEGIN
    FOR film IN (SELECT id_film, denumire
                 FROM filme) LOOP
        DBMS_OUTPUT.PUT_LINE('-------------------------------');
        DBMS_OUTPUT.PUT_LINE(UPPER(film.denumire));
        DBMS_OUTPUT.PUT_LINE('-------------------------------');
        cnt := 0;
        salar_total := 0;
        FOR act IN c(film.id_film) LOOP
            cnt := cnt + 1;
            IF act.nume_personaj IS NOT NULL THEN
      DBMS_OUTPUT.PUT_LINE(cnt || '. ' || act.prenume || ' ' || act.nume || ' a avut rolul '
                           || act.nume_personaj || ' si a castigat ' || NVL(act.castig, 0));
            ELSE
      DBMS_OUTPUT.PUT_LINE(cnt || '. ' || act.prenume || ' ' || act.nume || ' a castigat '
                           || NVL(act.castig, 0));
            END IF;
            salar_total := salar_total + act.castig;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('');
        DBMS_OUTPUT.PUT_LINE('Suma totala cheltuita: ' || salar_total);
    END LOOP;
END;
/

BEGIN
    afisare_filme();
END;
/
```



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.

Am creat o functie care pentru o categorie data returneaza numele actorului care a jucat in cele mai multe filme care apartin categoriei respective. Am tratat exceptiile cand nu exista categoria data si cand exista mai multi actori care au jucat in numarul maxim de filme din respectiva categorie.

```
CREATE OR REPLACE FUNCTION gaseste_actor(categ categorii.denumire%TYPE) RETURN VARCHAR2 IS
    id_categ categorii.id_categorie%TYPE;
    ans VARCHAR2(50);
BEGIN
    SELECT id_categorie INTO id_categ
    FROM categorii
    WHERE UPPER(denumire) = UPPER(categ);

    SELECT prenume ||  ' ' || nume INTO ans
    FROM actori act
    WHERE  (SELECT COUNT(DISTINCT f.id_film)
            FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
                          JOIN filme f ON (f.id_film = af.id_film)
                          JOIN categorie_film cf ON (cf.id_film = f.id_film)
            WHERE cf.id_categorie = id_categ AND a.id_actor = act.id_actor
            GROUP BY a.id_actor) =
             (SELECT MAX(COUNT(DISTINCT f.id_film))
              FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
                                JOIN filme f ON (f.id_film = af.id_film)
                                JOIN categorie_film cf ON (cf.id_film = f.id_film)
                 WHERE cf.id_categorie = id_categ
                 GROUP BY a.id_actor);

    RETURN ans;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Nu exista o categorie cu acest nume!');
        WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.PUT_LINE('Exista mai multi actori!');
        WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Alta eroare!');
END;
/

DECLARE
    den VARCHAR2(20) := '&nume_categ';
BEGIN
    DBMS_OUTPUT.PUT_LINE(UPPER(den));
    DBMS_OUTPUT.PUT_LINE(gaseste_actor(den));
END;
/
```

```oracle
CREATE OR REPLACE FUNCTION gaseste_actor(categ categorii.denumire%TYPE) RETURN VARCHAR2 IS
    id_categ categorii.id_categorie%TYPE;
    ans VARCHAR2(50);
BEGIN
    SELECT id_categorie INTO id_categ
    FROM categorii
    WHERE UPPER(denumire) = UPPER(categ);

    SELECT prenume || ' ' || nume INTO ans
    FROM actori act
    WHERE  (SELECT COUNT(DISTINCT f.id_film)
            FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
                          JOIN filme f ON (f.id_film = af.id_film)
                          JOIN categorie_film cf ON (cf.id_film = f.id_film)
            WHERE cf.id_categorie = id_categ AND a.id_actor = act.id_actor
            GROUP BY a.id_actor) = (SELECT MAX(COUNT(DISTINCT f.id_film))
                                    FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
                                                  JOIN filme f ON (f.id_film = af.id_film)
                                                  JOIN categorie_film cf ON (cf.id_film = f.id_film)
                                    WHERE cf.id_categorie = id_categ
                                    GROUP BY a.id_actor);

    RETURN ans;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Nu exista o categorie cu acest nume!');
        WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.PUT_LINE('Exista mai multi actori!');
        WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Alta eroare!');
END;
/
```

```
Function GASESTE_ACTOR compiled
```

```oracle
                        JOIN categorie_film cf ON (cf.id_film = f.id_film)
            WHERE cf.id_categorie = id_categ AND a.id_actor = act.id_actor
            GROUP BY a.id_actor) = (SELECT MAX(COUNT(DISTINCT f.id_film))
                                    FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
                                                  JOIN filme f ON (f.id_film = af.id_film)
                                                  JOIN categorie_film cf ON (cf.id_film = f.id_film)
                                    WHERE cf.id_categorie = id_categ
                                    GROUP BY a.id_actor);

    RETURN ans;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Nu exista o categorie cu acest nume!');
        WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.PUT_LINE('Exista mai multi actori!');
        WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Alta eroare!');
END;
/

DECLARE
    den VARCHAR2(20) := '&nume_categ';
BEGIN
    DBMS_OUTPUT.PUT_LINE(UPPER(den));
    DBMS_OUTPUT.PUT_LINE(gaseste_actor(den));
END;
/
```

```
06503. 00000 -  "PL/SQL: Function returned without value"
*Cause:    A call to PL/SQL function completed, but no RETURN statement was
           executed.
*Action:   Rewrite PL/SQL function, making sure that it always returns
           a value of a proper type.
```

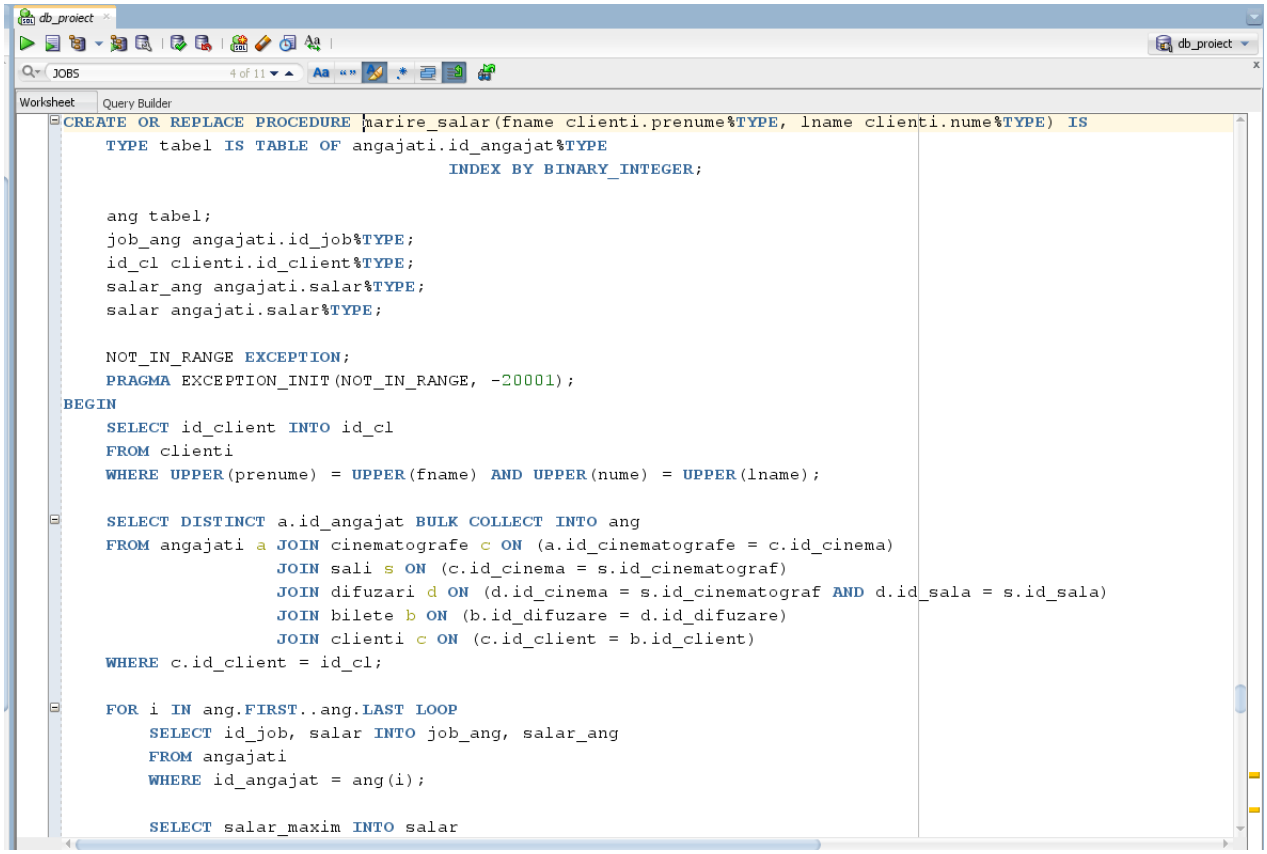
```
COMEDIE
Leonardo DiCaprio

SF
Nu exista o categorie cu acest nume!

DRAMA
Exista mai multi actori!
```

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

Am facut o procedura care primeste ca parametru numele si prenumele unui client, si mareste cu 5% salriile tuturor angajtilor care lucreaza intr-un cinematograf in care a fost persoana respectiva. Am tratat urmatoarele exceptii: nu exista o persoana cu numele dat, exista mai multe persoane cu numele dat, dupa marirea salariului noul salar depaseste salariul maxim pentru job-ul respectiv.

```
CREATE OR REPLACE PROCEDURE
           marire_salar(fname clienti.prenume%TYPE, lname clienti.nume%TYPE) IS
    TYPE tabel IS TABLE OF angajati.id_angajat%TYPE
                                    INDEX BY BINARY_INTEGER;

    ang tabel;
    job_ang angajati.id_job%TYPE;
    id_cl clienti.id_client%TYPE;
    salar_ang angajati.salar%TYPE;
    salar angajati.salar%TYPE;

    NOT_IN_RANGE EXCEPTION;
    PRAGMA EXCEPTION_INIT(NOT_IN_RANGE, -20001);
BEGIN
    SELECT id_client INTO id_cl
    FROM clienti
    WHERE UPPER(prenume) = UPPER(fname) AND UPPER(nume) = UPPER(lname);

    SELECT DISTINCT a.id_angajat BULK COLLECT INTO ang
    FROM angajati a JOIN cinematografe c ON (a.id_cinematografe = c.id_cinema)
                    JOIN sali s ON (c.id_cinema = s.id_cinematograf)
            JOIN difuzari d ON (d.id_cinema = s.id_cinematograf AND d.id_sala = s.id_sala)
                    JOIN bilete b ON (b.id_difuzare = d.id_difuzare)
                    JOIN clienti c ON (c.id_client = b.id_client)
    WHERE c.id_client = id_cl;

    FOR i IN ang.FIRST..ang.LAST LOOP
        SELECT id_job, salar INTO job_ang, salar_ang
        FROM angajati
        WHERE id_angajat = ang(i);

        SELECT salar_maxim INTO salar
        FROM jobs
        WHERE id_job = job_ang;

        IF salar_ang * 1.05 > salar THEN RAISE NOT_IN_RANGE;
        END IF;
    END LOOP;
    FOR i IN ang.FIRST..ang.LAST LOOP
        UPDATE angajati
        SET salar = 1.05 * salar
        WHERE id_angajat = ang(i);
```

```
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('S-a realizat update-ul cu succes!');
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu numele dat!');
        WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu numele dat!');
        WHEN NOT_IN_RANGE THEN
            DBMS_OUTPUT.PUT_LINE('Noul salar nu respecta restrictiile de salar!');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Alta eroare!');
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE('GIGEL GIGEL: ');
    marire_salar('gigel', 'gigel');
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE('ANDREI IONESCU: ');
    marire_salar('andrei', 'ionescu');
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE('IONEL GIGEL: ');
    marire_salar('ionel', 'gigel');
END;
/
```



```
    FOR i IN ang.FIRST..ang.LAST LOOP
        SELECT id_job, salar INTO job_ang, salar_ang
        FROM angajati
        WHERE id_angajat = ang(i);

        SELECT salar_maxim INTO salar
        FROM jobs
        WHERE id_job = job_ang;

        IF salar_ang * 1.05 > salar THEN RAISE NOT_IN_RANGE;
        END IF;
    END LOOP;

    FOR i IN ang.FIRST..ang.LAST LOOP
        UPDATE angajati
        SET salar = 1.05 * salar
        WHERE id_angajat = ang(i);
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('S-a realizat update-ul cu succes!');
    EXCEPTION
        WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu numele dat!');
        WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu numele dat!');
        WHEN NOT_IN_RANGE THEN DBMS_OUTPUT.PUT_LINE('Noul salar nu respecta restrictiile de salar!');
        WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Alta eroare!');
END;
/

BEGIN
    DBMS_OUTPUT.PUT_LINE('GIGEL GIGEL: ');
    marire_salar('gigel', 'gigel');
```

```
        WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Nu exista niciun clie
        WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.PUT_LINE('Exista mai multi clie
        WHEN NOT_IN_RANGE THEN DBMS_OUTPUT.PUT_LINE('Noul salar nu respecta
        WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Alta eroare!');
END;
/

BEGIN
    DBMS_OUTPUT.PUT_LINE('GIGEL GIGEL: ');
    marire_salar('gigel', 'gigel');
END;
/

BEGIN
    DBMS_OUTPUT.PUT_LINE('ANDREI IONESCU: ');
    marire_salar('andrei', 'ionescu');
END;
/

BEGIN
    DBMS_OUTPUT.PUT_LINE('IONEL GIGEL: ');
    marire_salar('ionel', 'gigel');
END;
/
```

Task completed in 0.271 seconds

```
Procedure MARIRE_SALAR compiled


PL/SQL procedure successfully completed.
```

```
GIGEL GIGEL:
Noul salar nu respecta restrictiile de salar!

ANDREI IONESCU:
S-a realizat update-ul cu succes!

IONEL GIGEL:
Nu exista niciun client cu numele dat!
```

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

Am definit un trigger pentru a nu permite sa fie mai mult de un cinematograf dintr-o anumite categorie in fiecare locatie.

```
CREATE OR REPLACE PACKAGE pachet_trigger AS
    TYPE tip IS RECORD (id_loc cinematografe.id_locatie%TYPE,
                        id_categ cinematografe.id_categorie%TYPE,
                        nr NUMBER);
    TYPE tabel_idx IS TABLE OF tip
            INDEX BY PLS_INTEGER;
    t tabel_idx;
    contor NUMBER:=0;
END;
/

CREATE OR REPLACE TRIGGER modif_cinema_comanda
BEFORE UPDATE OR INSERT
            ON cinematografe
BEGIN
    SELECT id_locatie, id_categorie, COUNT(*)
    BULK COLLECT INTO pachet_trigger.t
    FROM cinematografe
    GROUP BY id_locatie, id_categorie;
END;
/

CREATE OR REPLACE TRIGGER modif_cinema_linie
    BEFORE UPDATE OR INSERT ON cinematografe
    FOR EACH ROW
BEGIN
    FOR i IN pachet_trigger.t.FIRST..pachet_trigger.t.LAST LOOP
        IF pachet_trigger.t(i).id_loc = :NEW.id_locatie AND
           pachet_trigger.t(i).id_categ = :NEW.id_categorie AND
           pachet_trigger.t(i).nr + pachet_trigger.contor > 0 THEN
                RAISE_APPLICATION_ERROR(-20001, 'Nu poate fi mai mult de un cinematograf
                                                de la fiecare firma in fiecare locatie');
        END IF;
    END LOOP;
    pachet_trigger.contor := pachet_trigger.contor + 1;
END;
/

INSERT INTO cinematografe
SELECT 27, 1, 1, SYSDATE
FROM DUAL;

INSERT INTO cinematografe
SELECT 28, 1, 4, SYSDATE
FROM DUAL;
```

db_proiect.sql | Welcome Page

SQL Worksheet | History

db_proiect

Worksheet | Query Builder

```
CREATE OR REPLACE PACKAGE pachet_trigger AS
    TYPE tip IS RECORD (id_loc cinematografe.id_locatie%TYPE,
                        id_categ cinematografe.id_categorie%TYPE,
                        nr NUMBER);
    TYPE tabel_idx IS TABLE OF tip
            INDEX BY PLS_INTEGER;
    t tabel_idx;
    contor NUMBER:=0;
END;
/

CREATE OR REPLACE TRIGGER modif_cinema_comanda
BEFORE UPDATE OR INSERT
            ON cinematografe
BEGIN
    SELECT id_locatie, id_categorie, COUNT(*)
    BULK COLLECT INTO pachet_trigger.t
    FROM cinematografe
    GROUP BY id_locatie, id_categorie;
END;
/

CREATE OR REPLACE TRIGGER modif_cinema_linie
    BEFORE UPDATE OR INSERT ON cinematografe
    FOR EACH ROW
BEGIN
    FOR i IN pachet_trigger.t.FIRST..pachet_trigger.t.LAST LOOP
        IF pachet_trigger.t(i).id_loc = :NEW.id_locatie AND
           pachet_trigger.t(i).id_categ = :NEW.id_categorie AND
           pachet_trigger.t(i).nr + pachet_trigger.contor > 0 THEN
                RAISE_APPLICATION_ERROR(-20001, 'Nu poate fi mai mult de un cinematograf de la fiecare firma i
        END IF;
```

db_proiect.sql | Welcome Page

SQL Worksheet | History

db_proiect

Worksheet | Query Builder

```
BEGIN
    FOR i IN pachet_trigger.t.FIRST..pachet_trigger.t.LAST LOOP
        IF pachet_trigger.t(i).id_loc = :NEW.id_locatie AND
           pachet_trigger.t(i).id_categ = :NEW.id_categorie AND
           pachet_trigger.t(i).nr + pachet_trigger.contor > 0 THEN
                RAISE_APPLICATION_ERROR(-20001, 'Nu poate fi mai mult de un cinematograf de la fiecare firma i
        END IF;
    END LOOP;
    pachet_trigger.contor := pachet_trigger.contor + 1;
END;
/

INSERT INTO cinematografe
SELECT 27, 1, 1, SYSDATE
FROM DUAL;

INSERT INTO cinematografe
SELECT 28, 1, 4, SYSDATE
FROM DUAL;

ROLLBACK;
```

Script Output | Query Result

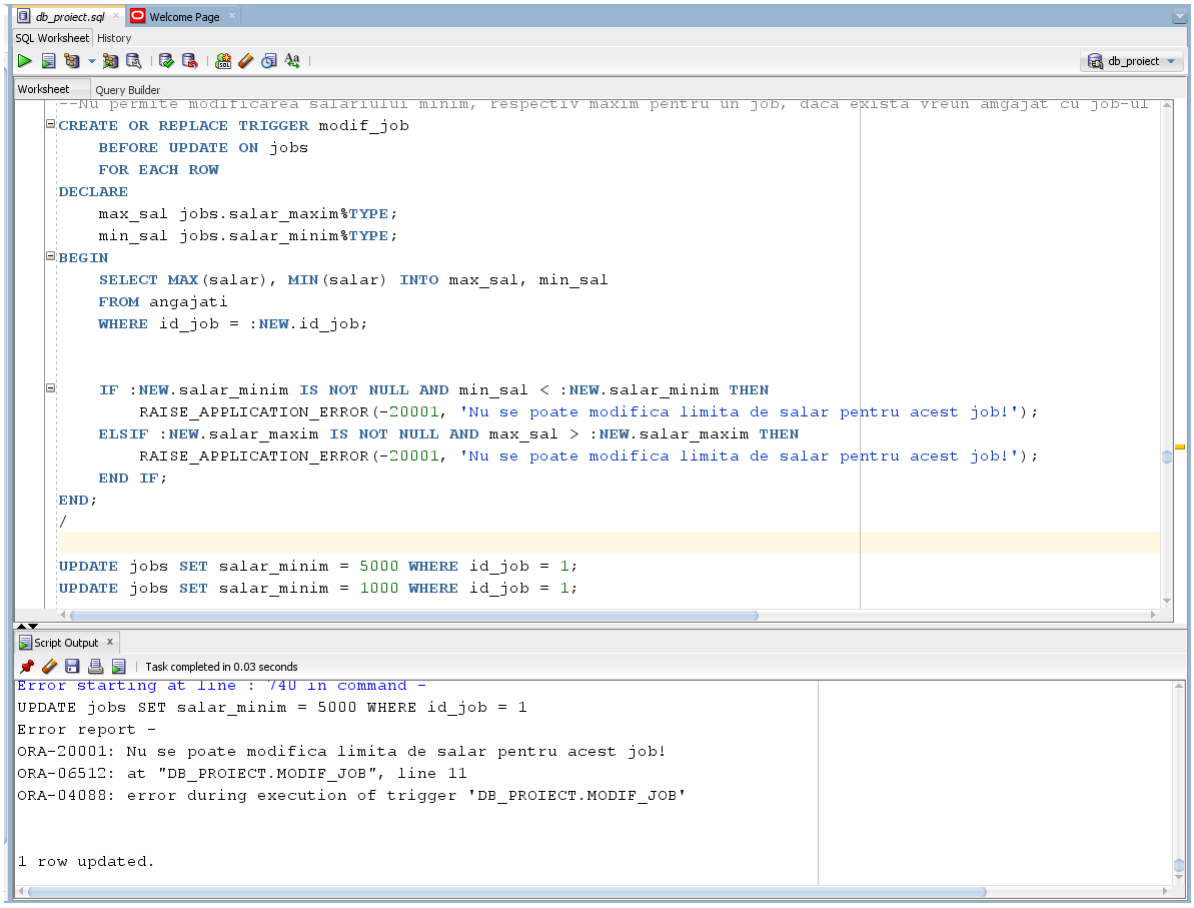
Task completed in 0.026 seconds

```
INSERT INTO cinematografe
SELECT 27, 1, 1, SYSDATE
FROM DUAL
Error report -
ORA-20001: Nu poate fi mai mult de un cinematograf de la fiecare firma in fiecare locatie
ORA-06512: at "DB_PROIECT.MODIF_CINEMA_LINIE", line 6
ORA-04088: error during execution of trigger 'DB_PROIECT.MODIF_CINEMA_LINIE'


1 row inserted.
```

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

Am definit un trigger care sa nu permita modificarea limitei inferioare, respectiv superioare a salariului unui job, decat daca toate salariile angajatilor cu acel job se afla in intervalul respectiv.

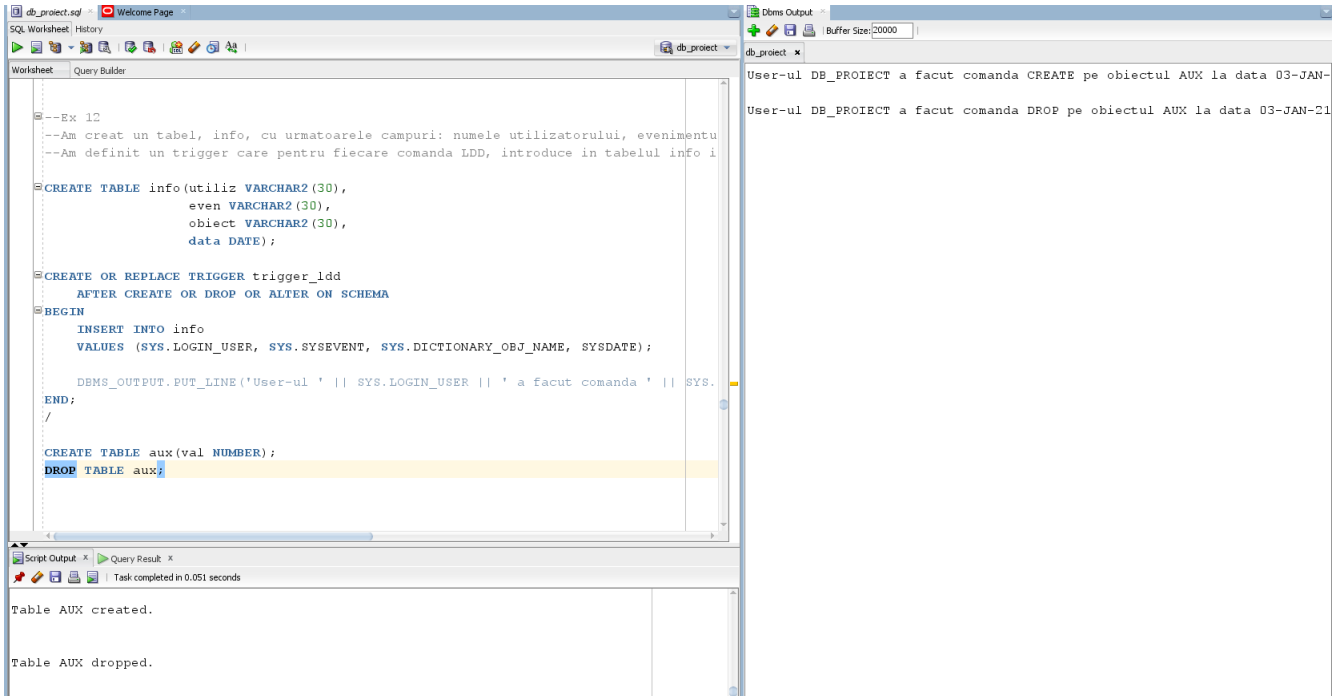
```
CREATE OR REPLACE TRIGGER modif_job
    BEFORE UPDATE ON jobs
    FOR EACH ROW
DECLARE
    max_sal jobs.salar_maxim%TYPE;
    min_sal jobs.salar_minim%TYPE;
BEGIN
    SELECT MAX(salar), MIN(salar) INTO max_sal, min_sal
    FROM angajati
    WHERE id_job = :NEW.id_job;


    IF :NEW.salar_minim IS NOT NULL AND min_sal < :NEW.salar_minim THEN
  RAISE_APPLICATION_ERROR(-20001, 'Nu se poate modifica limita de salar pentru acest job!');
    ELSIF :NEW.salar_maxim IS NOT NULL AND max_sal > :NEW.salar_maxim THEN
  RAISE_APPLICATION_ERROR(-20001, 'Nu se poate modifica limita de salar pentru acest job!');
    END IF;
END;
/
UPDATE jobs SET salar_minim = 5000 WHERE id_job = 1;
UPDATE jobs SET salar_minim = 1000 WHERE id_job = 1;
```



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

Am creat un tabel, info, cu urmatoarele campuri: numele utilizatorului, evenimentul de sistem, numele obiectului si data.
Am definit un trigger care pentru fiecare comanda LDD, introduce in tabelul info informatiile corespunzatoare.

```
CREATE TABLE info(utiliz VARCHAR2(30),
                  even VARCHAR2(30),
                  obiect VARCHAR2(30),
                  data DATE);

CREATE OR REPLACE TRIGGER trigger_ldd
    AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
    INSERT INTO info
    VALUES (SYS.LOGIN_USER, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE);

    DBMS_OUTPUT.PUT_LINE('User-ul ' || SYS.LOGIN_USER || ' a facut comanda ' ||
    SYS.SYSEVENT || ' pe obiectul ' || SYS.DICTIONARY_OBJ_NAME || ' la data ' || SYSDATE);
END;
/

CREATE TABLE aux(val NUMBER);
DROP TABLE aux;
```



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

```
CREATE OR REPLACE PACKAGE pachet_ex13 AS
    PROCEDURE mareste_salar;
    PROCEDURE afisare_filme;
    FUNCTION gaseste_actor(categ categorii.denumire%TYPE) RETURN VARCHAR2;
    PROCEDURE marire_salar(fname clienti.prenume%TYPE, lname clienti.nume%TYPE);

    TYPE tabel_cinema IS TABLE OF cinematografe.id_cinema%TYPE
                INDEX BY BINARY_INTEGER;
    TYPE tabel_ang IS TABLE OF angajati.id_angajat%TYPE
                INDEX BY BINARY_INTEGER;
END;
/

CREATE OR REPLACE PACKAGE BODY pachet_ex13 AS
    PROCEDURE mareste_salar IS
        cinemas tabel_cinema;
    BEGIN
        SELECT DISTINCT id_cinema BULK COLLECT INTO cinemas
        FROM difuzari dif JOIN bilete bil ON (dif.id_difuzare = bil.id_difuzare)
        WHERE (SELECT COUNT(id_cinema)
               FROM cinematografe cin
               WHERE (SELECT SUM(pret * (1 - NVL(discount, 0)))
                      FROM difuzari d JOIN bilete b ON (d.id_difuzare = b.id_difuzare)
                                      JOIN clienti c ON (b.id_client = c.id_client)
                      WHERE d.id_cinema = cin.id_cinema
                      GROUP BY d.id_cinema) >
                      (SELECT SUM(pret * (1 - NVL(discount, 0)))
                       FROM difuzari d JOIN bilete b ON (d.id_difuzare = b.id_difuzare)
                                       JOIN clienti c ON (b.id_client = c.id_client)
                                       WHERE d.id_cinema = dif.id_cinema
                                       GROUP BY d.id_cinema)) <= 2;
        FOR i IN cinemas.FIRST..cinemas.LAST LOOP
            UPDATE angajati
            SET salar = (SELECT
                            CASE
                                WHEN salar * 1.1 > salar_maxim THEN salar_maxim
                                ELSE salar * 1.1
                            END
                         FROM jobs
                         WHERE jobs.id_job = angajati.id_job)
            WHERE id_cinematografe = cinemas(i);
        END LOOP;
    END;
    PROCEDURE afisare_filme IS
        CURSOR c(f_id filme.id_film%TYPE) IS
            SELECT prenume, nume, nume_personaj, castig
            FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
            WHERE id_film = f_id;
        cnt NUMBER;
        salar_total NUMBER;
```

```
BEGIN
    FOR film IN (SELECT id_film, denumire
                 FROM filme) LOOP
        DBMS_OUTPUT.PUT_LINE('-------------------------------');
        DBMS_OUTPUT.PUT_LINE(UPPER(film.denumire));
        DBMS_OUTPUT.PUT_LINE('-------------------------------');

        cnt := 0;
        salar_total := 0;
        FOR act IN c(film.id_film) LOOP
            cnt := cnt + 1;
            IF act.nume_personaj IS NOT NULL THEN
                DBMS_OUTPUT.PUT_LINE(cnt || '. ' || act.prenume || ' ' || act.nume
   || ' a avut rolul ' || act.nume_personaj || ' si a castigat ' || NVL(act.castig, 0));
            ELSE
                DBMS_OUTPUT.PUT_LINE(cnt || '. ' || act.prenume || ' ' || act.nume
                || ' a castigat ' || NVL(act.castig, 0));
            END IF;
            salar_total := salar_total + act.castig;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('');
        DBMS_OUTPUT.PUT_LINE('Suma totala cheltuita: ' || salar_total);
    END LOOP;
END;
FUNCTION gaseste_actor(categ categorii.denumire%TYPE) RETURN VARCHAR2 IS
    id_categ categorii.id_categorie%TYPE;
    ans VARCHAR2(50);
BEGIN
    SELECT id_categorie INTO id_categ
    FROM categorii
    WHERE UPPER(denumire) = UPPER(categ);

    SELECT prenume ||  ' ' || nume INTO ans
    FROM actori act
    WHERE  (SELECT COUNT(DISTINCT f.id_film)
            FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
                          JOIN filme f ON (f.id_film = af.id_film)
                          JOIN categorie_film cf ON (cf.id_film = f.id_film)
            WHERE cf.id_categorie = id_categ AND a.id_actor = act.id_actor
            GROUP BY a.id_actor) =
            (SELECT MAX(COUNT(DISTINCT f.id_film))
             FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
                           JOIN filme f ON (f.id_film = af.id_film)
                           JOIN categorie_film cf ON (cf.id_film = f.id_film)
             WHERE cf.id_categorie = id_categ
             GROUP BY a.id_actor);

    RETURN ans;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
             DBMS_OUTPUT.PUT_LINE('Nu exista o categorie cu acest nume!');
        WHEN TOO_MANY_ROWS THEN
```

```
            DBMS_OUTPUT.PUT_LINE('Exista mai multi actori!');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Alta eroare!');
END;
PROCEDURE marire_salar(fname clienti.prenume%TYPE, lname clienti.nume%TYPE) IS
    ang tabel_ang;
    job_ang angajati.id_job%TYPE;
    id_cl clienti.id_client%TYPE;
    salar_ang angajati.salar%TYPE;
    salar angajati.salar%TYPE;

    NOT_IN_RANGE EXCEPTION;
    PRAGMA EXCEPTION_INIT(NOT_IN_RANGE, -20001);
BEGIN
    SELECT id_client INTO id_cl
    FROM clienti
    WHERE UPPER(prenume) = UPPER(fname) AND UPPER(nume) = UPPER(lname);

    SELECT DISTINCT a.id_angajat BULK COLLECT INTO ang
    FROM angajati a JOIN cinematografe c ON (a.id_cinematografe = c.id_cinema)
                    JOIN sali s ON (c.id_cinema = s.id_cinematograf)
        JOIN difuzari d ON (d.id_cinema = s.id_cinematograf AND d.id_sala = s.id_sala)
                    JOIN bilete b ON (b.id_difuzare = d.id_difuzare)
                    JOIN clienti c ON (c.id_client = b.id_client)
    WHERE c.id_client = id_cl;

    FOR i IN ang.FIRST..ang.LAST LOOP
        SELECT id_job, salar INTO job_ang, salar_ang
        FROM angajati
        WHERE id_angajat = ang(i);

        SELECT salar_maxim INTO salar
        FROM jobs
        WHERE id_job = job_ang;

        IF salar_ang * 1.05 > salar THEN RAISE NOT_IN_RANGE;
        END IF;
    END LOOP;

    FOR i IN ang.FIRST..ang.LAST LOOP
        UPDATE angajati
        SET salar = 1.05 * salar
        WHERE id_angajat = ang(i);
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('S-a realizat update-ul cu succes!');
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu numele dat!');
        WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu numele dat!');
        WHEN NOT_IN_RANGE THEN
            DBMS_OUTPUT.PUT_LINE('Noul salar nu respecta restrictiile de salar!');
```

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

BEGIN
    pachet_ex13.mareste_salar();

    DBMS_OUTPUT.PUT_LINE('Ex7');
    DBMS_OUTPUT.PUT_LINE('-------------------------');
    pachet_ex13.afisare_filme();

    DBMS_OUTPUT.PUT_LINE('Ex8');
    DBMS_OUTPUT.PUT_LINE('-------------------------');
    DBMS_OUTPUT.PUT_LINE(pachet_ex13.gaseste_actor('Comedie'));

    DBMS_OUTPUT.PUT_LINE('Ex9');
    DBMS_OUTPUT.PUT_LINE('-------------------------');
    pachet_ex13.marire_salar('Andrei', 'Ionescu');
END;
/
```

db_proiect.sql | Welcome Page

SQL Worksheet | History

Worksheet | Query Builder

```
--Ex 13
CREATE OR REPLACE PACKAGE pachet_ex13 AS
    PROCEDURE mareste_salar;
    PROCEDURE afisare_filme;
    FUNCTION gaseste_actor(categ categorii.denumire%TYPE) RETURN VARCHAR2;
    PROCEDURE marire_salar(fname clienti.prenume%TYPE, lname clienti.nume%TYPE);

    TYPE tabel_cinema IS TABLE OF cinematografe.id_cinema%TYPE
                INDEX BY BINARY_INTEGER;
    TYPE tabel_ang IS TABLE OF angajati.id_angajat%TYPE
                INDEX BY BINARY_INTEGER;
END;
/

CREATE OR REPLACE PACKAGE BODY pachet_ex13 AS
    PROCEDURE mareste_salar IS
        cinemas tabel_cinema;
    BEGIN
        SELECT DISTINCT id_cinema BULK COLLECT INTO cinemas
        FROM difuzari dif JOIN bilete bil ON (dif.id_difuzare = bil.id_difuzare)
        WHERE (SELECT COUNT(id_cinema)
               FROM cinematografe cin
               WHERE (SELECT SUM(pret * (1 - NVL(discount, 0)))
                      FROM difuzari d JOIN bilete b ON (d.id_difuzare = b.id_difuzare)
                                      JOIN clienti c ON (b.id_client = c.id_client)
                      WHERE d.id_cinema = cin.id_cinema
                      GROUP BY d.id_cinema) > (SELECT SUM(pret * (1 - NVL(discount, 0)))
                                               FROM difuzari d JOIN bilete b ON (d.id_difuzare = b.id_difuzare)
                                                               JOIN clienti c ON (b.id_client = c.id_client)
                                               WHERE d.id_cinema = dif.id_cinema
                                               GROUP BY d.id_cinema)) <= 2;
        FOR i IN cinemas.FIRST..cinemas.LAST LOOP
```

db_proiect.sql | Welcome Page

SQL Worksheet | History

Worksheet | Query Builder

```
                                        GROUP BY d.id_cinema)) <= 2;
        FOR i IN cinemas.FIRST..cinemas.LAST LOOP
            UPDATE angajati
            SET salar = (SELECT
                            CASE
                                WHEN salar * 1.1 > salar_maxim THEN salar_maxim
                                ELSE salar * 1.1
                            END
                        FROM jobs
                        WHERE jobs.id_job = angajati.id_job)
            WHERE id_cinematografe = cinemas(i);
        END LOOP;
    END;
    PROCEDURE afisare_filme IS
        CURSOR c(f_id filme.id_film%TYPE) IS
            SELECT prenume, nume, nume_personaj, castig
            FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
            WHERE id_film = f_id;
        cnt NUMBER;
        salar_total NUMBER;
    BEGIN
        FOR film IN (SELECT id_film, denumire
                     FROM filme) LOOP
            DBMS_OUTPUT.PUT_LINE('-------------------------------');
            DBMS_OUTPUT.PUT_LINE(UPPER(film.denumire));
            DBMS_OUTPUT.PUT_LINE('-------------------------------');

            cnt := 0;
            salar_total := 0;
            FOR act IN c(film.id_film) LOOP
                cnt := cnt + 1;
                IF act.nume_personaj IS NOT NULL THEN
```

db_proiect.sql | Welcome Page

SQL Worksheet | History

Worksheet | Query Builder

```
                IF act.nume_personaj IS NOT NULL THEN
                    DBMS_OUTPUT.PUT_LINE(cnt || '. ' || act.prenume || ' ' || act.nume || ' a avut rolul ' || act.nume_pe
                ELSE
                    DBMS_OUTPUT.PUT_LINE(cnt || '. ' || act.prenume || ' ' || act.nume || ' a castigat ' || NVL(act.cast
                END IF;
                salar_total := salar_total + act.castig;
            END LOOP;
            DBMS_OUTPUT.PUT_LINE('');
            DBMS_OUTPUT.PUT_LINE('Suma totala cheltuita: ' || salar_total);
        END LOOP;
    END;
    FUNCTION gaseste_actor(categ categorii.denumire%TYPE) RETURN VARCHAR2 IS
        id_categ categorii.id_categorie%TYPE;
        ans VARCHAR2(50);
    BEGIN
        SELECT id_categorie INTO id_categ
        FROM categorii
        WHERE UPPER(denumire) = UPPER(categ);

        SELECT prenume ||  ' ' || nume INTO ans
        FROM actori act
        WHERE  (SELECT COUNT(DISTINCT f.id_film)
                FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
                              JOIN filme f ON (f.id_film = af.id_film)
                              JOIN categorie_film cf ON (cf.id_film = f.id_film)
                WHERE cf.id_categorie = id_categ AND a.id_actor = act.id_actor
                GROUP BY a.id_actor) = (SELECT MAX(COUNT(DISTINCT f.id_film))
                                        FROM actori a JOIN actor_film af ON (a.id_actor = af.id_actor)
                                                      JOIN filme f ON (f.id_film = af.id_film)
                                                      JOIN categorie_film cf ON (cf.id_film = f.id_film)
                                        WHERE cf.id_categorie = id_categ
                                        GROUP BY a.id_actor);
```

db_proiect.sql | Welcome Page

SQL Worksheet | History

Worksheet | Query Builder

```
        RETURN ans;

        EXCEPTION
            WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Nu exista o categorie cu acest nume!');
            WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.PUT_LINE('Exista mai multi actori!');
            WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Alta eroare!');
    END;
    PROCEDURE marire_salar(fname clienti.prenume%TYPE, lname clienti.nume%TYPE) IS
        ang tabel_ang;
        job_ang angajati.id_job%TYPE;
        id_cl clienti.id_client%TYPE;
        salar_ang angajati.salar%TYPE;
        salar angajati.salar%TYPE;

        NOT_IN_RANGE EXCEPTION;
        PRAGMA EXCEPTION_INIT(NOT_IN_RANGE, -20001);
    BEGIN
        SELECT id_client INTO id_cl
        FROM clienti
        WHERE UPPER(prenume) = UPPER(fname) AND UPPER(nume) = UPPER(lname);

        SELECT DISTINCT a.id_angajat BULK COLLECT INTO ang
        FROM angajati a JOIN cinematografe c ON (a.id_cinematografe = c.id_cinema)
                        JOIN sali s ON (c.id_cinema = s.id_cinematograf)
                        JOIN difuzari d ON (d.id_cinema = s.id_cinematograf AND d.id_sala = s.id_sala)
                        JOIN bilete b ON (b.id_difuzare = d.id_difuzare)
                        JOIN clienti c ON (c.id_client = b.id_client)
        WHERE c.id_client = id_cl;

        FOR i IN ang.FIRST..ang.LAST LOOP
            SELECT id_job, salar INTO job_ang, salar_ang
            FROM angajati
```

db_proiect.sql | Welcome Page

SQL Worksheet | History

Worksheet | Query Builder

```
                        JOIN clienti c ON (c.id_client = b.id_client)
        WHERE c.id_client = id_cl;

        FOR i IN ang.FIRST..ang.LAST LOOP
            SELECT id_job, salar INTO job_ang, salar_ang
            FROM angajati
            WHERE id_angajat = ang(i);

            SELECT salar_maxim INTO salar
            FROM jobs
            WHERE id_job = job_ang;

            IF salar_ang * 1.05 > salar THEN RAISE NOT_IN_RANGE;
            END IF;
        END LOOP;

        FOR i IN ang.FIRST..ang.LAST LOOP
            UPDATE angajati
            SET salar = 1.05 * salar
            WHERE id_angajat = ang(i);
        END LOOP;

        DBMS_OUTPUT.PUT_LINE('S-a realizat update-ul cu succes!');
        EXCEPTION
            WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu numele dat!');
            WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu numele dat!');
            WHEN NOT_IN_RANGE THEN DBMS_OUTPUT.PUT_LINE('Noul salar nu respecta restrictiile de salar!');
            WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Alta eroare!');
    END;
END;
/
```

```
db_proiect.sql   Welcome Page
SQL Worksheet   History
Worksheet   Query Builder
            WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Alta eroare!');
        END;
    END;
    /

    BEGIN
        pachet_ex13.mareste_salar();

        DBMS_OUTPUT.PUT_LINE('Ex7');
        DBMS_OUTPUT.PUT_LINE('-------------------------');
        pachet_ex13.afisare_filme();

        DBMS_OUTPUT.PUT_LINE('Ex8');
        DBMS_OUTPUT.PUT_LINE('-------------------------');
        DBMS_OUTPUT.PUT_LINE(pachet_ex13.gaseste_actor('Comedie'));

        DBMS_OUTPUT.PUT_LINE('Ex9');
        DBMS_OUTPUT.PUT_LINE('-------------------------');
        pachet_ex13.marire_salar('Andrei', 'Ionescu');
    END;
    /

    ROLLBACK;
```

```
Script Output   Query Result
Task completed in 0.096 seconds
PL/SQL procedure successfully completed.


Rollback complete.


PL/SQL procedure successfully completed.
```

```
Dbms Output   Buffer Size: 20000
db_proiect
Ex7
-------------------------
--------------------------------
PIRATES OF THE CARIBBEAN
--------------------------------
1. Johnny Depp a avut rolul Jack Sparrow si a castigat 5000000
2. Penelope Cruz a avut rolul Angelica si a castigat 1000000
3. Sam Claflin a avut rolul Philip si a castigat 200000
4. Margot Robbie a avut rolul Gillette si a castigat 20000

Suma totala cheltuita: 6220000
--------------------------------
MURDER ON THE ORIENT EXPRESS
--------------------------------
1. Penelope Cruz a castigat 10000
2. Sam Claflin a avut rolul Hercule Poirot si a castigat 10000
3. Leonardo DiCaprio a avut rolul Young Policeman si a castigat 70000

Suma totala cheltuita: 90000
--------------------------------
THE LORD OF THE RINGS
--------------------------------
1. Orlando Bloom a avut rolul Legolas si a castigat 300000
2. Alan Howard a avut rolul Voice of the Ring si a castigat 900000

Suma totala cheltuita: 1200000
--------------------------------
THE GODFATHER
--------------------------------
1. Marlon Brando a avut rolul Don Vito Corleone si a castigat 7000000
2. Al Pacino a avut rolul Michael Corleone si a castigat 6000000

Suma totala cheltuita: 13000000
--------------------------------
SCHINDLER LIST
--------------------------------
1. Liam Neeson a avut rolul Oskar Schindler si a castigat 3000000
```

```
db_proiect.sql   Welcome Page
SQL Worksheet   History
Worksheet   Query Builder
            WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Alta eroare!');
        END;
    END;
    /

    BEGIN
        pachet_ex13.mareste_salar();

        DBMS_OUTPUT.PUT_LINE('Ex7');
        DBMS_OUTPUT.PUT_LINE('-------------------------');
        pachet_ex13.afisare_filme();

        DBMS_OUTPUT.PUT_LINE('Ex8');
        DBMS_OUTPUT.PUT_LINE('-------------------------');
        DBMS_OUTPUT.PUT_LINE(pachet_ex13.gaseste_actor('Comedie'));

        DBMS_OUTPUT.PUT_LINE('Ex9');
        DBMS_OUTPUT.PUT_LINE('-------------------------');
        pachet_ex13.marire_salar('Andrei', 'Ionescu');
    END;
    /

    ROLLBACK;
```

```
Script Output   Query Result
Task completed in 0.096 seconds
PL/SQL procedure successfully completed.


Rollback complete.


PL/SQL procedure successfully completed.
```

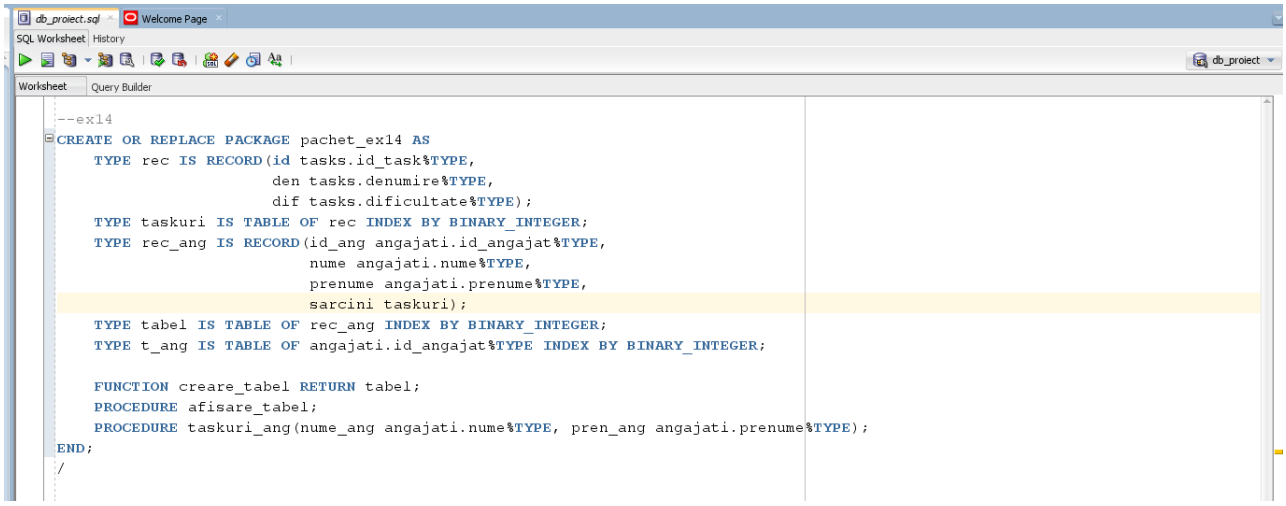
```
Dbms Output   Buffer Size: 20000
db_proiect
Suma totala cheltuita: 1200000
--------------------------------
THE GODFATHER
--------------------------------
1. Marlon Brando a avut rolul Don Vito Corleone si a castigat 7000000
2. Al Pacino a avut rolul Michael Corleone si a castigat 6000000

Suma totala cheltuita: 13000000
--------------------------------
SCHINDLER LIST
--------------------------------
1. Liam Neeson a avut rolul Oskar Schindler si a castigat 3000000
2. Caroline Goodall a avut rolul Emilie Schindler si a castigat 1000000

Suma totala cheltuita: 4000000
--------------------------------
INCEPTION
--------------------------------
1. Leonardo DiCaprio a avut rolul Cobb si a castigat 400000
2. Orlando Bloom a castigat 5000

Suma totala cheltuita: 405000
--------------------------------
THE WOLF OF WALL STREET
--------------------------------
1. Leonardo DiCaprio a avut rolul Jordan Belfort si a castigat 4000000
2. Margot Robbie a avut rolul Naomi Lapaglia si a castigat 800000

Suma totala cheltuita: 4800000
Ex8
-------------------------
Leonardo DiCaprio
Ex9
-------------------------
S-a realizat update-ul cu succes!
```

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

Am creat un pachet in care am folosit urmatorul tip de date: un tabel indexat cu elemente de tip record, in care elemtele de tip record contin un tabel indexat cu elemente de tip record.
Am introdus in acest tabel informatii despre angajati. Un element din tabel contine id-ul unui angajat, numele si prenumele acestuia si un tabel indexat in care se afla informatii despre taskurile pe care le are de facut angajatul respectiv.

```
CREATE OR REPLACE PACKAGE pachet_ex14 AS
    TYPE rec IS RECORD(id tasks.id_task%TYPE,
                       den tasks.denumire%TYPE,
                       dif tasks.dificultate%TYPE);
    TYPE taskuri IS TABLE OF rec INDEX BY BINARY_INTEGER;
    TYPE rec_ang IS RECORD(id_ang angajati.id_angajat%TYPE,
                           nume angajati.nume%TYPE,
                           prenume angajati.prenume%TYPE,
                           sarcini taskuri);
    TYPE tabel IS TABLE OF rec_ang INDEX BY BINARY_INTEGER;
    TYPE t_ang IS TABLE OF angajati.id_angajat%TYPE INDEX BY BINARY_INTEGER;

    FUNCTION creare_tabel RETURN tabel;
    PROCEDURE afisare_tabel;
    PROCEDURE taskuri_ang(nume_ang angajati.nume%TYPE, pren_ang angajati.prenume%TYPE);
END;
/

CREATE OR REPLACE PACKAGE BODY pachet_ex14 AS
    FUNCTION creare_tabel RETURN tabel IS
        aux taskuri;
        ang t_ang;
        ans tabel;
        nume_ang angajati.nume%TYPE;
        pren_ang angajati.prenume%TYPE;
    BEGIN
        SELECT id_angajat BULK COLLECT INTO ang
        FROM angajati;

        FOR i IN ang.FIRST..ang.LAST LOOP
            SELECT t.id_task, denumire, dificultate BULK COLLECT INTO aux
            FROM task_ang ta JOIN tasks t ON (ta.id_task = t.id_task)
            WHERE id_angajat = ang(i);

            SELECT nume, prenume INTO nume_ang, pren_ang
            FROM angajati
            WHERE id_angajat = ang(i);

            ans(i).id_ang := ang(i);
            ans(i).nume := nume_ang;
            ans(i).prenume := pren_ang;
            ans(i).sarcini := aux;
        END LOOP;

        RETURN ans;
    END;

    PROCEDURE afisare_tabel IS
        ans tabel;
    BEGIN
        ans := creare_tabel;
        FOR i IN ans.FIRST..ans.LAST LOOP
            DBMS_OUTPUT.PUT_LINE(ans(i).id_ang || ' ' || ans(i).nume
```

```
            || ' ' || ans(i).prenume);
            DBMS_OUTPUT.PUT_LINE('Taskuri: ');

            FOR j in ans(i).sarcini.FIRST..ans(i).sarcini.LAST LOOP
                DBMS_OUTPUT.PUT_LINE(ans(i).sarcini(j).id || ' '
                || ans(i).sarcini(j).den || ' ' || ans(i).sarcini(j).dif);
            END LOOP;
            DBMS_OUTPUT.PUT_LINE('-------------------------');
        END LOOP;
    END;

    PROCEDURE taskuri_ang(nume_ang angajati.nume%TYPE, pren_ang angajati.prenume%TYPE) IS
        ans tabel;
    BEGIN
        ans := creare_tabel;
        FOR i IN ans.FIRST..ans.LAST LOOP
            IF UPPER(ans(i).nume) = UPPER(nume_ang) AND
               UPPER(ans(i).prenume) = UPPER(pren_ang) THEN

                DBMS_OUTPUT.PUT_LINE(ans(i).id_ang || ' ' || ans(i).nume || ' '
                                     || ans(i).prenume);
                DBMS_OUTPUT.PUT_LINE('Taskuri: ');

                FOR j in ans(i).sarcini.FIRST..ans(i).sarcini.LAST LOOP
                    DBMS_OUTPUT.PUT_LINE(ans(i).sarcini(j).id || ' ' ||
                    ans(i).sarcini(j).den || ' ' || ans(i).sarcini(j).dif);
                END LOOP;
                DBMS_OUTPUT.PUT_LINE('-------------------------');
            END IF;
        END LOOP;
    END;
END;
/
BEGIN
    pachet_ex14.taskuri_ang('Popescu', 'IOn');
    DBMS_OUTPUT.PUT_LINE('');
    pachet_ex14.afisare_tabel();
END;
/
```



db_proiect.sql | Welcome Page

SQL Worksheet | History

db_proiect

Worksheet | Query Builder

```oracle_plsql
CREATE OR REPLACE PACKAGE BODY pachet_ex14 AS
    FUNCTION creare_tabel RETURN tabel IS
        aux taskuri;
        ang t_ang;
        ans tabel;
        nume_ang angajati.nume%TYPE;
        pren_ang angajati.prenume%TYPE;
    BEGIN
        SELECT id_angajat BULK COLLECT INTO ang
        FROM angajati;

        FOR i IN ang.FIRST..ang.LAST LOOP
            SELECT t.id_task, denumire, dificultate BULK COLLECT INTO aux
            FROM task_ang ta JOIN tasks t ON (ta.id_task = t.id_task)
            WHERE id_angajat = ang(i);

            SELECT nume, prenume INTO nume_ang, pren_ang
            FROM angajati
            WHERE id_angajat = ang(i);

            ans(i).id_ang := ang(i);
            ans(i).nume := nume_ang;
            ans(i).prenume := pren_ang;
            ans(i).sarcini := aux;
        END LOOP;

        RETURN ans;
    END;
```

db_proiect.sql | Welcome Page

SQL Worksheet | History

db_proiect

Worksheet | Query Builder

```oracle_plsql
    PROCEDURE afisare_tabel IS
        ans tabel;
    BEGIN
        ans := creare_tabel;
        FOR i IN ans.FIRST..ans.LAST LOOP
            DBMS_OUTPUT.PUT_LINE(ans(i).id_ang || ' ' || ans(i).nume || ' ' || ans(i).prenume);
            DBMS_OUTPUT.PUT_LINE('Taskuri: ');

            FOR j in ans(i).sarcini.FIRST..ans(i).sarcini.LAST LOOP
                DBMS_OUTPUT.PUT_LINE(ans(i).sarcini(j).id || ' ' || ans(i).sarcini(j).den || ' ' || ans(i).sarcini(j).dif);
            END LOOP;
            DBMS_OUTPUT.PUT_LINE('-------------------------');
        END LOOP;
    END;

    PROCEDURE taskuri_ang(nume_ang angajati.nume%TYPE, pren_ang angajati.prenume%TYPE) IS
        ans tabel;
    BEGIN
        ans := creare_tabel;
        FOR i IN ans.FIRST..ans.LAST LOOP
            IF UPPER(ans(i).nume) = UPPER(nume_ang) AND UPPER(ans(i).prenume) = UPPER(pren_ang) THEN
                DBMS_OUTPUT.PUT_LINE(ans(i).id_ang || ' ' || ans(i).nume || ' ' || ans(i).prenume);
                DBMS_OUTPUT.PUT_LINE('Taskuri: ');

                FOR j in ans(i).sarcini.FIRST..ans(i).sarcini.LAST LOOP
                    DBMS_OUTPUT.PUT_LINE(ans(i).sarcini(j).id || ' ' || ans(i).sarcini(j).den || ' ' || ans(i).sarcini(j).dif);
                END LOOP;
                DBMS_OUTPUT.PUT_LINE('-------------------------');
            END IF;
        END LOOP;
    END;
END;
```

```
                DBMS_OUTPUT.PUT_LINE(ans(i).id_ang || ' ' || ans(i).nume || ' ' || ans(
                DBMS_OUTPUT.PUT_LINE('Taskuri: ');

                FOR j in ans(i).sarcini.FIRST..ans(i).sarcini.LAST LOOP
                    DBMS_OUTPUT.PUT_LINE(ans(i).sarcini(j).id || ' ' || ans(i).sarcini(
                END LOOP;
                DBMS_OUTPUT.PUT_LINE('-------------------------');
            END IF;
        END LOOP;
    END;
END;
/

BEGIN
    pachet_ex14.taskuri_ang('Popescu', 'IOn');
    DBMS_OUTPUT.PUT_LINE('');
    pachet_ex14.afisare_tabel();
END;
/
```

```
Table AUX dropped.


Rollback complete.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

```
1 Popescu Ion
Taskuri:
2 Vinde bilete 2
-------------------------
23 Popescu Ion
Taskuri:
6 Imparte task-uri 4
-------------------------

1 Popescu Ion
Taskuri:
2 Vinde bilete 2
-------------------------
2 Georgescu Ana
Taskuri:
6 Imparte task-uri 4
-------------------------
3 Popescu Vasile
Taskuri:
4 Proiecteza filmul 3
5 Actualizeaza site-ul 2
-------------------------
4 Ionel Ion
Taskuri:
1 Curata sala 1
3 Verifica bilete la intrare 1
-------------------------
5 Vasilica Gigel
Taskuri:
6 Imparte task-uri 4
-------------------------
6 Ionescu Maria
Taskuri:
5 Actualizeaza site-ul 2
-------------------------
7 Steven King
Taskuri:
```