

UNIVERSITATEA BUCUREŞTI
Facultatea de Matematică și Informatică
Programul de studii: *Informatică*



**UNIVERSITATEA DIN
BUCUREŞTI**
— VIRTUTE ET SAPIENTIA —

Disciplina: Sisteme de Gestiune a Bazelor de Date

GESTIUNEA CINEMATOGRAFELOR

COORDONATOR ȘTIINȚIFIC:
Lect.dr. Gabriela MIHAI
Asist. Ioana-Delia GHERGHE

Student:
Ştefania RÎNCU

Grupa 232

BUCUREŞTI
2022

Cuprins

| | | |
|------|--|-----|
| 1. | Prezentarea pe scurt a bazei de date | 3 |
| 1.1 | Prezentarea constrângerilor impuse asupra modelului..... | 4 |
| 1.2 | Descrierea entităților, incluzând precizarea cheii primare | 5 |
| 1.3 | Descrierea relațiilor..... | 6 |
| 1.4 | Descrierea atributelor..... | 9 |
| 1.5. | Schemele relaționale | 15 |
| 2. | Diagrama Entitate-Relație | 16 |
| 3. | Diagrama Conceptuală..... | 17 |
| 4. | Crearea tabelelor | 18 |
| 5. | Inserarea datelor în tabele | 28 |
| 6. | Problemă subprogram stocat independent + colecții | 58 |
| 7. | Problemă subprogram stocat independent + cursoare..... | 65 |
| 8. | Problemă funcție stocată independent + excepții + 3 tabele | 76 |
| 9. | Problemă procedură stocată independent + excepții + 5 tabele | 83 |
| 10. | Trigger LMD la nivel de comandă..... | 93 |
| 11. | Trigger LMD la nivel de linie..... | 97 |
| 12. | Trigger LDD la nivel de schemă..... | 111 |
| 13. | Pachet ce include toate obiectele definite în proiect | 113 |
| 14. | Pachet complex | 130 |

1. Prezentarea pe scurt a bazei de date

Bazele de date sunt foarte utile în gestionarea centralizată și în timp real a informațiilor din diverse domenii de activitate. În cadrul acestei lucrări se va prezenta proiectarea unei baze de date care să se ocupe cu gestiunea cinematografelor ce aparțin companiei „Cinema City”. În acest domeniu o bază de date ar fi utilă în gestionarea informațiilor pentru organizarea procesului de administrare a locațiilor unde se vor viziona filmele incluzând și personalul necesar pentru deservirea acestora. De asemenea, este esențial să știm despre un cinematograf ce filme rulează în cadrul său și cine a fost implicat în producția acestora (regzori, actori).

Pentru proiectarea bazei de date am impus o serie de reguli necesare pentru buna funcționare a acesteia. Regulile de funcționare ale acestui model sunt următoarele:

1. În diagramă sunt incluse și cinematograf care nu rulează filme
2. Cinematografele rulează filme din întreaga lume
3. În acest model se presupune că ca există și cinematografe în care nu a lucrat și nu lucrează nimeni
4. Angajații pot fi: casieri (se ocupă cu vânzarea de bilete, rezervări), plasatori (se ocupă cu direcționarea corectă a clienților în cinema, verificarea biletelor de acces, implementarea politicii referitoare la restricțiile de vârstă), barman cafenea sau barman (se ocupă cu aranjarea, prepararea și servirea produselor de la bar), agent de curățare (se ocupă cu menținerea curățeniei), ingineri sunet, ingineri tehnic, manager
5. Joburile sunt: casier, plasator, barman cafenea, barman, agent de curățare, inginer sunet, inginer tehnic, manager
6. Actorii sunt persoanele care joacă într-un film, fiind gestionați actorii mai cunoscuți, despre care știm că vor stârni interesul publicului
7. Pentru actori și regzori vrem să știm țara natală
8. Ca și premii pentru actori și regzori se vor număra doar premiile importante
9. Un cinematograf se poate afla în cadrul unui singur complex comercial
10. În diagramă sunt inclusi doar angajații care lucrează sau au lucrat într-un cinematograf din cele incluse în baza de date
11. În diagramă sunt incluse și filme pentru care nu avem detalii despre difuzare sau rulare
12. În diagramă sunt incluse și locații în care nu există cinematografe
13. În diagramă sunt incluse doar biletele care au fost achiziționate pentru o anumita difuzare din baza de date

1.1 Prezentarea constrângerilor impuse asupra modelului

Deoarece această diagramă nu oglindește perfect realitatea am introdus o serie de constrângeri pe care le-am aplicat asupra modelului, astfel încât datele pe care le voi introduce și relațiile descrise ulterior să fie coerente.

Restricțiile și regulile impuse asupra modelului acesta sunt următoarele:

1. Un cinematograf se poate afla într-o singură locație
2. Un angajat nu poate lucra simultan în cadrul mai multor cinematografe ce aparțin de „Cinema City” și nu poate avea simultan mai multe joburi
3. Se vor memora date despre angajații care au schimbat un job sau au lucrat la mai multe cinematografe, pentru a se reține un istoric al angajaților
4. Un cinematograf poate să nu aibă niciun angajat (abia a fost introdus în baza de date)
5. În diagramă sunt incluse și cinematografe care nu rulează filme
6. Joburile sunt universale, aşadar toate cinematografele au aceleași joburi disponibile pentru angajați
7. La o locație nu putem avea mai multe cinematografe, dar putem să nu avem niciunul
8. Un film poate fi îndrumat de mai mulți regizori
9. Un regizor trebuie să regizeze minim un film pentru a fi recunoscut ca regizor
10. Un film poate să nu fie îndrumat de niciun regizor, dacă este film animat
11. Un film poate exista fără niciun actor, dacă este un film animat
12. Un actor trebuie să apară în minim un film pentru a fi recunoscut ca actor
13. O sală poate avea minim un loc, altfel nu au unde să se așeze spectatorii și nu putem vorbi despre o sală de difuzare a filmelor
14. Pe bilet apare codul difuzării căreia îl este asignat
15. Atât actorii, cât și regizorii pot avea o singură țară natală
16. Pentru o difuzare nu este obligatoriu să se afle bilete cumpărate înregistrate în baza de date, existând cazul în care nicio persoană nu a cumpărat un bilet la acea difuzare de film

1.2 Descrierea entităților, incluzând precizarea cheii primare

Baza de date proiectată în cadrul acestei lucrări conține entități ce desemnează componente care ajută la favorizarea bunei funcționări a unui cinematograf (spre exemplu sălile din acesta sau angajații care muncesc acolo).

Entitățile folosite în realizarea acestui proiect sunt:

1. **CINEMATOGRAF**(entitate independentă) – loc cu săli de spectacole destinate proiecției filmelor cinematografice în fața publicului spectator, cheia primară este „**cod_cinema**”
2. **SALA** (entitate dependentă) – încăpere în care sunt proiectate filmele în fața spectatorilor, nu poate exista fără să aparțină de un cinematograf, cheia primară este compusă din „**cod_cinema**” (cheie străină, ce aparține entității CINEMATOGRAF) și „**nr_sala**”
3. **ANGAJAT** (entitate independentă) – persoană care lucrează în cadrul cinematografului și care servește la buna funcționare a acestuia, cheia primară este „**cod_angajat**”
4. **JOB** (entitate independentă) – meserie pe care o poate avea un angajat, cheia primară este „**cod_job**”
5. **LOCATIE** (entitate independentă) – adresă completă, descriindu-se poziționarea geografică unde se află un cinematograf, cheia primară este „**id_locatie**”
6. **FILM** (entitate independentă) – înregistrare ce cuprinde momente succesive constând în imagini statice sau în mișcare, precum și sunete, cu scopul de a fi mai apoi proiectată pe un ecran, cheia primară este „**cod_film**”
7. **BILET_CUMPARAT** (entitate independentă) - bucată mică de hârtie sau de carton imprimat care dă dreptul la intrarea, ocuparea unui loc într-o sală și vizionarea unui film, cheia primară este „**cod_bilet**”
8. **DIFUZARE** (entitate dependentă) – acțiunea de a proiecta un film, într-o sală dintr-un cinematograf, cheia primară este „**cod_difuzare**”
9. **ACTOR** (entitate independentă) - persoană care interpretează diferite roluri în filme, cheia primară este „**id_actor**”
10. **REGIZOR** (entitate independentă) – specialist calificat profesional care se ocupă cu regia filmelor, cheia primară este „**id_regizor**”
11. **TARA** (entitate independentă) – regiune geografică ce reține numele unei țări existente pe globul pământesc, cheia primară este „**id_tara**”

1.3 Descrierea relațiilor

Descrierea relațiilor care au fost folosite în cadrul acestui proiect, respectând regulile modelului descrise mai sus și constrângerile impuse pentru o bună funcționare a sistemului de gestiune a bazei de date.

Detalierea acestor relații, incluzând precizarea cardinalității acestora:

1. **detine** (CINEMATOGRAF, SALA) – relație de dependență ce indică faptul că o sală aparține de un cinematograf, deoarece o sală (după cum am definit-o, ca fiind special creață pentru proiectarea filmelor) nu poate exista fără un cinematograf. Cardinalitățile sunt deduse din întrebările: „Câte săli poate detine un cinematograf? **M.** Câte săli trebuie să dețină un cinematograf? Cel puțin **1**, deoarece altfel nu ar avea unde să difuzeze filme. În câte cinematografe se poate afla o sală? În câte cinematografe trebuie să se afle o sală? În **1** cinematograf (pentru ambele întrebări).”
=>**Cardinalitate:M(1)** (Cinematograf – Sală), **1** (Sală – Cinematograf)
2. **ruleaza** (CINEMATOGRAF, FILM) – relație ce indică faptul că un film poate fi rulat într-un cinematograf. Cardinalitățile sunt deduse din întrebările: „Câte filme pot rula într-un cinematograf? **M.** Câte filme trebuie să ruleze într-un cinematograf? **0**, încă în această diagramă sunt incluse și cinematografe în care nu rulează filme. În câte cinematografe poate rula un film? **M.** În câte cinematografe trebuie să ruleze un film? **0**, dacă încă nu avem detalii despre rulare.”
=>**Cardinalitate:M(0)** (Cinematograf – Film), **M(0)** (Film- Cinematograf)
3. **are_loc** (SALA, DIFUZARE) – relație ce indică faptul că fiecare difuzare din cinematograf are loc într-o sală. Cardinalitățile sunt deduse din întrebările: „Câte difuzări pot avea loc într-o sală? **M.** Câte difuzări trebuie să aibă loc într-o sală? **0**, deoarece există posibilitatea să nu difuzeze nimic în acel moment. În câte săli poate avea loc o difuzare? În câte săli trebuie să aibă loc o difuzare? **1** (pentru ambele întrebări) deoarece pentru a vorbi despre o difuzare, aceasta trebuie să aibă loc într-o sală.”
=>**Cardinalitate:M(0)** (Sală - Difuzare), **1** (Difuzare - Sală)
4. **se_proiecteaza**(FILM, DIFUZARE) – relație ce indică faptul că fiecare film se proiectează în cadrul unei difuzări. Cardinalitățile sunt deduse din întrebările: „La câte difuzări se poate proiecta un film? **M.** La câte difuzări trebuie să se proiecteze un film? **0**, deoarece în această diagramă pot fi incluse și filme ce nu au fost difuzate încă în vreun cinematograf din baza de date. Câte filme se pot proiecta la o difuzare? Câte filme trebuie să se proiecteze la o difuzare? **1** (pentru ambele întrebări) deoarece pentru a vorbi despre o difuzare, aceasta trebuie să prezinte un film.”
=>**Cardinalitate:M(0)** (Film - Difuzare), **1** (Difuzare - Film)
5. **se_intra** (BILET_CUMPARAT, DIFUZARE) – relație ce indică faptul că pe baza unui bilet cumpărat se intră la o difuzare de film. Cardinalitățile sunt deduse din întrebările: „La câte difuzări se poate intra pe baza unui bilet cumpărat? La câte

difuzări trebuie să se intre pe baza unui bilet cumpărat? **1** (pentru ambele întrebări). Cu câte bilete cumpărate se poate intra la o difuzare? **M** (cumpărate de mai multe persoane pentru aceeași difuzare). Cu câte bilete cumpărate trebuie să se intre la o difuzare? **0**, deoarece există posibilitatea să nu cumpere nimeni bilet.”

=>**Cardinalitățile:1** (Bilet_cumparat - Difuzare), **M(0)** (Difuzare – Bilet_cumparat)

6. **regizeaza** (REGIZOR, FILM) – relație ce ne indică faptul că filmele sunt îndrumate de către regizori. Cardinalitățile sunt deduse din întrebările: „Câte filme pot fi regizate de un regizor? **M**. Câte filme trebuie să fie regizate de un regizor? Cel puțin **1**, întrucât am stabilit că un regizor trebuie să aibă măcar un film regizat pentru a fi recunoscut. Câți regizori pot să regizeze un film? **M**. Câți regizori trebuie să regizeze un film? **0**, deoarece putem vorbi despre un film animat.”
=>**Cardinalitățile:M(0)** (Film - Regizor), **M(1)** (Regizor - Film)
7. **joaca** (ACTOR, FILM) – relație ce indică faptul că actorii trebuie să aibă filme în care să joace pentru a fi recunoscuți, iar filmele au nevoie de actori care să dea contur acțiunii. Cardinalitățile sunt deduse din întrebările: „În câte filme poate să joace un actor? **M**. În câte filme trebuie să joace un actor? Cel puțin **1**, deoarece am stabilit că pentru a fi recunoscut un actor trebuie să joace în cel puțin un film. Câți actori pot juca într-un film? **M**. Câți actori trebuie să joace într-un film? **0**, deoarece putem vorbi despre un film animat.”
=>**Cardinalitățile:M(1)** (Actor – Film), **M(0)** (Film – Actor)
8. **a_lucrat** (CINEMATOGRAF, ANGAJAT, JOB) – relație de tip 3 ce indică fapul că este posibil ca un angajat să fi lucrat în trecut pe alte joburi sau în cadrul altor cinematografe. Cardinalitățile sunt deduse din întrebările: „Câți angajați se poate să fi lucrat într-un cinematograf? Câți angajați se poate să fi lucrat pe un anumit job? **M**. Câți angajați trebuie să fi lucrat într-un cinematograf? Câți angajați trebuie să fi lucrat pe un anumit job? **0**, întrucât există situația în care nimeni nu a demisionat de pe jobul sau de la cinematograful curent. Câte joburi se poate să se fi lucrat în cadrul unui cinematograf? Câte joburi se poate să se fi lucrat de un angajat? **M**. Câte joburi trebuie să se fi lucrat în cadrul unui cinematograf? Câte joburi trebuie să se fi lucrat de un angajat? **0**, întrucât există posibilitatea în care niciun angajat nu a demisionat de pe niciun job dintr-un cinematograf. În câte cinematografe se poate să se fi lucrat un angajat? În câte cinematografe se poate să se fi lucrat un anumit job? **M**. În câte cinematografe trebuie să se fi lucrat un angajat? În câte cinematografe trebuie să se fi lucrat un anumit job? **0**, întrucât există posibilitatea în care niciun angajat nu a demisionat de pe niciun job dintr-un cinematograf.”
=>**Cardinalitățile:M(0)** (Cinematograf - Angajat, Job - Angajat), **M(0)** (Angajat – Job, Cinematograf - Job), **M(0)**(Angajat – Cinematograf, Cinematograf - Job)
9. **lucreaza_in** (ANGAJAT, CINEMATOGRAF) – relație ce indică faptul că un angajat lucrează în cadrul unui cinematograf. Cardinalitățile sunt deduse din întrebările: „Câți angajați pot lucra într-un cinematograf? **M**. Câți angajați trebuie să lucreze într-un cinematograf? **0**, întrucât am stabilit că un cinematograf poate exista fără nici un angajat. În câte cinematografe poate să lucreze un angajat? În câte cinematografe trebuie să lucreze un angajat? **1**, întrucât am stabilit că în diagramă apar doar angajații

care lucrează în cadrul unui cinematograf, iar un angajat poate lucra în cadrul unui singur cinematograf la un moment de timp.”

=>**Cardinalitățile:M(0)** (Cinematograf - Angajat), **1** (Angajat – Cinematograf)

10. **are** (ANGAJAT, JOB) – relație ce indică faptul că un angajat are un anumit job.

Cardinalitățile sunt deduse din întrebările: „Câți angajați pot avea un job? **M**. Câți angajați trebuie să aibă un job? **0**. Câte joburi poate avea un angajat? Câte joburi trebuie să aibă un angajat? **1**, întrucât am stabilit că în diagramă apar doar angajații care lucrează în cadrul unui cinematograf, deci au un job, iar un angajat poate avea un singur job la un moment de timp.”

=>**Cardinalitățile:M(0)** (Job - Angajat), **1** (Angajat – Job)

11. **are_superior** (ANGAJAT, ANGAJAT) – relație ce indică faptul că un angajat poate avea un superior. Cardinalitățile sunt deduse din întrebările: „Câți angajați pot avea asignați un superior? **M**. Câți angajați trebuie să aibă asignați un superior? **0**. Câți superiori poate avea asignat un angajat? **1**. Câți superiori trebuie să aibă asignat un angajat? **0**.”

=>**Cardinalitățile:M(0)** (Angajat(Superior) – Angajat),

1(0) (Angajat – Angajat(Superior))

12. **se_afla**(CINEMATOGRAF, LOCATIE) – relație ce indică poziționarea unui cinematograf, arătându-ne adresa acestuia. Cardinalitățile sunt deduse din întrebările: „Câte cinematografe se pot afla într-o locație? **1**. Câte cinematografe trebuie să se afle la o locație? **0**, întrucât într-un punct anume se poate să nu fie niciun cinematograf construit. În câte locații se poate afla un cinematograf? În câte locații trebuie să se afle un cinematograf? **1** locație (pentru ambele întrebări), deoarece un cinematograf nu se poate afla simultan în mai multe locații, iar pentru a exista atunci trebuie să se regăsească undeva.”

=>**Cardinalitățile:1(0)** (Locație - Cinematograf), **1** (Cinematograf – Locație)

13. **se_afla**(LOCATIE, TARA) – relație ce indică poziționarea unei locații, indicându-ne țara. Cardinalitățile sunt deduse din întrebările: „Câte locații se pot afla într-o țară? **M**. Câte locații trebuie să se afle într-o țară? **0**. În câte țări se poate afla o locație? În câte țări trebuie să se afle o locație? **1**.”

=>**Cardinalitățile:1** (Locație - Țară), **M(0)** (Țară – Locație)

14. **provine**(ACTOR, TARA) – relație ce indică țara de proveniență a unui actor. Cardinalitățile sunt deduse din întrebările: „Câți actori pot proveni dintr-o țară? **M**. Câți actori trebuie să provină dintr-o țară? **0**. Din câte țări poate să provină un actor? Din câte țări trebuie să provină un actor? **1**.”

=>**Cardinalitățile:1** (Actor - Țară), **M(0)** (Țară – Actor)

15. **provine**(REGIZOR, TARA) – relație ce indică țara de proveniență a unui regizor. Cardinalitățile sunt deduse din întrebările: „Câți regizori pot proveni dintr-o țară? **M**. Câți regizori trebuie să provină dintr-o țară? **0**. Din câte țări poate să provină un regizor? Din câte țări trebuie să provină un regizor? **1**.”

=>**Cardinalitățile:1** (Regizor - Țară), **M(0)** (Țară – Regizor)

1.4 Descrierea atributelor

Descrierea atributelor s-a făcut ținând cont și de tipul de date, eventualele constrângeri, valori implicate, valori posibile ale acestora.

Atributele ce se regăsesc în cadrul acestui proiect sunt:

1. CINEMATOGRAF

- a. **cod_cinema** - număr de maxim 4 cifre (NUMBER(4)), care reprezintă codul cinematografului, fiind unic pentru fiecare cinematograf în parte (exemplu: 132116). Aceasta reprezintă cheia primară a entității CINEMATOGRAF
- b. **nume_cinema** – sir de maxim 40 de caractere (litere mari, mici, „ ”) (VARCHAR2(40)), care precizează numele cinematografului, aşa cum se regăsește în acte (exemplu: „Cinema City Brasov”). Acesta este diferit de „null”, deoarece un cinematograf care există obligatoriu are un nume
- c. **id_locatie** – număr de maxim 4 cifre (NUMBER (4)), care precizează id-ul locației unde se regăsește cinematograful (exemplu: 113). Acesta este diferit de „null”, deoarece dacă un cinematograf există atunci acesta se regăsește într-o locație. Acest id are corespondent în tabela LOCATIE („id_locatie”)
- d. **complex** - sir de maxim 40 de caractere (VARCHAR2(40)), care dă denumirea complexului comercial în care cinematograful se regăsește (litere mari, mici, „ ”) (exemplu: ”Afi Palace Cotroceni”)
- e. **telefon** - sir de maxim 20 de caractere(VARCHAR2(20)), precizează numărul de telefon la care poate fi contactat cinematograful (exemplu: 0741.321.133)
- f. **email** – sir de maxim 25 de caractere(litere mari, mici, „_”, „@”, „”, cifre)(CHAR(25)), precizează emailul la care poate fi contactat cinematograful (exemplu: happy.cinema@gmail.com). Aceasta trebuie să fie diferit de valoarea „null” și este unic
- g. **data_desch** - este o dată calendaristică (DATE) care arată data în care s-a deschis cinematograful respectiv (exemplu: 3 mai 2022). Aceasta este diferită de „null” și are valoarea „sysdate” by default

2. SALA

- a. **cod_cinema** - număr de maxim 4 cifre (NUMBER(4)), care reprezintă codul cinematografului în care se află sala respectivă, codul fiind unic. Aceasta reprezintă o cheie primară externă a entității SALA și are corespondent în tabela CINEMATOGRAF, având o valoare diferită de „null”
- b. **nr_sala** - număr de maxim 2 cifre (NUMBER(2)), mai mic decât 40 și mai mare ca 0, care reprezintă numărul sălii. Aceasta reprezintă cheia primară a entității SALA, împreună cu atributul „cod_cinema”

c. **tip** –șir de maxim 10 caractere (VARCHAR2(10)) care definește tipul unei săli de cinema (exemplu: VIP)

3. FILM

- a. **cod_film** - număr de maxim 4 cifre (NUMBER(4)), care reprezintă codul filmului, acesta fiind unic pentru fiecare film în parte (de exemplu: 1652). Acesta reprezintă cheia primară a entității FILM
- b. **nume_film** - șir de maxim 25 de caractere (litere mari, mici, „ ”, cifre) (VARCHAR 2(25)), care precizează numele complet al filmului, aşa cum se regăsește pe afișul filmului (exemplu: Uncharted). Acesta este diferit de „null”, deoarece un film care există obligatoriu are un nume și este unic (în asociere cu anul apariției)
- c. **gen_princ** - șir de maxim 20 de caractere (VARCHAR2 (20)), care precizează genul principal al filmului (litere mari, mici, „ ”) (exemplu: Actiune). Acesta este diferit de „null”, deoarece orice film aparține unei tipologii
- d. **durata** – număr de maxim 3 cifre (NUMBER(3)), care precizează durata filmului (în minute) (exemplu: 93). Acesta are valoarea implicită 0 minute
- e. **rating** - număr cu virgulă (de tipul „cifră.cifră”), float (NUMBER(2, 1)), care precizează aprecierea publicului (exemplu: 7.5), acesta are o valoare din intervalul 0 - 10
- f. **an_aparitie** - număr de 4 cifre, int (NUMBER(4)), care reprezintă anul în care a apărut filmul respectiv (exemplu: 2020). Acesta este diferit de „null” și mai mare decât 0, deoarece pentru ca un film să apară la cinematograf atunci trebuie să aibă un an în care a fost difuzat prima dată și unic

4. DIFUZARE

- a. **cod_difuzare** - număr de maxim 4 cifre (NUMBER(4)), care reprezintă codul difuzării, acesta fiind unic pentru fiecare difuzare în parte în parte (de exemplu: 1655). Acesta reprezintă cheia primară a entității DIFUZARE
- b. **cod_film** - număr de maxim 4 cifre (NUMBER(4)), care reprezintă codul filmului (de exemplu: 1652). Acesta este diferit de „null” și referențiază cheia primară a entității FILM
- c. **nr_sala** - număr de maxim 2 cifre (NUMBER(2)), mai mic sau egal decât 40 și mai mare ca 0, care reprezintă numărul sălii. Acesta este diferit de „null”
- d. **cod_cinema** – număr de maxim 4 cifre (NUMBER(4)), care reprezintă codul cinematografului în care se află difuzarea respectivă. Acesta are o valoare diferită de „null” și împreună cu atributul „nr_sala” referențiază cheia primară a entității SALA
- e. **data_inc** - este o dată calendaristică (DATE) care arată data în care are loc difuzarea respectivă (exemplu: 3 mai 2022). Aceasta este diferită de „null” și are valoarea „sysdate” by default

f. **ora_inc** - număr de maxim 2 cifre (NUMBER(2)), care reprezintă ora la care începe difuzarea respectivă (exemplu: 7). Aceasta este diferit de „null”, mai mare ca 0 și mai mic decât 24

5. BILET_CUMPARAT

- a. **cod_bilet** – număr de maxim 4 cifre (NUMBER(4)), care reprezintă codul biletului vândut, acest cod fiind unic. Aceasta reprezintă cheia primară a entității BILET și nu poate avea valoarea „null”, deoarece atunci când este vândut un bilet primește un cod
- b. **pret** - număr cu virgulă (de tipul „cifre.cifre”), float (NUMBER(8, 2)), care precizează prețul biletului achiziționat, preț care diferă de la film la film (exemplu: 18.50). Aceasta este mai mare sau egal decât 0.0, având valoare implicită 0.0
- c. **nr_loc** – număr de maxim 3 cifre (NUMBER(3)), mai mic sau egal decât 300 și mai mare ca 0, precizează numărul locului pe care îl va avea asignat spectatorul care se află în posesia aceluia bilet. Aceasta nu poate avea valoarea „null”, deoarece o dată cumpărat, orice bilet are atribuit un loc pentru cumpărător
- d. **nr_rand** – număr de maxim 2 cifre (NUMBER(2)), mai mic sau egal decât 20 și mai mare ca 0, precizează numărul rândului pe care se află locul pe care îl va avea asignat spectatorul care se află în posesia aceluia bilet. Aceasta nu poate avea valoarea „null”
- e. **cod_diffuzare** - număr de maxim 4 cifre (NUMBER(4)), care reprezintă codul difuzării căreia îi este asignat acest bilet. Aceasta are valoarea diferită de „null” și referențiază cheia primară a entității DIFUZARE („cod_diffuzare”)

6. ANGAJAT

- a. **cod_angajat** - număr de maxim 4 cifre (NUMBER(4)), care reprezintă codul angajatului, acest cod fiind unic (exemplu: 420). Aceasta reprezintă cheia primară a entității ANGAJAT
- b. **nume** - sir de maxim 25 de caractere (VARCHAR2 (25)), care precizează numele de familie al angajatului, aşa cum se regăseşte în cartea de identitate a acestuia (litere mari, mici, „-“) (exemplu: Popescu). Aceasta este unic și diferit de „null”, deoarece un angajat trebuie să apară în baza de date măcar cu numele de familie
- c. **prenume** - sir de maxim 25 de caractere (litere mari, mici, „-“) (VARCHAR2(25)), care precizează prenumele angajatului, aşa cum se regăseşte în cartea de identitate a acestuia.
- d. **email** – sir de maxim 25 de caractere (CHAR(25)) (litere mari, mici, „_“, „@“, „.“, cifre), precizează emailul la care poate fi contactat angajatul (exemplu: popescu@gmail.com). Aceasta trebuie să fie diferit de valoarea „null” și este unic

- e. **telefon** - sir de maxim 20 de caractere (VARCHAR2(20)), precizeaza numarul de telefon la care poate fi contactat angajatul, dar acesta nu este obligat sa ofere aceasta informatie (exemplu: 0723.999.234)
- f. **data_ang** – data calendaristica (DATE), precizeaza data la care s-a angajat angajatul respectiv (exemplu: 3 mai 2022). Aceasta este diferita de „null” si are valoarea „sysdate” by default
- g. **salariu** - numar cu virgula (de tipul „cifre.cifre”) (NUMBER(8, 2)), care reprezinta salariul unui angajat (exemplu: 16212.21). Acesta este diferit de „null” si mai mare decat 0, deoarece un salariu nu poate fi negativ
- h. **nr_ore** – numar de maxim 2 cifre (NUMBER(2)), care reprezinta numarul de ore pe care le lucreaza angajatul respectiv intr-o zi (norma standard, nu se tine cont de orele suplimentare) (exemplu: 7). Acesta este diferit de „null”, mai mare ca 0 si mai mic decat 24
- i. **id_superior** - de maxim 4 cifre (NUMBER(4)), care reprezinta codul superiorului fiecarui angajat (exemplu: 420). Acesta poate primi valori din multimea „cod_angajat” din tabela ANGAJAT, deoarece un superior trebuie sa fie angajat la randul sau
- j. **cod_cinema** - numar de maxim 4 cifre (NUMBER(4)), care reprezinta codul cinematografului la care este angajata persoana respectiva. Acesta are o valoare diferita de „null” si referinta cheia primara a entitatii CINEMATOGRAF („cod_cinema”)
- k. **cod_job** - sir de maxim 6 caractere (VARCHAR2(6)), care reprezinta codul jobului pe care il are asignat un cinematograf. Acesta este diferit de „null” si referinta cheia primara a entitatii JOB („cod_job”)

7. JOB

- a. **cod_job** - sir de maxim 6 caractere (VARCHAR2(6)), care reprezinta codul jobului, format din initialele reprezentative ale acestui job, acest cod fiind unic (exemplu: CAS (= casier), contraexemplu: 117). Acesta reprezinta cheia primara a entitatii JOB
- b. **nume_job** - sir de maxim 20 de caractere (litere mari, mici, „_”) (VARCHAR2(20)), care reprezinta numele jobului, asa cum apare acesta in fisiera de post (exemplu: casier). Acesta nu poate avea valoarea „null” si este unic
- c. **salariu_min** - numar cu virgula (de tipul „cifre.cifre”) (NUMBER(8.2)), care reprezinta salariul minim asignat jobului respectiv (exemplu: 1000.9). Acesta este diferit de „null” si mai mare decat 0, deoarece un salariu nu poate fi negativ
- d. **salariu_max** - numar cu virgula (de tipul „cifre.cifre”) (NUMBER(8.2)), care reprezinta salariul maxim asignat unui job (exemplu: 6000.9). Acesta este diferit de „null” si mai mare decat 0, deoarece un salariu nu poate fi negativ

8. LOCATIE

- a. **id_locatie** – număr de maxim 4 cifre (NUMBER(4)), care reprezintă id-ul locație, acest id fiind unic (exemplu: 177). Aceasta reprezintă cheia primară a entității LOCATIE
- b. **id_tara** - sir de maxim 3 caractere (VARCHAR2(3)), care reprezintă id-ul țării în care se află o locație (exemplu: RO). Acesta nu poate avea valoarea „null” și referențiază cheia primară a entității TARA („cod_tara”)
- c. **oras** – sir de maxim 20 de caractere (litere mari, mici, „-“) (VARCHAR2(20)), care reprezintă numele orașului unde se regăsește o anumită locație (exemplu: București). Aceasta nu poate avea valoarea „null”
- d. **strada** – sir de maxim 20 de caractere (litere mari, mici, „-“) (VARCHAR2(20)), care reprezintă numele străzii pe care se regăsește o anumită locație (exemplu: Arefu). Aceasta nu poate avea valoarea „null”
- e. **nr_strada** – număr de maxim 3 cifre (NUMBER(3)), care indică numărul străzii pe care se regăsește o anumită locație (exemplu: 59)

9. ACTOR

- a. **id_actor** – număr de maxim 4 cifre (NUMBER(4)), care reprezintă id-ul unui actor, acest id fiind unic (exemplu: 777123). Aceasta reprezintă cheia primară a entității ACTOR și nu poate avea valoarea „null”
- b. **nume** – sir de maxim 25 de caractere (litere mari, mici, „-“) (VARCHAR2(25)), care precizează numele de familie al actorului sau numele acestuia de scenă, după cel care este mai recunoscut (exemplu: Depp sau Coluche). Aceasta este diferit de „null”, deoarece trebuie să existe un nume după care să poată fi identificat, fiind unic
- c. **prenume** – sir de maxim 25 de caractere (litere mari, mici, „-“) (VARCHAR2(25)), care precizează prenumele actorului, fiind opțională precizarea acestuia
- d. **id_tara** - sir de maxim 3 caractere (VARCHAR2(3)), care reprezintă id-ul țării din care provine un actor (exemplu: RO). Aceasta nu poate avea valoarea „null” și referențiază cheia primară a entității TARA („cod_tara”)
- e. **nr_premii** – număr de maxim 3 cifre (NUMBER(3)), care reprezintă numărul de premii importante (Oscar, Grammy etc) primite de actorul respectiv până în prezent (exemplu: 3). Are valoare implicită 0 și este mai mare sau egal decât 0
- f. **an_debut** – număr de 4 cifre (NUMBER(4)), care reprezintă anul în care și-a făcut debutul actorul respectiv (exemplu: 1999). Aceasta este diferit de „null”, deoarece pentru ca un actor să fie recunoscut trebuie să debuteze mai întâi, de asemenea, este mai mare ca 0

10. REGIZOR

- a. **id_regizor** – număr de maxim 4 cifre (NUMBER(4)), care reprezintă id-ul unui regizor, acest id fiind unic. Aceasta reprezintă cheia primară a entității REGIZOR și nu poate avea valoarea „null”

- b.**nume** – sir de maxim 25 de caractere(litere mari, mici, „-“) (VARCHAR2(25)), care precizeaza numele de familie al regizorului. Acesta este diferit de „null”, deoarece trebuie sa existe un nume după care să poată fi identificat
- c.**prenume** – sir de maxim 25 de caractere (litere mari, mici, „-“) (VARCHAR 2(25)), care precizează prenumele regizorului, fiind optională precizarea acestuia
- d.**id_tara** – sir de maxim 3 caractere (VARCHAR2(3)), care reprezintă id-ul țării din care provine un regizor (exemplu: RO). Acesta nu poate avea valoarea „null” și referențiază cheia primară a entității TARA („cod_tara”)
- e.**nr_premii** – număr de maxim 3 cifre (NUMBER(3)), care reprezintă numărul de premii importante (Oscar, Gopo etc) primite de regizorul respectiv până în prezent. Are valoare implicită 0 și este mai mare sau egal decât 0

11. TARA

- a.**id_tara** – sir de maxim 3 caractere (VARCHAR2(3)), care reprezintă id-ul unei țări, acest id fiind unic. Aceasta reprezintă cheia primară a entității TARA și nu poate avea valoarea „null”
- b.**nume_tara** – sir de maxim 50 de caractere(litere mari, mici, „-“) (VARCHAR2(50)), care precizează numele complet al țării. Acesta este diferit de „null”, deoarece trebuie să existe un nume după care să poată fi identificată țara

1.5. Schemele relaționale

Enumerarea schemelor relaționale

CINEMATOGRAF(cod_cinema#, nume_cinema, id_locatie, complex, telefon,
email, data_desch)

SALA(cod_cinema#, nr_sala#, tip)

RULEAZA(cod_rulare#, cod_cinema, cod_film, data_inceput, data_final)

FILM(cod_film#, nume_film, gen_princ, durata, rating, an_aparitie)

DIFUZARE(cod_difuzare#, cod_film, nr_sala, cod_cinema, data_inc, ora_inc)

BILET_CUMPARAT(cod_bilet#, pret, nr_loc, nr_rand, cod_difuzare)

REGIZOR(id_regizor#, nume, prenume, id_tara, nr_premii)

REGIZEAZA(cod_film#, id_regizor#, an_inceput)

ACTOR(id_actor#, nume, prenume, id_tara, nr_premii, an_debut)

JOACA(cod_joaca#, cod_film, id_actor, rol)

ANGAJAT(cod_angajat#, nume, prenume, email, telefon, data_ang, salariu, nr_ore,
id_superior, cod_cinema, cod_job)

ISTORIC_LUCRU(cod_angajat#, data_start#, data_demisie, cod_cinema, cod_job)

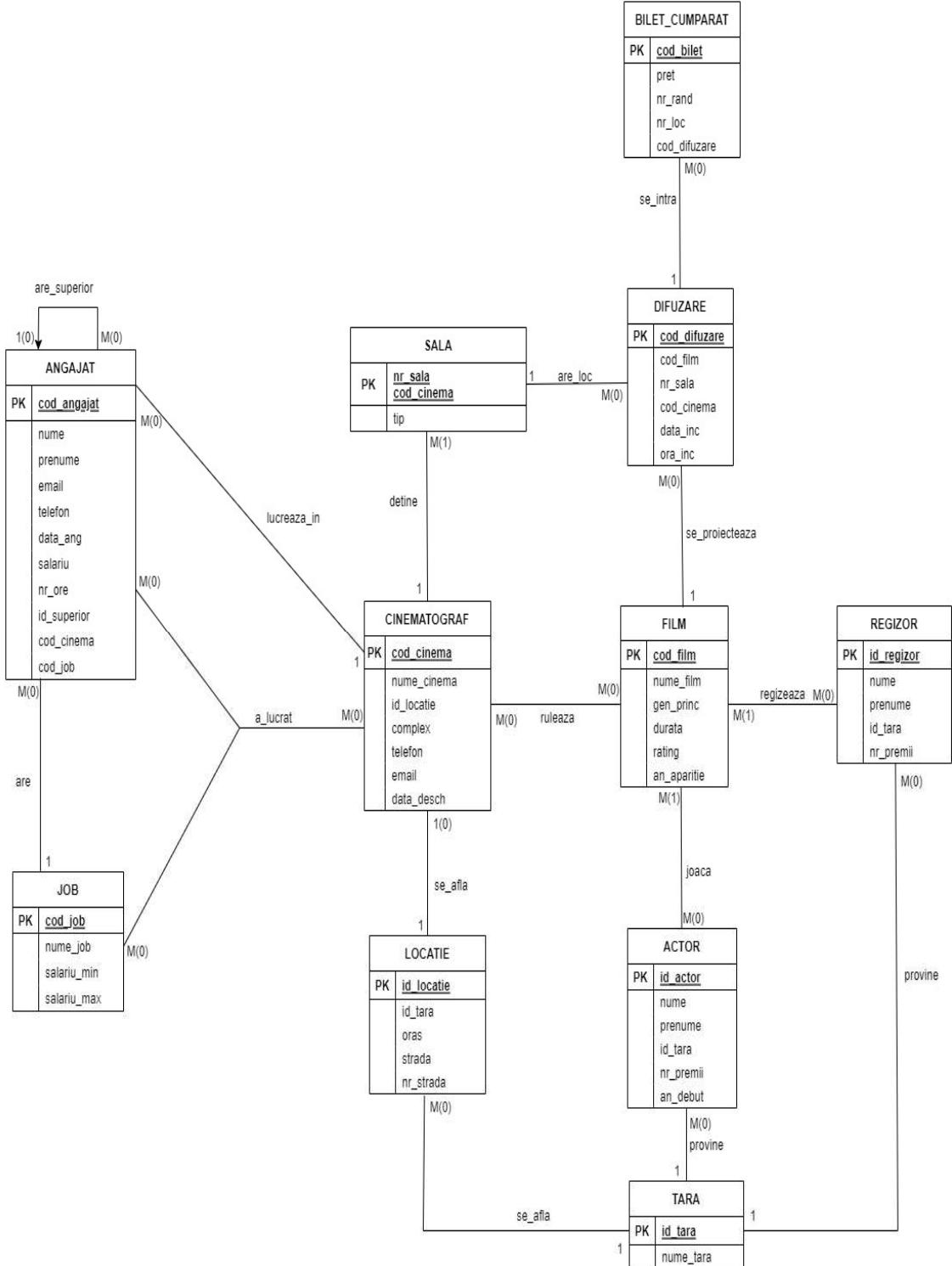
JOB(cod_job#, nume_job, salariu_min, salariu_max)

LOCATIE(id_locatie#, id_tara, oras, strada, nr_strada)

TARA(id_tara#, nume_tara)

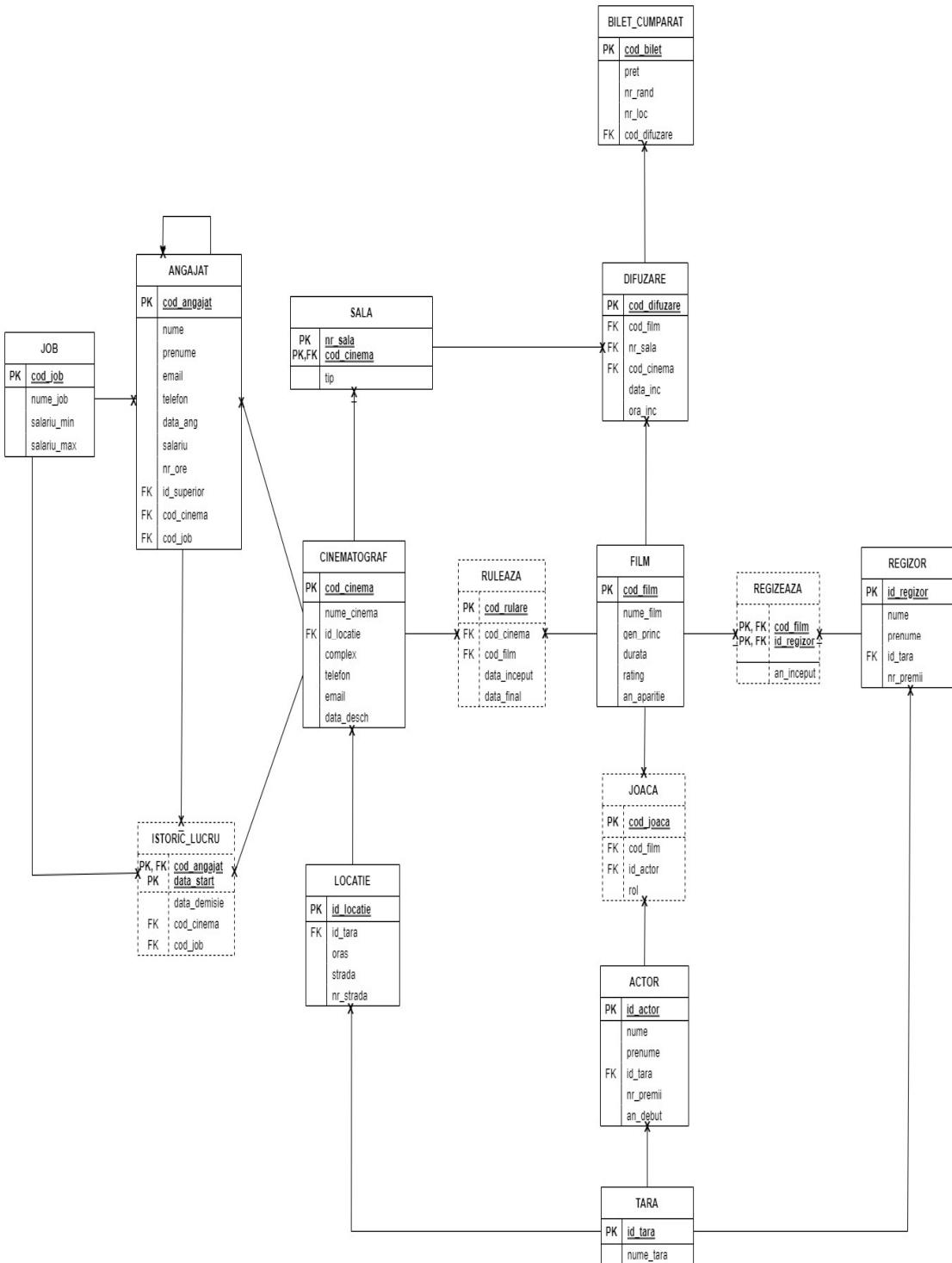
2. Diagrama Entitate-Relație

Prezentarea Diagramei Entitate-Relație



3. Diagrama Conceptuală

Prezentarea Diagramei Conceptuale



4. Crearea tabelelor

Implementarea în Oracle a diagramei conceptuale realizate, prin definirea tuturor tabelelor și aplicarea tuturor constrângerilor de integritate necesare (chei primare, cheile externe etc).

Crearea tabelei TARA:

CREATE TABLE tara

```
(id_tara VARCHAR2(3) CONSTRAINT pk_tara PRIMARY KEY,  
  nume_tara VARCHAR2(50) CONSTRAINT null_nume_tara NOT NULL);
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a code editor window containing the SQL command to create the TARA table. Below it, a 'Script Output' window displays the message 'Table TARA created.' and indicates a task completion time of 0.223 seconds.

```
CREATE TABLE tara  
(id_tara VARCHAR2(3) CONSTRAINT pk_tara PRIMARY KEY,  
  nume_tara VARCHAR2(50) CONSTRAINT null_nume_tara NOT NULL);
```

Script Output X | Task completed in 0.223 seconds

Table TARA created.

Crearea tabelei LOCATIE:

CREATE TABLE locatie

```
(id_locatie NUMBER(4) CONSTRAINT pk_locatie PRIMARY KEY,  
  id_tara VARCHAR2(3) CONSTRAINT null_loc_tara NOT NULL,  
  oras VARCHAR2(20) CONSTRAINT null_oras_loc NOT NULL,  
  strada VARCHAR2(20) CONSTRAINT null_strada_loc NOT NULL,  
  nr_strada NUMBER(3),  
  CONSTRAINT fk_loc_tara FOREIGN KEY(id_tara) REFERENCES  
tara(id_tara));
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a code editor window containing the SQL command to create the LOCATIE table. Below it, a 'Script Output' window displays the message 'Table LOCATIE created.' and indicates a task completion time of 0.042 seconds.

```
CREATE TABLE locatie  
(id_locatie NUMBER(4) CONSTRAINT pk_locatie PRIMARY KEY,  
  id_tara VARCHAR2(3) CONSTRAINT null_loc_tara NOT NULL,  
  oras VARCHAR2(20) CONSTRAINT null_oras_loc NOT NULL,  
  strada VARCHAR2(20) CONSTRAINT null_strada_loc NOT NULL,  
  nr_strada NUMBER(3),  
  CONSTRAINT fk_loc_tara FOREIGN KEY(id_tara) REFERENCES tara(id_tara));
```

Script Output X | Task completed in 0.042 seconds

Table LOCATIE created.

Crearea tabelei **JOB**:

```
CREATE TABLE job
  (cod_job VARCHAR2(6) CONSTRAINT pk_job PRIMARY KEY,
   nume_job VARCHAR2(20) CONSTRAINT null_nume_job NOT NULL,
   salariu_min NUMBER(8, 2) CONSTRAINT null_sal_min NOT NULL,
   salariu_max NUMBER(8, 2) CONSTRAINT null_sal_max NOT NULL,
   CONSTRAINT unq_nume_job UNIQUE(nume_job),
   CONSTRAINT ck_sal_min CHECK(salariu_min > 0.0),
   CONSTRAINT ck_sal_max CHECK(salariu_max > 0.0));
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a code editor window containing the SQL script to create the 'job' table. The code is identical to the one provided above. Below the code editor, in the bottom-left pane, there is a 'Script Output' tab. The output shows the command 'Table JOB created.' and a note indicating the task completed in 0.037 seconds.

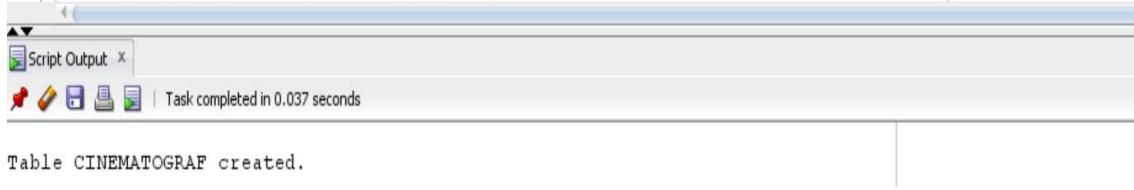
Crearea tabelei **CINEMATOGRAF**:

```
CREATE TABLE cinematograf
  (cod_cinema NUMBER(4) CONSTRAINT pk_cinema PRIMARY KEY,
   nume_cinema VARCHAR2(40) CONSTRAINT null_nume_cinema NOT NULL,
   id_locatie NUMBER(4) CONSTRAINT null_loc_cinema NOT NULL,
   complex VARCHAR2(40),
   telefon VARCHAR2(20),
   email CHAR(25) CONSTRAINT null_email_cinema NOT NULL,
   data_desch DATE DEFAULT SYSDATE,
   CONSTRAINT unq_email_cinema UNIQUE(email),
   CONSTRAINT fk_cinem_loc FOREIGN KEY(id_locatie) REFERENCES
   locatie(id_locatie),
   CONSTRAINT unqid_loc_cin UNIQUE(id_locatie));
```

```

CREATE TABLE cinematograf
(
cod_cinema NUMBER(4) CONSTRAINT pk_cinema PRIMARY KEY,
nume_cinema VARCHAR2(40) CONSTRAINT null_nume_cinema NOT NULL,
id_locatie NUMBER(4) CONSTRAINT null_loc_cinema NOT NULL,
complex VARCHAR2(40),
telefon VARCHAR2(20),
email CHAR(25) CONSTRAINT null_email_cinema NOT NULL,
data_desch DATE DEFAULT SYSDATE,
CONSTRAINT unq_email_cinema UNIQUE(email),
CONSTRAINT fk_cinem_loc FOREIGN KEY(id_locatie) REFERENCES locatie(id_locatie),
CONSTRAINT unqid_loc_cin UNIQUE(id_locatie));

```



Crearea tabelei ANGAJAT:

CREATE TABLE angajat

```

(cod_angajat NUMBER(4) CONSTRAINT pk_angajat PRIMARY KEY,
nume VARCHAR2(25) CONSTRAINT null_nume_ang NOT NULL,
prenume VARCHAR2(25),
email CHAR(25) CONSTRAINT null_email_ang NOT NULL,
telefon VARCHAR2(20),
data_ang DATE DEFAULT SYSDATE,
salariu NUMBER(8,2) CONSTRAINT null_sal_ang NOT NULL,
nr_ore NUMBER(2) CONSTRAINT null_nr_ore NOT NULL,
id_superior NUMBER(4) CONSTRAINT fk_ang_ang_sup REFERENCES
angajat(cod_angajat),
cod_cinema NUMBER(4) CONSTRAINT null_ang_cin NOT NULL,
cod_job VARCHAR2(6) CONSTRAINT null_ang_job NOT NULL,
CONSTRAINT unq_nume_pren_ang UNIQUE(nume, prenume),
CONSTRAINT unq_email_ang UNIQUE(email),
CONSTRAINT ck_nr_ore_ang CHECK(nr_ore > 0 AND nr_ore < 24),
CONSTRAINT ck_sal_ang CHECK(salariu > 0),
CONSTRAINT fk_ang_cin FOREIGN KEY(cod_cinema) REFERENCES
cinematograf(cod_cinema),

```

```

CONSTRAINT fk_ang_job FOREIGN KEY(cod_job) REFERENCES
job(cod_job));

```

```

CREATE TABLE angajat
(
cod_angajat NUMBER(4) CONSTRAINT pk_angajat PRIMARY KEY,
nume VARCHAR2(25) CONSTRAINT null_nume_ang NOT NULL,
prenume VARCHAR2(25),
email CHAR(25) CONSTRAINT null_email_ang NOT NULL,
telefon VARCHAR2(20),
data_ang DATE DEFAULT SYSDATE,
salariu NUMBER(8,2) CONSTRAINT null_sal_ang NOT NULL,
nr_ore NUMBER(2) CONSTRAINT null_nr_ore NOT NULL,
id_superior NUMBER(4) CONSTRAINT fk_ang_ang_sup REFERENCES angajat(cod_angajat),
cod_cinema NUMBER(4) CONSTRAINT null_ang_cin NOT NULL,
cod_job VARCHAR2(6) CONSTRAINT null_ang_job NOT NULL,
CONSTRAINT unq_nume_pren_ang UNIQUE(nume, prenume),
CONSTRAINT unq_email_ang UNIQUE(email),
CONSTRAINT ck_nr_ore_ang CHECK(nr_ore > 0 AND nr_ore < 24),
CONSTRAINT ck_sal_ang CHECK(salariu > 0),
CONSTRAINT fk_ang_cin FOREIGN KEY(cod_cinema) REFERENCES cinematograf(cod_cinema),
CONSTRAINT fk_ang_job FOREIGN KEY(cod_job) REFERENCES job(cod_job));

```

Script Output | Task completed in 0.09 seconds

Table ANGAJAT created.

Crearea tabeliei asociative **ISTORIC_LUCRU**:

```

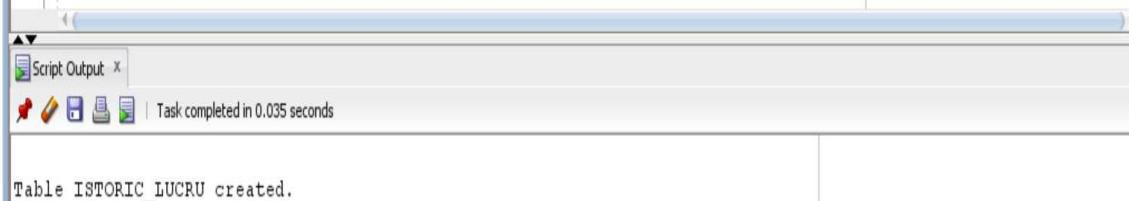
CREATE TABLE istoric_lucru
(
cod_angajat NUMBER(4) CONSTRAINT fk_ang_lucr REFERENCES
angajat(cod_angajat) ON DELETE CASCADE,
data_start DATE CONSTRAINT null_data_start NOT NULL,
data_demisie DATE DEFAULT SYSDATE,
cod_cinema NUMBER(4) CONSTRAINT null_ist_cin NOT NULL,
cod_job VARCHAR2(6) CONSTRAINT null_ist_job NOT NULL,
CONSTRAINT fk_cin_lucr FOREIGN KEY(cod_cinema) REFERENCES
cinematograf(cod_cinema) ON DELETE CASCADE,
CONSTRAINT fk_job_lucr FOREIGN KEY(cod_job) REFERENCES
job(cod_job) ON DELETE CASCADE,
CONSTRAINT pk_ang_ist_lucr PRIMARY KEY(cod_angajat, data_start));

```

```

CREATE TABLE istoric_lucru
(cod_angajat NUMBER(4) CONSTRAINT fk_ang_lucr REFERENCES angajat(cod_angajat) ON DELETE CASCADE,
data_start DATE CONSTRAINT null_data_start NOT NULL,
data_demisie DATE DEFAULT SYSDATE,
cod_cinema NUMBER(4) CONSTRAINT null_ist_cin NOT NULL,
cod_job VARCHAR2(6) CONSTRAINT null_ist_job NOT NULL,
CONSTRAINT fk_cin_lucr FOREIGN KEY(cod_cinema) REFERENCES cinematograf(cod_cinema) ON DELETE CASCADE,
CONSTRAINT fk_job_lucr FOREIGN KEY(cod_job) REFERENCES job(cod_job) ON DELETE CASCADE,
CONSTRAINT pk_ang_ist_lucr PRIMARY KEY(cod_angajat, data_start));

```



Crearea tabelei FILM:

CREATE TABLE film

```

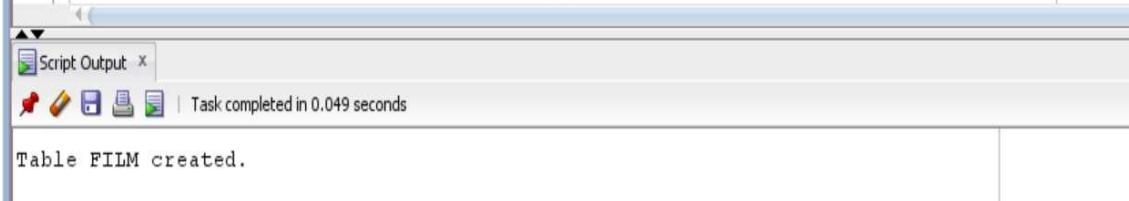
(cod_film NUMBER(4) CONSTRAINT pk_film PRIMARY KEY,
nume_film VARCHAR2(25) CONSTRAINT null_nume_film NOT NULL,
gen_princ VARCHAR2(20) CONSTRAINT null_gen NOT NULL,
durata NUMBER(3) DEFAULT 0,
rating NUMBER(2, 1) CONSTRAINT ck_rating CHECK(rating > 0.0 AND
rating <= 10.0),
an_aparitie NUMBER(4) CONSTRAINT null_an_aparitie NOT NULL,
CONSTRAINT ck_an_ap CHECK(an_aparitie > 0),
CONSTRAINT unq_nume_film_an_ap UNIQUE(nume_film, an_aparitie));

```

```

CREATE TABLE film
(cod_film NUMBER(4) CONSTRAINT pk_film PRIMARY KEY,
nume_film VARCHAR2(25) CONSTRAINT null_nume_film NOT NULL,
gen_princ VARCHAR2(20) CONSTRAINT null_gen NOT NULL,
durata NUMBER(3) DEFAULT 0,
rating NUMBER(2, 1) CONSTRAINT ck_rating CHECK(rating > 0.0 AND rating <= 10.0),
an_aparitie NUMBER(4) CONSTRAINT null_an_aparitie NOT NULL,
CONSTRAINT ck_an_ap CHECK(an_aparitie > 0),
CONSTRAINT unq_nume_film_an_ap UNIQUE(nume_film, an_aparitie));

```



Crearea tabeliei asociative RULEAZA:

CREATE TABLE ruleaza

```
(cod_rulare NUMBER(4) CONSTRAINT pk_ruleaza PRIMARY KEY,  
cod_cinema NUMBER(4) CONSTRAINT null_rul_cin NOT NULL,  
cod_film NUMBER(4) CONSTRAINT null_rul_film NOT NULL,  
data_inceput DATE DEFAULT SYSDATE,  
data_final DATE DEFAULT SYSDATE + 15,  
CONSTRAINT fk_cin_rul FOREIGN KEY(cod_cinema) REFERENCES  
cinematograf(cod_cinema) ON DELETE CASCADE,  
CONSTRAINT fk_film_rul FOREIGN KEY(cod_film) REFERENCES  
film(cod_film) ON DELETE CASCADE);
```

The screenshot shows the Oracle SQL Developer interface. In the top pane, there is a code editor containing the SQL script for creating the 'ruleaza' table. The table has columns for cod_rulare (primary key), cod_cinema, cod_film, data_inceput, and data_final. It includes two foreign key constraints: 'fk_cin_rul' linking to the 'cinematograf' table and 'fk_film_rul' linking to the 'film' table, both with cascade delete options. In the bottom pane, there is a 'Script Output' window showing the message 'Table RULEAZA created.' and a note indicating the task completed in 0.033 seconds.

Crearea tabeliei ACTOR:

CREATE TABLE actor

```
(id_actor NUMBER(4) CONSTRAINT pk_actor PRIMARY KEY,  
nume VARCHAR2(25) CONSTRAINT null_nume_actor NOT NULL,  
prenume VARCHAR2(25),  
id_tara VARCHAR2(3) CONSTRAINT null_tara_act NOT NULL,  
nr_premii NUMBER(3) DEFAULT 0,  
an_debut NUMBER(4) CONSTRAINT null_an_debut_act NOT NULL,  
CONSTRAINT unq_nume_pren_act UNIQUE(nume, prenume),  
CONSTRAINT ck_nr_premii_act CHECK(nr_premii >= 0),  
CONSTRAINT ck_an_debut_act CHECK(an_debut > 0),  
CONSTRAINT fk_act_tara FOREIGN KEY(id_tara) REFERENCES  
tara(id_tara));
```

```

CREATE TABLE actor
  (id_actor NUMBER(4) CONSTRAINT pk_actor PRIMARY KEY,
  nume VARCHAR2(25) CONSTRAINT null_nume_actor NOT NULL,
  prenume VARCHAR2(25),
  id_tara VARCHAR2(3) CONSTRAINT null_tara_act NOT NULL,
  nr_premii NUMBER(3) DEFAULT 0,
  an_debut NUMBER(4) CONSTRAINT null_an_debut_act NOT NULL,
  CONSTRAINT unq_nume_pren_act UNIQUE (nume, prenume),
  CONSTRAINT ck_nr_premii_act CHECK (nr_premii >= 0),
  CONSTRAINT ck_an_debut_act CHECK (an_debut > 0),
  CONSTRAINT fk_act_tara FOREIGN KEY(id_tara) REFERENCES tara(id_tara));

```

Table ACTOR created.

Crearea tabelei asociative JOACA:

```

CREATE TABLE joaca
  (cod_joaca NUMBER(4) CONSTRAINT pk_joaca PRIMARY KEY,
  cod_film NUMBER(4) CONSTRAINT null_film_joaca NOT NULL,
  id_actor NUMBER(4) CONSTRAINT null_act_joaca NOT NULL,
  rol VARCHAR2(20) CONSTRAINT null_rol_joaca NOT NULL,
  CONSTRAINT fk_film_joaca FOREIGN KEY(cod_film) REFERENCES
film(cod_film) ON DELETE CASCADE,
  CONSTRAINT fk_act_joaca FOREIGN KEY(id_actor) REFERENCES
actor(id_actor) ON DELETE CASCADE);

```

```

CREATE TABLE joaca
  (cod_joaca NUMBER(4) CONSTRAINT pk_joaca PRIMARY KEY,
  cod_film NUMBER(4) CONSTRAINT null_film_joaca NOT NULL,
  id_actor NUMBER(4) CONSTRAINT null_act_joaca NOT NULL,
  rol VARCHAR2(20) CONSTRAINT null_rol_joaca NOT NULL,
  CONSTRAINT fk_film_joaca FOREIGN KEY(cod_film) REFERENCES film(cod_film) ON DELETE CASCADE,
  CONSTRAINT fk_act_joaca FOREIGN KEY(id_actor) REFERENCES actor(id_actor) ON DELETE CASCADE);

```

Table JOACA created.

Crearea tabelei REGIZOR:

```

CREATE TABLE regizor
  (id_regizor NUMBER(4) CONSTRAINT pk_regizor PRIMARY KEY,

```

```

        nume VARCHAR2(25) CONSTRAINT null_nume_registor NOT NULL,
        prenume VARCHAR2(25),
        id_tara VARCHAR2(3) CONSTRAINT null_tara_reg NOT NULL,
        nr_premii NUMBER(3) DEFAULT 0,
        CONSTRAINT unq_nume_pren_reg UNIQUE(nume, prenume),
        CONSTRAINT ck_nr_premii_reg CHECK(nr_premii >= 0),
        CONSTRAINT fk_tara_reg FOREIGN KEY(id_tara) REFERENCES
tara(id_tara));

```

```

CREATE TABLE regizor
(
    id_regizor NUMBER(4) CONSTRAINT pk_regizor PRIMARY KEY,
    nume VARCHAR2(25) CONSTRAINT null_nume_registor NOT NULL,
    prenume VARCHAR2(25),
    id_tara VARCHAR2(3) CONSTRAINT null_tara_reg NOT NULL,
    nr_premii NUMBER(3) DEFAULT 0,
    CONSTRAINT unq_nume_pren_reg UNIQUE(nume, prenume),
    CONSTRAINT ck_nr_premii_reg CHECK(nr_premii >= 0),
    CONSTRAINT fk_tara_reg FOREIGN KEY(id_tara) REFERENCES tara(id_tara));

```

Script Output X

Table completed in 0.032 seconds

Table REGIZOR created.

Crearea tablei asociative REGIZEAZA:

```

CREATE TABLE regizeaza
(
    cod_film NUMBER(4) CONSTRAINT fk_film_regiz REFERENCES
film(cod_film) ON DELETE CASCADE,
    id_regizor NUMBER(4) CONSTRAINT fk_regizor_regiz REFERENCES
regizor(id_regizor) ON DELETE CASCADE,
    an_inceput NUMBER(4) CONSTRAINT null_an_inc_reg NOT NULL,
    CONSTRAINT pk_regizor_film PRIMARY KEY(cod_film, id_regizor),
    CONSTRAINT ck_an_inc_reg CHECK(an_inceput > 0));

```

```

CREATE TABLE regizeaza
(
    cod_film NUMBER(4) CONSTRAINT fk_film_regiz REFERENCES film(cod_film) ON DELETE CASCADE,
    id_regizor NUMBER(4) CONSTRAINT fk_regizor_regiz REFERENCES regizor(id_regizor) ON DELETE CASCADE,
    an_inceput NUMBER(4) CONSTRAINT null_an_inc_reg NOT NULL,
    CONSTRAINT pk_regizor_film PRIMARY KEY(cod_film, id_regizor),
    CONSTRAINT ck_an_inc_reg CHECK(an_inceput > 0));

```

Script Output X

Table completed in 0.026 seconds

Table REGIZEAZA created.

Crearea tabelei SALA:

```
CREATE TABLE sala
  (nr_sala NUMBER(2),
   cod_cinema NUMBER(4) CONSTRAINT fk_sala_cinema REFERENCES
   cinematograf(cod_cinema) ON DELETE CASCADE,
   tip VARCHAR2(10),
   CONSTRAINT ck_nr_sala CHECK(nr_sala <= 40 AND nr_sala > 0),
   CONSTRAINT pk_sala_cin PRIMARY KEY(nr_sala, cod_cinema));
```

The screenshot shows the Oracle SQL Developer interface. In the top pane, there is a code editor window containing the SQL script to create the 'sala' table. The table has columns for 'nr_sala' (NUMBER(2)), 'cod_cinema' (NUMBER(4)), 'tip' (VARCHAR2(10)), and three constraints: 'ck_nr_sala' (CHECK(nr_sala <= 40 AND nr_sala > 0)), 'pk_sala_cin' (PRIMARY KEY(nr_sala, cod_cinema)), and 'fk_sala_cinema' (FOREIGN KEY(cod_cinema) REFERENCES cinematograf(cod_cinema) ON DELETE CASCADE). Below the code editor is a 'Script Output' window showing the results of the execution. It displays a message indicating the task was completed in 0.033 seconds and a confirmation message: 'Table SALA created.'

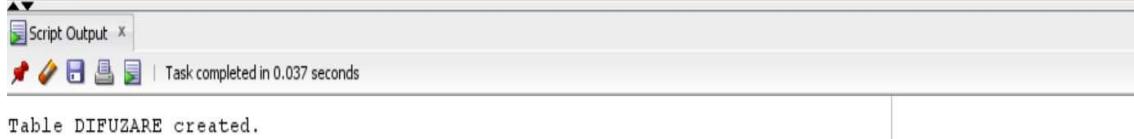
Crearea tabelei DIFUZARE:

```
CREATE TABLE difuzare
  (cod_difuzare NUMBER(4) CONSTRAINT pk_difuzare PRIMARY KEY,
   cod_film NUMBER(4) CONSTRAINT null_dif_film NOT NULL,
   nr_sala NUMBER(2) CONSTRAINT null_sala_dif NOT NULL,
   cod_cinema NUMBER(4) CONSTRAINT null_cin_dif NOT NULL,
   data_inc DATE DEFAULT SYSDATE,
   ora_inc NUMBER(2) CONSTRAINT null_ora_inc NOT NULL,
   CONSTRAINT ck_ora_inc CHECK(ora_inc > 0 AND ora_inc < 24),
   CONSTRAINT fk_film_dif FOREIGN KEY(cod_film) REFERENCES
   film(cod_film) ON DELETE CASCADE,
   CONSTRAINT fk_nr_sala_cin_dif FOREIGN KEY(nr_sala, cod_cinema)
   REFERENCES sala(nr_sala, cod_cinema) ON DELETE CASCADE);
```

```

CREATE TABLE difuzare
(
    cod_difuzare NUMBER(4) CONSTRAINT pk_difuzare PRIMARY KEY,
    cod_film NUMBER(4) CONSTRAINT null_dif_film NOT NULL,
    nr_sala NUMBER(2) CONSTRAINT null_sala_dif NOT NULL,
    cod_cinema NUMBER(4) CONSTRAINT null_cin_dif NOT NULL,
    data_inc DATE DEFAULT SYSDATE,
    ora_inc NUMBER(2) CONSTRAINT null_ora_inc NOT NULL,
    CONSTRAINT ck_ora_inc CHECK(ora_inc > 0 AND ora_inc < 24),
    CONSTRAINT fk_film_dif FOREIGN KEY(cod_film) REFERENCES film(cod_film) ON DELETE CASCADE,
    CONSTRAINT fk_nr_sala_cin_dif FOREIGN KEY(nr_sala, cod_cinema)
        REFERENCES sala(nr_sala, cod_cinema) ON DELETE CASCADE);

```



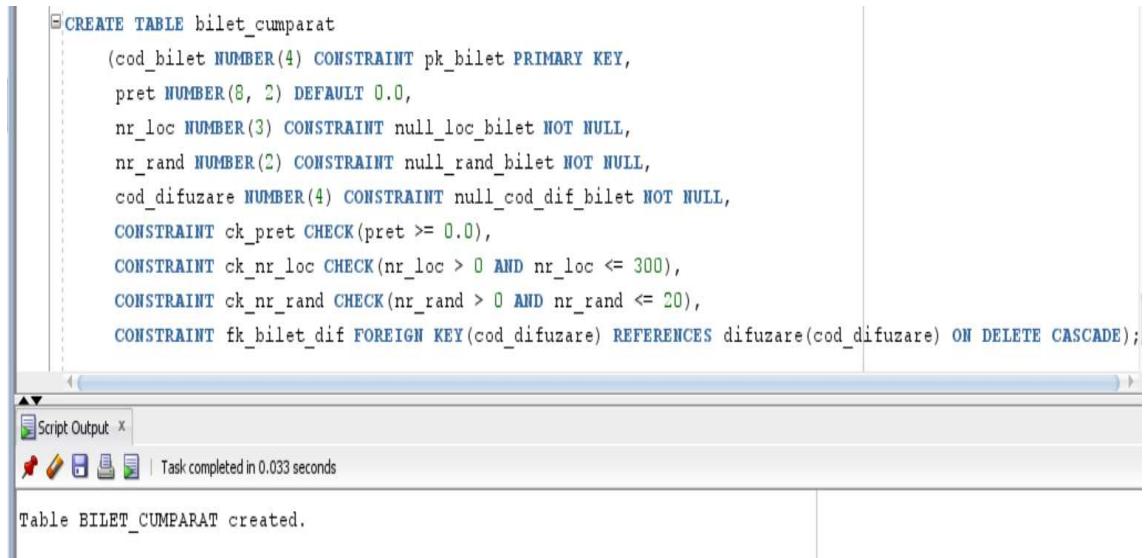
Crearea tabelei BILET_CUMPARAT:

CREATE TABLE bilet_cumparat

```

(cod_bilet NUMBER(4) CONSTRAINT pk_bilet PRIMARY KEY,
 pret NUMBER(8, 2) DEFAULT 0.0,
 nr_loc NUMBER(3) CONSTRAINT null_loc_bilet NOT NULL,
 nr_rand NUMBER(2) CONSTRAINT null_rand_bilet NOT NULL,
 cod_difuzare NUMBER(4) CONSTRAINT null_cod_dif_bilet NOT NULL,
 CONSTRAINT ck_pret CHECK(pret >= 0.0),
 CONSTRAINT ck_nr_loc CHECK(nr_loc > 0 AND nr_loc <= 300),
 CONSTRAINT ck_nr_rand CHECK(nr_rand > 0 AND nr_rand <= 20),
 CONSTRAINT fk_bilet_dif FOREIGN KEY(cod_difuzare) REFERENCES
 difuzare(cod_difuzare) ON DELETE CASCADE);

```



5. Inserarea datelor în tabele

Adăugarea de informații coerente în tabelele create mai sus (minim 5 înregistrări pentru fiecare entitate independentă și minim 10 înregistrări pentru fiecare tabelă asociativă).

Inserări în tabela **TARA**:

```
INSERT INTO tara
VALUES('RO', 'Romania');

INSERT INTO tara
VALUES('SUA', 'Statele Unite ale Americii');

INSERT INTO tara
VALUES('EN', 'Anglia');

INSERT INTO tara
VALUES('CA', 'Canada');

INSERT INTO tara
VALUES('IL', 'Israel');

INSERT INTO tara
VALUES('AR', 'Argentina');

INSERT INTO tara
VALUES('MD', 'Moldova');

INSERT INTO tara
VALUES('NO', 'Norvegia');
```

The screenshot shows a database management interface with a script editor and an output window. The script editor contains the SQL code for inserting data into the 'tara' table. The output window shows the results of the execution, indicating that 1 row was inserted.

```
INSERT INTO tara
VALUES ('RO', 'Romania');
INSERT INTO tara
VALUES ('SUA', 'Statele Unite ale Americii');
INSERT INTO tara
VALUES ('EN', 'Anglia');
INSERT INTO tara
VALUES ('CA', 'Canada');
INSERT INTO tara
VALUES ('IL', 'Israel');
INSERT INTO tara
VALUES ('AR', 'Argentina');
INSERT INTO tara
VALUES ('MD', 'Moldova');
INSERT INTO tara
VALUES ('NO', 'Norvegia');
```

Script Output | Task completed in 0.129 seconds

1 row inserted.

The screenshot shows the SQL Server Management Studio interface. In the top pane, there is a code editor with the following SQL query:

```
SELECT * FROM tara;
```

In the bottom pane, there are two tabs: "Script Output" and "Query Result". The "Query Result" tab is selected and displays the following data:

| ID_TARA | NUME_TARA |
|---------|--------------------------------|
| 1 | RO Romania |
| 2 | SUA Statele Unite ale Americii |
| 3 | EN Anglia |
| 4 | CA Canada |
| 5 | IL Israel |
| 6 | AR Argentina |
| 7 | MD Moldova |
| 8 | NO Norvegia |

Inserări în tabela LOCATIE:

INSERT INTO locatie

VALUES(100, 'RO', 'Bucuresti', 'Bd Doina Cornea', 4);

INSERT INTO locatie

VALUES(105, 'RO', 'Bucuresti', 'Bd Timisoara', 26);

INSERT INTO locatie

VALUES(110, 'RO', 'Bucuresti', 'Mihai Eminescu', null);

INSERT INTO locatie

VALUES(115, 'RO', 'Sibiu', 'Aurel Vlaicu', 11);

INSERT INTO locatie

VALUES(120, 'RO', 'Suceava', 'Energeticului', null);

INSERT INTO locatie

VALUES(125, 'RO', 'Iasi', 'Tudor Vladimirescu', 2);

INSERT INTO locatie

VALUES(130, 'RO', 'Constanta', 'Aurel Vlaicu', 220);

INSERT INTO locatie

VALUES(135, 'RO', 'Cluj-Napoca', 'Bd Eroilor', 51);

INSERT INTO locatie

VALUES(140, 'RO', 'Baia Mare', 'Victoriei', 73);

INSERT INTO locatie

VALUES(145, 'RO', 'Pitesti', 'Calea Bucuresti', 36);

INSERT INTO locatie

VALUES(150, 'RO', 'Drobeta Turnu Severin', 'Bd Alunis', 78);

```

INSERT INTO locatie
VALUES(100, 'RO', 'Bucuresti', 'Bd Doina Cornea', 4);
INSERT INTO locatie
VALUES(105, 'RO', 'Bucuresti', 'Bd Timisoara', 26);
INSERT INTO locatie
VALUES(110, 'RO', 'Bucuresti', 'Mihai Eminescu', null);
INSERT INTO locatie
VALUES(115, 'RO', 'Sibiu', 'Aurel Vlaicu', 11);
INSERT INTO locatie
VALUES(120, 'RO', 'Suceava', 'Energeticului', null);
INSERT INTO locatie
VALUES(125, 'RO', 'Iasi', 'Tudor Vladimirescu', 2);
INSERT INTO locatie
VALUES(130, 'RO', 'Constanta', 'Aurel Vlaicu', 220);
INSERT INTO locatie
VALUES(135, 'RO', 'Cluj-Napoca', 'Bd Eroilor', 51);
INSERT INTO locatie
VALUES(140, 'RO', 'Baia Mare', 'Victoriei', 73);
INSERT INTO locatie
VALUES(145, 'RO', 'Pitesti', 'Calea Bucuresti', 36);
INSERT INTO locatie
VALUES(150, 'RO', 'Drobeta Turnu Severin', 'Bd Alunis', 78);

```

```
SELECT * FROM locatie;
```

Script Output | Query Result | SQL | All Rows Fetched: 11 in 0.003 seconds

| ID_LOCATIE | ID_TARA | ORAS | STRADA | NR_STRADA |
|------------|---------|------|-----------------------|-----------------------|
| 1 | 100 | RO | Bucuresti | Bd Doina Cornea 4 |
| 2 | 105 | RO | Bucuresti | Bd Timisoara 26 |
| 3 | 110 | RO | Bucuresti | Mihai Eminescu (null) |
| 4 | 115 | RO | Sibiu | Aurel Vlaicu 11 |
| 5 | 120 | RO | Suceava | Energeticului (null) |
| 6 | 125 | RO | Iasi | Tudor Vladimirescu 2 |
| 7 | 130 | RO | Constanta | Aurel Vlaicu 220 |
| 8 | 135 | RO | Cluj-Napoca | Bd Eroilor 51 |
| 9 | 140 | RO | Baia Mare | Victoriei 73 |
| 10 | 145 | RO | Pitesti | Calea Bucuresti 36 |
| 11 | 150 | RO | Drobeta Turnu Severin | Bd Alunis 78 |

Inserări în tabela **JOB**:

INSERT INTO job

VALUES('CAS', 'Casier', 3792.1, 6293.0);

INSERT INTO job

VALUES('AG_C', 'Agent de Curatenie', 1239.2, 1897.9);

INSERT INTO job

VALUES('MAN', 'Manager', 5439.22, 6934.2);

INSERT INTO job

VALUES('ING_S', 'Inginer Sunet', 3234.2, 4593.3);

INSERT INTO job

VALUES('ING_T', 'Inginer Tehnic', 2134.2, 5593.3);

```

INSERT INTO job
VALUES('PLAS', 'Plasator', 1231.4, 3432.43);

INSERT INTO job
VALUES('BAR_C', 'Barman Cafenea', 1312.3, 2313.3);

INSERT INTO job
VALUES('BAR', 'Barman', 1192.3, 1922.0);

```

The screenshot shows the Oracle SQL Developer interface with two panes. The top pane contains the SQL script for inserting data into the 'job' table. The bottom pane shows the results of a 'SELECT * FROM job;' query.

```

INSERT INTO job
VALUES('CAS', 'Casier', 3792.1, 6293.0);
INSERT INTO job
VALUES('AG_C', 'Agent de Curatenie', 1239.2, 1897.9);
INSERT INTO job
VALUES('MAN', 'Manager', 5439.22, 6934.2);
INSERT INTO job
VALUES('ING_S', 'Inginer Sunet', 3234.2, 4593.3);
INSERT INTO job
VALUES('ING_T', 'Inginer Tehnic', 2134.2, 5593.3);
INSERT INTO job
VALUES('PLAS', 'Plasator', 1231.4, 3432.43);
INSERT INTO job
VALUES('BAR_C', 'Barman Cafenea', 1312.3, 2313.3);
INSERT INTO job
VALUES('BAR', 'Barman', 1192.3, 1922.0);

1 row inserted.

| SELECT * FROM job; |
| Script Output | Query Result |
| Task completed in 0.073 seconds |
| 1 row inserted. |

| COD_JOB | NUME_JOB | SALARIU_MIN | SALARIU_MAX |
| 1 CAS | Casier | 3792.1 | 6293 |
| 2 AG_C | Agent de Curatenie | 1239.2 | 1897.9 |
| 3 MAN | Manager | 5439.22 | 6934.2 |
| 4 ING_S | Inginer Sunet | 3234.2 | 4593.3 |
| 5 ING_T | Inginer Tehnic | 2134.2 | 5593.3 |
| 6 PLAS | Plasator | 1231.4 | 3432.43 |
| 7 BAR_C | Barman Cafenea | 1312.3 | 2313.3 |
| 8 BAR | Barman | 1192.3 | 1922 |

```

Inserări în tabela CINEMATOGRAF:

INSERT INTO cinematograf

```

VALUES(120, 'Cinema City PLake', 105, 'Park Lake', '0314.032.700',
'cin.city.pklk@yahoo.com', '2 may 2016');

```

```

INSERT INTO cinematograf
VALUES(140, 'Cinema City Sibiu', 115, null, '0734.566.932', 'cin.city.arta@yahoo.com',
'27 december 2008');

INSERT INTO cinematograf
VALUES(160, 'Cinema City Cluj', 135, 'Iulius Mall', '0314.130.400',
'cin.city.ilm@yahoo.com', '16 april 2014');

INSERT INTO cinematograf
VALUES(180, 'Cinema City Constanta', 130, null, null, 'cin.city.ct@yahoo.com', '17
october 2018');

INSERT INTO cinematograf
VALUES(200, 'Cinema City Baia Mare', 140, null, '0362.403.030',
'cin.city.bm@yahoo.com', '7 may 2016');

INSERT INTO cinematograf
VALUES(220, 'Cinema City Afi Cotroceni', 100, 'Afi Cotroceni', '0355.671.129',
'cin.city.afi@yahoo.com', '13 april 2013');

INSERT INTO cinematograf
VALUES(240, 'Cinema City Iasi', 125, 'Iulius Mall', '0314.130.000',
'cin.city.iasi@yahoo.com', '9 july 2009');

INSERT INTO cinematograf
VALUES(260, 'Cinema City Pitesti', 145, 'VIVO!', '0314.130.080',
'cin.city.pit@yahoo.com', '12 may 2014');

INSERT INTO cinematograf
VALUES(280, 'Cinema City DrobetaTS', 150, 'Severin Shopping Center',
'031.413.0380', 'cin.city.dts@yahoo.com', '4 july 2016');

```

```

INSERT INTO cinematograf
VALUES(120, 'Cinema City PLake', 105, 'Park Lake', '0314.032.700', 'cin.city.pklk@yahoo.com', '2 may 2016');
INSERT INTO cinematograf
VALUES(140, 'Cinema City Sibiu', 115, null, '0734.566.932', 'cin.city.arta@yahoo.com', '27 december 2008');
INSERT INTO cinematograf
VALUES(160, 'Cinema City Cluj', 135, 'Iulius Mall', '0314.130.400', 'cin.city.ilm@yahoo.com', '16 april 2014');
INSERT INTO cinematograf
VALUES(180, 'Cinema City Constanta', 130, null, null, 'cin.city.ct@yahoo.com', '17 october 2018');
INSERT INTO cinematograf
VALUES(200, 'Cinema City Baia Mare', 140, null, '0362.403.030', 'cin.city.bm@yahoo.com', '7 may 2016');
INSERT INTO cinematograf
VALUES(220, 'Cinema City Afi Cotroceni', 100, 'Afi Cotroceni', '0355.671.129', 'cin.city.afi@yahoo.com', '13 april 2013');
INSERT INTO cinematograf
VALUES(240, 'Cinema City Iasi', 125, 'Iulius Mall', '0314.130.000', 'cin.city.iasi@yahoo.com', '9 july 2009');
INSERT INTO cinematograf
VALUES(260, 'Cinema City Pitesti', 145, 'VIVO!', '0314.130.080', 'cin.city.pit@yahoo.com', '12 may 2014');
INSERT INTO cinematograf
VALUES(280, 'Cinema City DrobetaTS', 150, 'Severin Shopping Center', '031.413.0380', 'cin.city.dts@yahoo.com', '4 july 2016');

```

1 row inserted.

SELECT * FROM cinematograf;

cript Output x Query Result x

All Rows Fetched: 9 in 0.017 seconds

| | COD_CINEMA | NUME_CINEMA | ID_LOCATIE | COMPLEX | TELEFON | EMAIL | DATA_DESCR |
|---|------------|---------------------------|------------|-------------------------|--------------|-------------------------|------------|
| 1 | 120 | Cinema City PLake | 105 | Park Lake | 0314.032.700 | cin.city.pklk@yahoo.com | 02-MAY-16 |
| 2 | 140 | Cinema City Sibiu | 115 | (null) | 0734.566.932 | cin.city.arta@yahoo.com | 27-DEC-08 |
| 3 | 160 | Cinema City Cluj | 135 | Iulius Mall | 0314.130.400 | cin.city.ilm@yahoo.com | 16-APR-14 |
| 4 | 180 | Cinema City Constanta | 130 | (null) | (null) | cin.city.ct@yahoo.com | 17-OCT-18 |
| 5 | 200 | Cinema City Baia Mare | 140 | (null) | 0362.403.030 | cin.city.bm@yahoo.com | 07-MAY-16 |
| 6 | 220 | Cinema City Afi Cotroceni | 100 | Afi Cotroceni | 0355.671.129 | cin.city.afi@yahoo.com | 13-APR-13 |
| 7 | 240 | Cinema City Iasi | 125 | Iulius Mall | 0314.130.000 | cin.city.iasi@yahoo.com | 09-JUL-09 |
| 8 | 260 | Cinema City Pitesti | 145 | VIVO! | 0314.130.080 | cin.city.pit@yahoo.com | 12-MAY-14 |
| 9 | 280 | Cinema City DrobetaTS | 150 | Severin Shopping Center | 031.413.0380 | cin.city.dts@yahoo.com | 04-JUL-16 |

Inserări în tabela ANGAJAT:

INSERT INTO angajat

```
VALUES(1000, 'Popescu', 'Ion', 'ion.popescu@gmail.com', '0741.311.314', '3 june 2018',
6023.5, 9, null, 120, 'MAN');
```

INSERT INTO angajat

```
VALUES(1017, 'Tudor', 'Maria', 'maria.tudor@gmail.com', '0712.133.292', '5
september 2017', 5998.8, 12, null, 140, 'MAN');
```

INSERT INTO angajat

```
VALUES(1034, 'Georgesc', 'Vasile', 'vasile.georgesc@gmail.com', null, '5 may 2019',
1555.1, 6, null, 140, 'AG_C');
```

INSERT INTO angajat

```
VALUES(1051, 'Matei', 'Andrei', 'andrei.matei@gmail.com', '0799.532.678', '27
january 2020', 4672.5, 8, 1000, 120, 'CAS');
```

INSERT INTO angajat

```
VALUES(1085, 'Iana', 'Gabriela', 'gabriela.iana@gmail.com', '0712.623.881', '12 april
2021', 3972.5, 8, 1051, 120, 'CAS');
```

INSERT INTO angajat

```
VALUES(1102, 'Miron', 'Costin', 'costin.miron@gmail.com', '0771.229.345', '19
november 2019', 5789.1, 9, null, 260, 'MAN');
```

INSERT INTO angajat

```
VALUES(1119, 'Banu', 'Codrin', 'codrin.banu@gmail.com', '0741.721.921', '19
november 2019', 1231.1, 4, 1102, 260, 'ING_S');
```

INSERT INTO angajat

```
VALUES(1136, 'Crivat', 'Gheorghe', 'gheo.crivat@gmail.com', null, '21 december
2018', 5772.6, 7, null, 180, 'MAN');
```

INSERT INTO angajat

```
VALUES(1153, 'Stan', 'Elena', 'elena.stan@gmail.com', '0789.123.433', '13 february  
2019', 4129.7, 10, 1136, 180, 'ING_S');  
INSERT INTO angajat  
VALUES(1170, 'Pitco', 'Natanael', 'nate.pitco@gmail.com', null, '2 may 2018', 6191.0, 6,  
null, 200, 'MAN');  
INSERT INTO angajat  
VALUES(1187, 'Florea', 'Ana', 'ana.florea@gmail.com', null, '9 july 2020', 5972.5, 8,  
null, 240, 'MAN');  
INSERT INTO angajat  
VALUES(1204, 'Nistor', 'Viviana', 'viv.nistor@gmail.com', '0712.385.763', '12 april  
2022', 3472.5, 6, 1187, 240, 'ING_T');  
INSERT INTO angajat  
VALUES(1221, 'Leona', 'Antonia', 'anto.leona@gmail.com', '0711.332.454', '19 august  
2020', 5789.1, 8, null, 260, 'MAN');  
INSERT INTO angajat  
VALUES(1238, 'Nils', 'Corina', 'corina.nils@gmail.com', null, '1 march 2019', 6100.5, 9,  
null, 160, 'MAN');  
INSERT INTO angajat  
VALUES(1255, 'Petre', 'Cosmin', 'cosmin.petre@gmail.com', '0718.223.344', '26 june  
2021', 5972.5, 7, 1255, 160, 'ING_S');  
INSERT INTO angajat  
VALUES(1272, 'Popa', 'Maria', 'maria.popa@yahoo.com', '0792.233.211', '15 may  
2017', 6132.1, 8, null, 220, 'MAN');  
INSERT INTO angajat  
VALUES(1289, 'Dumitran', 'George', 'george.dumi@gmail.com', '0793.557.876', '22  
june 2019', 4432.1, 4, null, 160, 'ING_T');  
INSERT INTO angajat  
VALUES(1306, 'Stanciu', 'Ecaterina', 'eca_stnc@gmail.com', '0732.323.233', '12  
december 2021', 3693.6, 6, 1170, 200, 'ING_S');  
INSERT INTO angajat  
VALUES(1323, 'Chesaru', 'Ana', 'ana.chesaru@yahoo.com', '0723.324.443', '6 october  
2022', 1922, 8, 1272, 220, 'BAR');  
INSERT INTO angajat  
VALUES(1340, 'Toth', 'Raluca', 'raluca_toth@gmail.com', '0799.112.321', '15 february  
2022', 1666.5, 4, 1272, 220, 'BAR');
```

INSERT INTO angajat

```
VALUES(1357, 'Luca', 'Ioan', 'ioan.luca@yahoo.com', '0791.211.322', '7 june 2022',  
4747.1, 8, 1289, 160, 'ING_T');
```

INSERT INTO angajat

```
VALUES(1394, 'Coca', 'Emanuel', 'emanuel.coca@gmail.com', '0790.134.755', '1  
january 2019', 2525.6, 4, 1136, 180, 'PLAS');
```

INSERT INTO angajat

```
VALUES(1411, 'Pitulan', 'Gabriela', 'gabitzap@gmail.com', '0700.212.332', '12  
december 2020', 2999.2, 6, 1238, 160, 'PLAS');
```

INSERT INTO angajat

```
VALUES(1428, 'Miron', 'Luminita', 'lumi.miron@yahoo.com', null, '6 october 2022',  
5555.5, 5, 1272, 220, 'CAS');
```

INSERT INTO angajat

```
VALUES(1445, 'Tanase', 'Vlad', 'vlad.t@gmail.com', '0721.345.788', '1 december 2022',  
5050.3, 6, 1428, 220, 'CAS');
```

INSERT INTO angajat

```
VALUES(1462, 'Tamas', 'Vasilica', 'v.tamas@gmail.com', null, '1 december 2022', 2002,  
6, 1238, 160, 'BAR');
```

```
INSERT INTO angajat  
VALUES(1000, 'Popescu', 'Ion', 'ion.popescu@gmail.com', '0741.311.314', '3 june 2018', 6023.5, 9, null, 120, 'MAN');  
INSERT INTO angajat  
VALUES(1017, 'Tudor', 'Maria', 'maria.tudor@gmail.com', '0712.133.292', '5 september 2017', 5998.8, 12, null, 140, 'MAN')  
INSERT INTO angajat  
VALUES(1034, 'Georgesc', 'Vasile', 'vasile.georgesc@gmail.com', null, '5 may 2019', 1555.1, 6, null, 140, 'AG_C');  
INSERT INTO angajat  
VALUES(1051, 'Matei', 'Andrei', 'andrei.matei@gmail.com', '0799.532.678', '27 january 2020', 4672.5, 8, 1000, 120, 'CAS')  
INSERT INTO angajat  
VALUES(1085, 'Iana', 'Gabriela', 'gabriela.iana@gmail.com', '0712.623.881', '12 april 2021', 3972.5, 8, 1051, 120, 'CAS')  
INSERT INTO angajat  
VALUES(1102, 'Miron', 'Costin', 'costin.miron@gmail.com', '0771.229.345', '19 november 2019', 5789.1, 9, null, 260, 'MAN')  
INSERT INTO angajat  
VALUES(1119, 'Banu', 'Codrin', 'codrin.banu@gmail.com', '0741.721.921', '19 november 2019', 1231.1, 4, 1102, 260, 'ING_S')  
INSERT INTO angajat  
VALUES(1136, 'Crivat', 'Gheorghe', 'gheo.crivat@gmail.com', null, '21 december 2018', 5772.6, 7, null, 180, 'MAN');  
INSERT INTO angajat  
VALUES(1153, 'Stan', 'Elena', 'elena.stan@gmail.com', '0789.123.433', '13 february 2019', 4129.7, 10, 1136, 180, 'ING_S')  
INSERT INTO angajat  
VALUES(1170, 'Pitco', 'Natanael', 'nate.pitco@gmail.com', null, '2 may 2018', 6191.0, 6, null, 200, 'MAN');  
INSERT INTO angajat  
VALUES(1187, 'Florea', 'Ana', 'ana.florea@qmail.com', null, '9 july 2020', 5972.5, 8, null, 240, 'MAN');
```

```

INSERT INTO angajat
VALUES(1204, 'Nistor', 'Viviana', 'viv.nistor@gmail.com', '0712.385.763', '12 april 2022', 3472.5, 6, 1187, 240, 'ING_T');
INSERT INTO angajat
VALUES(1221, 'Leona', 'Antonia', 'anto.leona@gmail.com', '0711.332.454', '19 august 2020', 5789.1, 8, null, 260, 'MAN');
INSERT INTO angajat
VALUES(1238, 'Nils', 'Corina', 'corina.nils@gmail.com', null, '1 march 2019', 6100.5, 9, null, 160, 'MAN');
INSERT INTO angajat
VALUES(1255, 'Petre', 'Cosmin', 'cosmin.petre@gmail.com', '0718.223.344', '26 june 2021', 5972.5, 7, 1255, 160, 'ING_S');
INSERT INTO angajat
VALUES(1272, 'Popa', 'Maria', 'maria.popa@yahoo.com', '0792.233.211', '15 may 2017', 6132.1, 8, null, 220, 'MAN');
INSERT INTO angajat
VALUES(1289, 'Dumitran', 'George', 'george.dumi@gmail.com', '0793.557.876', '22 june 2019', 4432.1, 4, null, 160, 'ING_T');
INSERT INTO angajat
VALUES(1306, 'Stanciu', 'Ecaterina', 'eca_stnc@gmail.com', '0732.323.233', '12 december 2021', 3693.6, 6, 1170, 200, 'ING_S');
INSERT INTO angajat
VALUES(1323, 'Chesaru', 'Ana', 'ana.chesaru@yahoo.com', '0723.324.443', '6 october 2022', 1922, 8, 1272, 220, 'BAR');
INSERT INTO angajat
VALUES(1340, 'Toth', 'Raluca', 'raluca_toth@gmail.com', '0799.112.321', '15 february 2022', 1666.5, 4, 1272, 220, 'BAR');
INSERT INTO angajat
VALUES(1357, 'Luca', 'Ioan', 'ioan.luca@yahoo.com', '0791.211.322', '7 june 2022', 4747.1, 8, 1289, 160, 'ING_T');


```

```

INSERT INTO angajat
VALUES(1394, 'Coca', 'Emanuel', 'emanuel.coca@gmail.com', '0790.134.755', '1 january 2019', 2525.6, 4, 1136, 180, 'PLAS');
INSERT INTO angajat
VALUES(1411, 'Pitulan', 'Gabriela', 'gabitzap@gmail.com', '0700.212.332', '12 december 2020', 2999.2, 6, 1238, 160, 'PLAS');
INSERT INTO angajat
VALUES(1428, 'Miron', 'Luminita', 'lumi.miron@yahoo.com', null, '6 october 2022', 5555.5, 5, 1272, 220, 'CAS');
INSERT INTO angajat
VALUES(1445, 'Tanase', 'Vlad', 'vlad.t@gmail.com', '0721.345.788', '1 december 2022', 5050.3, 6, 1428, 220, 'CAS');
INSERT INTO angajat
VALUES(1462, 'Tamas', 'Vasilica', 'v.tamas@gmail.com', null, '1 december 2022', 2002, 6, 1238, 160, 'BAR');


```

Script Output | Query Result | Task completed in 0.18 seconds

1 row inserted.

1 row inserted.

```
SELECT * FROM angajat;
```

Script Output | Query Result | All Rows Fetched: 26 in 0.004 seconds

| | COD_ANGAJAT | NUME | PRENUME | EMAIL | TELEFON | DATA_ANG | SALARIU | NR_ORE | ID_SUPERIOR | COD_CINEMA | COD_JOB |
|----|-------------|----------|----------|---------------------------|--------------|-----------|---------|--------|-------------|------------|---------|
| 1 | 1000 | Popescu | Ion | ion.popescu@gmail.com | 0741.311.314 | 03-JUN-18 | 6023.5 | 9 | (null) | 120 | MAN |
| 2 | 1017 | Tudor | Maria | maria.tudor@gmail.com | 0712.133.292 | 05-SEP-17 | 5998.8 | 12 | (null) | 140 | MAN |
| 3 | 1034 | Georgesc | Vasile | vasile.georgesc@gmail.com | (null) | 05-MAY-19 | 1555.1 | 6 | (null) | 140 | AG_C |
| 4 | 1051 | Matei | Andrei | andrei.matei@gmail.com | 0799.532.678 | 27-JAN-20 | 4672.5 | 8 | 1000 | 120 | CAS |
| 5 | 1085 | Iana | Gabriela | gabriela.iana@gmail.com | 0712.623.881 | 12-APR-21 | 3972.5 | 8 | 1051 | 120 | CAS |
| 6 | 1102 | Miron | Costin | costin.miron@gmail.com | 0771.229.345 | 19-NOV-19 | 5789.1 | 9 | (null) | 260 | MAN |
| 7 | 1119 | Banu | Codrin | codrin.banu@gmail.com | 0741.721.921 | 19-NOV-19 | 1231.1 | 4 | 1102 | 260 | ING_S |
| 8 | 1136 | Crivat | Gheorghe | gheo.crivat@gmail.com | (null) | 21-DEC-18 | 5772.6 | 7 | (null) | 180 | MAN |
| 9 | 1153 | Stan | Elena | elena.stan@gmail.com | 0789.123.433 | 13-FEB-19 | 4129.7 | 10 | 1136 | 180 | ING_S |
| 10 | 1170 | Pitco | Natanael | nate.pitco@gmail.com | (null) | 02-MAY-18 | 6191 | 6 | (null) | 200 | MAN |
| 11 | 1187 | Florea | Ana | ana.florea@gmail.com | (null) | 09-JUL-20 | 5972.5 | 8 | (null) | 240 | MAN |
| 12 | 1204 | Nistor | Viviana | viv.nistor@gmail.com | 0712.385.763 | 12-APR-22 | 3472.5 | 6 | 1187 | 240 | ING_T |
| 13 | 1221 | Leona | Antonia | anto.leona@gmail.com | 0711.332.454 | 19-AUG-20 | 5789.1 | 8 | (null) | 260 | MAN |
| 14 | 1238 | Nils | Corina | corina.nils@gmail.com | (null) | 01-MAR-19 | 6100.5 | 9 | (null) | 160 | MAN |

| | | | | | | | | | | | |
|----|------|----------|-----------|------------------------|--------------|-----------|--------|---|--------|-----|-------|
| 15 | 1255 | Petre | Cosmin | cosmin.petre@gmail.com | 0718.223.344 | 26-JUN-21 | 5972.5 | 7 | 1255 | 160 | ING_S |
| 16 | 1272 | Popa | Maria | maria.popa@yahoo.com | 0792.233.211 | 15-MAY-17 | 6132.1 | 8 | (null) | 220 | MAN |
| 17 | 1289 | Dumitran | George | george.dumi@gmail.com | 0793.557.876 | 22-JUN-19 | 4432.1 | 4 | (null) | 160 | ING_T |
| 18 | 1306 | Stanciu | Ecaterina | eca_stnc@gmail.com | 0732.323.233 | 12-DEC-21 | 3693.6 | 6 | 1170 | 200 | ING_S |
| 19 | 1323 | Chesaru | Ana | ana.chesaru@yahoo.com | 0723.324.443 | 06-OCT-22 | 1922 | 8 | 1272 | 220 | BAR |
| 20 | 1340 | Toth | Raluca | raluca_toth@gmail.com | 0799.112.321 | 15-FEB-22 | 1666.5 | 4 | 1272 | 220 | BAR |
| 21 | 1357 | Luca | Ioan | ioan.luca@yahoo.com | 0791.211.322 | 07-JUN-22 | 4747.1 | 8 | 1289 | 160 | ING_T |
| 22 | 1394 | Coca | Emanuel | emanuel.coca@gmail.com | 0790.134.755 | 01-JAN-19 | 2525.6 | 4 | 1136 | 180 | PLAS |
| 23 | 1411 | Pitulan | Gabriela | gabitzap@gmail.com | 0700.212.332 | 12-DEC-20 | 2999.2 | 6 | 1238 | 160 | PLAS |
| 24 | 1428 | Miron | Luminita | lumi.miron@yahoo.com | (null) | 06-OCT-22 | 5555.5 | 5 | 1272 | 220 | CAS |
| 25 | 1445 | Tanase | Vlad | vlad.t@gmail.com | 0721.345.788 | 01-DEC-22 | 5050.3 | 6 | 1428 | 220 | CAS |
| 26 | 1462 | Tamas | Vasilica | v.tamas@gmail.com | (null) | 01-DEC-22 | 2002 | 6 | 1238 | 160 | BAR |

Inserări în tabela asociativă **ISTORIC_LUCRU**:

INSERT INTO istoric_lucru

VALUES(1034, '3 may 2018', '5 may 2019', 120, 'AG_C');

INSERT INTO istoric_lucru

VALUES(1102, '17 july 2019', '18 november 2019', 120, 'PLAS');

INSERT INTO istoric_lucru

VALUES(1221, '19 november 2018', '19 august 2020', 260, 'MAN');

INSERT INTO istoric_lucru

VALUES(1255, '7 november 2020', '20 june 2021', 160, 'BAR');

INSERT INTO istoric_lucru

VALUES(1119, '7 may 2014', '7 june 2015', 260, 'ING_S');

INSERT INTO istoric_lucru

VALUES(1119, '8 june 2015', '18 november 2019', 260, 'CAS');

INSERT INTO istoric_lucru

VALUES(1289, '20 march 2019', '13 june 2019', 240, 'ING_T');

INSERT INTO istoric_lucru

VALUES(1462, '15 july 2022', '29 november 2022', 220, 'BAR_C');

INSERT INTO istoric_lucru

VALUES(1411, '7 may 2018', '11 december 2020', 260, 'PLAS');

INSERT INTO istoric_lucru

VALUES(1340, '9 december 2021', '9 february 2022', 160, 'BAR_C');

```

INSERT INTO istoric_lucru
VALUES(1034, '3 may 2018', '5 may 2019', 120, 'AG_C');
INSERT INTO istoric_lucru
VALUES(1102, '17 july 2019', '18 november 2019', 120, 'PLAS');
INSERT INTO istoric_lucru
VALUES(1221, '19 november 2018', '19 august 2020', 260, 'MAN');
INSERT INTO istoric_lucru
VALUES(1255, '7 november 2020', '20 june 2021', 160, 'BAR');
INSERT INTO istoric_lucru
VALUES(1119, '7 may 2014', '7 june 2015', 260, 'ING_S');
INSERT INTO istoric_lucru
VALUES(1119, '8 june 2015', '18 november 2019', 260, 'CAS');
INSERT INTO istoric_lucru
VALUES(1289, '20 march 2019', '13 june 2019', 240, 'ING_T');
INSERT INTO istoric_lucru
VALUES(1462, '15 july 2022', '29 november 2022', 220, 'BAR_C');
INSERT INTO istoric_lucru
VALUES(1411, '7 may 2018', '11 december 2020', 260, 'PLAS');
INSERT INTO istoric_lucru
VALUES(1340, '9 december 2021', '9 february 2022', 160, 'BAR_C');

```

```

SELECT * FROM istoric_lucru;

```

Script Output x Query Result x

SQL | All Rows Fetched: 10 in 0.036 seconds

| | COD_ANGAJAT | DATA_START | DATA_DEMISIE | COD_CINEMA | COD_JOB |
|----|-------------|------------|--------------|------------|---------|
| 1 | 1034 | 03-MAY-18 | 05-MAY-19 | 120 | AG_C |
| 2 | 1102 | 17-JUL-19 | 18-NOV-19 | 120 | PLAS |
| 3 | 1221 | 19-NOV-18 | 19-AUG-20 | 260 | MAN |
| 4 | 1255 | 07-NOV-20 | 20-JUN-21 | 160 | BAR |
| 5 | 1119 | 07-MAY-14 | 07-JUN-15 | 260 | ING_S |
| 6 | 1119 | 08-JUN-15 | 18-NOV-19 | 260 | CAS |
| 7 | 1289 | 20-MAR-19 | 13-JUN-19 | 240 | ING_T |
| 8 | 1462 | 15-JUL-22 | 29-NOV-22 | 220 | BAR_C |
| 9 | 1411 | 07-MAY-18 | 11-DEC-20 | 260 | PLAS |
| 10 | 1340 | 09-DEC-21 | 09-FEB-22 | 160 | BAR_C |

Inserări în tabela SALA:

INSERT INTO sala

VALUES(1, 120, 'VIP');

INSERT INTO sala

VALUES(4, 120, 'Multiplex');

INSERT INTO sala

VALUES(7, 120, 'Megaplex');

INSERT INTO sala

VALUES(10, 140, 'Multiplex');

```
INSERT INTO sala
VALUES(13, 160, 'Complex');

INSERT INTO sala
VALUES(16, 160, 'VIP');

INSERT INTO sala
VALUES(19, 180, 'Megaplex');

INSERT INTO sala
VALUES(22, 200, 'Multiplex');

INSERT INTO sala
VALUES(25, 240, 'Complex');

INSERT INTO sala
VALUES(28, 220, 'VIP');

INSERT INTO sala
VALUES(31, 220, 'VIP');

INSERT INTO sala
VALUES(34, 200, 'Megaplex');

INSERT INTO sala
VALUES(37, 260, 'Multiplex');

INSERT INTO sala
VALUES(40, 260, 'Complex');

INSERT INTO sala
VALUES(4, 140, 'Complex');

INSERT INTO sala
VALUES(7, 200, 'Complex');

INSERT INTO sala
VALUES(10, 120, 'Megaplex');

INSERT INTO sala
VALUES(13, 120, 'VIP');

INSERT INTO sala
VALUES(16, 260, 'VIP');

INSERT INTO sala
VALUES(19, 220, 'Multiplex');
```

```

INSERT INTO sala
VALUES(1, 120, 'VIP');
INSERT INTO sala
VALUES(4, 120, 'Multiplex');
INSERT INTO sala
VALUES(7, 120, 'Megaplex');
INSERT INTO sala
VALUES(10, 140, 'Multiplex');
INSERT INTO sala
VALUES(13, 160, 'Complex');
INSERT INTO sala
VALUES(16, 160, 'VIP');
INSERT INTO sala
VALUES(19, 180, 'Megaplex');
INSERT INTO sala
VALUES(22, 200, 'Multiplex');
INSERT INTO sala
VALUES(25, 240, 'Complex');
INSERT INTO sala
VALUES(28, 220, 'VIP');

```

```

INSERT INTO sala
VALUES(31, 220, 'VIP');
INSERT INTO sala
VALUES(34, 200, 'Megaplex');
INSERT INTO sala
VALUES(37, 260, 'Multiplex');
INSERT INTO sala
VALUES(40, 260, 'Complex');
INSERT INTO sala
VALUES(4, 140, 'Complex');
INSERT INTO sala
VALUES(7, 200, 'Complex');
INSERT INTO sala
VALUES(10, 120, 'Megaplex');
INSERT INTO sala
VALUES(13, 120, 'VIP');
INSERT INTO sala
VALUES(16, 260, 'VIP');
INSERT INTO sala
VALUES(19, 220, 'Multiplex');

```

```
SELECT * FROM sala;
```

Script Output x Query Result x

SQL | All Rows Fetched: 20 in 0.012 seconds

| | NR_SALA | COD_CINEMA | TIP |
|---|---------|------------|-----------|
| 1 | 1 | 120 | VIP |
| 2 | 4 | 120 | Multiplex |
| 3 | 7 | 120 | Megaplex |
| 4 | 10 | 140 | Multiplex |
| 5 | 13 | 160 | Complex |
| 6 | 16 | 160 | VIP |
| 7 | 19 | 180 | Megaplex |
| 8 | 22 | 200 | Multiplex |
| 9 | 25 | 240 | Complex |

| | | |
|----|----|---------------|
| 10 | 28 | 220 VIP |
| 11 | 31 | 220 VIP |
| 12 | 34 | 200 Megaplex |
| 13 | 37 | 260 Multiplex |
| 14 | 40 | 260 Complex |
| 15 | 4 | 140 Complex |
| 16 | 7 | 200 Complex |
| 17 | 10 | 120 Megaplex |
| 18 | 13 | 120 VIP |
| 19 | 16 | 260 VIP |
| 20 | 19 | 220 Multiplex |

Inserări în tabela FILM:

INSERT INTO film

VALUES(10, 'Uncharted', 'Actiune', 116, 6.4, 2022);

INSERT INTO film

VALUES(110, 'Spiderman: No Way Home', 'Actiune', 148, 8.4, 2021);

INSERT INTO film

VALUES(210, 'Free Guy', 'Actiune', 115, 7.1, 2021);

INSERT INTO film

VALUES(310, 'Little Women', 'Romance', 135, 7.8, 2019);

INSERT INTO film

VALUES(410, 'It', 'Horror', 135, 7.4, 2017);

INSERT INTO film

VALUES(510, 'Encanto', 'Animatie', 109, 7.8, 2021);

INSERT INTO film

VALUES(610, 'Star Wars: Episode III', 'Science Fiction', 140, 7.6, 2005);

INSERT INTO film

VALUES(710, 'Black Widow', 'Actiune', 134, 6.4, 2021);

INSERT INTO film

VALUES(810, 'Titanic', 'Drama', 194, 7.9, 1997);

INSERT INTO film

VALUES(910, 'The Batman', 'Crima', 176, 8.0, 2022);

INSERT INTO film

VALUES(1010, 'Red Notice', 'Comedie', 118, 6.3, 2021);

INSERT INTO film

VALUES(1110, 'Venom', 'Actiune', 112, 7.9, 2018);

```

INSERT INTO film
VALUES(1210, 'Crimson Peak', 'Horror', 119, 7.0, 2015);

INSERT INTO film
VALUES(1310, 'Yes Day', 'Familie', 146, 5.7, 2021);

INSERT INTO film
VALUES(1410, 'Zootopia', 'Animatie', 108, 8.5, 2016);

INSERT INTO film
VALUES(1510, 'Doctor Strange', 'Science Fiction', 116, 7.5, 2016);

INSERT INTO film
VALUES(1610, 'It chapter 2', 'Horror', 169, 6.5, 2019);

INSERT INTO film
VALUES(1710, 'Death on the Nile', 'Mister', 127, 6.9, 2022);

INSERT INTO film
VALUES(1810, 'Sherlock Holmes', 'Mister', 128, 8.6, 2009);

INSERT INTO film
VALUES(1910, 'Divergent', 'Romance', 140, 8.1, 2014);

```

| |
|--|
| <pre> INSERT INTO film VALUES(10, 'Uncharted', 'Actiune', 116, 6.4, 2022); INSERT INTO film VALUES(110, 'Spiderman: No Way Home', 'Actiune', 148, 8.4, 2021); INSERT INTO film VALUES(210, 'Free Guy', 'Actiune', 115, 7.1, 2021); INSERT INTO film VALUES(310, 'Little Women', 'Romance', 135, 7.8, 2019); INSERT INTO film VALUES(410, 'It', 'Horror', 135, 7.4, 2017); INSERT INTO film VALUES(510, 'Encanto', 'Animatie', 109, 7.8, 2021); INSERT INTO film VALUES(610, 'Star Wars: Episode III', 'Science Fiction', 140, 7.6, 2005); INSERT INTO film VALUES(710, 'Black Widow', 'Actiune', 134, 6.4, 2021); INSERT INTO film VALUES(810, 'Titanic', 'Drama', 194, 7.9, 1997); INSERT INTO film VALUES(910, 'The Batman', 'Crima', 176, 8.0, 2022); INSERT INTO film VALUES(1010, 'Red Notice', 'Comedie', 118, 6.3, 2021); </pre> |
|--|

```

INSERT INTO film
VALUES(1110, 'Venom', 'Actiune', 112, 7.9, 2018);
INSERT INTO film
VALUES(1210, 'Crimson Peak', 'Horror', 119, 7.0, 2015);
INSERT INTO film
VALUES(1310, 'Yes Day', 'Familie', 146, 5.7, 2021);
INSERT INTO film
VALUES(1410, 'Zootopia', 'Animatie', 108, 8.5, 2016);
INSERT INTO film
VALUES(1510, 'Doctor Strange', 'Science Fiction', 116, 7.5, 2016);
INSERT INTO film
VALUES(1610, 'It chapter 2', 'Horror', 169, 6.5, 2019);
INSERT INTO film
VALUES(1710, 'Death on the Nile', 'Mister', 127, 6.9, 2022);
INSERT INTO film
VALUES(1810, 'Sherlock Holmes', 'Mister', 128, 8.6, 2009);
INSERT INTO film
VALUES(1910, 'Divergent', 'Romance', 140, 8.1, 2014);

```

SELECT * FROM film;

| COD_FILM | NUME_FILM | GEN_PRINC | DURATA | RATING | AN_APARITIE |
|----------|----------------------------|-----------------|--------|--------|-------------|
| 1 | 10 Uncharted | Actiune | 116 | 6.4 | 2022 |
| 2 | 110 Spiderman: No Way Home | Actiune | 148 | 8.4 | 2021 |
| 3 | 210 Free Guy | Actiune | 115 | 7.1 | 2021 |
| 4 | 310 Little Women | Romance | 135 | 7.8 | 2019 |
| 5 | 410 It | Horror | 135 | 7.4 | 2017 |
| 6 | 510 Encanto | Animatie | 109 | 7.8 | 2021 |
| 7 | 610 Star Wars: Episode III | Science Fiction | 140 | 7.6 | 2005 |
| 8 | 710 Black Widow | Actiune | 134 | 6.4 | 2021 |
| 9 | 810 Titanic | Drama | 194 | 7.9 | 1997 |
| 10 | 910 The Batman | Crima | 176 | 8 | 2022 |
| 11 | 1010 Red Notice | Comedie | 118 | 6.3 | 2021 |
| 12 | 1110 Venom | Actiune | 112 | 7.9 | 2018 |
| 13 | 1210 Crimson Peak | Horror | 119 | 7 | 2015 |
| 14 | 1310 Yes Day | Familie | 146 | 5.7 | 2021 |
| 15 | 1410 Zootopia | Animatie | 108 | 8.5 | 2016 |
| 16 | 1510 Doctor Strange | Science Fiction | 116 | 7.5 | 2016 |
| 17 | 1610 It chapter 2 | Horror | 169 | 6.5 | 2019 |
| 18 | 1710 Death on the Nile | Mister | 127 | 6.9 | 2022 |
| 19 | 1810 Sherlock Holmes | Mister | 128 | 8.6 | 2009 |
| 20 | 1910 Divergent | Romance | 140 | 8.1 | 2014 |

Inserări în tabela asociativă RULEAZA:

INSERT INTO ruleaza

VALUES(260, 120, 10, '2 march 2022', '13 may 2022');

INSERT INTO ruleaza

VALUES(300, 120, 1210, '21 june 2018', '13 january 2019');

INSERT INTO ruleaza

VALUES(340, 140, 1110, '17 july 2020', '23 november 2020');

INSERT INTO ruleaza

VALUES(380, 180, 1110, '17 july 2020', '23 november 2020');

INSERT INTO ruleaza

VALUES(420, 200, 810, '15 april 2018', '19 may 2019');

```
INSERT INTO ruleaza
VALUES(460, 200, 410, '5 september 2017', '16 december 2017');

INSERT INTO ruleaza
VALUES(500, 200, 1610, '15 november 2019', '8 february 2020');

INSERT INTO ruleaza
VALUES(540, 160, 1610, '15 november 2019', '8 february 2020');

INSERT INTO ruleaza
VALUES(580, 140, 710, '28 may 2021', '13 october 2021');

INSERT INTO ruleaza
VALUES(620, 180, 1210, '23 december 2015', '27 may 2016');

INSERT INTO ruleaza
VALUES(660, 180, 510, '1 january 2022', '23 april 2022');

INSERT INTO ruleaza
VALUES(700, 220, 910, '15 march 2022', '23 july 2022');

INSERT INTO ruleaza
VALUES(740, 220, 1710, '15 april 2022', '3 september 2022');

INSERT INTO ruleaza
VALUES(780, 200, 610, '5 june 2015', '18 august 2015');

INSERT INTO ruleaza
VALUES(820, 240, 610, '6 june 2015', '18 september 2015');

INSERT INTO ruleaza
VALUES(860, 160, 1310, '6 april 2021', '14 november 2021');

INSERT INTO ruleaza
VALUES(900, 120, 1610, '15 november 2019', '8 february 2020');

INSERT INTO ruleaza
VALUES(940, 200, 710, '28 may 2021', '13 october 2021');

INSERT INTO ruleaza
VALUES(980, 240, 1210, '23 december 2015', '27 may 2016');

INSERT INTO ruleaza
VALUES(1020, 240, 510, '1 january 2022', '23 april 2022');

INSERT INTO ruleaza
VALUES(1060, 260, 1510, '19 may 2017', '21 december 2017');

INSERT INTO ruleaza
```

```
VALUES(1100, 260, 310, '15 august 2019', '5 january 2020');  
INSERT INTO ruleaza  
VALUES(1140, 120, 310, '15 august 2019', '5 january 2020');  
INSERT INTO ruleaza  
VALUES(1180, 160, 310, '16 august 2019', '5 january 2020');  
INSERT INTO ruleaza  
VALUES(1220, 220, 1010, '15 september 2021', '14 february 2022');  
INSERT INTO ruleaza  
VALUES(1260, 240, 1010, '17 september 2021', '14 february 2022');  
INSERT INTO ruleaza  
VALUES(1300, 120, 1010, '16 september 2021', '11 february 2022');  
INSERT INTO ruleaza  
VALUES(1340, 140, 1310, '6 april 2021', '14 november 2021');  
INSERT INTO ruleaza  
VALUES(1380, 140, 110, '16 december 2021', '14 february 2022');  
INSERT INTO ruleaza  
VALUES(1420, 160, 210, '6 july 2021', '19 october 2021');  
INSERT INTO ruleaza  
VALUES(1460, 120, 810, '7 may 2021', '13 september 2021');  
INSERT INTO ruleaza  
VALUES(1500, 120, 1410, '14 february 2017', '18 may 2017');  
INSERT INTO ruleaza  
VALUES(1540, 140, 410, '7 december 2017', '12 june 2018');  
INSERT INTO ruleaza  
VALUES(1580, 140, 510, '3 june 2022', '2 september 2022');  
INSERT INTO ruleaza  
VALUES(1620, 140, 310, '1 january 2020', '15 march 2020');  
INSERT INTO ruleaza  
VALUES(1660, 140, 1810, '31 october 2020', '30 december 2020');  
INSERT INTO ruleaza  
VALUES(1700, 160, 510, '1 january 2022', '3 march 2022');  
INSERT INTO ruleaza  
VALUES(1740, 180, 1110, '14 january 2021', '14 march 2021');
```

```

INSERT INTO ruleaza
VALUES(1780, 180, 610, '21 august 2021', '15 november 2021');

INSERT INTO ruleaza
VALUES(1820, 180, 1910, '5 july 2017', '15 november 2017');

INSERT INTO ruleaza
VALUES(1860, 180, 710, '3 june 2021', '3 december 2021');

INSERT INTO ruleaza
VALUES(1900, 200, 1010, '14 february 2022', '15 may 2022');

INSERT INTO ruleaza
VALUES(1940, 240, 10, '14 july 2022', '14 august 2022');

INSERT INTO ruleaza
VALUES(1980, 240, 1910, '4 april 2021', '14 july 2021');

```

```

INSERT INTO ruleaza
VALUES(260, 120, 10, '2 march 2022', '13 may 2022');
INSERT INTO ruleaza
VALUES(300, 120, 1210, '21 june 2018', '13 january 2019');
INSERT INTO ruleaza
VALUES(340, 140, 1110, '17 july 2020', '23 november 2020');
INSERT INTO ruleaza
VALUES(380, 180, 1110, '17 july 2020', '23 november 2020');
INSERT INTO ruleaza
VALUES(420, 200, 810, '15 april 2018', '19 may 2019');
INSERT INTO ruleaza
VALUES(460, 200, 410, '5 september 2017', '16 december 2017');
INSERT INTO ruleaza
VALUES(500, 200, 1610, '15 november 2019', '8 february 2020');
INSERT INTO ruleaza
VALUES(540, 160, 1610, '15 november 2019', '8 february 2020');
INSERT INTO ruleaza
VALUES(580, 140, 710, '28 may 2021', '13 october 2021');
INSERT INTO ruleaza
VALUES(620, 180, 1210, '23 december 2015', '27 may 2016');
INSERT INTO ruleaza
VALUES(660, 180, 510, '1 january 2022', '23 april 2022');

```

```

INSERT INTO ruleaza
VALUES(700, 220, 910, '15 march 2022', '23 july 2022');
INSERT INTO ruleaza
VALUES(740, 220, 1710, '15 april 2022', '3 september 2022');
INSERT INTO ruleaza
VALUES(780, 200, 610, '5 june 2015', '18 august 2015');
INSERT INTO ruleaza
VALUES(820, 240, 610, '6 june 2015', '18 september 2015');
INSERT INTO ruleaza
VALUES(860, 160, 1310, '6 april 2021', '14 november 2021');
INSERT INTO ruleaza
VALUES(900, 120, 1610, '15 november 2019', '8 february 2020');
INSERT INTO ruleaza
VALUES(940, 200, 710, '28 may 2021', '13 october 2021');
INSERT INTO ruleaza
VALUES(980, 240, 1210, '23 december 2015', '27 may 2016');
INSERT INTO ruleaza
VALUES(1020, 240, 510, '1 january 2022', '23 april 2022');
INSERT INTO ruleaza
VALUES(1060, 260, 1510, '19 may 2017', '21 december 2017');
INSERT INTO ruleaza
VALUES(1100, 260, 310, '15 august 2019', '5 january 2020');

```

```

INSERT INTO ruleaza
VALUES(1140, 120, 310, '15 august 2019', '5 january 2020');
INSERT INTO ruleaza
VALUES(1180, 160, 310, '16 august 2019', '5 january 2020');
INSERT INTO ruleaza
VALUES(1220, 220, 1010, '15 september 2021', '14 february 2022');
INSERT INTO ruleaza
VALUES(1260, 240, 1010, '17 september 2021', '14 february 2022');
INSERT INTO ruleaza
VALUES(1300, 120, 1010, '16 september 2021', '11 february 2022');
INSERT INTO ruleaza
VALUES(1340, 140, 1310, '6 april 2021', '14 november 2021');
INSERT INTO ruleaza
VALUES(1380, 140, 110, '16 december 2021', '14 february 2022');
INSERT INTO ruleaza
VALUES(1420, 160, 210, '6 july 2021', '19 october 2021');
INSERT INTO ruleaza
VALUES(1460, 120, 810, '7 may 2021', '13 september 2021');
INSERT INTO ruleaza
VALUES(1500, 120, 1410, '14 february 2017', '18 may 2017');
INSERT INTO ruleaza
VALUES(1540, 140, 410, '7 december 2017', '12 june 2018');

```

```

INSERT INTO ruleaza
VALUES(1620, 140, 310, '1 january 2020', '15 march 2020');
INSERT INTO ruleaza
VALUES(1660, 140, 1810, '31 october 2020', '30 december 2020');
INSERT INTO ruleaza
VALUES(1700, 160, 510, '1 january 2022', '3 march 2022');
INSERT INTO ruleaza
VALUES(1740, 180, 1110, '14 january 2021', '14 march 2021');
INSERT INTO ruleaza
VALUES(1780, 180, 610, '21 august 2021', '15 november 2021');
INSERT INTO ruleaza
VALUES(1820, 180, 1910, '5 july 2017', '15 november 2017');
INSERT INTO ruleaza
VALUES(1860, 180, 710, '3 june 2021', '3 december 2021');
INSERT INTO ruleaza
VALUES(1900, 200, 1010, '14 february 2022', '15 may 2022');
INSERT INTO ruleaza
VALUES(1940, 240, 10, '14 july 2022', '14 august 2022');
INSERT INTO ruleaza
VALUES(1980, 240, 1910, '4 april 2021', '14 july 2021');

```

```
SELECT * FROM ruleaza;
```

Script Output x | Query Result x

SQL | All Rows Fetched: 44 in 0.016 seconds

| | COD_RULARE | COD_CINEMA | COD_FILM | DATA_INCEPUT | DATA_FINAL |
|----|------------|------------|----------------|--------------|------------|
| 1 | 260 | 120 | 10 02-MAR-22 | 13-MAY-22 | |
| 2 | 300 | 120 | 1210 21-JUN-18 | 13-JAN-19 | |
| 3 | 340 | 140 | 1110 17-JUL-20 | 23-NOV-20 | |
| 4 | 380 | 180 | 1110 17-JUL-20 | 23-NOV-20 | |
| 5 | 420 | 200 | 810 15-APR-18 | 19-MAY-19 | |
| 6 | 460 | 200 | 410 05-SEP-17 | 16-DEC-17 | |
| 7 | 500 | 200 | 1610 15-NOV-19 | 08-FEB-20 | |
| 8 | 540 | 160 | 1610 15-NOV-19 | 08-FEB-20 | |
| 9 | 580 | 140 | 710 28-MAY-21 | 13-OCT-21 | |
| 10 | 620 | 180 | 1210 23-DEC-15 | 27-MAY-16 | |
| 11 | 660 | 180 | 510 01-JAN-22 | 23-APR-22 | |
| 12 | 700 | 220 | 910 15-MAR-22 | 23-JUL-22 | |
| 13 | 740 | 220 | 1710 15-APR-22 | 03-SEP-22 | |
| 14 | 780 | 200 | 610 05-JUN-15 | 18-AUG-15 | |
| 15 | 820 | 240 | 610 06-JUN-15 | 18-SEP-15 | |
| 16 | 860 | 160 | 1310 06-APR-21 | 14-NOV-21 | |
| 17 | 900 | 120 | 1610 15-NOV-19 | 08-FEB-20 | |
| 18 | 940 | 200 | 710 28-MAY-21 | 13-OCT-21 | |

| | | | | |
|----|------|-----|----------------|-----------|
| 19 | 980 | 240 | 1210 23-DEC-15 | 27-MAY-16 |
| 20 | 1020 | 240 | 510 01-JAN-22 | 23-APR-22 |
| 21 | 1060 | 260 | 1510 19-MAY-17 | 21-DEC-17 |
| 22 | 1100 | 260 | 310 15-AUG-19 | 05-JAN-20 |
| 23 | 1140 | 120 | 310 15-AUG-19 | 05-JAN-20 |
| 24 | 1180 | 160 | 310 16-AUG-19 | 05-JAN-20 |
| 25 | 1220 | 220 | 1010 15-SEP-21 | 14-FEB-22 |
| 26 | 1260 | 240 | 1010 17-SEP-21 | 14-FEB-22 |
| 27 | 1300 | 120 | 1010 16-SEP-21 | 11-FEB-22 |
| 28 | 1340 | 140 | 1310 06-APR-21 | 14-NOV-21 |
| 29 | 1380 | 140 | 110 16-DEC-21 | 14-FEB-22 |
| 30 | 1420 | 160 | 210 06-JUL-21 | 19-OCT-21 |
| 31 | 1460 | 120 | 810 07-MAY-21 | 13-SEP-21 |
| 32 | 1500 | 120 | 1410 14-FEB-17 | 18-MAY-17 |
| 33 | 1540 | 140 | 410 07-DEC-17 | 12-JUN-18 |
| 34 | 1580 | 140 | 510 03-JUN-22 | 02-SEP-22 |
| 35 | 1620 | 140 | 310 01-JAN-20 | 15-MAR-20 |
| 36 | 1660 | 140 | 1810 31-OCT-20 | 30-DEC-20 |
| 37 | 1700 | 160 | 510 01-JAN-22 | 03-MAR-22 |
| 38 | 1740 | 180 | 1110 14-JAN-21 | 14-MAR-21 |
| 39 | 1780 | 180 | 610 21-AUG-21 | 15-NOV-21 |
| 40 | 1820 | 180 | 1910 05-JUL-17 | 15-NOV-17 |
| 41 | 1860 | 180 | 710 03-JUN-21 | 03-DEC-21 |
| 42 | 1900 | 200 | 1010 14-FEB-22 | 15-MAY-22 |
| 43 | 1940 | 240 | 10 14-JUL-22 | 14-AUG-22 |
| 44 | 1980 | 240 | 1910 04-APR-21 | 14-JUL-21 |

Inserări în tabela ACTOR:

INSERT INTO actor

VALUES(5, 'Zendaya', null, 'SUA', 22, 2009);

INSERT INTO actor

VALUES(10, 'Holland', 'Tom', 'EN', 7, 2008);

INSERT INTO actor

VALUES(15, 'Reynolds', 'Ryan', 'CA', 9, 1991);

INSERT INTO actor

VALUES(20, 'Watson', 'Emma', 'EN', 24, 2001);

INSERT INTO actor

VALUES(25, 'Wolfhard', 'Finn', 'CA', 3, 2017);

INSERT INTO actor

VALUES(30, 'Gadot', 'Gal', 'IL', 3, 2004);

INSERT INTO actor

VALUES(35, 'Downey Jr', 'Robert', 'SUA', 29, 1970);

INSERT INTO actor

VALUES(40, 'Pattinson', 'Robert', 'EN', 30, 2005);

```

INSERT INTO actor
VALUES(5, 'Zendaya', null, 'SUA', 22, 2009);
INSERT INTO actor
VALUES(10, 'Holland', 'Tom', 'EN', 7, 2008);
INSERT INTO actor
VALUES(15, 'Reynolds', 'Ryan', 'CA', 9, 1991);
INSERT INTO actor
VALUES(20, 'Watson', 'Emma', 'EN', 24, 2001);
INSERT INTO actor
VALUES(25, 'Wolfhard', 'Finn', 'CA', 3, 2017);
INSERT INTO actor
VALUES(30, 'Gadot', 'Gal', 'IL', 3, 2004);
INSERT INTO actor
VALUES(35, 'Downey Jr', 'Robert', 'SUA', 29, 1970);
INSERT INTO actor
VALUES(40, 'Pattinson', 'Robert', 'EN', 30, 2005);

```

Script Output | Query Result | Task completed in 0.049 seconds
1 row inserted.

```

SELECT * FROM actor;

```

Script Output | Query Result | All Rows Fetched: 8 in 0.004 seconds

| ID_ACTOR | NUME | PRENUME | TARA | NR_PREMII | AN_DEBUT |
|----------|--------------|---------|------|-----------|----------|
| 1 | 5 Zendaya | (null) | SUA | 22 | 2009 |
| 2 | 10 Holland | Tom | EN | 7 | 2008 |
| 3 | 15 Reynolds | Ryan | CA | 9 | 1991 |
| 4 | 20 Watson | Emma | EN | 24 | 2001 |
| 5 | 25 Wolfhard | Finn | CA | 3 | 2017 |
| 6 | 30 Gadot | Gal | IL | 3 | 2004 |
| 7 | 35 Downey Jr | Robert | SUA | 29 | 1970 |
| 8 | 40 Pattinson | Robert | EN | 30 | 2005 |

Inserări în tabela asociativă **JOACA**:

INSERT INTO joaca

VALUES(30, 10, 10, 'principal');

INSERT INTO joaca

VALUES(50, 110, 10, 'principal');

INSERT INTO joaca

VALUES(70, 410, 25, 'principal');

INSERT INTO joaca

VALUES(90, 1610, 25, 'secundar');

INSERT INTO joaca

VALUES(110, 210, 15, 'principal');

INSERT INTO joaca

VALUES(130, 1010, 30, 'principal');

```

INSERT INTO joaca
VALUES(150, 310, 20, 'principal');

INSERT INTO joaca
VALUES(170, 110, 5, 'principal');

INSERT INTO joaca
VALUES(190, 910, 40, 'principal');

INSERT INTO joaca
VALUES(210, 1710, 30, 'principal');

INSERT INTO joaca
VALUES(230, 1810, 35, 'principal');

```

```

INSERT INTO joaca
VALUES(50, 110, 10, 'principal');
INSERT INTO joaca
VALUES(70, 410, 25, 'principal');
INSERT INTO joaca
VALUES(90, 1610, 25, 'secundar');
INSERT INTO joaca
VALUES(110, 210, 15, 'principal');
INSERT INTO joaca
VALUES(130, 1010, 30, 'principal');
INSERT INTO joaca
VALUES(150, 310, 20, 'principal');
INSERT INTO joaca
VALUES(170, 110, 5, 'principal');
INSERT INTO joaca
VALUES(190, 910, 40, 'principal');
INSERT INTO joaca
VALUES(210, 1710, 30, 'principal');
INSERT INTO joaca
VALUES(230, 1810, 35, 'principal');

```

SELECT * FROM joaca;

The screenshot shows the MySQL Workbench interface with the following details:

- Script Output:** Shows the SQL code used to insert data into the 'joaca' table.
- Query Result:** Shows the results of the SELECT query. The table has four columns: COD_JOACA, COD_FILM, ID_ACTOR, and ROL.
- Table Data:**

| COD_JOACA | COD_FILM | ID_ACTOR | ROL |
|-----------|----------|----------|--------------|
| 1 | 30 | 10 | 10 principal |
| 2 | 50 | 110 | 10 principal |
| 3 | 70 | 410 | 25 principal |
| 4 | 90 | 1610 | 25 secundar |
| 5 | 110 | 210 | 15 principal |
| 6 | 130 | 1010 | 30 principal |
| 7 | 150 | 310 | 20 principal |
| 8 | 170 | 110 | 5 principal |
| 9 | 190 | 910 | 40 principal |
| 10 | 210 | 1710 | 30 principal |
| 11 | 230 | 1810 | 35 principal |
- Statistics:** All Rows Fetched: 11 in 0.008 seconds.

Inserări în tabela REGIZOR:

```
INSERT INTO regizor
VALUES(30, 'Watts', 'Jon', 'SUA', 0);
INSERT INTO regizor
VALUES(50, 'Levy', 'Shawn', 'CA', 0);
INSERT INTO regizor
VALUES(70, 'Reeves', 'Matt', 'SUA', 1);
INSERT INTO regizor
VALUES(90, 'Gerwig', 'Greta', 'SUA', 6);
INSERT INTO regizor
VALUES(110, 'Lucas', 'George', 'SUA', 7);
INSERT INTO regizor
VALUES(130, 'Fleischer', 'Ruben', 'SUA', 0);
INSERT INTO regizor
VALUES(150, 'Muschietti', 'Andy', 'AR', 0);
INSERT INTO regizor
VALUES(170, 'Branagh', 'Kenneth', 'EN', 22);
INSERT INTO regizor
VALUES(190, 'Ritchie', 'Guy', 'EN', 5);
INSERT INTO regizor
VALUES(210, 'Serkis', 'Andy', 'EN', 8);
```

```
INSERT INTO regizor
VALUES(30, 'Watts', 'Jon', 'SUA', 0);
INSERT INTO regizor
VALUES(50, 'Levy', 'Shawn', 'CA', 0);
INSERT INTO regizor
VALUES(70, 'Reeves', 'Matt', 'SUA', 1);
INSERT INTO regizor
VALUES(90, 'Gerwig', 'Greta', 'SUA', 6);
INSERT INTO regizor
VALUES(110, 'Lucas', 'George', 'SUA', 7);
INSERT INTO regizor
VALUES(130, 'Fleischer', 'Ruben', 'SUA', 0);
INSERT INTO regizor
VALUES(150, 'Muschietti', 'Andy', 'AR', 0);
INSERT INTO regizor
VALUES(170, 'Branagh', 'Kenneth', 'EN', 22);
INSERT INTO regizor
VALUES(190, 'Ritchie', 'Guy', 'EN', 5);
INSERT INTO regizor
VALUES(210, 'Serkis', 'Andy', 'EN', 8);
```

The screenshot shows the MySQL Workbench interface. In the top-left pane, there is a SQL editor window with the following content:

```
SELECT * FROM regizor;
```

In the bottom-right pane, there is a "Query Result" tab showing the output of the query. The results are presented in a table with the following columns: ID_REGIZOR, NUME, PRENUME, TARA, and NR_PREMII. The data is as follows:

| ID_REGIZOR | NUME | PRENUME | TARA | NR_PREMII |
|------------|----------------|---------|------|-----------|
| 1 | 30 Watts | Jon | SUA | 0 |
| 2 | 50 Levy | Shawn | CA | 0 |
| 3 | 70 Reeves | Matt | SUA | 1 |
| 4 | 90 Gerwig | Greta | SUA | 6 |
| 5 | 110 Lucas | George | SUA | 7 |
| 6 | 130 Fleischer | Ruben | SUA | 0 |
| 7 | 150 Muschietti | Andy | AR | 0 |
| 8 | 170 Branagh | Kenneth | EN | 22 |
| 9 | 190 Ritchie | Guy | EN | 5 |
| 10 | 210 Serkis | Andy | EN | 8 |

Inserări în tabela asociativă REGIZEAZA:

```
INSERT INTO regizeaza
VALUES(1110, 210, 2017);

INSERT INTO regizeaza
VALUES(1110, 130, 2017);

INSERT INTO regizeaza
VALUES(610, 110, 2003);

INSERT INTO regizeaza
VALUES(110, 30, 2020);

INSERT INTO regizeaza
VALUES(210, 50, 2020);

INSERT INTO regizeaza
VALUES(910, 70, 2021);

INSERT INTO regizeaza
VALUES(310, 90, 2017);

INSERT INTO regizeaza
VALUES(10, 130, 2021);

INSERT INTO regizeaza
VALUES(410, 150, 2015);

INSERT INTO regizeaza
VALUES(1610, 150, 2017);

INSERT INTO regizeaza
VALUES(1710, 170, 2020);

INSERT INTO regizeaza
VALUES(1810, 190, 2008);
```

```

INSERT INTO regizeaza
VALUES(1110, 210, 2017);
INSERT INTO regizeaza
VALUES(1110, 130, 2017);
INSERT INTO regizeaza
VALUES(610, 110, 2003);
INSERT INTO regizeaza
VALUES(110, 30, 2020);
INSERT INTO regizeaza
VALUES(210, 50, 2020);
INSERT INTO regizeaza
VALUES(910, 70, 2021);
INSERT INTO regizeaza
VALUES(310, 90, 2017);
INSERT INTO regizeaza
VALUES(10, 130, 2021);
INSERT INTO regizeaza
VALUES(410, 150, 2015);

VALUES(1610, 150, 2017);
INSERT INTO regizeaza
VALUES(1710, 170, 2020);
INSERT INTO regizeaza
VALUES(1810, 190, 2008);

```



1 row inserted.

SELECT * FROM regizeaza;

| | COD_FILM | ID_REGIZOR | AN_INCEPUT |
|----|----------|------------|------------|
| 1 | 1110 | 210 | 2017 |
| 2 | 1110 | 130 | 2017 |
| 3 | 610 | 110 | 2003 |
| 4 | 110 | 30 | 2020 |
| 5 | 210 | 50 | 2020 |
| 6 | 910 | 70 | 2021 |
| 7 | 310 | 90 | 2017 |
| 8 | 10 | 130 | 2021 |
| 9 | 410 | 150 | 2015 |
| 10 | 1610 | 150 | 2017 |
| 11 | 1710 | 170 | 2020 |
| 12 | 1810 | 190 | 2008 |

Inserări în tabela DIFUZARE:

INSERT INTO difuzare

VALUES(50, 10, 1, 120, '3 april 2022', 19);

INSERT INTO difuzare

VALUES(71, 1210, 7, 120, '9 july 2018', 12);

INSERT INTO difuzare

VALUES(92, 710, 10, 140, '5 august 2021', 15);

INSERT INTO difuzare

```
VALUES(113, 1510, 37, 260, '19 may 2017', 9);
INSERT INTO difuzare
VALUES(155, 1110, 4, 140, '3 august 2022', 10);
INSERT INTO difuzare
VALUES(176, 1010, 28, 220, '19 december 2021', 11);
INSERT INTO difuzare
VALUES(197, 610, 34, 200, '18 august 2015', 14);
INSERT INTO difuzare
VALUES(218, 410, 22, 200, '15 november 2017', 9);
INSERT INTO difuzare
VALUES(239, 1310, 16, 160, '9 july 2021', 19);
INSERT INTO difuzare
VALUES(260, 1610, 16, 160, '1 december 2019', 22);
INSERT INTO difuzare
VALUES(281, 1010, 25, 240, '18 september 2021', 11);
INSERT INTO difuzare
VALUES(323, 1210, 25, 240, '1 march 2016', 18);
INSERT INTO difuzare
VALUES(344, 910, 28, 220, '3 july 2022', 17);
INSERT INTO difuzare
VALUES(365, 1210, 4, 120, '7 october 2018', 11);
INSERT INTO difuzare
VALUES(386, 710, 7, 200, '27 june 2021', 16);
INSERT INTO difuzare
VALUES(407, 1510, 40, 260, '7 august 2017', 12);
INSERT INTO difuzare
VALUES(428, 1710, 31, 220, '1 september 2022', 13);
INSERT INTO difuzare
VALUES(449, 1310, 10, 140, '6 june 2021', 22);
INSERT INTO difuzare
VALUES(470, 510, 19, 180, '2 february 2022', 19);
INSERT INTO difuzare
VALUES(491, 1210, 19, 180, '2 february 2022', 12);
```

```

INSERT INTO difuzare
VALUES(50, 10, 1, 120, '3 april 2022', 19);
INSERT INTO difuzare
VALUES(71, 1210, 7, 120, '9 july 2018', 12);
INSERT INTO difuzare
VALUES(92, 710, 10, 140, '5 august 2021', 15);
INSERT INTO difuzare
VALUES(113, 1510, 37, 260, '19 may 2017', 9);
INSERT INTO difuzare
VALUES(155, 1110, 4, 140, '3 august 2022', 10);
INSERT INTO difuzare
VALUES(176, 1010, 28, 220, '19 december 2021', 11);
INSERT INTO difuzare
VALUES(197, 610, 34, 200, '18 august 2015', 14);
INSERT INTO difuzare
VALUES(218, 410, 22, 200, '15 november 2017', 9);
INSERT INTO difuzare
VALUES(239, 1310, 16, 160, '9 july 2021', 19);
INSERT INTO difuzare
VALUES(260, 1610, 16, 160, '1 december 2019', 22);
INSERT INTO difuzare
VALUES(281, 1010, 25, 240, '18 september 2021', 11);

INSERT INTO difuzare
VALUES(323, 1210, 25, 240, '1 march 2016', 18);
INSERT INTO difuzare
VALUES(344, 910, 28, 220, '3 july 2022', 17);
INSERT INTO difuzare
VALUES(365, 1210, 4, 120, '7 october 2018', 11);
INSERT INTO difuzare
VALUES(386, 710, 7, 200, '27 june 2021', 16);
INSERT INTO difuzare
VALUES(407, 1510, 40, 260, '7 august 2017', 12);
INSERT INTO difuzare
VALUES(428, 1710, 31, 220, '1 september 2022', 13);
INSERT INTO difuzare
VALUES(449, 1310, 10, 140, '6 june 2021', 22);
INSERT INTO difuzare
VALUES(470, 510, 19, 180, '2 february 2022', 19);
INSERT INTO difuzare
VALUES(491, 1210, 19, 180, '2 february 2022', 12);

```

Script Output x | Query Result x
 Task completed in 0.271 seconds

1 row inserted.

SELECT * FROM difuzare;

Script Output x | Query Result x
 SQL | All Rows Fetched: 20 in 0.01 seconds

| COD_DIFUZARE | COD_FILM | NR_SALA | COD_CINEMA | DATA_INC | ORA_INC |
|--------------|----------|---------|------------|--------------|---------|
| 1 | 50 | 10 | 1 | 12003-APR-22 | 19 |
| 2 | 71 | 1210 | 7 | 12009-JUL-18 | 12 |
| 3 | 92 | 710 | 10 | 14005-AUG-21 | 15 |
| 4 | 113 | 1510 | 37 | 26019-MAY-17 | 9 |
| 5 | 155 | 1110 | 4 | 14003-AUG-22 | 10 |
| 6 | 176 | 1010 | 28 | 22019-DEC-21 | 11 |
| 7 | 197 | 610 | 34 | 20018-AUG-15 | 14 |
| 8 | 218 | 410 | 22 | 20015-NOV-17 | 9 |
| 9 | 239 | 1310 | 16 | 16009-JUL-21 | 19 |

| | | | | | |
|----|-----|------|----|---------------|----|
| 10 | 260 | 1610 | 16 | 160 01-DEC-19 | 22 |
| 11 | 281 | 1010 | 25 | 240 18-SEP-21 | 11 |
| 12 | 323 | 1210 | 25 | 240 01-MAR-16 | 18 |
| 13 | 344 | 910 | 28 | 220 03-JUL-22 | 17 |
| 14 | 365 | 1210 | 4 | 120 07-OCT-18 | 11 |
| 15 | 386 | 710 | 7 | 200 27-JUN-21 | 16 |
| 16 | 407 | 1510 | 40 | 260 07-AUG-17 | 12 |
| 17 | 428 | 1710 | 31 | 220 01-SEP-22 | 13 |
| 18 | 449 | 1310 | 10 | 140 06-JUN-21 | 22 |
| 19 | 470 | 510 | 19 | 180 02-FEB-22 | 19 |
| 20 | 491 | 1210 | 19 | 180 02-FEB-22 | 12 |

Inserări în tabela **BILET_CUMPARAT**:

```

INSERT INTO bilet_cumparat
VALUES(11, 17.1, 17, 3, 50);
INSERT INTO bilet_cumparat
VALUES(16, 22.5, 1, 1, 386);
INSERT INTO bilet_cumparat
VALUES(21, 18.6, 78, 10, 323);
INSERT INTO bilet_cumparat
VALUES(26, 36.9, 22, 5, 491);
INSERT INTO bilet_cumparat
VALUES(31, 0.0, 15, 2, 491);
INSERT INTO bilet_cumparat
VALUES(36, 24.3, 98, 14, 113);
INSERT INTO bilet_cumparat
VALUES(41, 22.5, 56, 4, 176);
INSERT INTO bilet_cumparat
VALUES(46, 40.1, 27, 3, 92);
INSERT INTO bilet_cumparat
VALUES(51, 10.4, 61, 7, 260);
INSERT INTO bilet_cumparat
VALUES(56, 12.2, 15, 3, 407);

```

```

INSERT INTO bilet_cumparat
VALUES(11, 17.1, 17, 3, 50);
INSERT INTO bilet_cumparat
VALUES(16, 22.5, 1, 1, 386);
INSERT INTO bilet_cumparat
VALUES(21, 18.6, 78, 10, 323);
INSERT INTO bilet_cumparat
VALUES(26, 36.9, 22, 5, 491);
INSERT INTO bilet_cumparat
VALUES(31, 0.0, 15, 2, 491);
INSERT INTO bilet_cumparat
VALUES(36, 24.3, 98, 14, 113);
INSERT INTO bilet_cumparat
VALUES(41, 22.5, 56, 4, 176);
INSERT INTO bilet_cumparat
VALUES(46, 40.1, 27, 3, 92);
INSERT INTO bilet_cumparat
VALUES(51, 10.4, 61, 7, 260);
INSERT INTO bilet_cumparat
VALUES(56, 12.2, 15, 3, 407);

```

```

SELECT * FROM bilet_cumparat;

```

Script Output x Query Result x

SQL | All Rows Fetched: 10 in 0.009 seconds

| | COD_BILET | PRET | NR_LOC | NR_RAND | COD_DIFUZARE |
|----|-----------|------|--------|---------|--------------|
| 1 | 11 | 17.1 | 17 | 3 | 50 |
| 2 | 16 | 22.5 | 1 | 1 | 386 |
| 3 | 21 | 18.6 | 78 | 10 | 323 |
| 4 | 26 | 36.9 | 22 | 5 | 491 |
| 5 | 31 | 0 | 15 | 2 | 491 |
| 6 | 36 | 24.3 | 98 | 14 | 113 |
| 7 | 41 | 22.5 | 56 | 4 | 176 |
| 8 | 46 | 40.1 | 27 | 3 | 92 |
| 9 | 51 | 10.4 | 61 | 7 | 260 |
| 10 | 56 | 12.2 | 15 | 3 | 407 |

Permanentizarea inserărilor:

COMMIT;

```

COMMIT;

```

Script Output x Query Result x

Task completed in 0.03 seconds

Commit complete.

6. Problemă subprogram stocat independent + colecții

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze două tipuri diferite de colecții studiate. Apelați subprogramul.

Pentru fiecare cinematograf din baza de date să se afișeze un top al 3 celor mai bine cotate filme (criteriul de departajare și de clasificare în acest top se va face pe baza ratingului) care au rulat în cadrul acestuia.

Se vor face afișări corespunzătoare și verificări pentru:

- I. cazul în care un cinematograf nu a rulat cel puțin 3 filme până în prezent
- II. cazul în care un cinematograf nu a rulat cel puțin 3 filme DISTINCTE până în prezent
- III. cazul în care un cinematograf a rulat cel puțin 3 filme DISTINCTE până în prezent, dar nu se poate realiza un top 3 deoarece filmele din cadrul acelui cinematograf au ratinguri comune, ce nu permit departajarea pe cel puțin 3 locuri
- IV. cazul în care un cinematograf a rulat cel puțin 3 filme DISTINCTE până în prezent și se poate realiza departajarea acestora pe 3 locuri distincte

--PROCEDURĂ (subprogram stocat), TABEL IMBRICAT ce conține date de tip RECORD, VARRAY, TABEL INDEXAT

```
CREATE OR REPLACE PROCEDURE proc_proiect_ex6_rs IS
  CURSOR c(parametru cinematograf.cod_cinema%TYPE) IS
    SELECT c.nume_cinema, f.nume_film, f.rating, f.an_aparitie
      FROM film f
     JOIN (SELECT DISTINCT cod_cinema, cod_film
           FROM ruleaza) r ON r.cod_film = f.cod_film
    JOIN (SELECT cod_cinema, nume_cinema
          FROM cinematograf) c ON r.cod_cinema = c.cod_cinema
   WHERE c.cod_cinema = parametru
 ORDER BY c.cod_cinema, f.rating DESC, f.nume_film, f.an_aparitie;
```

```
TYPE record_CF IS RECORD
  (poz NUMBER,
   nume_cinema cinematograf.nume_cinema%TYPE,
   nume_film film.nume_film%TYPE,
   rating film.rating%TYPE,
   an_aparitie film.an_aparitie%TYPE);
```

```
TYPE tab_imb_CF IS TABLE OF record_CF;
t_CF tab_imb_CF := tab_imb_CF();
```

```
TYPE varr_cinemaC IS VARRAY(20) OF cinematograf.cod_cinema%TYPE;
vect_cod_cin varr_cinemaC;
```

```

TYPE varr_cinemaN IS VARRAY(20) OF cinematograf.nume_cinema%TYPE;
vect_num_cin varr_cinemaN;

TYPE tab_ind_cnt IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
t_cntFilm tab_ind_cnt;

cnt_FilmDist NUMBER;
v_prec film.rating%TYPE;
v_top NUMBER := 1;
v_cnt NUMBER := 1;

BEGIN
  SELECT c.cod_cinema, nume_cinema, COUNT(cod_film)
  BULK COLLECT INTO vect_cod_cin, vect_num_cin, t_cntFilm
  FROM cinematograf c, ruleaza r
  WHERE c.cod_cinema = r.cod_cinema(+)
  GROUP BY c.cod_cinema, nume_cinema;

  FOR i IN vect_cod_cin.FIRST..vect_cod_cin.LAST LOOP
    dbms_output.put_line('-----');
    IF t_cntFilm(i) = 0 THEN
      dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' NU a rulat
NICIUN film pana in prezent.');
      dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3 filme care
au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');
    ELSIF t_cntFilm(i) = 1 THEN
      dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' a rulat un
singur film pana in prezent.');
      dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3 filme care
au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');
    ELSIF t_cntFilm(i) = 2 THEN
      dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat doar 2
filme pana in prezent.');
      dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3 filme care
au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');
    ELSE
      v_top := 1;
      v_cnt := 1;
      SELECT COUNT(f.cod_film) INTO cnt_FilmDist
      FROM film f
      JOIN (SELECT DISTINCT cod_cinema, cod_film
            FROM ruleaza) r ON r.cod_film = f.cod_film
      JOIN (SELECT cod_cinema, nume_cinema
            FROM cinematograf) c ON r.cod_cinema = c.cod_cinema
    END IF;
  END LOOP;
END;

```

```

WHERE c.cod_cinema = vect_cod_cin(i)
ORDER BY c.cod_cinema, f.rating DESC, f.nume_film, f.an_aparitie;
IF cnt_FilmDist = 1 THEN
    dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat ' ||
t_cntFilm(i) || ' filme, dar eliminand duplicatele doar un singur film DISTINCT pana in
prezent.');
    dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3 filme
care au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');
ELSIF cnt_FilmDist = 2 THEN
    dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat ' ||
t_cntFilm(i) || ' filme, dar dar eliminand duplicatele doar 2 filme DISTINCTE pana in
prezent.');
    dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3 filme
care au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');
ELSE
    OPEN c(vect_cod_cin(i));
    t_CF.EXTEND;
    FETCH c INTO t_CF(v_cnt).nume_cinema, t_CF(v_cnt).nume_film,
t_CF(v_cnt).rating, t_CF(v_cnt).an_aparitie;
    t_CF(v_cnt).poz := v_top;
    v_prec := t_CF(v_cnt).rating;
    LOOP
        t_CF.EXTEND;
        v_cnt := v_cnt + 1;
        FETCH c INTO t_CF(v_cnt).nume_cinema, t_CF(v_cnt).nume_film,
t_CF(v_cnt).rating, t_CF(v_cnt).an_aparitie;
        EXIT WHEN c%NOTFOUND;
        IF t_CF(v_cnt).rating <> v_prec THEN
            v_top := v_top + 1;
        END IF;
        EXIT WHEN v_top = 4;
        t_CF(v_cnt).poz := v_top;
        v_prec := t_CF(v_cnt).rating;
    END LOOP;
    IF v_top <> 4 THEN
        dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat ' ||
|| t_cntFilm(i) || ' filme, iar eliminand duplicatele ' || cnt_FilmDist || ' filme DISTINCTE, dar
ratingurile nu pot determina un top 3 cele mai indragite filme.');
        dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3 filme
care au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');
    ELSE
        dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat ' ||
|| t_cntFilm(i) || ' filme, iar eliminand duplicatele ' || cnt_FilmDist || ' filme DISTINCTE.');
        dbms_output.put_line('---Top 3 cele mai bine cotate filme care au rulat in
cinematograful ' || UPPER(vect_num_cin(i)) || '---');
        v_top := 0;
    END IF;
END IF;

```

```

FOR j IN t_CF.FIRST..t_CF.LAST LOOP
    IF t_CF(j).poz <> v_top THEN
        v_top := t_CF(j).poz;
        dbms_output.new_line;
        dbms_output.put_line('-----LOCUL ' || v_top || ' -----');
        dbms_output.put_line('--Ratingul: ' || t_CF(j).rating|| ' ---');
    END IF;
    dbms_output.put_line('Filmul: ' || t_CF(j).nume_film || ', aparut in anul: ' ||
t_CF(j).an_aparitie);
    END LOOP;
    END IF;
    t_CF.DELETE;
    CLOSE c;
    END IF;
    END IF;
    dbms_output.put_line('-----');
    dbms_output.new_line();
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare');
END proc_proiect_ex6_rs;
/

```

```

CREATE OR REPLACE PROCEDURE proc_proiect_ex6_rs IS
CURSOR c(parametru cinematograf.cod_cinema%TYPE) IS
    SELECT c.nume_cinema, f.nume_film, f.rating, f.an_aparitie
    FROM film f
    JOIN (SELECT DISTINCT cod_cinema, cod_film
          FROM ruleaza) r ON r.cod_film = f.cod_film
    JOIN (SELECT cod_cinema, nume_cinema
          FROM cinematograf) c ON r.cod_cinema = c.cod_cinema
    WHERE c.cod_cinema = parametru
    ORDER BY c.cod_cinema, f.rating DESC, f.nume_film, f.an_aparitie;

TYPE record_CF IS RECORD
(pozi NUMBER,
 nume_cinema cinematograf.nume_cinema%TYPE,
 nume_film film.nume_film%TYPE,
 rating film.rating%TYPE,
 an_aparitie film.an_aparitie%TYPE);

TYPE tab_imb_CF IS TABLE OF record_CF;
t_CF tab_imb_CF := tab_imb_CF();

TYPE varr_cinemaC IS VARRAY(20) OF cinematograf.cod_cinema%TYPE;
vect_cod_cin varr_cinemaC;

```

```

TYPE varr_cinemaN IS VARRAY(20) OF cinematograf.numcine$TYPE;
vect_num_cin varr_cinemaN;

TYPE tab_ind_cnt IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
t_cntFilm tab_ind_cnt;

cnt_FilmDist NUMBER;
v_prec film.rating$TYPE;
v_top NUMBER := 1;
v_cnt NUMBER := 1;

BEGIN
  SELECT c.cod_cinema, nume_cinema, COUNT(cod_film)
  BULK COLLECT INTO vect_cod_cin, vect_num_cin, t_cntFilm
  FROM cinematograf c, ruleaza r
  WHERE c.cod_cinema = r.cod_cinema(+)
  GROUP BY c.cod_cinema, nume_cinema;

  FOR i IN vect_cod_cin.FIRST..vect_cod_cin.LAST LOOP
    dbms_output.put_line('-----');
    IF t_cntFilm(i) = 0 THEN
      dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' NU a rulat NICIUN film pana in prezent.');
      dbms_output.put_line('----NU se poate reliza un top al celor mai bine cotate 3 filme care au rulat in cinematograful ');
    ELSIF t_cntFilm(i) = 1 THEN
      dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' a rulat un singur film pana in prezent.');
      dbms_output.put_line('----NU se poate reliza un top al celor mai bine cotate 3 filme care au rulat in cinematograful ');
    ELSIF t_cntFilm(i) = 2 THEN
      dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat doar 2 filme pana in prezent.');
      dbms_output.put_line('----NU se poate reliza un top al celor mai bine cotate 3 filme care au rulat in cinematograful ');
    ELSE
      v_top := 1;
      v_cnt := 1;
      SELECT COUNT(f.cod_film) INTO cnt_FilmDist
      FROM film f
      JOIN (SELECT DISTINCT cod_cinema, cod_film
            FROM ruleaza) r ON r.cod_film = f.cod_film
      JOIN (SELECT cod_cinema, nume_cinema
            FROM cinematograf) c ON r.cod_cinema = c.cod_cinema
      WHERE c.cod_cinema = vect_cod_cin(i)
      ORDER BY c.cod_cinema, f.rating DESC, f.nume_film, f.an_aparitie;
      IF cnt_FilmDist = 1 THEN
        dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat ' || t_cntFilm(i) || ' filme, dar ');
        dbms_output.put_line('----NU se poate reliza un top al celor mai bine cotate 3 filme care au rulat in cinematograful ');
      ELSIF cnt_FilmDist = 2 THEN
        dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat ' || t_cntFilm(i) || ' filme, dar ');
        dbms_output.put_line('----NU se poate reliza un top al celor mai bine cotate 3 filme care au rulat in cinematograful ');
      ELSE
        OPEN c(vect_cod_cin(i));
        t_CF.EXTEND;
        FETCH c INTO t_CF(v_cnt).nume_cinema, t_CF(v_cnt).nume_film, t_CF(v_cnt).rating, t_CF(v_cnt).an_aparitie;
        t_CF(v_cnt).poz := v_top;
        v_prec := t_CF(v_cnt).rating;
        LOOP
          t_CF.EXTEND;
          v_cnt := v_cnt + 1;
          FETCH c INTO t_CF(v_cnt).nume_cinema, t_CF(v_cnt).nume_film, t_CF(v_cnt).rating, t_CF(v_cnt).an_aparitie;
          EXIT WHEN c%NOTFOUND;
          IF t_CF(v_cnt).rating <> v_prec THEN
            v_top := v_top + 1;
          END IF;
          EXIT WHEN v_top = 4;
          t_CF(v_cnt).poz := v_top;
          v_prec := t_CF(v_cnt).rating;
        END LOOP;
        IF v_top <> 4 THEN
          dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat ' || t_cntFilm(i) || ' filme,
          dbms_output.put_line('----NU se poate reliza un top al celor mai bine cotate 3 filme care au rulat in cinematogr
        ELSE
          dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat ' || t_cntFilm(i) || ' filme,
          dbms_output.put_line('----Top 3 cele mai bine cotate filme care au rulat in cinematograful ' || UPPER(vect_num_c
          v_top := 0;
          FOR j IN t_CF.FIRST..t_CF.LAST LOOP
            IF t_CF(j).poz <> v_top THEN
              v_top := t_CF(j).poz;
              dbms_output.new_line;
              dbms_output.put_line('-----LOCUL ' || v_top || ' -----');
            END IF;
          END LOOP;
        END IF;
      END IF;
    END IF;
  END LOOP;
END;

```

```

        dbms_output.put_line('---Ratingul: ' || t_CF(j).rating|| ' ---');
    END IF;
    dbms_output.put_line('Filmul: ' || t_CF(j).nume_film || ', aparut in anul: ' || t_CF(j).an_aparitie);
END LOOP;
END IF;
t_CF.DELETE;
CLOSE c;
END IF;
dbms_output.put_line('-----');
dbms_output.new_line();
END LOOP;
EXCEPTION
WHEN OTHERS THEN
RAISE_APPLICATION_ERROR(-20002, 'Alta eroare');
END proc_proiect_ex6_rs;
/

```

Script Output x | Task completed in 0.528 seconds
Procedure PROC_PROJECT_EX6_RS compiled

EXECUTE proc_proiect_ex6_rs;

```

Worksheet Query Builder
EXECUTE proc_proiect_ex6_rs;

```

Script Output x | Task completed in 0.058 seconds
PL/SQL procedure successfully completed.

Dbms Output | Buffer Size:20000 | P_SGBD x

In cinematograful CINEMA CITY PLAKE au rulat 7 filme, iar eliminand duplicatele 7 filme DISTINCTE.
---Top 3 cele mai bine cotate filme care au rulat in cinematograful CINEMA CITY PLAKE---

-----LOCUL 1 -----
---Ratingul: 8.5 ---
Filmul: Zootopia, aparut in anul: 2016

-----LOCUL 2 -----
---Ratingul: 7.9 ---
Filmul: Titanic, aparut in anul: 1997

-----LOCUL 3 -----
---Ratingul: 7.8 ---
Filmul: Little Women, aparut in anul: 2019
Filmul: Crimson Peak, aparut in anul: 2015

In cinematograful CINEMA CITY SIBIU au rulat 8 filme, iar eliminand duplicatele 8 filme DISTINCTE.
---Top 3 cele mai bine cotate filme care au rulat in cinematograful CINEMA CITY SIBIU---

-----LOCUL 1 -----
---Ratingul: 8.6 ---
Filmul: Sherlock Holmes, aparut in anul: 2009

-----LOCUL 2 -----
---Ratingul: 8.4 ---
Filmul: Spiderman: No Way Home, aparut in anul: 2021

-----LOCUL 3 -----
---Ratingul: 7.9 ---
Filmul: Venom, aparut in anul: 2018
Filmul: Encanto, aparut in anul: 2021

In cinematograful CINEMA CITY CLUJ au rulat 5 filme, iar eliminand duplicatele 5 filme DISTINCTE.
 ---Top 3 cele mai bine cotate filme care au rulat in cinematograful CINEMA CITY CLUJ---

-----LOCUL 1 -----
 ---Ratingul: 7.8 ---
 Filmul: Encanto, aparut in anul: 2021
 Filmul: Little Women, aparut in anul: 2019

-----LOCUL 2 -----
 ---Ratingul: 7.1 ---
 Filmul: Free Guy, aparut in anul: 2021

-----LOCUL 3 -----
 ---Ratingul: 6.5 ---
 Filmul: It chapter 2, aparut in anul: 2019
 Filmul: Yes Day, aparut in anul: 2021

In cinematograful CINEMA CITY CONSTANTA au rulat 7 filme, iar eliminand duplicatele 6 filme DISTINCTE.
 ---Top 3 cele mai bine cotate filme care au rulat in cinematograful CINEMA CITY CONSTANTA---

-----LOCUL 1 -----
 ---Ratingul: 8.1 ---
 Filmul: Divergent, aparut in anul: 2014

-----LOCUL 2 -----
 ---Ratingul: 7.9 ---
 Filmul: Venom, aparut in anul: 2018

-----LOCUL 3 -----
 ---Ratingul: 7.8 ---
 Filmul: Encanto, aparut in anul: 2021
 Filmul: Star Wars: Episode III, aparut in anul: 2005

In cinematograful CINEMA CITY BAIA MARE au rulat 6 filme, iar eliminand duplicatele 6 filme DISTINCTE.
 ---Top 3 cele mai bine cotate filme care au rulat in cinematograful CINEMA CITY BAIA MARE---

-----LOCUL 1 -----
 ---Ratingul: 7.9 ---
 Filmul: Titanic, aparut in anul: 1997

-----LOCUL 2 -----
 ---Ratingul: 7.6 ---
 Filmul: Star Wars: Episode III, aparut in anul: 2005

-----LOCUL 3 -----
 ---Ratingul: 7.4 ---
 Filmul: It, aparut in anul: 2017
 Filmul: It chapter 2, aparut in anul: 2019

In cinematograful CINEMA CITY API COTROCENI au rulat 3 filme, iar eliminand duplicatele 3 filme DISTINCTE, dar ratingurile nu pot fi determinate.
 ---NU se poate realiza un top al celor mai bine cotate 3 filme care au rulat in cinematograful CINEMA CITY API COTROCENI!---

In cinematograful CINEMA CITY IASI au rulat 6 filme, iar eliminand duplicatele 6 filme DISTINCTE.
 ---Top 3 cele mai bine cotate filme care au rulat in cinematograful CINEMA CITY IASI---

-----LOCUL 1 -----
 ---Ratingul: 8.1 ---
 Filmul: Divergent, aparut in anul: 2014

-----LOCUL 2 -----
 ---Ratingul: 7.8 ---
 Filmul: Encanto, aparut in anul: 2021

-----LOCUL 3 -----
 ---Ratingul: 7.6 ---
 Filmul: Star Wars: Episode III, aparut in anul: 2005
 Filmul: Crimson Peak, aparut in anul: 2015

In cinematograful CINEMA CITY PITESTI au rulat doar 2 filme pana in prezent.
 ---NU se poate realiza un top al celor mai bine cotate 3 filme care au rulat in cinematograful CINEMA CITY PITESTI!---

In cinematograful CINEMA CITY DROBETATSI NU a rulat NICIUN film pana in prezent.
 ---NU se poate realiza un top al celor mai bine cotate 3 filme care au rulat in cinematograful CINEMA CITY DROBETATSI!---

7. Problemă subprogram stocat independent + cursoare

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

Să se creeze o procedură care primește ca parametru un număr ce reprezintă opțiunea dorită de utilizator și afișează informații coerente, în concordanță cu alegerea introdusa de acesta. Opțiunile existente sunt 1 și 2.

I. Pentru opțiunea 1 se vor lista toate cinematografele existente în baza de date, împreună cu informații coerente despre situația de angajare din cadrul cinematografului respectiv, ceea ce înseamnă că se vor afișa joburile care au posturi ocupate, împreună cu angajații care lucrează pe jobul respectiv, în acel cinematograf sau un mesaj care să indice faptul că pentru acel cinematograf nu avem înregistrat niciun angajat.

II. Pentru opțiunea 2, se vor lista toate cinematografele existente în baza de date, precizându-se numărul angajaților care lucrează în prezent în acel cinematograf și numărul angajaților care au lucrat în trecut în acel cinematograf. De asemenea, împreună cu cinematografele se afișează informații coerente despre situația de angajare din cadrul cinematografului respectiv, ceea ce înseamnă că se vor afișa joburile care au posturi ocupate în prezent sau care au fost ocupate în trecut și angajații care lucrează pe jobul respectiv, în acel cinematograf și/sau care au lucrat în trecut pe jobul respectiv, în acel cinematograf, împreună cu un mesaj care să precizeze ce fel de angajat este (Actual / Istoric).

II. Orice alt număr reprezintă o opțiune incorectă.

--PROCEDURĂ (subprogram stocat), EXPRESIE CURSOR IMRICATĂ, CURSOR EXPLICIT, CURSOR PARAMETRIZAT, CICLU CURSOR, CICLU CURSOR CU SUBCERERI , REF CURSOR

```
CREATE OR REPLACE PROCEDURE proc_proiect_ex7_rs( optiune NUMBER ) IS
    --expresie cursor imbricata
    CURSOR c_expr
        IS SELECT nume_cinema,
               CURSOR (SELECT nume_job,
                           CURSOR (SELECT nume || ' ' || prenume
                                    FROM angajat a
                                   WHERE a.cod_cinema = c.cod_cinema AND j.cod_job = a.cod_job)
                           FROM job j)
        FROM cinematograf c;
    v_nume_cinema cinematograf.nume_cinema%TYPE;
    v_nume_job job.nume_job%TYPE;
    v_nume_ang VARCHAR2(70);
    v_nr_ang NUMBER;
    cnt NUMBER;
    TYPE refcursor IS REF CURSOR;
    v_cursor_imbr refcursor;
    v_cursor refcursor;
```

```

--cursor explicit
CURSOR c_expl IS SELECT c.cod_cinema v_cod_cinema, c.nume_cinema v_nume_cinema,
    COUNT(a.cod_angajat) v_cnt_ang, (SELECT COUNT(*)
        FROM istoric_lucru il
        WHERE c.cod_cinema = il.cod_cinema(+))v_hist_ang
    FROM cinematograf c, angajat a
    WHERE c.cod_cinema = a.cod_cinema(+)
    GROUP BY c.cod_cinema, c.nume_cinema;

--cursor parametrizat
CURSOR c_param(param_cin cinematograf.cod_cinema%TYPE, param_job
job.nume_job%TYPE) IS SELECT nume || ' ' || prenume
    FROM angajat a, job j
    WHERE cod_cinema = param_cin AND a.cod_job = j.cod_job
        AND j.nume_job = param_job;

--cursor parametrizat
CURSOR c_param_hist(param_cin cinematograf.cod_cinema%TYPE, param_job
job.nume_job%TYPE) IS SELECT nume || ' ' || prenume
    FROM angajat a, job j, istoric_lucru il
    WHERE il.cod_cinema = param_cin AND il.cod_job = j.cod_job
        AND j.nume_job = param_job AND il.cod_angajat = a.cod_angajat;

BEGIN
    IF optiune = 1 THEN
        OPEN c_expr;
        LOOP
            FETCH c_expr INTO v_nume_cinema, v_cursor_imbr;
            EXIT WHEN c_expr%NOTFOUND;
            dbms_output.put_line('-----');
            dbms_output.put_line('Cinematograful ' || UPPER(v_nume_cinema));
            dbms_output.new_line();
            cnt := 0;
            LOOP
                FETCH v_cursor_imbr INTO v_nume_job, v_cursor;
                EXIT WHEN v_cursor_imbr%NOTFOUND;

                FETCH v_cursor INTO v_nume_ang;
                IF v_cursor%ROWCOUNT > 0 THEN
                    dbms_output.put_line('-----Jobul ' || UPPER(v_nume_job) || ' -----');
                    dbms_output.put_line('  Angajatul ' || v_nume_ang);
                    LOOP
                        FETCH v_cursor INTO v_nume_ang;
                        EXIT WHEN v_cursor%NOTFOUND;
                        dbms_output.put_line('  Angajatul ' || v_nume_ang);
                END IF;
            END LOOP;
        END LOOP;
    END IF;
END;

```

```

        END LOOP;
        dbms_output.new_line();
    END IF;
    IF v_cursor%ROWCOUNT = 0 THEN
        cnt := cnt + 1;
    END IF;
END LOOP;
IF v_cursor_imbr%ROWCOUNT = cnt THEN
    dbms_output.put_line('  NICIUN ANGAJAT ');
    dbms_output.new_line();
END IF;
dbms_output.put_line('-----');
-----');

END LOOP;
CLOSE c_expr;
ELSIF optiune = 2 THEN
--ciclu cursor
FOR k IN c_expl LOOP
    dbms_output.put_line('-----');
-----');

    IF k.v_cnt_ang = 0 THEN
        IF k.v_hist_ang = 0 THEN
            dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' nu
lucreaza si nu a lucrat NICIUN angajat');
        ELSIF k.v_hist_ang = 1 THEN
            dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' nu
lucreaza in prezent NICIUN angajat, insa in trecut a mai lucrat un singur angajat');
        ELSE
            dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' nu
lucreaza in prezent NICIUN angajat, insa in trecut au mai lucrat ' || k.v_hist_ang || ' angajati');
        END IF;
    ELSE
        IF k.v_cnt_ang = 1 THEN
            IF k.v_hist_ang = 0 THEN
                dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent un singur angajat, iar in trecut nu a mai lucrat NICIUN angajat');
            ELSIF k.v_hist_ang = 1 THEN
                dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent un singur angajat, iar in trecut a mai lucrat un singur angajat');
            ELSE
                dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent un singur angajat, iar in trecut au mai lucrat ' || k.v_hist_ang || ' angajati');
            END IF;
        ELSE
            IF k.v_hist_ang = 0 THEN

```

```

        dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent ' || k.v_cnt_ang || ' angajati, iar in trecut nu a mai lucrat NICIUN angajat');

ELSIF k.v_hist_ang = 1 THEN
    dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent ' || k.v_cnt_ang || ' angajati, iar in trecut a mai lucrat un singur angajat');
ELSE
    dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent ' || k.v_cnt_ang || ' angajati, iar in trecut au mai lucrat ' || k.v_hist_ang || '
angajati');

END IF;
END IF;
--ciclu cursor cu subcereri
FOR i IN (SELECT nume_job, (SELECT COUNT(a.cod_angajat)
                            FROM angajat a
                           WHERE a.cod_job(+) = j.cod_job
                           AND a.cod_cinema = k.v_cod_cinema) nr_ang,
                           (SELECT COUNT(il.cod_angajat)
                            FROM istoric_lucru il
                           WHERE j.cod_job = il.cod_job(+)
                           AND il.cod_cinema = k.v_cod_cinema) hist_ang
                           FROM job j) LOOP
IF i.nr_ang != 0 THEN
    OPEN c_param(k.v_cod_cinema, i.nume_job);
    dbms_output.new_line();
    dbms_output.put_line('-----Jobul ' || UPPER(i.nume_job) || ' -----');
LOOP
    FETCH c_param INTO v_nume_ang;
    EXIT WHEN c_param%NOTFOUND;
    dbms_output.put_line(' (Actual) Angajatul ' || v_nume_ang);
    END LOOP;
CLOSE c_param;
IF i.hist_ang != 0 THEN
    OPEN c_param_hist(k.v_cod_cinema, i.nume_job);
    LOOP
        FETCH c_param_hist INTO v_nume_ang;
        EXIT WHEN c_param_hist%NOTFOUND;
        dbms_output.put_line(' (Istoric) Angajatul ' || v_nume_ang);
    END LOOP;
    CLOSE c_param_hist;
END IF;
ELSIF i.hist_ang != 0 THEN
    OPEN c_param_hist(k.v_cod_cinema, i.nume_job);
    dbms_output.new_line();
    dbms_output.put_line('-----Jobul ' || UPPER(i.nume_job) || ' -----');

```

```

LOOP
    FETCH c_param_hist INTO v_nume_ang;
    EXIT WHEN c_param_hist%NOTFOUND;
    dbms_output.put_line(' (istoric) Angajatul ' || v_nume_ang);
END LOOP;
CLOSE c_param_hist;
END IF;
END LOOP;
END IF;
dbms_output.put_line('-----');
dbms_output.new_line();
END LOOP;
ELSE
    dbms_output.put_line('Optiunea nu e valida');
END IF;
EXCEPTION
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002, 'Alta eroare');
END proc_proiect_ex7_rs;
/

```

```

CREATE OR REPLACE PROCEDURE proc_proiect_ex7_rs(optiune NUMBER) IS
--expresie cursor imbriicata
CURSOR c_expr
    IS SELECT nume_cinema, CURSOR (SELECT nume_job, CURSOR (SELECT nume || ' ' || prenume
        FROM angajat a
        WHERE a.cod_cinema = c.cod_cinema AND j.cod_job = a.cod_job)
        FROM job j)
    FROM cinematograf c;
v_nume_cinema cinematograf.nume_cinema%TYPE;
v_nume_job job.nume_job%TYPE;
v_nume_ang VARCHAR2(70);
v_nr_ang NUMBER;
cnt NUMBER;
TYPE refcursor IS REF CURSOR;
v_cursor_imbr refcursor;
v_cursor refcursor;

--cursor explicit
CURSOR c_expl IS SELECT c.cod_cinema v_cod_cinema, c.nume_cinema v_nume_cinema, COUNT(a.cod_angajat) v_cnt_ang,
    (SELECT COUNT(*)
     FROM istoric_lucru il
     WHERE c.cod_cinema = il.cod_cinema(+)) v_hist_ang
    FROM cinematograf c, angajat a
    WHERE c.cod_cinema = a.cod_cinema(+)
    GROUP BY c.cod_cinema, c.nume_cinema;

```

```

--cursor parametrizat
CURSOR c_param(param_cin cinematograf.cod_cinema$TYPE, param_job job.nume_job$TYPE)
IS SELECT nume || ' ' || prenume
  FROM angajat a, job j
 WHERE cod_cinema = param_cin AND a.cod_job = j.cod_job AND j.nume_job = param_job;

--cursor parametrizat
CURSOR c_param_hist(param_cin cinematograf.cod_cinema$TYPE, param_job job.nume_job$TYPE)
IS SELECT nume || ' ' || prenume
  FROM angajat a, job j, istoric_lucru il
 WHERE il.cod_cinema = param_cin AND il.cod_job = j.cod_job
       AND j.nume_job = param_job AND il.cod_angajat = a.cod_angajat;

BEGIN
  IF optiune = 1 THEN
    OPEN c_expr;
    LOOP
      FETCH c_expr INTO v_nume_cinema, v_cursor_imbr;
      EXIT WHEN c_expr%NOTFOUND;
      dbms_output.put_line('-----');
      dbms_output.put_line('Cinematograful ' || UPPER(v_nume_cinema));
      dbms_output.new_line();
      cnt := 0;
    LOOP
      FETCH v_cursor_imbr INTO v_nume_job, v_cursor;
      EXIT WHEN v_cursor_imbr%NOTFOUND;
      FETCH v_cursor INTO v_nume_ang;
      IF v_cursor%ROWCOUNT > 0 THEN
        dbms_output.put_line('-----Jobul ' || UPPER(v_nume_job) || ' -----');
        dbms_output.put_line('      Angajatul ' || v_nume_ang);
      LOOP
        FETCH v_cursor INTO v_nume_ang;
        EXIT WHEN v_cursor%NOTFOUND;
        dbms_output.put_line('      Angajatul ' || v_nume_ang);
      END LOOP;
      dbms_output.new_line();
    END IF;
    IF v_cursor%ROWCOUNT = 0 THEN
      cnt := cnt + 1;
    END IF;
  END LOOP;
  IF v_cursor_imbr%ROWCOUNT = cnt THEN
    dbms_output.put_line('      NICIUN ANGAJAT ');
    dbms_output.new_line();
  END IF;
  dbms_output.put_line('-----');
END LOOP;
CLOSE c_expr;
ELSIF optiune = 2 THEN
  --ciclu cursor
  FOR k IN c_expl LOOP
    dbms_output.put_line('-----');
    IF k.v_cnt_ang = 0 THEN
      IF k.v_hist_ang = 0 THEN
        dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' nu lucreaza si nu a lucrat NICIUN angajat');
      ELSIF k.v_hist_ang = 1 THEN
        dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' nu lucreaza in prezent NICIUN angajat, in');
      ELSE
        dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' nu lucreaza in prezent NICIUN angajat, in');
      END IF;
    ELSE
      IF k.v_cnt_ang = 1 THEN
        IF k.v_hist_ang = 0 THEN
          dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' lucreaza in prezent un singur angajat');
        ELSIF k.v_hist_ang = 1 THEN
          dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' lucreaza in prezent un singur angajat');
        ELSE
          dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' lucreaza in prezent un singur angajat');
        END IF;
      ELSE
        IF k.v_hist_ang = 0 THEN
          dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' lucreaza in prezent ' || k.v_cnt_ang);
        ELSIF k.v_hist_ang = 1 THEN
          dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' lucreaza in prezent ' || k.v_cnt_ang);
        END IF;
      END IF;
    END IF;
  END LOOP;

```

```

        ELSE
            dbms_output.put_line('In cinematograful ' || UPPER(k.v_num_cinema) || ' lucreaza in prezent ' || k.v_cnt_ang
        END IF;
    END IF;
    --ciclu cursor cu subcereri
    FOR i IN (SELECT nume_job, (SELECT COUNT(a.cod_angajat)
                                    FROM angajat a
                                   WHERE a.cod_job(+) = j.cod_job
                                     AND a.cod_cinema = k.v_cod_cinema) nr_ang,
                  (SELECT COUNT(il.cod_angajat)
                    FROM istoric_lucru il
                   WHERE j.cod_job = il.cod_job(+)
                     AND il.cod_cinema = k.v_cod_cinema) hist_ang
               FROM job j) LOOP
        IF i.nr_ang != 0 THEN
            OPEN c_param(k.v_cod_cinema, i.nume_job);
            dbms_output.new_line();
            dbms_output.put_line('-----Jobul ' || UPPER(i.nume_job) || ' -----');
            LOOP
                FETCH c_param INTO v_nume_ang;
                EXIT WHEN c_param%NOTFOUND;
                dbms_output.put_line(' (Actual) Angajatul ' || v_nume_ang);
            END LOOP;
            CLOSE c_param;
        IF i.hist_ang != 0 THEN
            OPEN c_param_hist(k.v_cod_cinema, i.nume_job);
            LOOP
                FETCH c_param_hist INTO v_nume_ang;
                EXIT WHEN c_param_hist%NOTFOUND;
                dbms_output.put_line(' (Istoric) Angajatul ' || v_nume_ang);
            END LOOP;
            CLOSE c_param_hist;
        END IF;
        ELSIF i.hist_ang != 0 THEN
            OPEN c_param_hist(k.v_cod_cinema, i.nume_job);
            dbms_output.new_line();
            dbms_output.put_line('-----Jobul ' || UPPER(i.nume_job) || ' -----');
            LOOP
                FETCH c_param_hist INTO v_nume_ang;
                EXIT WHEN c_param_hist%NOTFOUND;
                dbms_output.put_line(' (Istoric) Angajatul ' || v_nume_ang);
            END LOOP;
            CLOSE c_param_hist;
        END IF;
        END LOOP;
    END IF;
    dbms_output.put_line('-----');
    dbms_output.new_line();
END LOOP;
ELSE
    dbms_output.put_line('Optiunea nu e valida');
END IF;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare');
END proc_proiect_ex7_rs;
/

```

Script Output x | Task completed in 0.255 seconds

Procedure PROC_PROIECT_EX7_RS compiled

```

DECLARE
    optiune NUMBER := &optiune;
BEGIN
    proc_proiect_ex7_rs(optiune);
END;
/

```

Opțiunea 1

The screenshot shows the Oracle SQL Developer interface with the following details:

- Worksheet Tab:** Contains the PL/SQL code:

```
DECLARE
    optiune NUMBER := &optiune;
BEGIN
    proc_proiect_ex7_rs(optiune);
```
- Script Output Tab:** Shows the execution results:

```
new:DECLARE
    optiune NUMBER := 1;
BEGIN
    proc_proiect_ex7_rs(optiune);
END;

PL/SQL procedure successfully completed.
```
- Dbms Output Tab:** Shows the buffer size set to 20000.
- P_SGBD Tab:** Displays the output of the procedure execution, listing employees by job title and department:
 - Cinematograful CINEMA CITY PLAKE**
 - Jobul CASIER -----
Angajatul Matei Andrei
Angajatul Iana Gabriela
 - Jobul MANAGER -----
Angajatul Popescu Ion
 - Cinematograful CINEMA CITY SIBIU**
 - Jobul AGENT DE CURATENIE -----
Angajatul Georgesc Vasile
 - Jobul MANAGER -----
Angajatul Tudor Maria
 - Cinematograful CINEMA CITY CLUJ**
 - Jobul BARMAN -----
Angajatul Tamas Vasilica
 - Jobul INGINER SUNET -----
Angajatul Petre Cosmin
 - Jobul INGINER TEHNIC -----
Angajatul Dumitran George
Angajatul Luca Ioan
 - Jobul MANAGER -----
Angajatul Nils Corina
 - Jobul PLASATOR -----
Angajatul Pitulan Gabriela
 - Cinematograful CINEMA CITY CONSTANTA**
 - Jobul INGINER SUNET -----
Angajatul Stan Elena
 - Jobul MANAGER -----
Angajatul Crivat Gheorghe
 - Jobul PLASATOR -----
Angajatul Coca Emanuel

```

-----  

:cinematograful CINEMA CITY BAIA MARE  

-----  

-----Jobul INGINER SUNET -----  

Angajatul Stanciu Ecaterina  

-----  

-----Jobul MANAGER -----  

Angajatul Pitco Natanael  

-----  

-----  

:cinematograful CINEMA CITY AFI COTROCENI  

-----  

-----Jobul BARMAN -----  

Angajatul Chesaru Ana  

Angajatul Toth Raluca  

-----  

-----Jobul CASIER -----  

Angajatul Miron Luminita  

Angajatul Tanase Vlad  

-----  

-----Jobul MANAGER -----  

Angajatul Popa Maria  

-----  

Cinematograful CINEMA CITY IASI  

-----  

-----Jobul INGINER TEHNIC -----  

Angajatul Nistor Viviana  

-----  

-----Jobul MANAGER -----  

Angajatul Florea Ana  

-----  

-----  

Cinematograful CINEMA CITY PITESTI  

-----  

-----Jobul INGINER SUNET -----  

Angajatul Banu Codrin  

-----  

-----Jobul MANAGER -----  

Angajatul Miron Costin  

Angajatul Leona Antonia  

-----  

-----  

Cinematograful CINEMA CITY DROBETATSI  

-----  

NICIUN ANGAJAT

```

Opțiunea 2

```

DECLARE
    optiune NUMBER := &optiune;
BEGIN
    proc_proiect_ex7_rs(optiune);
END;

```

Script Output x | Task completed in 2.22 seconds

new:DECLARE
 optiune NUMBER := 2;
BEGIN
 proc_proiect_ex7_rs(optiune);
END;

PL/SQL procedure successfully completed.

DBMS Output
Buffer Size:20000 |

_SGBD x

In cinematograful CINEMA CITY PLAKE lucreaza in prezent 3 angajati, iar in trecut au mai lucrat 2 angajati

-----Jobul AGENT DE CURATENIE -----

(Istoric) Angajatul Georgesc Vasile

-----Jobul CASIER -----

(Actual) Angajatul Matei Andrei
(Actual) Angajatul Iana Gabriela

-----Jobul MANAGER -----

(Actual) Angajatul Popescu Ion

-----Jobul PLASATOR -----

(Istoric) Angajatul Miron Costin

In cinematograful CINEMA CITY SIBIU lucreaza in prezent 2 angajati, iar in trecut nu a mai lucrat NICIUN angajat

-----Jobul AGENT DE CURATENIE -----

(Actual) Angajatul Georgesc Vasile

-----Jobul MANAGER -----

(Actual) Angajatul Tudor Maria

In cinematograful CINEMA CITY CLUJ lucreaza in prezent 6 angajati, iar in trecut au mai lucrat 2 angajati

-----Jobul BARMAN -----

(Actual) Angajatul Tamas Vasilica
(Istoric) Angajatul Petre Cosmin

-----Jobul BARMAN CAFENA -----

(Istoric) Angajatul Toth Raluca

-----Jobul INGINER SUNET -----

(Actual) Angajatul Petre Cosmin

-----Jobul INGINER TEHNIC -----

(Actual) Angajatul Dumitran George
(Actual) Angajatul Luca Ioan

-----Jobul MANAGER -----

(Actual) Angajatul Nils Corina

-----Jobul PLASATOR -----

(Actual) Angajatul Pitulan Gabriela

In cinematograful CINEMA CITY CONSTANTA lucreaza in prezent 3 angajati, iar in trecut nu a mai lucrat NICIUN angajat

-----Jobul INGINER SUNET -----

(Actual) Angajatul Stan Elena

-----Jobul MANAGER -----

(Actual) Angajatul Crivat Gheorghe

-----Jobul PLASATOR -----

(Actual) Angajatul Coca Emanuel

In cinematograful CINEMA CITY BAIA MARE lucreaza in prezent 2 angajati, iar in trecut nu a mai lucrat NICIUN angajat

-----Jobul INGINER SUNET -----

(Actual) Angajatul Stanciu Ecaterina

-----Jobul MANAGER -----

(Actual) Angajatul Pitco Natanael

In cinematograful CINEMA CITY API COTROCENI lucreaza in prezent 5 angajati, iar in trecut a mai lucrat un singur angajat

```

-----Jobul BARMAN -----
(Actual)    Angajatul Chesaru Ana
(Actual)    Angajatul Toth Raluca

-----Jobul BARMAN CAFENEIA -----
(Istoric)   Angajatul Tamas Vasilica

-----Jobul CASIER -----
(Actual)    Angajatul Miron Luminita
(Actual)    Angajatul Tanase Vlad

-----Jobul MANAGER -----
(Actual)    Angajatul Popa Maria

-----
In cinematograful CINEMA CITY IASI lucreaza in prezent 2 angajati, iar in trecut a mai lucrat un singur angajat

-----Jobul INGINER TEHNIC -----
(Actual)    Angajatul Nistor Viviana
(Istoric)   Angajatul Dumitran George

-----Jobul MANAGER -----
(Actual)    Angajatul Florea Ana

-----
In cinematograful CINEMA CITY PITESTI lucreaza in prezent 3 angajati, iar in trecut au mai lucrat 4 angajati

-----Jobul CASIER -----
(Istoric)   Angajatul Banu Codrin

-----Jobul INGINER SUNET -----
(Actual)    Angajatul Banu Codrin
(Istoric)   Angajatul Banu Codrin

-----Jobul MANAGER -----
(Actual)    Angajatul Miron Costin
(Actual)    Angajatul Leona Antonia
(Istoric)   Angajatul Leona Antonia

-----Jobul PLASATOR -----
(Istoric)   Angajatul Pitulan Gabriela

-----
In cinematograful CINEMA CITY DROBETATI nu lucreaza si nu a lucrat NICIUN angajat

```

Opțiunea 99 => inexistentă

```

DECLARE
    optiune NUMBER := &optiune;
BEGIN
    proc_proiect_ex7_rs(optiune);
END;

```

Script Output x | Task completed in 2.842 seconds

```

new:DECLARE
    optiune NUMBER := 99;
BEGIN
    proc_proiect_ex7_rs(optiune);
END;

PL/SQL procedure successfully completed.

```

DBMS Output x | Buffer Size:20000

P_SGBD x | Optiunea nu e valida

8. Problemă funcție stocată independent + excepții + 3 tabele

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

Fiind dat genul unui film și un an să se afișeze suma prețurilor tuturor biletelor cumpărate pentru filmele care se încadrează în genul respectiv și pe baza cărora s-a intrat la difuzari ce au avut ca dată de început un an mai mare decât anul dat ca parametru.

Se vor trata excepțiile:

- I. anul dat ca parametru este negativ
- II. anul dat ca parametru este mai mare decât cel curent
- III. pentru genul dat ca parametru nu există filme introduse în baza de date
- IV. nu există nicio difuzare pentru niciun film cu genul dat ca parametru, care să aibă loc după anul dat ca parametru
- V. nu există niciun bilet cumpărat la vreo difuzare ce a avut loc după anul dat ca parametru a vreunui film ce are genul dat ca parametru

--JOIN ce include 3 tabele (FILM, DIFUZARE, BILET_CUMPARAT)

```
CREATE OR REPLACE FUNCTION funct_proiect_ex8_rs(gen IN film.gen_princ%TYPE, an IN CHAR)
  RETURN NUMBER IS
    exc_err_an_neg EXCEPTION;
    exc_err_an EXCEPTION;
    exc_err_gen EXCEPTION;
    exc_err_dif EXCEPTION;
    exc_err_bilet EXCEPTION;
    cnt NUMBER;
    pret NUMBER;
BEGIN
  IF an < 0 THEN
    RAISE exc_err_an_neg;
  END IF;
  IF an > TO_CHAR(SYSDATE, 'YYYY') THEN
    RAISE exc_err_an;
  END IF;
  SELECT COUNT(*) INTO cnt
  FROM film
  WHERE UPPER(gen_princ) = UPPER(gen);
  IF cnt = 0 THEN
    RAISE exc_err_gen;
  END IF;
  SELECT COUNT(cod_difuzare) INTO cnt
```

```

    FROM film f, difuzare d
    WHERE UPPER(gen_princ) = UPPER(gen) AND f.cod_film = d.cod_film AND
TO_CHAR(data_inc, 'YYYY') >= an;
    IF cnt = 0 THEN
        RAISE exc_err_dif;
    END IF;
--JOIN ce include 3 tabele (FILM, DIFUZARE, BILET_CUMPARAT)
    SELECT SUM(bc.pret) INTO pret
    FROM film f, difuzare d, bilet_cumparat bc
    WHERE UPPER(f.gen_princ) = UPPER(gen) AND f.cod_film = d.cod_film
        AND bc.cod_difuzare(+) = d.cod_difuzare AND TO_CHAR(d.data_inc, 'YYYY') >= an;
    IF pret is null THEN
        RAISE exc_err_bilet;
    END IF;
    RETURN pret;
EXCEPTION
    WHEN exc_err_an_neg THEN
        RAISE_APPLICATION_ERROR(-20003, 'Nu puteti introduce un an negativ!');
    WHEN exc_err_an THEN
        RAISE_APPLICATION_ERROR(-20004, 'Ati introdus un an mai mare decat anul curent,
si nu exista date despre difuzari din viitor!');
    WHEN exc_err_gen THEN
        RAISE_APPLICATION_ERROR(-20005, 'Nu exista niciun film care sa aiba genul ' ||
UPPER(gen) || '!');
    WHEN exc_err_dif THEN
        RAISE_APPLICATION_ERROR(-20006, 'Nu exista nicio difuzare pentru filme de genul '
|| UPPER(gen) || ' care sa fi avut loc dupa anul ' || an || '!');
    WHEN exc_err_bilet THEN
        RAISE_APPLICATION_ERROR(-20007, 'Nu exista niciun bilet vandut la difuzare care a
avut loc dupa anul ' || an || ' a vreunui film cu genul ' || UPPER(gen) || '!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare');
END funct_proiect_ex8_rs;
/

```

```

CREATE OR REPLACE FUNCTION funct_proiect_ex8_rs(gen IN film.gen_princ%TYPE, an IN CHAR)
  RETURN NUMBER IS
    exc_err_an_neg EXCEPTION;
    exc_err_an EXCEPTION;
    exc_err_gen EXCEPTION;
    exc_err_dif EXCEPTION;
    exc_err_bilet EXCEPTION;
    cnt NUMBER;
    pret NUMBER;
BEGIN
  IF an < 0 THEN
    RAISE exc_err_an_neg;
  END IF;
  IF an > TO_CHAR(SYSDATE, 'YYYY') THEN
    RAISE exc_err_an;
  END IF;
  SELECT COUNT(*) INTO cnt
  FROM film
  WHERE UPPER(gen_princ) = UPPER(gen);
  IF cnt = 0 THEN
    RAISE exc_err_gen;
  END IF;
  SELECT COUNT(cod_difuzare) INTO cnt
  FROM film f, difuzare d
  WHERE UPPER(gen_princ) = UPPER(gen) AND f.cod_film = d.cod_film AND TO_CHAR(data_inc, 'YYYY') >= an;
  IF cnt = 0 THEN
    RAISE exc_err_dif;
  END IF;
  SELECT SUM(bc.pret) INTO pret
  FROM film f, difuzare d, bilet_cumparat bc
  WHERE UPPER(f.gen_princ) = UPPER(gen) AND f.cod_film = d.cod_film
    AND bc.cod_difuzare(+) = d.cod_difuzare AND TO_CHAR(d.data_inc, 'YYYY') >= an;
  IF pret is null THEN
    RAISE exc_err_bilet;
  END IF;
  RETURN pret;
EXCEPTION
  WHEN exc_err_an_neg THEN
    RAISE_APPLICATION_ERROR(-20003, 'Nu puteti introduce un an negativ!');
  WHEN exc_err_an THEN
    RAISE_APPLICATION_ERROR(-20004, 'Ati introdus un an mai mare decat anul curent, si nu exista date despre difuzari din viitor!');
  WHEN exc_err_gen THEN
    RAISE_APPLICATION_ERROR(-20005, 'Nu exista niciun film care sa aiba genul ' || UPPER(gen) || '!');
  WHEN exc_err_dif THEN
    RAISE_APPLICATION_ERROR(-20006, 'Nu exista nicio difuzare pentru filme de genul ' || UPPER(gen) || ' care sa fi avut loc dupa anul curent');
  WHEN exc_err_bilet THEN
    RAISE_APPLICATION_ERROR(-20007, 'Nu exista niciun bilet vandut la difuzare care a avut loc dupa anul ' || an || ' a vreunui film');
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002, 'Alta eroare');
END funct_proiect_ex8_rs;
/

```

Script Output X | Task completed in 0.024 seconds

Function FUNCT_PROIECT_EX8_RS compiled

Model apelare funcție prin introducerea de la tastatură a parametrilor:

```

SET SERVEROUTPUT ON
ACCEPT p_gen PROMPT 'Introduceti genul filmului: '
ACCEPT p_an PROMPT 'Introduceti anul: '
DECLARE
  gen film.gen_princ%TYPE := '&p_gen';
  an NUMBER := &p_an;
  pret NUMBER;
BEGIN
  pret := funct_proiect_ex8_rs(gen, an);

```

```

    dbms_output.put_line('Suma preturilor biletelor cumpărate pentru difuzari de după anul ' || 
an || ' ale filmelor ce au genul ' || UPPER(gen) || ' este de: ' || pret);
END;
/

```

SET SERVEROUTPUT OFF

The screenshot shows the Oracle SQL Developer interface. In the top window (Worksheet), there is a PL/SQL block. In the bottom window (Script Output), the results of the execution are displayed.

```

SET SERVEROUTPUT ON
ACCEPT p_gen PROMPT 'Introduceti genul filmului: '
ACCEPT p_an PROMPT 'Introduceti anul: '
DECLARE
    gen film.gen_princ%TYPE := '&p_gen';
    an NUMBER := &p_an;
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs(gen, an);
    dbms_output.put_line('Suma preturilor biletelor cumpărate pentru difuzari de după anul ' || an || ' ale filmelor ce au genul ' || gen);
END;
/
SET SERVEROUTPUT OFF

```

Script Output:

```

old:DECLARE
gen film.gen_princ%TYPE := '&p_gen';
an NUMBER := &p_an;
pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs(gen, an);
    dbms_output.put_line('Suma preturilor biletelor cumpărate pentru difuzari de după anul ' || an || ' ale filmelor ce au genul ' || gen);
END;
Suma preturilor biletelor cumpărate pentru difuzari de după anul 2019 ale filmelor ce au genul ACTIUNE este de: 79.7

PL/SQL procedure successfully completed.

```

Evidențiere exceptii:

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Actiune', '2024');
    dbms_output.put_line('Suma preturilor biletelor vândute care să respecte criteriile impuse este de: ' || pret);
END;
/

```

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Actiune', '2024');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;

```

Script Output x | Task completed in 0.044 seconds

Error starting at line : 1,065 in command -

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Actiune', '2024');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
Error report -
ORA-20004: Ati introdus un an mai mare decat anul curent, si nu exista date despre difuzari din viitor!
ORA-06512: at "C##SGBD_STEF.FUNCT_PROIECT_EX8_RS", line 41
ORA-06512: at line 4

```

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Teatru', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
/

```

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Teatru', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;

```

Script Output x | Task completed in 0.045 seconds

Error starting at line : 1,073 in command -

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Teatru', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
Error report -
ORA-20005: Nu exista niciun film care sa aiba genul TEATRU!
ORA-06512: at "C##SGBD_STEF.FUNCT_PROIECT_EX8_RS", line 43
ORA-06512: at line 4

```

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Romance', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
/

```

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Romance', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;

```

Script Output x
Task completed in 0.045 seconds

Error starting at line : 1,081 in command -

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Romance', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;

```

Error report -

ORA-20006: Nu exista nicio difuzare pentru filme de genul ROMANCE care sa fi avut loc dupa anul 2019!
ORA-06512: at "C##SGBD_STEF.FUNCT_PROIECT_EX8_RS", line 45
ORA-06512: at line 4

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Romance', 'scara');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
/

```

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Romance', 'scara');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;

```

Script Output x
Task completed in 0.041 seconds

Error starting at line : 1,089 in command -

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Romance', 'scara');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;

```

Error report -

ORA-20002: Alta eroare
ORA-06512: at "C##SGBD_STEF.FUNCT_PROIECT_EX8_RS", line 49
ORA-06512: at line 4

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Comedie', '-2022');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
/

```

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Comedie', '-2022');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
/

```

Script Output x | Task completed in 0.044 seconds

Error starting at line : 1,097 in command -

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Comedie', '-2022');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
Error report -
ORA-20003: Nu puteti introduce un an negativ!
ORA-06512: at "C##SGBD_STEF.FUNCT_PROIECT_EX8_RS", line 39
ORA-06512: at line 4

```

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Animatie', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
/

```

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Animatie', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
/

```

Script Output x | Task completed in 0.034 seconds

Error starting at line : 1,105 in command -

```

DECLARE
    pret NUMBER;
BEGIN
    pret := funct_proiect_ex8_rs('Animatie', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
Error report -
ORA-20007: Nu exista niciun bilet vandut la difuzare care a avut loc dupa anul 2019 a vreunui film cu genul ANIMATIE!
ORA-06512: at "C##SGBD_STEF.FUNCT_PROIECT_EX8_RS", line 47
ORA-06512: at line 4

```

9. Problemă procedură stocată independent + excepții + 5 tabele

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

Să se implementeze o procedură care primește ca parametrii un an și un nume de actor. Vrem să căutăm date despre filmele care au rulat pe o perioadă mai lungă de 4 luni într-un cinematograf deschis luni sau sămbătă, ținând cont ca anul rularii respective să fie mai mare decât anul dat ca parametru, și de asemenea, acel film să aibă înregistrată o difuzare într-o sală de tipul 'vip' sau care conține în numele tipului subșirul 'multi'.

Să se modifice structura tabelei FILM astfel încât să se rețină într-o singură coloană date despre numărul de premii strâns de toți actorii din baza de date care au participat la distribuția filmului respectiv, numărul acestor actori, numărul de premii strâns de toți regizorii din baza de date care au participat la distribuția filmului respectiv și numărul acestor regizori. În această nouă coloană se vor număra doar premiile actorilor și numărul actorilor care au anul de debut diferit de anul de debut preluat de la actorul cu numele sau prenumele dat ca parametru în procedură.

În cadrul procedurii descrise mai sus, vrem să afișăm detalii despre cinematograf, ziua deschiderii acestuia, detalii despre sălile în care au avut loc difuzările filmului curent, numele filmului curent și ratingul acestuia. Folosind coloana adițională din tabela FILM vom calcula o medie dintre numărul de premii câștigate de persoanele care fac parte din distribuție și numărul acestor persoane, aceasta fiind media de premiere a filmului.

Se vor trata cazurile (excepțiile) în care:

- I. anul dat ca parametru este negativ
- II. anul dat ca parametru este mai mare decât cel curent
- III. pentru anul dat ca parametru nu există cinematografe care să se fi deschis luni sau sămbătă
- IV. nu există niciun actor cu numele dat ca parametru
- V. există mai mulți actori cu numele dat ca parametru

De asemenea, se vor afișa corespunzător mesaje pentru cazurile în care:

- I. media de premiere a filmului este 0 și
 - a. avem date despre mai mulți membrii care au participat în distribuție
 - b. avem date despre un singur membru care a participat în distribuție
 - c. nu avem date despre niciun membru care a participat în distribuție
- II. media de premiere a filmului este diferita de 0 și
 - a. avem date despre mai mulți membrii care au participat în distribuție
 - b. avem date despre un singur membru care a participat în distribuție
 - c. nu avem date despre niciun membru care a participat în distribuție

--comandă SQL ce include 5 tabele, prin intermediul subcererilor (ACTOR, JOACA, REGIZOR, REGIZEAZA, FILM)

--comandă SQL ce include 5 tabele, prin intermediul JOIN-ului (FILM, RULEAZA, DIFUZARE, CINEMATOGRAF, SALA), la care se adaugă prin intermediul subcererilor încă o dată tabela FILM și coloana adițională din tabela FILM preluată ca TABLE

```

CREATE OR REPLACE TYPE obj IS OBJECT
  (nr_premii NUMBER,
   part NUMBER);
/
CREATE OR REPLACE TYPE tab_imbr IS TABLE OF obj;
/
ALTER TABLE film
ADD (date_membrii tab_imbr) NESTED TABLE date_membrii STORE AS
tab_date_membrii;

```

The screenshot shows the Oracle SQL Developer interface. In the top pane, there is a 'Worksheet' tab with the DDL code. Below it is a 'Script Output' tab showing the results of the execution:

```

Type OBJ compiled
Type TAB_IMBR compiled
Table FILM altered.

```

CREATE OR REPLACE PROCEDURE proc_proiect_ex9_rs(an CHAR, numeA VARCHAR2)
IS

```

exc_an_neg EXCEPTION;
exc_an EXCEPTION;
exc_an_NDF EXCEPTION;
cnt NUMBER;
an_deb actor.an_debut%TYPE;
BEGIN
  IF an < 0 THEN
    RAISE exc_an_neg;
  END IF;
  IF an > TO_CHAR(SYSDATE, 'YYYY') THEN
    RAISE exc_an;
  END IF;

  SELECT COUNT(*) INTO cnt
  FROM cinematograf
  WHERE to_char(data_desch, 'YYYY') >= an AND (UPPER(to_char(data_desch, 'DAY'))
LIKE '%MONDAY%' OR UPPER(to_char(data_desch, 'DAY')) LIKE '%SATURDAY%');
  IF cnt = 0 THEN
    RAISE exc_an_NDF;
  END IF;

```

```

SELECT an_debut INTO an_deb
FROM actor
WHERE (LOWER(prenume) LIKE LOWER(numeA) OR LOWER(nume) LIKE
LOWER(numeA));
--comandă SQL ce include 5 tabele, prin intermediul subcererilor (ACTOR, JOACA, REGIZOR,
REGIZEAZA, FILM)
FOR i IN (SELECT f.cod_film cod, NVL((SELECT SUM(nr_premii)
FROM actor a, joaca j
WHERE a.id_actor = j.id_actor AND j.cod_film = f.cod_film
AND a.an_debut <> an_deb), 0) premii_act,
NVL((SELECT COUNT(a.id_actor)
FROM actor a, joaca j
WHERE a.id_actor = j.id_actor AND j.cod_film = f.cod_film
AND a.an_debut <> an_deb), 0) nr_act,
NVL((SELECT SUM(nr_premii)
FROM regizor r, regizeaza ra
WHERE r.id_regizor = ra.id_regizor AND ra.cod_film = f.cod_film), 0) premii_reg,
NVL((SELECT COUNT(nr_premii)
FROM regizor r, regizeaza ra
WHERE r.id_regizor = ra.id_regizor AND ra.cod_film = f.cod_film), 0) nr_reg
FROM film f) LOOP
UPDATE film
SET date_membrii = tab_imbr( obj(i.premii_act, i.nr_act), obj(i.premii_reg, i.nr_reg))
WHERE cod_film = i.cod;
END LOOP;
--comandă SQL ce include 5 tabele, prin intermediul JOIN-ului (FILM, RULEAZA, DIFUZARE,
CINEMATOGRAF, SALA), la care se adaugă prin intermediul subcererilor încă o dată tabela FILM
și coloana adițională din tabela FILM preluată ca TABLE
FOR i IN (SELECT c.nume_cinema numeC, to_char(data_desch, 'DAY') zi, f.nume_film
numeF, d.nr_sala salaD,
s.tip salaT, f.rating ratingF,
(SELECT SUM(b.part)
FROM film a, TABLE (a.date_membrui) b
WHERE a.cod_film = f.cod_film) nr_part,
(SELECT SUM(b.nr_premii)
FROM film a, TABLE (a.date_membrui) b
WHERE a.cod_film = f.cod_film) nr_premii,
(SELECT nvl(SUM(b.nr_premii)/ NULLIF(SUM(b.part), 0), 0)
FROM film a, TABLE (a.date_membrui) b
WHERE a.cod_film = f.cod_film) medie_premii
FROM film f, ruleaza r, difuzare d, cinematograf c, sala s
WHERE c.cod_cinema = r.cod_cinema AND r.cod_film = f.cod_film
AND (UPPER(to_char(data_desch, 'DAY')) LIKE '%MONDAY%' OR
UPPER(to_char(data_desch, 'DAY')) LIKE '%SATURDAY%')

```

```

        AND to_char(data_desch, 'YYYY') >= an
        AND f.cod_film = d.cod_film
        AND d.nr_sala = s.nr_sala AND s.cod_cinema = c.cod_cinema
        AND (LOWER(s.tip) LIKE '%multi%' OR LOWER(s.tip) LIKE 'vip')
        AND months_between(r.data_final, r.data_inceput) > 4
    ORDER BY 7 DESC, 6 DESC, 5 DESC, 4 DESC) LOOP
IF UPPER(i.zi) LIKE '%SATURDAY%' THEN
    i.zi := 'SAMBATA';
ELSE
    i.zi := 'LUNI';
END IF;
dbms_output.new_line();
IF i.medie_premii = 0 THEN
    IF i.nr_part = 1 THEN
        dbms_output.put_line('[Cinematograf: ' || UPPER(i.numeC) || '] [Zi deschidere: ' || i.zi
        || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
        UPPER(i.numeF)|| '] [Rating: ' || i.ratingF || ']');
        dbms_output.put_line('Despre film se stie ca a avut ca a avut un singur membru in
productie, iar acesta nu a castigat NICIUN premiu');
    ELSIF i.nr_part > 0 THEN
        dbms_output.put_line('[Cinematograf: ' || UPPER(i.numeC) || '] [Zi deschidere: ' || i.zi
        || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
        UPPER(i.numeF)|| '] [Rating: ' || i.ratingF || ']');
        dbms_output.put_line('Despre film se stie ca a avut ' || i.nr_part || ' membrii
in productie, iar acestia nu a castigat NICIUN premiu');
    ELSE
        dbms_output.put_line('[Cinematograf: ' || UPPER(i.numeC) || '] [Zi deschidere: ' || i.zi
        || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
        UPPER(i.numeF)|| '] [Rating: ' || i.ratingF || ']');
        dbms_output.put_line('Pentru acest film nu s-au putut colectiona date despre actori
sau regizori');
    END IF;
ELSE
    IF i.nr_premii = 1 THEN
        IF i.nr_part = 1 THEN
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numeC) || '] [Zi deschidere: ' || i.zi
            || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
            UPPER(i.numeF)|| '] [Rating: ' || i.ratingF || ']');
            dbms_output.put_line('Despre film se stie ca a avut ca a avut un singur membru in
productie, iar acesta a castigat in total un singur premiu');
            dbms_output.put_line('----- Media de premiere a filmului -----> ' ||
            i.medie_premii);
        ELSE
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numeC) || '] [Zi deschidere: ' || i.zi
            || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
            UPPER(i.numeF)|| '] [Rating: ' || i.ratingF || ']');
        END IF;
    END IF;
END IF;

```

```

        dbms_output.put_line('Despre film se stie ca a avut ca a avut ' || i.nr_part
                            || ' membrii in productie, iar acestia au castigat in total un singur premiu');
        dbms_output.put_line('----- Media de premiere a filmului -----> ' || i.medie_premii);
    END IF;
    ELSE
        IF i.nr_part = 1 THEN
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numeC) || '] [Zi deschidere: ' || i.zi || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' || UPPER(i.numeF)|| '] [Rating: ' || i.ratingF || ']');
            dbms_output.put_line('Despre film se stie ca a avut ca a avut un singur membru in
productie, iar acesta a castigat in total ' || i.nr_premii || ' premii');
            dbms_output.put_line('----- Media de premiere a filmului -----> ' || i.medie_premii);
        ELSE
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numeC) || '] [Zi deschidere: ' || i.zi || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' || UPPER(i.numeF)|| '] [Rating: ' || i.ratingF || ']');
            dbms_output.put_line('Despre film se stie ca a avut ca a avut ' || i.nr_part || '
membrii in productie, iar acestia au castigat in total ' || i.nr_premii || ' premii');
            dbms_output.put_line('----- Media de premiere a filmului -----> ' || i.medie_premii);
        END IF;
    END IF;
    END IF;
END LOOP;

EXCEPTION
    WHEN exc_an_neg THEN
        RAISE_APPLICATION_ERROR(-20003, 'Nu puteti introduce un an negativ!');
    WHEN exc_an THEN
        RAISE_APPLICATION_ERROR(-20004, 'Ati introdus un an mai mare decat anul curent,
si nu exista date despre cinematografe deschise in viitor!');
    WHEN exc_an_NDF THEN
        RAISE_APPLICATION_ERROR(-20005, 'Nu exista niciun cinematograf deschis luni sau
sambata, dupa anul ' || an || '!');
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20006, 'Nu exista niciun actor cu numele sau
prenumele ' || UPPER(numeA) || '!');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20007, 'Exista mai multi actori cu numele sau
prenumele ' || UPPER(numeA) || '!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20008, 'Alta eroare!');
END proc_proiect_ex9_rs;
/

```

```

CREATE OR REPLACE PROCEDURE proc_proiect_ex9_rs(an CHAR, numeA VARCHAR2) IS
    exc_an_neg EXCEPTION;
    exc_an EXCEPTION;
    exc_an_NDF EXCEPTION;
    cnt NUMBER;
    an_debut actor.an_debut%TYPE;
BEGIN
    IF an < 0 THEN
        RAISE exc_an_neg;
    END IF;
    IF an > TO_CHAR(SYSDATE, 'YYYY') THEN
        RAISE exc_an;
    END IF;

    SELECT COUNT(*) INTO cnt
    FROM cinematograf
    WHERE to_char(data_desch, 'YYYY') >= an AND (UPPER(to_char(data_desch, 'DAY')) LIKE '%MONDAY%' OR UPPER(to_char(data_desch, 'DAY')) LIKE '%SATURDAY%');

    IF cnt = 0 THEN
        RAISE exc_an_NDF;
    END IF;

    SELECT an_debut INTO an_deb
    FROM actor
    WHERE (LOWER(prenume) LIKE LOWER(numeA) OR LOWER(nume) LIKE LOWER(numeA));
    --comanda SQL ce include 5 table, prin intermediul subcererilor (ACTOR, JOACA, REGIZOR, REGIZEAZA, FILM)
    FOR i IN (SELECT f.cod_film cod, NVL((SELECT SUM(nr_premii)
                                                FROM actor a, joaca j
                                                WHERE a.id_actor = j.id_actor AND j.cod_film = f.cod_film
                                                AND a.an_debut <> an_deb), 0) premii_act,
                  NVL((SELECT COUNT(a.id_actor)
                                                FROM actor a, joaca j
                                                WHERE a.id_actor = j.id_actor AND j.cod_film = f.cod_film
                                                AND a.an_debut <> an_deb), 0) nr_act,
                  NVL((SELECT SUM(nr_premii)
                                                FROM regizor r, regizeaza ra
                                                WHERE r.id_regizor = ra.id_regizor AND ra.cod_film = f.cod_film), 0) premii_reg,
                  NVL((SELECT COUNT(nr_premii)
                                                FROM regizor r, regizeaza ra
                                                WHERE r.id_regizor = ra.id_regizor AND ra.cod_film = f.cod_film), 0) nr_reg
              FROM film f) LOOP
        UPDATE film
        SET date_membrii = tab_imbr( obj(i.premii_act, i.nr_act), obj(i.premii_reg, i.nr_reg))
        WHERE cod_film = i.cod;
    END LOOP;

    --comanda SQL ce include 5 table, prin intermediul JOIN-ului (FILM, RULEAZA, DIFUZARE, CINEMATOGRAP, SALA), la care se adauga prin intermediul subcererilor inca o data tabela FILM si coloana aditionala din tabela film preluata ca TABLE
    FOR i IN (SELECT c.nume_cinema numeC, to_char(data_desch, 'DAY') zi, f.nume_film numeF, d.nr_sala salaD,
                  s.tip salaT, f.rating ratingF,
                  (SELECT SUM(b.part)
                   FROM film a, TABLE (a.date_membrii) b
                   WHERE a.cod_film = f.cod_film) nr_part,
                  (SELECT SUM(b.nr_premii)
                   FROM film a, TABLE (a.date_membrii) b
                   WHERE a.cod_film = f.cod_film) nr_premii,
                  (SELECT nvl(SUM(b.nr_premii)/ NULLIF(SUM(b.part), 0), 0)
                   FROM film a, TABLE (a.date_membrii) b
                   WHERE a.cod_film = f.cod_film) medie_premii
              FROM film f, ruleaza r, difuzare d, cinematograf c, sala s
              WHERE c.cod_cinema = r.cod_cinema AND r.cod_film = f.cod_film
                AND (UPPER(to_char(data_desch, 'DAY')) LIKE '%MONDAY%' OR UPPER(to_char(data_desch, 'DAY')) LIKE '%SATURDAY%')
                AND to_char(data_desch, 'YYYY') >= an
                AND f.cod_film = d.cod_film
                AND d.nr_sala = s.nr_sala AND s.cod_cinema = c.cod_cinema
                AND (LOWER(s.tip) LIKE '%multi%' OR LOWER(s.tip) LIKE 'vip')
                AND months_between(r.data_final, r.data_inceput) > 4
              ORDER BY 7 DESC, 6 DESC, 5 DESC, 4 DESC) LOOP
        IF UPPER(i.zi) LIKE '%SATURDAY%' THEN
            i.zi := 'SAMBATA';
        END IF;
    END LOOP;

```

```

    ELSE
        i.zi := 'LUNDI';
    END IF;
    dbms_output.new_line();
    IF i.medie_premii = 0 THEN
        IF i.nr_part = 1 THEN
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' || i.zi
                || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD
                || '] [Film: ' || UPPER(i.numef)|| '] [Rating: ' || i.ratingF || ']');
            dbms_output.put_line('Despre film se stie ca a avut ca a avut un singur membru in productie, iar acesta nu a castiga');
        ELSIF i.nr_part > 0 THEN
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' || i.zi
                || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD
                || '] [Film: ' || UPPER(i.numef)|| '] [Rating: ' || i.ratingF || ']');
            dbms_output.put_line('Despre film se stie ca a avut ca a avut ' || i.nr_part || ' membrii in productie, iar acestia nu au putut colectiona date despre actori sau regizori');
        ELSE
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' || i.zi
                || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD
                || '] [Film: ' || UPPER(i.numef)|| '] [Rating: ' || i.ratingF || ']');
            dbms_output.put_line('Pentru acest film nu s-au putut colectiona date despre actori sau regizori');
        END IF;
    ELSE
        IF i.nr_premii = 1 THEN
            IF i.nr_part = 1 THEN
                dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' || i.zi
                    || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD
                    || '] [Film: ' || UPPER(i.numef)|| '] [Rating: ' || i.ratingF || ']');
                dbms_output.put_line('Despre film se stie ca a avut ca a avut un singur membru in productie, iar acesta a castigat');
                dbms_output.put_line('----- Media de premiere a filmului -----> ' || i.medie_premii);
            ELSE
                dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' || i.zi
                    || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD
                    || '] [Film: ' || UPPER(i.numef)|| '] [Rating: ' || i.ratingF || ']');
                dbms_output.put_line('Despre film se stie ca a avut ca a avut ' || i.nr_part
                    || ' membrii in productie, iar acestia au castigat in total un singur premiu');
                dbms_output.put_line('----- Media de premiere a filmului -----> ' || i.medie_premii);
            END IF;
        ELSE
            IF i.nr_part = 1 THEN
                dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' || i.zi
                    || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD
                    || '] [Film: ' || UPPER(i.numef)|| '] [Rating: ' || i.ratingF || ']');
                dbms_output.put_line('Despre film se stie ca a avut ca a avut un singur membru in productie, iar acesta a castigat');
                dbms_output.put_line('----- Media de premiere a filmului -----> ' || i.medie_premii);
            ELSE
                dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' || i.zi
                    || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD
                    || '] [Film: ' || UPPER(i.numef)|| '] [Rating: ' || i.ratingF || ']');
                dbms_output.put_line('Despre film se stie ca a avut ca a avut ' || i.nr_part || ' membrii in productie, iar acesti au castigat');
                dbms_output.put_line('----- Media de premiere a filmului -----> ' || i.medie_premii);
            END IF;
        END IF;
    END LOOP;
EXCEPTION
    WHEN exc_an_neg THEN
        RAISE_APPLICATION_ERROR(-20003, 'Nu puteti introduce un an negativ!');
    WHEN exc_an THEN
        RAISE_APPLICATION_ERROR(-20004, 'Ati introdus un an mai mare decat anul curent, si nu exista date despre cinematografe deschise');
    WHEN exc_an_NDF THEN
        RAISE_APPLICATION_ERROR(-20005, 'Nu exista niciun cinematograf deschis luni sau sambata, dupa anul ' || an || '!');
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20006, 'Nu exista niciun actor cu numele sau prenumele ' || UPPER(numea) || '!');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20007, 'Există mai multe actori cu numele sau prenumele ' || UPPER(numea) || '!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20008, 'Alta eroare!');
END proc_proiect_ex9_rs;
/

```

Script Output x | Query Result x

 Task completed in 0.282 seconds
 Procedure PROC_PROJECT_EX9_RS compiled

EXECUTE proc_project_ex9_rs('2000', 'Ryan');

```

EXECUTE proc_project_ex9_rs('2000', 'Ryan');

PL/SQL procedure successfully completed.

Dbsns Output
P_SGBD x

[Cinematograf: CINEMA CITY API COTROCENI] [Zi deschidere: SAMBATA] [Tip sala difuzare: VIP] [Numar sala difuzare: 28] [Film: THE BATMAN]
Despre film se stie ca a avut ca a avut 2 membrii in productie, iar acestia au castigat in total 31 premii
----- Media de premiere a filmului -----> 15.5

[Cinematograf: CINEMA CITY API COTROCENI] [Zi deschidere: SAMBATA] [Tip sala difuzare: VIP] [Numar sala difuzare: 31] [Film: DEATH ON THE N
Despre film se stie ca a avut ca a avut 2 membrii in productie, iar acestia au castigat in total 25 premii
----- Media de premiere a filmului -----> 12.5

[Cinematograf: CINEMA CITY API COTROCENI] [Zi deschidere: SAMBATA] [Tip sala difuzare: VIP] [Numar sala difuzare: 28] [Film: RED NOTICE] [R
Despre film se stie ca a avut ca a avut un singur membru in productie, iar acesta a castigat in total 3 premii
----- Media de premiere a filmului -----> 3

[Cinematograf: CINEMA CITY PITESTI] [Zi deschidere: LUNI] [Tip sala difuzare: Multiplex] [Numar sala difuzare: 37] [Film: DOCTOR STRANGE] []
Pentru acest film nu s-au putut colectiona date despre actori sau regizori

[Cinematograf: CINEMA CITY PLAKE] [Zi deschidere: LUNI] [Tip sala difuzare: Multiplex] [Numar sala difuzare: 4] [Film: CRIMSON PEAK] [Rating: 8.5]
Pentru acest film nu s-au putut colectiona date despre actori sau regizori

[Cinematograf: CINEMA CITY SIBIU] [Zi deschidere: SAMBATA] [Tip sala difuzare: Multiplex] [Numar sala difuzare: 10] [Film: BLACK WIDOW] [Rating: 7.5]
Pentru acest film nu s-au putut colectiona date despre actori sau regizori

[Cinematograf: CINEMA CITY SIBIU] [Zi deschidere: SAMBATA] [Tip sala difuzare: Multiplex] [Numar sala difuzare: 10] [Film: YES DAY] [Rating: 7.5]
Pentru acest film nu s-au putut colectiona date despre actori sau regizori

```

Tabela FILM după executarea procedurii:

| COD_FILM | NUME_FILM | GEN_PRINC | DURATA | RATING | AN_APARITIE | DATE_MEMBRII |
|----------|----------------------------|-----------------|--------|--------|-------------|---|
| 1 | 10 Uncharted | Actiune | 116 | 6.4 | 2022 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 2 | 110 Spiderman: No Way Home | Actiune | 148 | 8.4 | 2021 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 3 | 210 Free Guy | Actiune | 115 | 7.1 | 2021 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 4 | 310 Little Women | Romance | 135 | 7.8 | 2019 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 5 | 410 It | Horror | 135 | 7.4 | 2017 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 6 | 510 Encanto | Animatie | 109 | 7.8 | 2021 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 7 | 610 Star Wars: Episode III | Science Fiction | 140 | 7.6 | 2005 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 8 | 710 Black Widow | Actiune | 134 | 6.4 | 2021 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 9 | 810 Titanic | Drama | 194 | 7.9 | 1997 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 10 | 910 The Batman | Crima | 176 | 8 | 2022 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 11 | 1010 Red Notice | Comedie | 118 | 6.3 | 2021 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 12 | 1110 Venom | Actiune | 112 | 7.9 | 2018 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 13 | 1210 Crimson Peak | Horror | 119 | 7 | 2015 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |
| 14 | 1310 Yes Day | Familie | 146 | 5.7 | 2021 | C##SGBD_STEF.TAB_IMBR([C##SGBD_STEF.OBJ], |

Evidențiere exceptii:

EXECUTE proc_proiect_ex9_rs('-10', 'Zendaya');

The screenshot shows the Oracle SQL Developer interface. In the top bar, there is a button labeled "EXECUTE proc_proiect_ex9_rs(' -10 ', 'Zendaya ')". Below the toolbar, the "Script Output" tab is selected, showing the command and its execution details. The output window displays the following error message:

```
Error starting at line : 1,316 in command -
BEGIN proc_proiect_ex9_rs(' -10 ', 'Zendaya '); END;
Error report -
ORA-20003: Nu puteti introduce un an negativ!
ORA-06512: at "C##SGBD_STEF.PROC_PROIECT_EX9_RS", line 128
ORA-06512: at line 1
```

EXECUTE proc_proiect_ex9_rs('2033', 'Zendaya');

The screenshot shows the Oracle SQL Developer interface. In the top bar, there is a button labeled "EXECUTE proc_proiect_ex9_rs('2033', 'Zendaya ')". Below the toolbar, the "Script Output" tab is selected, showing the command and its execution details. The output window displays the following error message:

```
Error starting at line : 1,319 in command -
BEGIN proc_proiect_ex9_rs('2033', 'Zendaya '); END;
Error report -
ORA-20004: Ati introdus un an mai mare decat anul curent, si nu exista date despre cinematografe deschise in viitor!
ORA-06512: at "C##SGBD_STEF.PROC_PROIECT_EX9_RS", line 130
ORA-06512: at line 1
```

EXECUTE proc_proiect_ex9_rs('2000', 'Gigi');

The screenshot shows the Oracle SQL Developer interface. In the top bar, there is a button labeled "EXECUTE proc_proiect_ex9_rs('2000', 'Gigi ')". Below the toolbar, the "Script Output" tab is selected, showing the command and its execution details. The output window displays the following error message:

```
Error starting at line : 1,322 in command -
BEGIN proc_proiect_ex9_rs('2000', 'Gigi '); END;
Error report -
ORA-20006: Nu exista niciun actor cu numele sau prenumele GIGI!
ORA-06512: at "C##SGBD_STEF.PROC_PROIECT_EX9_RS", line 134
ORA-06512: at line 1
```

Exceptia NO_DATA_FOUND

EXECUTE proc_proiect_ex9_rs('2000', 'Robert');

The screenshot shows the Oracle SQL Developer interface. In the top bar, there is a button labeled "EXECUTE proc_proiect_ex9_rs('2000', 'Robert ')". Below the toolbar, the "Script Output" tab is selected, showing the command and its execution details. The output window displays the following error message:

```
Error starting at line : 1,325 in command -
BEGIN proc_proiect_ex9_rs('2000', 'Robert '); END;
Error report -
ORA-20007: Există mai mulți actori cu numele sau prenumele ROBERT!
ORA-06512: at "C##SGBD_STEF.PROC_PROIECT_EX9_RS", line 136
ORA-06512: at line 1
```

Exceptia TOO_MANY_ROWS

```
EXECUTE proc_proiect_ex9_rs('scara', 'Ryan');
```

The screenshot shows the Oracle SQL Developer interface. In the top bar, there is a tab labeled "EXECUTE proc_proiect_ex9_rs('scara', 'Ryan');". Below the tabs, there are two panes: "Script Output" and "Query Result". The "Script Output" pane contains the following text:

```
Error starting at line : 1,328 in command -
BEGIN proc_proiect_ex9_rs('scara', 'Ryan'); END;
Error report -
ORA-20008: Alta eroare!
ORA-06512: at "C##SGBD_STEF.PROC_PROIECT_EX9_RS", line 138
ORA-06512: at line 1
```

The "Query Result" pane is empty.

Revenirea la structura inițială a tebelei FILM

The screenshot shows the Oracle SQL Developer interface. In the top bar, there is a tab labeled "ALTER TABLE film". Below the tabs, there are two panes: "Script Output" and "Query Result". The "Script Output" pane contains the following text:

```
ALTER TABLE film
DROP COLUMN date_membrii;

DROP TYPE tab_imbr;
DROP TYPE obj;
```

The "Query Result" pane contains the following text:

```
Table FILM altered.

Type TAB_IMBR dropped.

Type OBJ dropped.
```

The "Script Output" tab is highlighted in blue, indicating the current tab.

10. Trigger LMD la nivel de comandă

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

Vrem să definim un declanșator care să verifice anumite proprietăți înainte de orice tip de inserare în tabela ACTOR. Trebuie să verificăm dacă toți actorii care se află momentan stocați în baza de date sunt înregistrați ca jucând în cel puțin un film, dacă datele despre anul de debut ale fiecarui actor din baza de date sunt corecte, în concordanță cu filmele în care acesta a jucat (anul de debut să nu fie mai târziu decât cel de apariție al filmului în care joacă) și ca datele despre numărul de premii obținute de fiecare actor să fie coerente în raport cu anul de debut (50% din numărul de premii să nu fie mai mare decât (anul actual - anul de debut) * 100)

```
CREATE OR REPLACE TRIGGER trig_proiect_ex10_rs
    BEFORE INSERT ON actor
DECLARE
    cnt NUMBER;
BEGIN
    FOR i IN (SELECT COUNT(j.id_actor) cnt
               FROM actor a, joaca j
              WHERE a.id_actor = j.id_actor(+)
            GROUP BY a.id_actor) LOOP
        IF i.cnt = 0 THEN
            RAISE_APPLICATION_ERROR(-20002, 'Nu puteti introduce un nou actor pana cand
nu aveti date despre cel putin o aparitie intr-un film pentru fiecare actor din baza de date!');
        END IF;
    END LOOP;

    SELECT COUNT(a.id_actor) INTO cnt
    FROM actor a, joaca j, film f
   WHERE a.id_actor = j.id_actor AND j.cod_film = f.cod_film AND an_debut > an_aparitie;
    IF cnt <> 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nu puteti introduce un nou actor, deoarece in
baza de date exista informatii eronate cu privire la anul de debut al unui actor sau filmele in
care acesta a jucat!');
    END IF;

    SELECT COUNT(id_actor) INTO cnt
    FROM actor
   WHERE (to_char(SYSDATE, 'YYYY') - an_debut) * 100 < nr_premii * 0.5 ;
    IF cnt <> 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nu puteti introduce un nou actor, deoarece in
baza de date exista informatii eronate cu privire la numarul de premii sau anul de debut al unui
actor!');
    END IF;
```

```

END trig_proiect_ex10_rs;
/

```

```

CREATE OR REPLACE TRIGGER trig_proiect_ex10_rs
    BEFORE INSERT ON actor
DECLARE
    cnt NUMBER;
BEGIN
    FOR i IN (SELECT COUNT(j.id_actor) cnt
              FROM actor a, joaca j
              WHERE a.id_actor = j.id_actor(+)
              GROUP BY a.id_actor) LOOP
        IF i.cnt = 0 THEN
            RAISE_APPLICATION_ERROR(-20002, 'Nu puteti introduce un nou actor pana cand nu aveti date despre cel putin o aparitie in film');
        END IF;
    END LOOP;

    SELECT COUNT(a.id_actor) INTO cnt
    FROM actor a, joaca j, film f
    WHERE a.id_actor = j.id_actor AND j.cod_film = f.cod_film
          AND an_debut > an_aparitie;
    IF cnt <> 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nu puteti introduce un nou actor, deoarece in baza de date exista informatii eronate cu privire la anul debutului');
    END IF;

    SELECT COUNT(id_actor) INTO cnt
    FROM actor
    WHERE (to_char(SYSDATE, 'YYYY') - an_debut) * 100 < nr_premii * 0.5 ;
    IF cnt <> 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nu puteti introduce un nou actor, deoarece in baza de date exista informatii eronate cu privire la varsta');
    END IF;
END;
/

```

Trigger TRIG_PROIECT_EX10_RS compiled

Declansare trigger:

```

INSERT INTO actor
VALUES(1, 'ana', 'lugojana', 'RO', 220, 2022);

```

```

INSERT INTO actor
VALUES(2, 'ana', 'blandiana', 'RO', 2, 2017);

```

```

INSERT INTO actor
VALUES(1, 'ana', 'lugojana', 'RO', 220, 2022);

INSERT INTO actor
VALUES(2, 'ana', 'blandiana', 'RO', 2, 2017);

```

1 row inserted.

Error starting at line : 1,388 in command -

```

INSERT INTO actor
VALUES(2, 'ana', 'blandiana', 'RO', 2, 2017)
Error report -
ORA-20002: Nu puteti introduce un nou actor pana cand nu aveti date despre cel putin o aparitie intr-un film pentru fiecare actor din baza de date
ORA-06512: at "C##SGBD_STEF.TRIG_PROJECT_EX10_RS", line 9
ORA-06512: at "C##SGBD_STEF.TRIG_PROJECT_EX10_RS", line 9
ORA-04088: error during execution of trigger 'C##SGBD_STEF.TRIG_PROJECT_EX10_RS'

```

Trigger declanșat deoarece am introdus un nou actor, fără să avem informații despre participarea acestuia în cadrul vreunui film.

INSERT INTO joaca

```
VALUES(1, 910, 1, 'principal');
```

INSERT INTO actor

```
VALUES(2, 'ana', 'blandiana', 'RO', 2, 2017);
```

```

INSERT INTO joaca
VALUES(1, 910, 1, 'principal');

INSERT INTO actor
VALUES(2, 'ana', 'blandiana', 'RO', 2, 2017);

```

1 row inserted.

Error starting at line : 1,400 in command -

```

INSERT INTO actor
VALUES(2, 'ana', 'blandiana', 'RO', 2, 2017)
Error report -
ORA-20002: Nu puteti introduce un nou actor, deoarece in baza de date exista informatii eronate cu privire la numarul de premii sau anul
ORA-06512: at "C##SGBD_STEF.TRIG_PROJECT_EX10_RS", line 25
ORA-04088: error during execution of trigger 'C##SGBD_STEF.TRIG_PROJECT_EX10_RS'

```

Trigger declanșat deoarece deși am introdus date despre participarea noului actor în cadrul unui film, acest nou actor are un procent mult prea mare de premii față de anul de debut.

ROLLBACK;

INSERT INTO actor

```
VALUES(1, 'ana', 'lugojana', 'RO', 2, 2020);
```

```
INSERT INTO joaca
VALUES(1, 1810, 1, 'principal');
```

```
INSERT INTO actor
VALUES(2, 'ana', 'blandiana', 'RO', 2, 2017);
```

The screenshot shows a SQL developer interface with a query editor and a script output window.

In the query editor, the following code is present:

```
ROLLBACK;
INSERT INTO actor
VALUES(1, 'ana', 'lugojana', 'RO', 2, 2020);
INSERT INTO joaca
VALUES(1, 1810, 1, 'principal');
INSERT INTO actor
VALUES(2, 'ana', 'blandiana', 'RO', 2, 2017);
```

The script output window shows the results of the rollback command:

```
Script Output x | Query Result x
ROLLBACK complete.

1 row inserted.

1 row inserted.

Error starting at line : 1,416 in command -
INSERT INTO actor
VALUES(2, 'ana', 'blandiana', 'RO', 2, 2017)
Error report -
ORA-20002: Nu puteti introduce un nou actor, deoarece in baza de date exista informatii eronate cu privire la anul de debut al unui actor
ORA-06512: at "C##SGBD_STEF.TRIG_PROIECT_EX10_RS", line 18
ORA-04088: error during execution of trigger 'C##SGBD_STEF.TRIG_PROIECT_EX10_RS'
```

Rollback efectuat pentru a elimina actorul cu date eronate.

Trigger declanșat deoarece deși am introdus un actor cu date corecte și i-am atribuit acestuia cel puțin un film în care să fi jucat, când încercăm să inserăm un nou actor observăm că pentru actorul precedent dat de debut și cea în care a apărut filmul în care a jucat nu corespund.

Dezactivare trigger:

The screenshot shows a SQL developer interface with a query editor and a script output window.

In the query editor, the following code is present:

```
DROP TRIGGER trig_proiect_ex10_rs;
```

The script output window shows the result of the trigger drop:

```
Script Output x | Query Result x
Trigger TRIG_PROIECT_EX10_RS dropped.
```

11. Trigger LMD la nivel de linie

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

Vrem să definim un declanșator care să verifice anumite proprietăți la nivel de linie înainte de o inserare, ștergere sau updateare a unui câmp ce aparține coloanei “cod_cinema”, “cod_job” sau “id_superior” în tabela ANGAJAT.

Înainte de a insera o nouă persoană în tabela ANGAJAT în cazul în care dorim să punem noua persoană pe postul de manager trebuie să ne asigurăm că nu mai există un alt manager în acel cinematograf, de asemenea, trebuie să avem grija ca id-ul superiorului să se regasească în același cinematograf ca cel ales pentru noul angajat și salariul atribuit să se încadreze în intervalul determinat de job.

Înainte de a șterge o valoare în tabela ANGAJAT trebuie să updatam coloana “id_superior” pentru subalternii săi deoarece este cheia externă care referențiază o valoare din tabela ANGAJAT.

Înainte de a update una din liniile tablei ANGAJAT, pentru una din coloanele precizate, trebuie să verificăm dacă în cinematograful în care dorim să îl mutăm, sau în vechiul cinematograf (deinde de coloanele asupra cărora facem update), în cazul în care pe acel angajat dorim să îl punem pe postul de manager sau a posedat postul de manager (deinde de coloanele asupra cărora facem update) să nu fie mai mulți manageri, de asemenea, trebuie să avem grija ca id-ul superiorului să se regăsească în același cinematograf ca cel în care se află sau se va afla angajatul actualizat (deinde de coloanele asupra cărora facem update).

De asemenea, după update sau delete vrem să modificam datele din tabela ISTORIC_LUCRU și coloana “id_superior” pentru subalternii angajatului curent, în funcție de cazurile determinate de coloanele asupra cărora facem update. În plus, pentru update trebuie modificate (dacă este cazul) data de angajare și salariul, iar această modificare se produce prin intermediul trigger-ului declarat.

```
CREATE OR REPLACE TRIGGER trig_before_proiect_ex11_rs
  BEFORE INSERT OR DELETE OR UPDATE OF cod_cinema, cod_job, id_superior ON
  angajat
    FOR EACH ROW
  DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
    cnt NUMBER;
    sal_min NUMBER;
    sal_max NUMBER;
  BEGIN
    IF INSERTING THEN
      SELECT COUNT(*) INTO cnt
      FROM angajat
      WHERE cod_cinema = :NEW.cod_cinema
        AND cod_job LIKE :NEW.cod_job;
      IF :NEW.cod_job LIKE ('%MAN%') THEN
        IF cnt + 1 > 1 THEN
          RAISE_APPLICATION_ERROR(-20002, 'Exista deja un manager in cinematograful
  dorit!');
        END IF;
      END IF;
    END IF;
```

END IF;

IF :NEW.id_superior IS NOT null THEN

 SELECT COUNT(*) INTO cnt

 FROM angajat

 WHERE cod_cinema = :NEW.cod_cinema AND cod_angajat = :NEW.id_superior;

 IF cnt = 0 THEN

 RAISE_APPLICATION_ERROR(-20002, 'Id-ul superiorului este gresit, deoarece nu
 există un angajat cu acest cod în cinematograful dorit!');

 END IF;

END IF;

SELECT COUNT(*) INTO cnt

FROM job

WHERE cod_job LIKE :NEW.cod_job;

IF cnt <> 0 THEN

 SELECT salariu_min, salariu_max INTO sal_min, sal_max

 FROM job

 WHERE cod_job LIKE :NEW.cod_job;

 IF sal_min > :NEW.salariu THEN

 RAISE_APPLICATION_ERROR(-20002, 'Salariul este mai mic decât cel minim
 pentru jobul ales!');

 ELSIF sal_max < :NEW.salariu THEN

 RAISE_APPLICATION_ERROR(-20002, 'Salariul este mai mare decât cel maxim
 pentru jobul ales!');

 END IF;

END IF;

ELSIF DELETING THEN

 UPDATE angajat

 SET id_superior = :OLD.id_superior

 WHERE id_superior = :OLD.cod_angajat;

ELSIF UPDATING('cod_cinema') AND UPDATING('cod_job') THEN

 IF UPDATING('id_superior') THEN

 IF :NEW.id_superior IS NOT null THEN

 SELECT COUNT(*) INTO cnt

 FROM angajat

 WHERE cod_cinema = :NEW.cod_cinema AND cod_angajat = :NEW.id_superior;

 IF cnt = 0 THEN

 RAISE_APPLICATION_ERROR(-20002, 'Id-ul superiorului este gresit, deoarece
 nu există un angajat cu acest cod în cinematograful dorit!');

 END IF;

 END IF;

 END IF;

```

SELECT COUNT(*) INTO cnt
FROM angajat
WHERE cod_job = :NEW.cod_job AND cod_cinema = :NEW.cod_cinema;
IF :NEW.cod_job = 'MAN' THEN
    IF cnt + 1 > 1 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Exista deja un manager in cinematograful
dorit!');
    END IF;
END IF;

ELSIF UPDATING('cod_cinema') THEN
    IF UPDATING('id_superior') THEN
        IF :NEW.id_superior IS NOT null THEN
            SELECT COUNT(*) INTO cnt
            FROM angajat
            WHERE cod_cinema = :NEW.cod_cinema AND cod_angajat = :NEW.id_superior;
            IF cnt = 0 THEN
                RAISE_APPLICATION_ERROR(-20002, 'Id-ul superiorului este gresit, deoarece
nu exista un angajat cu acest cod in cinematograful dorit!');
            END IF;
        END IF;
    END IF;
END IF;

SELECT COUNT(*) INTO cnt
FROM angajat
WHERE cod_job = :OLD.cod_job AND cod_cinema = :NEW.cod_cinema;
IF :OLD.cod_job = 'MAN' THEN
    IF cnt + 1 > 1 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Exista deja un manager in cinematograful
dorit!');
    END IF;
END IF;

ELSIF UPDATING('cod_job') THEN
    IF UPDATING('id_superior') THEN
        IF :NEW.id_superior IS NOT null THEN
            SELECT COUNT(*) INTO cnt
            FROM angajat
            WHERE cod_cinema = :OLD.cod_cinema AND cod_angajat = :NEW.id_superior;
            IF cnt = 0 THEN
                RAISE_APPLICATION_ERROR(-20002, 'Id-ul superiorului este gresit, deoarece
nu exista un angajat cu acest cod in cinematograful dorit!');
            END IF;
        END IF;
    END IF;
END IF;

```

```

SELECT COUNT(*) INTO cnt
FROM angajat
WHERE cod_job = :NEW.cod_job AND cod_cinema = :OLD.cod_cinema;
IF :OLD.cod_job = 'MAN' THEN
    IF cnt + 1 > 1 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Exista deja un manager in cinematograful
current!');
    END IF;
END IF;
ELSIF UPDATING('id_superior') THEN
    IF :NEW.id_superior IS NOT null THEN
        SELECT COUNT(*) INTO cnt
        FROM angajat
        WHERE cod_cinema = :OLD.cod_cinema AND cod_angajat = :NEW.id_superior;
        IF cnt = 0 THEN
            RAISE_APPLICATION_ERROR(-20002, 'Id-ul superiorului este gresit, deoarece nu
exista un angajat cu acest cod in cinematograful dorit!');
        END IF;
    END IF;
END IF;
COMMIT;
END trig_before_proiect_ex11_rs;
/

```

The screenshot shows the Oracle SQL Developer interface with the 'Query Builder' tab selected. The code area displays the trigger definition:

```

CREATE OR REPLACE TRIGGER trig_before_proiect_ex11_rs
    BEFORE INSERT OR DELETE OR UPDATE OF cod_cinema, cod_job, id_superior ON angajat
    FOR EACH ROW
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
    cnt NUMBER;
    sal_min NUMBER;
    sal_max NUMBER;
BEGIN
    IF INSERTING THEN
        SELECT COUNT(*) INTO cnt
        FROM angajat
        WHERE cod_cinema = :NEW.cod_cinema
            AND cod_job LIKE :NEW.cod_job;
        IF :NEW.cod_job LIKE ('%MAN%') THEN
            IF cnt + 1 > 1 THEN
                RAISE_APPLICATION_ERROR(-20002, 'Exista deja un manager in cinematograful dorit!');
            END IF;
        END IF;

        IF :NEW.id_superior IS NOT null THEN
            SELECT COUNT(*) INTO cnt
            FROM angajat
            WHERE cod_cinema = :NEW.cod_cinema AND cod_angajat = :NEW.id_superior;
            IF cnt = 0 THEN

```

```

        RAISE_APPLICATION_ERROR(-20002, 'Id-ul superiorului este gresit, deoarece nu exista un angajat cu acest cod in cinema');
    END IF;
END IF;

SELECT COUNT(*) INTO cnt
FROM job
WHERE cod_job LIKE :NEW.cod_job;
IF cnt <> 0 THEN
    SELECT salariu_min, salariu_max INTO sal_min, sal_max
    FROM job
    WHERE cod_job LIKE :NEW.cod_job;
    IF sal_min > :NEW.salariu THEN
        RAISE_APPLICATION_ERROR(-20002, 'Salariul este mai mic decat cel minim pentru jobul ales!');
    ELSIF sal_max < :NEW.salariu THEN
        RAISE_APPLICATION_ERROR(-20002, 'Salariul este mai mare decat cel maxim pentru jobul ales!');
    END IF;
END IF;
ELSIF DELETING THEN
    UPDATE angajat
    SET id_superior = :OLD.id_superior
    WHERE id_superior = :OLD.cod_angajat;
ELSIF UPDATING('cod_cinema') AND UPDATING('cod_job') THEN
    IF UPDATING('id_superior') THEN
        IF :NEW.id_superior IS NOT null THEN
            SELECT COUNT(*) INTO cnt
            FROM angajat
            WHERE cod_cinema = :NEW.cod_cinema AND cod_angajat = :NEW.id_superior;
            IF cnt = 0 THEN
                RAISE_APPLICATION_ERROR(-20002, 'Id-ul superiorului este gresit, deoarece nu exista un angajat cu acest cod in cinema');
            END IF;
        END IF;
        SELECT COUNT(*) INTO cnt
        FROM angajat
        WHERE cod_job = :NEW.cod_job AND cod_cinema = :NEW.cod_cinema;
        IF :NEW.cod_job = 'MAN' THEN
            IF cnt + 1 > 1 THEN
                RAISE_APPLICATION_ERROR(-20002, 'Exista deja un manager in cinematograful dorit!');
            END IF;
        END IF;
        ELSIF UPDATING('cod_cinema') THEN
            IF UPDATING('id_superior') THEN
                IF :NEW.id_superior IS NOT null THEN
                    SELECT COUNT(*) INTO cnt
                    FROM angajat
                    WHERE cod_cinema = :NEW.cod_cinema AND cod_angajat = :NEW.id_superior;
                    IF cnt = 0 THEN
                        RAISE_APPLICATION_ERROR(-20002, 'Id-ul superiorului este gresit, deoarece nu exista un angajat cu acest cod in cinema');
                    END IF;
                END IF;
            END IF;
            SELECT COUNT(*) INTO cnt
            FROM angajat
            WHERE cod_job = :OLD.cod_job AND cod_cinema = :NEW.cod_cinema;
            IF :OLD.cod_job = 'MAN' THEN
                IF cnt + 1 > 1 THEN
                    RAISE_APPLICATION_ERROR(-20002, 'Exista deja un manager in cinematograful dorit!');
                END IF;
            END IF;
            ELSIF UPDATING('cod_job') THEN
                IF UPDATING('id_superior') THEN
                    IF :NEW.id_superior IS NOT null THEN
                        SELECT COUNT(*) INTO cnt
                        FROM angajat
                        WHERE cod_cinema = :OLD.cod_cinema AND cod_angajat = :NEW.id_superior;
                        IF cnt = 0 THEN
                            RAISE_APPLICATION_ERROR(-20002, 'Id-ul superiorului este gresit, deoarece nu exista un angajat cu acest cod in cinema');
                        END IF;
                    END IF;
                END IF;
            END IF;
            SELECT COUNT(*) INTO cnt
            FROM angajat

```

```

    WHERE cod_job = :NEW.cod_job AND cod_cinema = :OLD.cod_cinema;
    IF :OLD.cod_job = 'MAN' THEN
        IF cnt + 1 > 1 THEN
            RAISE_APPLICATION_ERROR(-20002, 'Exista deja un manager in cinematograful curent!');
        END IF;
    END IF;
ELSIF UPDATING('id_superior') THEN
    IF :NEW.id_superior IS NOT null THEN
        SELECT COUNT(*) INTO cnt
        FROM angajat
        WHERE cod_cinema = :OLD.cod_cinema AND cod_angajat = :NEW.id_superior;
        IF cnt = 0 THEN
            RAISE_APPLICATION_ERROR(-20002, 'Id-ul superiorului este gresit, deoarece nu exista un angajat cu acest cod in cinema');
        END IF;
    END IF;
    END IF;
    COMMIT;
END trig_before_proiect_ex11_rs;
/

```

Script Output | Query Result | Task completed in 0.604 seconds
Trigger TRIG_BEFORE_PROJECT_EX11_RS compiled

CREATE OR REPLACE TRIGGER trig_after_proiect_ex11_rs
FOR DELETE OR UPDATE OF cod_cinema, cod_job, id_superior ON angajat
COMPOUND TRIGGER

TYPE rec_upd IS RECORD
 (id_ang angajat.cod_angajat%TYPE,
 id_sup angajat.id_superior%TYPE,
 v_cin angajat.cod_cinema%TYPE,
 n_cin angajat.cod_cinema%TYPE,
 v_job angajat.cod_job%TYPE,
 n_job angajat.cod_job%TYPE,
 d_ang angajat.data_ang%TYPE,
 v_sal angajat.salariu%TYPE,
 flag NUMBER);

TYPE tab_idx_upd IS TABLE OF rec_upd INDEX BY PLS_INTEGER;
 t_upd tab_idx_upd;

TYPE tab_idx_del IS TABLE OF angajat.cod_angajat%TYPE INDEX BY PLS_INTEGER;
 t_del tab_idx_del;
 poz NUMBER;
 sal_max NUMBER;
 sal_min NUMBER;

AFTER EACH ROW IS

BEGIN

IF DELETING THEN

poz := t_del.COUNT + 1;
 t_del(poz) := :OLD.cod_angajat;

ELSIF UPDATING('cod_cinema') AND UPDATING('cod_job') THEN
 poz := t_upd.COUNT + 1;
 t_upd(poz).id_ang := :OLD.cod_angajat;
 t_upd(poz).id_sup := :OLD.id_superior;

```

t_upd(poz).v_cin := :OLD.cod_cinema;
t_upd(poz).d_ang := :OLD.data_ang;
t_upd(poz).n_cin := :NEW.cod_cinema;
t_upd(poz).v_job := :OLD.cod_job;
t_upd(poz).n_job := :NEW.cod_job;
t_upd(poz).flag := 0;
t_upd(poz).v_sal := :OLD.salariu;
IF UPDATING('id_superior') THEN
    t_upd(poz).flag := 1;
END IF;
ELSIF UPDATING('cod_cinema')THEN
    poz := t_upd.COUNT + 1;
    t_upd(poz).id_ang := :OLD.cod_angajat;
    t_upd(poz).id_sup := :OLD.id_superior;
    t_upd(poz).v_cin := :OLD.cod_cinema;
    t_upd(poz).d_ang := :OLD.data_ang;
    t_upd(poz).n_cin := :NEW.cod_cinema;
    t_upd(poz).v_job := :OLD.cod_job;
    t_upd(poz).v_sal := :OLD.salariu;
    t_upd(poz).flag := 0;
    t_upd(poz).n_job := :OLD.cod_job;
    IF UPDATING('id_superior') THEN
        t_upd(poz).flag := 1;
    END IF;
ELSIF UPDATING('cod_job') THEN
    poz := t_upd.COUNT + 1;
    t_upd(poz).id_ang := :OLD.cod_angajat;
    t_upd(poz).id_sup := :OLD.id_superior;
    t_upd(poz).v_cin := :OLD.cod_cinema;
    t_upd(poz).d_ang := :OLD.data_ang;
    t_upd(poz).n_cin := :OLD.cod_cinema;
    t_upd(poz).v_job := :OLD.cod_job;
    t_upd(poz).v_sal := :OLD.salariu;
    t_upd(poz).n_job := :NEW.cod_job;
    t_upd(poz).flag := 1;
END IF;
END AFTER EACH ROW;

```

```

AFTER STATEMENT IS
BEGIN
    FOR i IN 1..t_del.COUNT LOOP
        DELETE FROM istoric_lucru
        WHERE cod_angajat = t_del(i);

```

```

END LOOP;
FOR i IN 1..t_upd.COUNT LOOP
    IF t_upd(i).v_cin <> t_upd(i).n_cin THEN
        UPDATE angajat
        SET id_superior = t_upd(i).id_sup
        WHERE id_superior = t_upd(i).id_ang;
        IF t_upd(i).flag = 0 THEN
            UPDATE angajat
            SET id_superior = null
            WHERE cod_angajat = t_upd(i).id_ang;
        END IF;
    END IF;

    INSERT INTO istoric_lucru
    VALUES(t_upd(i).id_ang, t_upd(i).d_ang, SYSDATE, t_upd(i).v_cin, t_upd(i).v_job);

    UPDATE angajat
    SET data_ang = SYSDATE
    WHERE cod_angajat = t_upd(i).id_ang;

    IF t_upd(i).v_job NOT LIKE t_upd(i).n_job THEN
        SELECT salariu_max, salariu_min INTO sal_max, sal_min
        FROM job
        WHERE cod_job LIKE t_upd(i).n_job;

        IF sal_max < t_upd(i).v_sal THEN
            UPDATE angajat
            SET salariu = sal_max
            WHERE cod_angajat = t_upd(i).id_ang;
        ELSIF sal_min > t_upd(i).v_sal THEN
            UPDATE angajat
            SET salariu = sal_min
            WHERE cod_angajat = t_upd(i).id_ang;
        END IF;
    END IF;
END LOOP;
END AFTER STATEMENT;
END trig_after_proiect_ex11_rs;
/

```

```

CREATE OR REPLACE TRIGGER trig_after_proiect_excl_rs
  FOR DELETE OR UPDATE OF cod_cinema, cod_job, id_superior ON angajat
  COMPOUND TRIGGER
    TYPE rec_upd IS RECORD
      (id_ang angajat.cod_angajat$TYPE,
       id_sup angajat.id_superior$TYPE,
       v_cin angajat.cod_cinema$TYPE,
       n_cin angajat.cod_cinema$TYPE,
       v_job angajat.cod_job$TYPE,
       n_job angajat.cod_job$TYPE,
       d_ang angajat.data_ang$TYPE,
       v_sal angajat.salariu$TYPE,
       flag NUMBER);
    TYPE tab_idx_upd IS TABLE OF rec_upd INDEX BY PLS_INTEGER;
    t_upd tab_idx_upd;

    TYPE tab_idx_del IS TABLE OF angajat.cod_angajat$TYPE INDEX BY PLS_INTEGER;
    t_del tab_idx_del;
    poz NUMBER;
    sal_max NUMBER;
    sal_min NUMBER;
  AFTER EACH ROW IS
  BEGIN
    IF DELETING THEN
      poz := t_del.COUNT + 1;
      t_del(poz) := :OLD.cod_angajat;
    ELSIF UPDATING('cod_cinema') AND UPDATING('cod_job') THEN
      poz := t_upd.COUNT + 1;
      t_upd(poz).id_ang := :OLD.cod_angajat;
      t_upd(poz).id_sup := :OLD.id_superior;
      t_upd(poz).v_cin := :OLD.cod_cinema;
      t_upd(poz).d_ang := :OLD.data_ang;
      t_upd(poz).n_cin := :NEW.cod_cinema;
      t_upd(poz).v_job := :OLD.cod_job;
      t_upd(poz).n_job := :NEW.cod_job;
      t_upd(poz).flag := 0;
      t_upd(poz).v_sal := :OLD.salariu;
      IF UPDATING('id_superior') THEN
        t_upd(poz).flag := 1;
      END IF;
    ELSIF UPDATING('cod_cinema') THEN
      poz := t_upd.COUNT + 1;
      t_upd(poz).id_ang := :OLD.cod_angajat;
      t_upd(poz).id_sup := :OLD.id_superior;
      t_upd(poz).v_cin := :OLD.cod_cinema;
      t_upd(poz).d_ang := :OLD.data_ang;
      t_upd(poz).n_cin := :NEW.cod_cinema;
      t_upd(poz).v_job := :OLD.cod_job;
      t_upd(poz).v_sal := :OLD.salariu;
      t_upd(poz).flag := 0;
      t_upd(poz).n_job := :OLD.cod_job;
      IF UPDATING('id_superior') THEN
        t_upd(poz).flag := 1;
      END IF;
    ELSIF UPDATING('cod_job') THEN
      poz := t_upd.COUNT + 1;
      t_upd(poz).id_ang := :OLD.cod_angajat;
      t_upd(poz).id_sup := :OLD.id_superior;
      t_upd(poz).v_cin := :OLD.cod_cinema;
      t_upd(poz).d_ang := :OLD.data_ang;
      t_upd(poz).n_cin := :OLD.cod_cinema;
      t_upd(poz).v_job := :OLD.cod_job;
      t_upd(poz).v_sal := :OLD.salariu;
      t_upd(poz).n_job := :NEW.cod_job;
      t_upd(poz).flag := 1;
    END IF;
  END AFTER EACH ROW;

  AFTER STATEMENT IS
  BEGIN
    FOR i IN 1..t_del.COUNT LOOP
      DELETE FROM istoric_lucru
      WHERE cod_angajat = t_del(i);
    END LOOP;
    FOR i IN 1..t_upd.COUNT LOOP

```

```

        IF t_upd(i).v_cin <> t_upd(i).n_cin THEN
            UPDATE angajat
            SET id_superior = t_upd(i).id_sup
            WHERE id_superior = t_upd(i).id_ang;
            IF t_upd(i).flag = 0 THEN
                UPDATE angajat
                SET id_superior = null
                WHERE cod_angajat = t_upd(i).id_ang;
            END IF;
        END IF;

        INSERT INTO istoric_lucru
        VALUES(t_upd(i).id_ang, t_upd(i).d_ang, SYSDATE, t_upd(i).v_cin, t_upd(i).v_job);

        UPDATE angajat
        SET data_ang = SYSDATE
        WHERE cod_angajat = t_upd(i).id_ang;

        IF t_upd(i).v_job NOT LIKE t_upd(i).n_job THEN
            SELECT salariu_max, salariu_min INTO sal_max, sal_min
            FROM job
            WHERE cod_job LIKE t_upd(i).n_job;

            IF sal_max < t_upd(i).v_sal THEN
                UPDATE angajat
                SET salariu = sal_max
                WHERE cod_angajat = t_upd(i).id_ang;
            ELSIF sal_min > t_upd(i).v_sal THEN
                UPDATE angajat
                SET salariu = sal_min
                WHERE cod_angajat = t_upd(i).id_ang;
            END IF;
        END IF;
    END LOOP;
END AFTER STATEMENT;
END trig_after_proiect_ex11_rs;
/

```

Trigger TRIG_AFTER_PROJECT_EX11_RS compiled

Declansare triggeri:

INSERT INTO angajat

VALUES (1, 'ana', 'blandiana', 'ana.bland@gmail', null, sysdate, 59827, 8, null, 120, 'MAN');

```

        INSERT INTO angajat
        VALUES (1, 'ana', 'blandiana', 'ana.bland@gmail', null, sysdate, 59827, 8, null, 120, 'MAN');

Error starting at line : 1,683 in command -
INSERT INTO angajat
VALUES (1, 'ana', 'blandiana', 'ana.bland@gmail', null, sysdate, 59827, 8, null, 120, 'MAN')
Error report -
ORA-20002: Există deja un manager în cinematograful dorit!
ORA-06512: at "C##SGBD_STEF.TRIG_BEFORE_PROJECT_EX11_RS", line 14
ORA-04088: error during execution of trigger 'C##SGBD_STEF.TRIG_BEFORE_PROJECT_EX11_RS'

```

Trigger BEFORE declansat de faptul că am încercat să introduc un nou manager într-un cinematograf care are deja manager asignat.

INSERT INTO angajat

```
VALUES (1, 'ana', 'blandiana', 'ana.bland@gmail', null, sysdate, 59827, 8, null, 120, 'AG_C');
```

The screenshot shows the Oracle SQL Developer interface. In the script editor, there is an attempt to insert a new row into the 'angajat' table:

```
INSERT INTO angajat
VALUES (1, 'ana', 'blandiana', 'ana.bland@gmail', null, sysdate, 59827, 8, null, 120, 'AG_C');
```

Below the editor, the 'Script Output' tab is active, displaying the error message:

```
Error starting at line : 1,690 in command -
INSERT INTO angajat
VALUES (1, 'ana', 'blandiana', 'ana.bland@gmail', null, sysdate, 59827, 8, null, 120, 'AG_C')
Error report -
ORA-20002: Salariul este mai mare decat cel maxim pentru jobul ales!
ORA-06512: at "C##SGBD_STEF.TRIG_BEFORE_PROIECT_EX11_RS", line 37
ORA-04088: error during execution of trigger 'C##SGBD_STEF.TRIG_BEFORE_PROIECT_EX11_RS'
```

Trigger BEFORE declanșat de faptul că am încercat să introduc un nou angajat cu un job, atribuindu-i un salariu mai mare decât cel maxim definit.

UPDATE angajat

```
SET cod_cinema = 140
```

```
WHERE cod_angajat = 1000;
```

The screenshot shows the Oracle SQL Developer interface. In the script editor, there is an attempt to update the 'cod_cinema' column for the employee with ID 1000:

```
UPDATE angajat
SET cod_cinema = 140
WHERE cod_angajat = 1000;
```

Below the editor, the 'Script Output' tab is active, displaying the error message:

```
Error starting at line : 1,697 in command -
UPDATE angajat
SET cod_cinema = 140
WHERE cod_angajat = 1000
Error report -
ORA-20002: Exista deja un manager in cinematograful doriti!
ORA-06512: at "C##SGBD_STEF.TRIG_BEFORE_PROIECT_EX11_RS", line 81
ORA-04088: error during execution of trigger 'C##SGBD_STEF.TRIG_BEFORE_PROIECT_EX11_RS'
```

Trigger BEFORE declanșat de faptul că am încercat să fac UPDATE unui angajat cu un job-ul de manager, mutându-l întrul cinematograf care are deja manager.

UPDATE angajat

```
SET cod_job = 'BAR'
```

```
WHERE cod_angajat = 1000;
```

The screenshot shows the Oracle SQL Developer interface. In the script editor, there is an attempt to update the 'cod_job' column for the employee with ID 1000:

```
UPDATE angajat
SET cod_job = 'BAR'
WHERE cod_angajat = 1000;
```

Below the editor, the 'Script Output' tab is active, displaying the confirmation message:

```
1 row updated.
```

Trigger AFTER declanșat de faptul că am făcut UPDATE unui angajat modificându-i job-ul.

Trigger-ul AFTER îmi va insera vechile valori ale angajatului modificat în tabela ISTORIC_LUCRU.

```
SELECT * FROM istoric_lucru;
```

Script Output | Query Result | SQL | All Rows Fetched: 11 in 0.024 seconds

| COD_ANGAJAT | DATA_START | DATA_DEMISIE | COD_CINEMA | COD_JOB |
|-------------|----------------|--------------|------------|---------|
| 1 | 1000 03-JUN-18 | 11-JAN-23 | 120 | MAN |
| 2 | 1034 03-MAY-18 | 05-MAY-19 | 120 | AG_C |
| 3 | 1102 17-JUL-19 | 18-NOV-19 | 120 | PLAS |
| ... | ... | ... | ... | ... |

**DELETE FROM angajat
WHERE cod_angajat = 1000;**

```
DELETE FROM angajat
WHERE cod_angajat = 1000;
```

Script Output | Query Result | SQL | Task completed in 0.063 seconds

1 row deleted.

Trigger AFTER declanșat de faptul că am un angajat din baza de date.

Triger-ul AFTER îmi va șterge din tabela ISTORIC_JOB toate informațiile despre vechile anajări ale persoanei șterse, fără a efectua modificări în tabela ANGAJAT, deoarece proprietățile vechi corespund cu verificările efectuate.

```
SELECT * FROM istoric_lucru;
```

Script Output | Query Result | SQL | All Rows Fetched: 10 in 0.002 seconds

| COD_ANGAJAT | DATA_START | DATA_DEMISIE | COD_CINEMA | COD_JOB |
|-------------|----------------|--------------|------------|---------|
| 1 | 1034 03-MAY-18 | 05-MAY-19 | 120 | AG_C |
| 2 | 1102 17-JUL-19 | 18-NOV-19 | 120 | PLAS |
| 3 | 1221 19-NOV-18 | 19-AUG-20 | 260 | MAN |
| 4 | 1255 07-NOV-20 | 20-JUN-21 | 160 | BAR |
| 5 | 1119 07-MAY-14 | 07-JUN-15 | 260 | ING_S |
| 6 | 1119 08-JUN-15 | 18-NOV-19 | 260 | CAS |
| 7 | 1289 20-MAR-19 | 13-JUN-19 | 240 | ING_T |
| 8 | 1462 15-JUL-22 | 29-NOV-22 | 220 | BAR_C |
| 9 | 1411 07-MAY-18 | 11-DEC-20 | 260 | PLAS |
| 10 | 1340 09-DEC-21 | 09-FEB-22 | 160 | BAR_C |

**UPDATE angajat
SET cod_cinema = 200, cod_job = 'PLAS', id_superior = 1170
WHERE cod_angajat = 1272;**

```
UPDATE angajat
SET cod_cinema = 200, cod_job = 'PLAS', id_superior = 1170
WHERE cod_angajat = 1272;
```

Script Output | Query Result | SQL | Task completed in 0.24 seconds

1 row updated.

Trigger AFTER declanșat de faptul că am făcut UPDATE unui angajat modificându-i job-ul, cinematograful și superiorul.

Triger-ul AFTER îmi va inseră vechile valori ale angajatului modificat în tabela ISTORIC_LUCRU și în plus va actualiza tabela ANGAJAT, modificându-se pentru subalterni coloana "id_superior", iar pentru angajat, salariul acestuia astfel încât să corespundă cu noul job, și de asemenea, data de angajare va fi actualizată.

```

SELECT * FROM istoric_lucru;

```

| | COD_ANGAJAT | DATA_START | DATA_DEMISIE | COD_CINEMA | COD_JOB |
|---|-------------|------------|--------------|------------|---------|
| 1 | 1272 | 15-MAY-17 | 11-JAN-23 | 220 | MAN |
| 2 | 1034 | 03-MAY-18 | 05-MAY-19 | 120 | AG_C |
| 3 | 1102 | 17-JUL-19 | 18-NOV-19 | 120 | PLAS |
| 4 | 1221 | 19-NOV-18 | 19-AUG-20 | 260 | MAN |
| 5 | 1255 | 07-NOV-20 | 20-JUN-21 | 160 | BAR |
| 6 | 1110 | 07-MAY-14 | 07-JUN-15 | 260 | TMC_S |


```

SELECT * FROM angajat;

```

| | COD_ANGAJAT | NUME | PRENUME | EMAIL | TELEFON | DATA_ANG | SALARIU | NR_ORE | ID_SUPERIOR | COD_CINEMA | COD_JOB |
|----|-------------|----------|-----------|------------------------|--------------|-----------|---------|--------|-------------|------------|---------|
| 14 | 1238 | Nils | Corina | corina.nils@gmail.com | (null) | 01-MAR-19 | 6100.5 | 9 | (null) | 160 | MAN |
| 15 | 1255 | Petre | Cosmin | cosmin.petre@gmail.com | 0718.223.344 | 26-JUN-21 | 5972.5 | 7 | 1255 | 160 | ING_S |
| 16 | 1272 | Popa | Maria | maria.popa@yahoo.com | 0792.233.011 | 15-MAY-17 | 6132.1 | 8 | (null) | 220 | MAN |
| 17 | 1289 | Dumitran | George | george.dumi@gmail.com | 0793.557.876 | 22-JUN-19 | 4432.1 | 4 | (null) | 160 | ING_T |
| 18 | 1306 | Stanciu | Ecaterina | eca_stnc@gmail.com | 0732.323.233 | 12-DEC-21 | 3693.6 | 6 | 1170 | 200 | ING_S |
| 19 | 1323 | Chesaru | Ana | ana.chesaru@yahoo.com | 0723.324.443 | 06-OCT-22 | 1922 | 8 | 1272 | 220 | BAR |
| 20 | 1340 | Toth | Raluca | raluca_toth@gmail.com | 0799.112.321 | 15-FEB-22 | 1666.5 | 4 | 1272 | 220 | BAR |
| 21 | 1357 | Luca | Ioan | ioan.luca@yahoo.com | 0791.211.322 | 07-JUN-22 | 4747.1 | 8 | 1289 | 160 | ING_T |
| 22 | 1394 | Coca | Emanuel | emanuel.coca@gmail.com | 0790.134.755 | 01-JAN-19 | 2525.6 | 4 | 1136 | 180 | PLAS |
| 23 | 1411 | Pitulan | Gabriela | gabitsa.p@gmail.com | 0700.212.332 | 12-DEC-20 | 2999.2 | 6 | 1238 | 160 | PLAS |
| 24 | 1428 | Miron | Luminita | lumi.miron@yahoo.com | (null) | 06-OCT-22 | 5555.5 | 5 | 1272 | 220 | CAS |
| 25 | 1445 | Tanase | Vlad | vlad.t@gmail.com | 0731.345.788 | 01-DEC-22 | 5050.3 | 6 | 1428 | 220 | CAS |

Înainte de trigger

| | COD_ANGAJAT | NUME | PRENUME | EMAIL | TELEFON | DATA_ANG | SALARIU | NR_ORE | ID_SUPERIOR | COD_CINEMA | COD_JOB |
|----|-------------|----------|-----------|------------------------|--------------|-----------|---------|--------|-------------|------------|---------|
| 14 | 1238 | Nils | Corina | corina.nils@gmail.com | (null) | 01-MAR-19 | 6100.5 | 9 | (null) | 160 | MAN |
| 15 | 1255 | Petre | Cosmin | cosmin.petre@gmail.com | 0718.223.344 | 26-JUN-21 | 5972.5 | 7 | 1255 | 160 | ING_S |
| 16 | 1272 | Popa | Maria | maria.popa@yahoo.com | 0792.233.011 | 11-JAN-23 | 3432.43 | 8 | 1170 | 200 | PLAS |
| 17 | 1289 | Dumitran | George | george.dumi@gmail.com | 0793.557.876 | 22-JUN-19 | 4432.1 | 4 | (null) | 160 | ING_T |
| 18 | 1306 | Stanciu | Ecaterina | eca_stnc@gmail.com | 0732.323.233 | 12-DEC-21 | 3693.6 | 6 | 1170 | 200 | ING_S |
| 19 | 1323 | Chesaru | Ana | ana.chesaru@yahoo.com | 0723.324.443 | 06-OCT-22 | 1922 | 8 | (null) | 220 | BAR |
| 20 | 1340 | Toth | Raluca | raluca_toth@gmail.com | 0799.112.321 | 15-FEB-22 | 1666.5 | 4 | (null) | 220 | BAR |
| 21 | 1357 | Luca | Ioan | ioan.luca@yahoo.com | 0791.211.322 | 07-JUN-22 | 4747.1 | 8 | 1289 | 160 | ING_T |
| 22 | 1394 | Coca | Emanuel | emanuel.coca@gmail.com | 0790.134.755 | 01-JAN-19 | 2525.6 | 4 | 1136 | 180 | PLAS |
| 23 | 1411 | Pitulan | Gabriela | gabitsa.p@gmail.com | 0700.212.332 | 12-DEC-20 | 2999.2 | 6 | 1238 | 160 | PLAS |
| 24 | 1428 | Miron | Luminita | lumi.miron@yahoo.com | (null) | 06-OCT-22 | 5555.5 | 5 | (null) | 220 | CAS |
| 25 | 1445 | Tanase | Vlad | vlad.t@gmail.com | 0731.345.788 | 01-DEC-22 | 5050.3 | 6 | 1428 | 220 | CAS |

După trigger

UPDATE angajat

```

SET cod_cinema = 200, cod_job = 'PLAS', id_superior = 1000
WHERE cod_angajat = 1272;

```

```
UPDATE angajat
SET cod_cinema = 200, cod_job = 'PLAS', id_superior = 1000
WHERE cod_angajat = 1272;

Script Output | Query Result
Task completed in 0.098 seconds

Error starting at line : 1,730 in command -
UPDATE angajat
SET cod_cinema = 200, cod_job = 'PLAS', id_superior = 1000
WHERE cod_angajat = 1272
Error report -
ORA-20002: Id-ul superiorului este gresit, deoarece nu exista un angajat cu acest cod in cinematograful dorit!
ORA-06512: at "C##SGBD_STEF.TRIG_BEFORE_PROJECT_EX11_RS", line 51
ORA-04088: error during execution of trigger 'C##SGBD_STEF.TRIG_BEFORE_PROJECT_EX11_RS'
```

Trigger BEFORE declanșat de faptul că am încercat să fac UPDATE unui angajat, modificându-i jobul, cinematograful și superiorul, dar noul superior atrăiuit nu se regăsește în cinematograful în care l-am mutat.

Dezactivare triggeri:

```
DROP TRIGGER trig_before_project_ex11_rs;
DROP TRIGGER trig_after_project_ex11_rs;

Script Output | Query Result
Task completed in 0.065 seconds
```

Trigger TRIG_BEFORE_PROJECT_EX11_RS dropped.

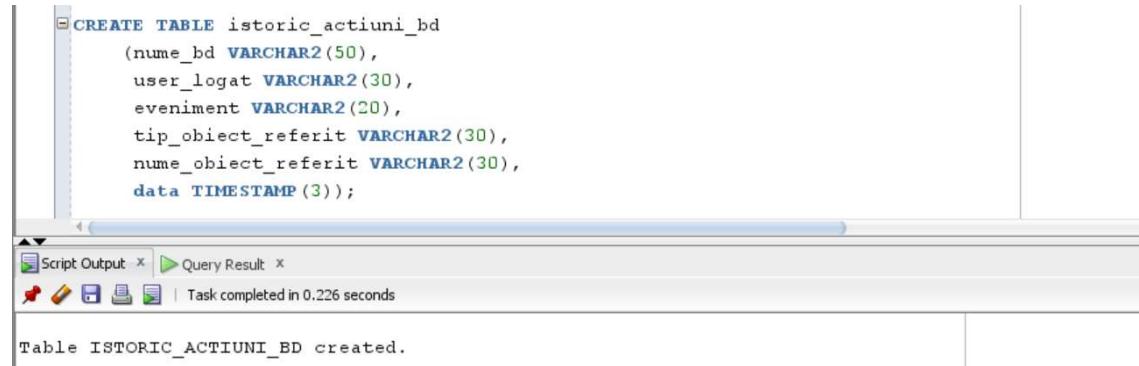
Trigger TRIG_AFTER_PROJECT_EX11_RS dropped.

12. Trigger LDD la nivel de schemă

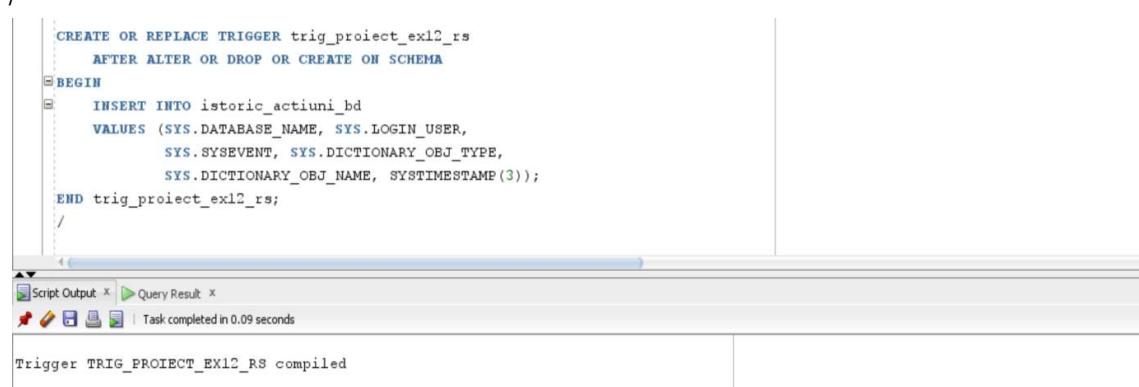
Definiți un trigger de tip LDD. Declanșați trigger-ul.

Vrem să creem un tabel de istoric acțiuni din baza de date, în care vom introduce date corespunzătoare folosindu-ne de un trigger de tip LDD.

```
CREATE TABLE istoric_actiuni_bd
  (nume_bd VARCHAR2(50),
  user_logat VARCHAR2(30),
  eveniment VARCHAR2(20),
  tip_object_referit VARCHAR2(30),
  nume_object_referit VARCHAR2(30),
  data TIMESTAMP(3));
```



```
CREATE OR REPLACE TRIGGER trig_proiect_ex12_rs
  AFTER ALTER OR DROP OR CREATE ON SCHEMA
BEGIN
  INSERT INTO istoric_actiuni_bd
    VALUES (SYS.DATABASE_NAME, SYS.LOGIN_USER,
            SYS.SYSEVENT, SYS.DICTIONARY_OBJ_TYPE,
            SYS.DICTIONARY_OBJ_NAME, SYSTIMESTAMP(3));
END trig_proiect_ex12_rs;
/
```



Declanșarea acestui trigger:

The screenshot shows the Oracle SQL Developer interface. A query is being run in the 'Query Result' tab:

```
SELECT * FROM ISTORIC_ACTIUNI_BD;
```

The results show 20 rows of data from the table, with columns: NUME_BD, USER_LOGAT, EVENIMENT, TIP_OBJECT_REFERIT, NUME_OBJECT_REFERIT, and DATA.

| NUME_BD | USER_LOGAT | EVENIMENT | TIP_OBJECT_REFERIT | NUME_OBJECT_REFERIT | DATA |
|---------|--------------|-----------|--------------------|-----------------------------|---------------------------------|
| 1 XE | C##SGBD_STEF | CREATE | PROCEDURE | PROC_PROJECT_EX6_RS | 11-JAN-23 02.25.07.588000000 PM |
| 2 XE | C##SGBD_STEF | DROP | PROCEDURE | PROC_PROJECT_EX6_RS | 11-JAN-23 02.25.15.302000000 PM |
| 3 XE | C##SGBD_STEF | CREATE | PROCEDURE | PROC_PROJECT_EX7_RS | 11-JAN-23 02.25.20.516000000 PM |
| 4 XE | C##SGBD_STEF | DROP | PROCEDURE | PROC_PROJECT_EX7_RS | 11-JAN-23 02.25.28.250000000 PM |
| 5 XE | C##SGBD_STEF | CREATE | FUNCTION | FUNCT_PROJECT_EX8_RS | 11-JAN-23 02.25.32.942000000 PM |
| 6 XE | C##SGBD_STEF | DROP | FUNCTION | FUNCT_PROJECT_EX8_RS | 11-JAN-23 02.26.07.810000000 PM |
| 7 XE | C##SGBD_STEF | CREATE | TYPE | OBJ | 11-JAN-23 02.26.12.864000000 PM |
| 8 XE | C##SGBD_STEF | CREATE | TYPE | TAB_IMBR | 11-JAN-23 02.26.14.762000000 PM |
| 9 XE | C##SGBD_STEF | CREATE | TABLE | TAB_DATE_MEMBRII | 11-JAN-23 02.26.17.002000000 PM |
| 10 XE | C##SGBD_STEF | CREATE | INDEX | SYS_PK0000084734N00007\$ | 11-JAN-23 02.26.17.034000000 PM |
| 11 XE | C##SGBD_STEF | CREATE | INDEX | SYS_C009675 | 11-JAN-23 02.26.17.061000000 PM |
| 12 XE | C##SGBD_STEF | ALTER | TABLE | FILM | 11-JAN-23 02.26.17.128000000 PM |
| 13 XE | C##SGBD_STEF | CREATE | PROCEDURE | PROC_PROJECT_EX9_RS | 11-JAN-23 02.26.19.163000000 PM |
| 14 XE | C##SGBD_STEF | ALTER | TABLE | FILM | 11-JAN-23 02.26.25.982000000 PM |
| 15 XE | C##SGBD_STEF | DROP | TYPE | TAB_IMBR | 11-JAN-23 02.26.28.566000000 PM |
| 16 XE | C##SGBD_STEF | DROP | TYPE | OBJ | 11-JAN-23 02.26.30.818000000 PM |
| 17 XE | C##SGBD_STEF | DROP | PROCEDURE | PROC_PROJECT_EX9_RS | 11-JAN-23 02.26.32.851000000 PM |
| 18 XE | C##SGBD_STEF | CREATE | TRIGGER | TRIG_PROJECT_EX10_RS | 11-JAN-23 02.26.36.035000000 PM |
| 19 XE | C##SGBD_STEF | DROP | TRIGGER | TRIG_PROJECT_EX10_RS | 11-JAN-23 02.26.41.535000000 PM |
| 20 XE | C##SGBD_STEF | CREATE | TRIGGER | TRIG_BEFORE_PROJECT_EX11_RS | 11-JAN-23 02.26.44.735000000 PM |

Trigger-ul definit va introduce automat valori în tabelul definit pentru a menține informații cu privire la operațiile de create/alter/drop asupra schemei.

Dezactivare trigger:

The screenshot shows the Oracle SQL Developer interface. A query is being run in the 'Query Result' tab:

```
DROP TRIGGER trig_project_ex12_rs;
```

The results show the trigger has been dropped:

Trigger TRIG_PROJECT_EX12_RS dropped.

13. Pachet ce include toate obiectele definite în proiect

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

Tipul OBJECT nu poate fi inclus într-un pachet, decât dacă este creat prin intermediul unei comenzi ce conține SQL dinamic. De asemenea, comanda ALTER TABLE, efectuată într-un pachet, trebuie introdusă prin intermediul unei comenzi de SQL dinamic.

Totuși, în cadrul procedurii ”proc_proiect_ex9_rs” folosesc tabela FILM modificată, ceea ce înseamnă ca aceasta trebuie alterată înainte de a compila corpul pachetului în cadrul căreia am inclus procedura ”proc_proiect_ex9_rs”, deoarece până nu se inițiază executarea pachetului, acesta nu apelează funcțiile din interiorul său, ci caută erori la compilare.

Pentru aceasta, am creat un pachet auxiliar care să se realizeze adăugarea și ștergerea coloanei auxiliare, precum și definirea tipului OBJECT și a tabelului ce conține înregistrări ale tipului definit mai sus.

```
CREATE OR REPLACE PACKAGE pach_aux AS
```

```
    PROCEDURE altering_film;
```

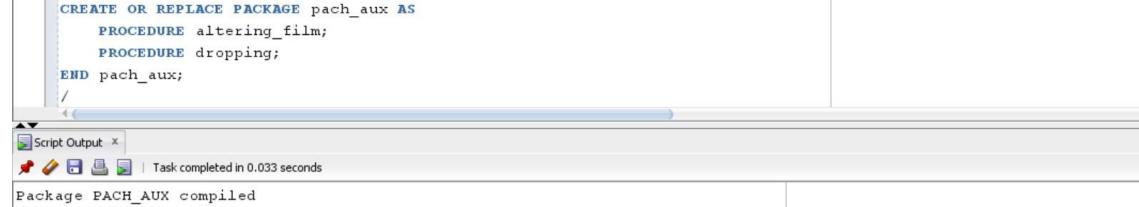
```
    PROCEDURE dropping;
```

```
END pach_aux;
```

```
/
```

```
CREATE OR REPLACE PACKAGE pach_aux AS
    PROCEDURE altering_film;
    PROCEDURE dropping;
END pach_aux;
```

```
/
```



```
Script Output X | Task completed in 0.033 seconds
```

```
Package PACH_AUX compiled
```

```
CREATE OR REPLACE PACKAGE BODY pach_aux AS
```

```
    PROCEDURE altering_film IS
```

```
        BEGIN
```

```
            EXECUTE IMMEDIATE 'CREATE OR REPLACE TYPE obj IS OBJECT(nr_premii
NUMBER,part NUMBER)';
```

```
            EXECUTE IMMEDIATE 'CREATE OR REPLACE TYPE tab_imbr IS TABLE OF obj';
```

```
            EXECUTE IMMEDIATE 'ALTER TABLE film ADD (date_membrii tab_imbr) NESTED
TABLE date_membrii STORE AS tab_date_membrii';
```

```
        END altering_film;
```

```
    PROCEDURE dropping IS
```

```
        BEGIN
```

```
            EXECUTE IMMEDIATE 'ALTER TABLE film DROP COLUMN date_membrii';
```

```
            EXECUTE IMMEDIATE 'DROP TYPE tab_imbr';
```

```
            EXECUTE IMMEDIATE 'DROP TYPE obj';
```

```
        END dropping;
```

```
    END pach_aux;
```

```
/
```

```

CREATE OR REPLACE PACKAGE BODY pach_aux AS
    PROCEDURE altering_film IS
        BEGIN
            EXECUTE IMMEDIATE 'CREATE OR REPLACE TYPE obj IS OBJECT(nr_premii NUMBER, part NUMBER)';
            EXECUTE IMMEDIATE 'CREATE OR REPLACE TYPE tab_imbr IS TABLE OF obj';
            EXECUTE IMMEDIATE 'ALTER TABLE film ADD (date_membrii tab_imbr) NESTED TABLE date_membrii STORE AS tab_date_membrii';
        END altering_film;

    PROCEDURE dropping IS
        BEGIN
            EXECUTE IMMEDIATE 'ALTER TABLE film DROP COLUMN date_membrii';
            EXECUTE IMMEDIATE 'DROP TYPE tab_imbr';
            EXECUTE IMMEDIATE 'DROP TYPE obj';
        END dropping;
    END pach_aux;
/

```

Script Output x | Task completed in 0.031 seconds
Package Body PACH_AUX compiled

EXECUTE pach_aux.altering_film;

```

EXECUTE pach_aux.altering_film;

```

Script Output x | Task completed in 0.08 seconds
PL/SQL procedure successfully completed.

Pachetul propriu-zis:

```

CREATE OR REPLACE PACKAGE pachet_proiect_ex13_rs AS
    PROCEDURE proc_proiect_ex6_rs;

    PROCEDURE proc_proiect_ex7_rs(optiune NUMBER);

    FUNCTION funct_proiect_ex8_rs(gen IN film.gen_princ%TYPE, an IN CHAR) RETURN NUMBER;

    PROCEDURE proc_proiect_ex9_rs(an CHAR, numeA VARCHAR2);

```

END pachet_proiect_ex13_rs;

```

/
CREATE OR REPLACE PACKAGE pachet_proiect_ex13_rs AS
    PROCEDURE proc_proiect_ex6_rs;

    PROCEDURE proc_proiect_ex7_rs(optiune NUMBER);

    FUNCTION funct_proiect_ex8_rs(gen IN film.gen_princ%TYPE, an IN CHAR) RETURN NUMBER;

    PROCEDURE proc_proiect_ex9_rs(an CHAR, numeA VARCHAR2);

    END pachet_proiect_ex13_rs;
/

```

Script Output x | Task completed in 0.049 seconds
Package PACHET_PROIECT_EX13_RS compiled

```

CREATE OR REPLACE PACKAGE BODY pachet_proiect_ex13_rs AS
  PROCEDURE proc_proiect_ex6_rs IS
    CURSOR c(parametru cinematograf.cod_cinema%TYPE) IS
      SELECT c.nume_cinema, f.nume_film, f.rating, f.an_aparitie
      FROM film f
      JOIN (SELECT DISTINCT cod_cinema, cod_film
            FROM ruleaza) r ON r.cod_film = f.cod_film
      JOIN (SELECT cod_cinema, nume_cinema
            FROM cinematograf) c ON r.cod_cinema = c.cod_cinema
      WHERE c.cod_cinema = parametru
      ORDER BY c.cod_cinema, f.rating DESC, f.nume_film, f.an_aparitie;

    TYPE record_CF IS RECORD
      (poz NUMBER,
       nume_cinema cinematograf.nume_cinema%TYPE,
       nume_film film.nume_film%TYPE,
       rating film.rating%TYPE,
       an_aparitie film.an_aparitie%TYPE);

    TYPE tab_imb_CF IS TABLE OF record_CF;
    t_CF tab_imb_CF := tab_imb_CF();

    TYPE varr_cinemaC IS VARRAY(20) OF cinematograf.cod_cinema%TYPE;
    vect_cod_cin varr_cinemaC;

    TYPE varr_cinemaN IS VARRAY(20) OF cinematograf.nume_cinema%TYPE;
    vect_num_cin varr_cinemaN;

    TYPE tab_ind_cnt IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
    t_cntFilm tab_ind_cnt;

    cnt_FilmDist NUMBER;
    v_prec film.rating%TYPE;
    v_top NUMBER := 1;
    v_cnt NUMBER := 1;

  BEGIN
    SELECT c.cod_cinema, nume_cinema, COUNT(cod_film)
    BULK COLLECT INTO vect_cod_cin, vect_num_cin, t_cntFilm
    FROM cinematograf c, ruleaza r
    WHERE c.cod_cinema = r.cod_cinema(+)
    GROUP BY c.cod_cinema, nume_cinema;

    FOR i IN vect_cod_cin.FIRST..vect_cod_cin.LAST LOOP

```

```

dbms_output.put_line('-----');
-----');

IF t_cntFilm(i) = 0 THEN
    dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' NU a rulat
NICIUN film pana in prezent.');
    dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3 filme
care au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');

ELSIF t_cntFilm(i) = 1 THEN
    dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' a rulat un
singur film pana in prezent.');
    dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3 filme
care au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');

ELSIF t_cntFilm(i) = 2 THEN
    dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat
doar 2 filme pana in prezent.');
    dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3 filme
care au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');

ELSE
    v_top := 1;
    v_cnt := 1;
    SELECT COUNT(f.cod_film) INTO cnt_FilmDist
    FROM film f
    JOIN (SELECT DISTINCT cod_cinema, cod_film
          FROM ruleaza) r ON r.cod_film = f.cod_film
    JOIN (SELECT cod_cinema, nume_cinema
          FROM cinematograf) c ON r.cod_cinema = c.cod_cinema
    WHERE c.cod_cinema = vect_cod_cin(i)
    ORDER BY c.cod_cinema, f.rating DESC, f.nume_film, f.an_aparitie;
    IF cnt_FilmDist = 1 THEN
        dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat '
|| t_cntFilm(i) || ' filme, dar eliminand duplicatele doar un singur film DISTINCT pana in
prezent.');
        dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3 filme
care au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');

ELSIF cnt_FilmDist = 2 THEN
        dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat '
|| t_cntFilm(i) || ' filme, dar dar eliminand duplicatele doar 2 filme DISTINCTE pana in
prezent.');
        dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3 filme
care au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');

ELSE
    OPEN c(vect_cod_cin(i));
    t_CF.EXTEND;
    FETCH c INTO t_CF(v_cnt).nume_cinema, t_CF(v_cnt).nume_film,
t_CF(v_cnt).rating, t_CF(v_cnt).an_aparitie;
    t_CF(v_cnt).poz := v_top;

```

```

v_prec := t_CF(v_cnt).rating;
LOOP
    t_CF.EXTEND;
    v_cnt := v_cnt + 1;
    FETCH c INTO t_CF(v_cnt).nume_cinema, t_CF(v_cnt).nume_film,
t_CF(v_cnt).rating, t_CF(v_cnt).an_aparitie;
        EXIT WHEN c%NOTFOUND;
        IF t_CF(v_cnt).rating <> v_prec THEN
            v_top := v_top + 1;
        END IF;
        EXIT WHEN v_top = 4;
        t_CF(v_cnt).poz := v_top;
        v_prec := t_CF(v_cnt).rating;
    END LOOP;
    IF v_top <> 4 THEN
        dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat
' || t_cntFilm(i) || ' filme, iar eliminand duplicatele ' || cnt_FilmDist || ' filme DISTINCTE, dar
ratingurile nu pot determina un top 3 cele mai indragite filme.');
        dbms_output.put_line('---NU se poate reliza un top al celor mai bine cotate 3
filme care au rulat in cinematograful ' || UPPER(vect_num_cin(i)) || '!---');
    ELSE
        dbms_output.put_line('In cinematograful ' || UPPER(vect_num_cin(i)) || ' au rulat
' || t_cntFilm(i) || ' filme, iar eliminand duplicatele ' || cnt_FilmDist || ' filme DISTINCTE.');
        dbms_output.put_line('---Top 3 cele mai bine cotate filme care au rulat in
cinematograful ' || UPPER(vect_num_cin(i)) || '---');
        v_top := 0;
    FOR j IN t_CF.FIRST..t_CF.LAST LOOP
        IF t_CF(j).poz <> v_top THEN
            v_top := t_CF(j).poz;
            dbms_output.new_line;
            dbms_output.put_line('-----LOCUL ' || v_top || ' -----');
            dbms_output.put_line('---Ratingul: ' || t_CF(j).rating || ' ---');
        END IF;
        dbms_output.put_line('Filmul: ' || t_CF(j).nume_film || ', aparut in anul: ' ||
t_CF(j).an_aparitie);
    END LOOP;
    END IF;
    t_CF.DELETE;
    CLOSE c;
    END IF;
END IF;
dbms_output.put_line('-----');
dbms_output.new_line();
END LOOP;

```

```

EXCEPTION
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002, 'Alta eroare');
END proc_proiect_ex6_rs;

PROCEDURE proc_proiect_ex7_rs(optiune NUMBER) IS
    --expresie cursor imbricata
    CURSOR c_expr IS SELECT nume_cinema,
        CURSOR (SELECT nume_job, CURSOR (SELECT nume || ' ' || prenume
            FROM angajat a
            WHERE a.cod_cinema = c.cod_cinema
            AND j.cod_job = a.cod_job)
            FROM job j)
        FROM cinematograf c;
    v_numecinema cinematograf.numecinema%TYPE;
    v_numejob job.numejob%TYPE;
    v_numearg VARCHAR2(70);
    v_nrarg NUMBER;
    cnt NUMBER;
    TYPE refcursor IS REF CURSOR;
    v_cursor_imbr refcursor;
    v_cursor refcursor;

    --cursor explicit
    CURSOR c_expl IS SELECT c.cod_cinema v_cod_cinema, c.nume_cinema
    v_numecinema, COUNT(a.cod_angajat) v_cnt_ang,
        (SELECT COUNT(*)
            FROM istoric_lucru il
            WHERE c.cod_cinema = il.cod_cinema(+))v_hist_ang
        FROM cinematograf c, angajat a
        WHERE c.cod_cinema = a.cod_cinema(+)
        GROUP BY c.cod_cinema, c.nume_cinema;

    --cursor parametrizat
    CURSOR c_param(param_cin cinematograf.cod_cinema%TYPE, param_job
    job.numejob%TYPE)
        IS SELECT nume || ' ' || prenume
        FROM angajat a, job j
        WHERE cod_cinema = param_cin AND a.cod_job = j.cod_job AND j.nume_job =
        param_job;
    --cursor parametrizat
    CURSOR c_param_hist(param_cin cinematograf.cod_cinema%TYPE, param_job
    job.numejob%TYPE)
        IS SELECT nume || ' ' || prenume

```

```

    FROM angajat a, job j, istoric_lucru il
    WHERE il.cod_cinema = param_cin AND il.cod_job = j.cod_job
        AND j.nume_job = param_job AND il.cod_angajat = a.cod_angajat;
BEGIN
    IF optiune = 1 THEN
        OPEN c_expr;
        LOOP
            FETCH c_expr INTO v_nume_cinema, v_cursor_imbr;
            EXIT WHEN c_expr%NOTFOUND;
            dbms_output.put_line('-----');
            dbms_output.put_line('Cinematograful ' || UPPER(v_nume_cinema));
            dbms_output.new_line();
            cnt := 0;
            LOOP
                FETCH v_cursor_imbr INTO v_nume_job, v_cursor;
                EXIT WHEN v_cursor_imbr%NOTFOUND;

                FETCH v_cursor INTO v_nume_ang;
                IF v_cursor%ROWCOUNT > 0 THEN
                    dbms_output.put_line('-----Jobul ' || UPPER(v_nume_job) || ' -----');
                    dbms_output.put_line('  Angajatul ' || v_nume_ang);
                    LOOP
                        FETCH v_cursor INTO v_nume_ang;
                        EXIT WHEN v_cursor%NOTFOUND;
                        dbms_output.put_line('  Angajatul ' || v_nume_ang);
                    END LOOP;
                    dbms_output.new_line();
                END IF;
                IF v_cursor%ROWCOUNT = 0 THEN
                    cnt := cnt + 1;
                END IF;
            END LOOP;
            IF v_cursor_imbr%ROWCOUNT = cnt THEN
                dbms_output.put_line('  NICIUN ANGAJAT ');
                dbms_output.new_line();
            END IF;
            dbms_output.put_line('-----');
        END LOOP;
        CLOSE c_expr;
    ELSIF optiune = 2 THEN
        --ciclul cursor
        FOR k IN c_expl LOOP

```

```

dbms_output.put_line('-----');
-----');

IF k.v_cnt_ang = 0 THEN
    IF k.v_hist_ang = 0 THEN
        dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' nu
lucreaza si nu a lucrat NICIUN angajat');

ELSIF k.v_hist_ang = 1 THEN
    dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' nu
lucreaza in prezent NICIUN angajat, insa in trecut a mai lucrat un singur angajat');

ELSE
    dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || ' nu
lucreaza in prezent NICIUN angajat, insa in trecut au mai lucrat ' || k.v_hist_ang || ' angajati');

END IF;

ELSE
    IF k.v_cnt_ang = 1 THEN
        IF k.v_hist_ang = 0 THEN
            dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent un singur angajat, iar in trecut nu a mai lucrat NICIUN angajat');

ELSIF k.v_hist_ang = 1 THEN
    dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent un singur angajat, iar in trecut a mai lucrat un singur angajat');

ELSE
    dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent un singur angajat, iar in trecut au mai lucrat ' || k.v_hist_ang || ' angajati');

END IF;

ELSE
    IF k.v_hist_ang = 0 THEN
        dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent ' || k.v_cnt_ang || ' angajati, iar in trecut nu a mai lucrat NICIUN angajat');

ELSIF k.v_hist_ang = 1 THEN
    dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent ' || k.v_cnt_ang || ' angajati, iar in trecut a mai lucrat un singur angajat');

ELSE
    dbms_output.put_line('In cinematograful ' || UPPER(k.v_nume_cinema) || '
lucreaza in prezent ' || k.v_cnt_ang || ' angajati, iar in trecut au mai lucrat ' || k.v_hist_ang || '
angajati');

END IF;

END IF;

--ciclul cursor cu subcereri

FOR i IN (SELECT nume_job, (SELECT COUNT(a.cod_angajat)
FROM angajat a
WHERE a.cod_job(+) = j.cod_job
AND a.cod_cinema = k.v_cod_cinema) nr_ang,
(SELECT COUNT(il.cod_angajat)
FROM istoric_lucru il

```

```

        WHERE j.cod_job = il.cod_job(+)
        AND il.cod_cinema = k.v_cod_cinema) hist_ang
    FROM job j) LOOP
    IF i.nr_ang != 0 THEN
        OPEN c_param(k.v_cod_cinema, i.nume_job);
        dbms_output.new_line();
        dbms_output.put_line('-----Jobul ' || UPPER(i.nume_job) || ' -----');
        LOOP
            FETCH c_param INTO v_nume_ang;
            EXIT WHEN c_param%NOTFOUND;
            dbms_output.put_line(' (Actual) Angajatul ' || v_nume_ang);
        END LOOP;
        CLOSE c_param;
        IF i.hist_ang != 0 THEN
            OPEN c_param_hist(k.v_cod_cinema, i.nume_job);
            LOOP
                FETCH c_param_hist INTO v_nume_ang;
                EXIT WHEN c_param_hist%NOTFOUND;
                dbms_output.put_line(' (Istoric) Angajatul ' || v_nume_ang);
            END LOOP;
            CLOSE c_param_hist;
        END IF;
        ELSIF i.hist_ang != 0 THEN
            OPEN c_param_hist(k.v_cod_cinema, i.nume_job);
            dbms_output.new_line();
            dbms_output.put_line('-----Jobul ' || UPPER(i.nume_job) || ' -----');
            LOOP
                FETCH c_param_hist INTO v_nume_ang;
                EXIT WHEN c_param_hist%NOTFOUND;
                dbms_output.put_line(' (Istoric) Angajatul ' || v_nume_ang);
            END LOOP;
            CLOSE c_param_hist;
        END IF;
        END LOOP;
    END IF;
    dbms_output.put_line('-----');
    dbms_output.new_line();
END LOOP;
ELSE
    dbms_output.put_line('Optiunea nu e valida');
END IF;
EXCEPTION
WHEN OTHERS THEN

```

```

        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare');
END proc_proiect_ex7_rs;

FUNCTION funct_proiect_ex8_rs(gen IN film.gen_princ%TYPE, an IN CHAR)
    RETURN NUMBER IS
    exc_err_an_neg EXCEPTION;
    exc_err_an EXCEPTION;
    exc_err_gen EXCEPTION;
    exc_err_dif EXCEPTION;
    exc_err_bilet EXCEPTION;
    cnt NUMBER;
    pret NUMBER;
BEGIN
    IF an < 0 THEN
        RAISE exc_err_an_neg;
    END IF;
    IF an > TO_CHAR(SYSDATE, 'YYYY') THEN
        RAISE exc_err_an;
    END IF;
    SELECT COUNT(*) INTO cnt
    FROM film
    WHERE UPPER(gen_princ) = UPPER(gen);
    IF cnt = 0 THEN
        RAISE exc_err_gen;
    END IF;
    SELECT COUNT(cod_difuzare) INTO cnt
    FROM film f, difuzare d
    WHERE UPPER(gen_princ) = UPPER(gen) AND f.cod_film = d.cod_film AND
    TO_CHAR(data_inc, 'YYYY') >= an;
    IF cnt = 0 THEN
        RAISE exc_err_dif;
    END IF;
    --JOIN ce include 3 table (FILM, DIFUZARE, BILET_CUMPARAT)
    SELECT SUM(bc.pret) INTO pret
    FROM film f, difuzare d, bilet_cumparat bc
    WHERE UPPER(f.gen_princ) = UPPER(gen) AND f.cod_film = d.cod_film
        AND bc.cod_difuzare(+) = d.cod_difuzare AND TO_CHAR(d.data_inc, 'YYYY') >= an;
    IF pret is null THEN
        RAISE exc_err_bilet;
    END IF;
    RETURN pret;
EXCEPTION
    WHEN exc_err_an_neg THEN
        RAISE_APPLICATION_ERROR(-20003, 'Nu puteti introduce un an negativ!');

```

```

WHEN exc_err_an THEN
    RAISE_APPLICATION_ERROR(-20004, 'Ati introdus un an mai mare decat anul
current, si nu exista date despre difuzari din viitor!');

WHEN exc_err_gen THEN
    RAISE_APPLICATION_ERROR(-20005, 'Nu exista niciun film care sa aiba genul ' ||
UPPER(gen) || '!');

WHEN exc_err_dif THEN
    RAISE_APPLICATION_ERROR(-20006, 'Nu exista nicio difuzare pentru filme de
genul ' || UPPER(gen) || ' care sa fi avut loc dupa anul ' || an || '!');

WHEN exc_err_bilet THEN
    RAISE_APPLICATION_ERROR(-20007, 'Nu exista niciun bilet vandut la difuzare care
a avut loc dupa anul ' || an || ' a vreunui film cu genul ' || UPPER(gen) || '!');

WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002, 'Alta eroare');

END funct_proiect_ex8_rs;

PROCEDURE proc_proiect_ex9_rs(an CHAR, numeA VARCHAR2) IS
    exc_an_neg EXCEPTION;
    exc_an EXCEPTION;
    exc_an_NDF EXCEPTION;
    cnt NUMBER;
    an_deb actor.an_debut%TYPE;
BEGIN
    IF an < 0 THEN
        RAISE exc_an_neg;
    END IF;
    IF an > TO_CHAR(SYSDATE, 'YYYY') THEN
        RAISE exc_an;
    END IF;

    SELECT COUNT(*) INTO cnt
    FROM cinematograf
    WHERE to_char(data_desch, 'YYYY') >= an AND (UPPER(to_char(data_desch, 'DAY')) LIKE '%MONDAY%' OR UPPER(to_char(data_desch, 'DAY')) LIKE '%SATURDAY%');
    IF cnt = 0 THEN
        RAISE exc_an_NDF;
    END IF;

    SELECT an_debut INTO an_deb
    FROM actor
    WHERE (LOWER(prenume) LIKE LOWER(numeA) OR LOWER(nume) LIKE
LOWER(numeA));

```

--comandă SQL ce include 5 tabele, prin intermediul subcererilor (ACTOR, JOACA, REGIZOR, REGIZEAZA, FILM)

```

FOR i IN (SELECT f.cod_film cod, NVL((SELECT SUM(nr_premii)
                                         FROM actor a, joaca j
                                         WHERE a.id_actor = j.id_actor AND j.cod_film = f.cod_film
                                         AND a.an_debut <> an_deb), 0) premii_act,
                  NVL((SELECT COUNT(a.id_actor)
                                         FROM actor a, joaca j
                                         WHERE a.id_actor = j.id_actor AND j.cod_film = f.cod_film
                                         AND a.an_debut <> an_deb), 0) nr_act,
                  NVL((SELECT SUM(nr_premii)
                                         FROM regizor r, regizeaza ra
                                         WHERE r.id_regizor = ra.id_regizor AND ra.cod_film = f.cod_film), 0)
premii_reg,
                  NVL((SELECT COUNT(nr_premii)
                                         FROM regizor r, regizeaza ra
                                         WHERE r.id_regizor = ra.id_regizor AND ra.cod_film = f.cod_film), 0) nr_reg
FROM film f) LOOP
UPDATE film
SET date_membrii = tab_imbr( obj(i.premii_act, i.nr_act), obj(i.premii_reg, i.nr_reg))
WHERE cod_film = i.cod;
END LOOP;
```

--comandă SQL ce include 5 tabele, prin intermediul JOIN-ului (FILM, RULEAZA, DIFUZARE, CINEMATOGRAF, SALA), la care se adaugă prin intermediul subcererilor încă o data tabela FILM și coloana adițională din tabela film preluată ca TABLE

```

FOR i IN (SELECT c.nume_cinema numeC, to_char(data_desch, 'DAY') zi, f.nume_film
numeF, d.nr_sala salaD,
s.tip salaT, f.rating ratingF,
(SELECT SUM(b.part)
FROM film a, TABLE (a.date_membrii) b
WHERE a.cod_film = f.cod_film) nr_part,
(SELECT SUM(b.nr_premii)
FROM film a, TABLE (a.date_membrii) b
WHERE a.cod_film = f.cod_film) nr_premii,
(SELECT nvl(SUM(b.nr_premii)/ NULLIF(SUM(b.part), 0), 0)
FROM film a, TABLE (a.date_membrii) b
WHERE a.cod_film = f.cod_film) medie_premii
FROM film f, ruleaza r, difuzare d, cinematograf c, sala s
WHERE c.cod_cinema = r.cod_cinema AND r.cod_film = f.cod_film
AND (UPPER(to_char(data_desch, 'DAY')) LIKE '%MONDAY%' OR
UPPER(to_char(data_desch, 'DAY')) LIKE '%SATURDAY%')
AND to_char(data_desch, 'YYYY') >= an
AND f.cod_film = d.cod_film
```

```

        AND d.nr_sala = s.nr_sala AND s.cod_cinema = c.cod_cinema
        AND (LOWER(s.tip) LIKE '%multi%' OR LOWER(s.tip) LIKE 'vip')
        AND months_between(r.data_final, r.data_inceput) > 4
    ORDER BY 7 DESC, 6 DESC, 5 DESC, 4 DESC LOOP

    IF UPPER(i.zi) LIKE '%SATURDAY%' THEN
        i.zi := 'SAMBATA';
    ELSE
        i.zi := 'LUNI';
    END IF;
    dbms_output.new_line();
    IF i.medie_premii = 0 THEN
        IF i.nr_part = 1 THEN
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' ||
i.zi || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
UPPER(i.numef) || '] [Rating: ' || i.ratingF || ']');
            dbms_output.put_line('Despre film se stie ca a avut ca a avut un singur membru in
productie, iar acesta nu a castigat NICIUN premiu');

        ELSIF i.nr_part > 0 THEN
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' ||
i.zi || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
UPPER(i.numef) || '] [Rating: ' || i.ratingF || ']');
            dbms_output.put_line('Despre film se stie ca a avut ca a avut ' || i.nr_part || '
membrui in productie, iar acestia nu a castigat NICIUN premiu');

        ELSE
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' ||
i.zi || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
UPPER(i.numef) || '] [Rating: ' || i.ratingF || ']');
            dbms_output.put_line('Pentru acest film nu s-au putut colectiona date despre actori
sau regizori');

        END IF;
    ELSE
        IF i.nr_premii = 1 THEN
            IF i.nr_part = 1 THEN
                dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' ||
i.zi || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
UPPER(i.numef) || '] [Rating: ' || i.ratingF || ']');
                dbms_output.put_line('Despre film se stie ca a avut ca a avut un singur membru
in productie, iar acesta a castigat in total un singur premiu');

            dbms_output.put_line('----- Media de premiere a filmului -----> ' ||
i.medie_premii);

        ELSE
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numec) || '] [Zi deschidere: ' ||
i.zi || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
UPPER(i.numef) || '] [Rating: ' || i.ratingF || ']');
            dbms_output.put_line('Despre film se stie ca a avut ca a avut ' || i.nr_part
|| ' membrii in productie, iar acestia au castigat in total un singur premiu');


```

```

        dbms_output.put_line('----- Media de premiere a filmului -----> ' ||
i.medie_premii);
        END IF;
    ELSE
        IF i.nr_part = 1 THEN
            dbms_output.put_line('[Cinematograf: ' || UPPER(i.numeC) || '] [Zi deschidere: ' ||
i.zi || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
UPPER(i.numeF)|| '] [Rating: ' || i.ratingF || ']');
            dbms_output.put_line('Despre film se stie ca a avut ca a avut un singur membru
in productie, iar acesta a castigat in total ' || i.nr_premii || ' premii');
            dbms_output.put_line('----- Media de premiere a filmului -----> ' ||
i.medie_premii);
            ELSE
                dbms_output.put_line('[Cinematograf: ' || UPPER(i.numeC) || '] [Zi deschidere: ' ||
i.zi || '] [Tip sala difuzare: ' || i.salaT || '] [Numar sala difuzare: ' || i.salaD || '] [Film: ' ||
UPPER(i.numeF)|| '] [Rating: ' || i.ratingF || ']');
                dbms_output.put_line('Despre film se stie ca a avut ' || i.nr_part || '
membruii in productie, iar acestia au castigat in total ' || i.nr_premii || ' premii');
                dbms_output.put_line('----- Media de premiere a filmului -----> ' ||
i.medie_premii);
                END IF;
            END IF;
        END IF;
    END LOOP;
EXCEPTION
    WHEN exc_an_neg THEN
        RAISE_APPLICATION_ERROR(-20003, 'Nu puteti introduce un an negativ!');
    WHEN exc_an THEN
        RAISE_APPLICATION_ERROR(-20004, 'Ati introdus un an mai mare decat anul
current, si nu exista date despre cinematografe deschise in viitor!');
    WHEN exc_an_NDF THEN
        RAISE_APPLICATION_ERROR(-20005, 'Nu exista niciun cinematograf deschis luni
sau sambata, dupa anul ' || an || '!');
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20006, 'Nu exista niciun actor cu numele sau
prenumele ' || UPPER(numeA) || '!');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20007, 'Există mai mulți actori cu numele sau
prenumele ' || UPPER(numeA) || '!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20008, 'Alta eroare!');
END proc_proiect_ex9_rs;
END packet_proiect_ex13_rs;
/

```

```

CREATE OR REPLACE PACKAGE BODY pachet_proiect_ex13_rs AS
    PROCEDURE proc_proiect_ex6_rs IS...
    PROCEDURE proc_proiect_ex7_rs(optiune NUMBER) IS ... 
    FUNCTION funct_proiect_ex8_rs(gen IN film.gen_princ%TYPE, an IN CHAR)...
    PROCEDURE proc_proiect_ex9_rs(an CHAR, numeA VARCHAR2) IS...
END pachet_proiect_ex13_rs;
/

```

Script Output | Task completed in 0.142 seconds

Package Body PACHET_PROIECT_EX13_RS compiled

Exemple de apelare a procedurilor și funcțiilor din interiorul pachetului:

EXECUTE pachet_proiect_ex13_rs.proc_proiect_ex6_rs;

```

EXECUTE pachet_proiect_ex13_rs.proc_proiect_ex6_rs;

```

Script Output | Query Result | Task completed in 0.065 seconds

PL/SQL procedure successfully completed.

Dbsm Output | Buffer Size:20000 | P_SGDB x

In cinematograful CINEMA CITY PLAKE au rulat 7 filme, iar eliminand duplicatele 7 filme DISTINCTE.
---Top 3 cele mai bine cotate filme care au rulat in cinematograful CINEMA CITY PLAKE---

-----LOCUL 1 -----
---Ratingul: 8.5 ---
Filmul: Zootopia, aparut in anul: 2016

-----LOCUL 2 -----
---Ratingul: 7.9 ---

EXECUTE pachet_proiect_ex13_rs.proc_proiect_ex7_rs(1);

```

EXECUTE pachet_proiect_ex13_rs.proc_proiect_ex7_rs(1);

```

Script Output | Query Result | Task completed in 0.043 seconds

PL/SQL procedure successfully completed.

Dbsm Output | Buffer Size:20000 | P_SGDB x

Cinematograful CINEMA CITY PLAKE

-----Jobul CASIER -----
Angajatul Matei Andrei
Angajatul Iana Gabriela

-----Jobul MANAGER -----
Angajatul Popescu Ion

EXECUTE pachet_proiect_ex13_rs.proc_proiect_ex7_rs(99);

```

EXECUTE pachet_proiect_ex13_rs.proc_proiect_ex7_rs(99);

```

Script Output | Query Result | Task completed in 0.03 seconds

PL/SQL procedure successfully completed.

Dbsm Output | Buffer Size:20000 | P_SGDB x

Optiunea nu e valida

```

DECLARE
    pret NUMBER;
BEGIN
    pret := pachet_proiect_ex13_rs.funct_proiect_ex8_rs('Teatru', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
/

```

The screenshot shows the Oracle SQL Developer interface with the code above. The code is executed successfully, and the output window displays the result:

```

DECLARE
    pret NUMBER;
BEGIN
    pret := pachet_proiect_ex13_rs.funct_proiect_ex8_rs('Teatru', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
/

```

PL/SQL procedure successfully completed.

DBMS Output:

```

Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: 79.7

```

```

DECLARE
    pret NUMBER;
BEGIN
    pret := pachet_proiect_ex13_rs.funct_proiect_ex8_rs('Actiune', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
/

```

The screenshot shows the Oracle SQL Developer interface with the code above. The code is executed successfully, and the output window displays the result:

```

DECLARE
    pret NUMBER;
BEGIN
    pret := pachet_proiect_ex13_rs.funct_proiect_ex8_rs('Actiune', '2019');
    dbms_output.put_line('Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: ' || pret);
END;
/

```

PL/SQL procedure successfully completed.

DBMS Output:

```

Suma preturilor biletelor vandute care sa respecte criteriile impuse este de: 79.7

```

```

EXECUTE pachet_proiect_ex13_rs.proc_proiect_ex9_rs('-10', 'Zendaya');

```

The screenshot shows the Oracle SQL Developer interface with the code above. An error occurs due to the negative value passed to the procedure. The error message is:

```

Error starting at line : 2,305 in command -
BEGIN pachet_proiect_ex13_rs.proc_proiect_ex9_rs('-10', 'Zendaya'); END;
Error report -
ORA-20003: Nu puteti introduce un an negativ!
ORA-06512: at "C##SGBD_STEF.PACHET_PROIECT_EX13_RS", line 454
ORA-06512: at line 1

```

```
EXECUTE pachet_proiect_ex13_rs.proc_proiect_ex9_rs('2000', 'Ryan');
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a script editor window containing the command: `EXECUTE pachet_proiect_ex13_rs.proc_proiect_ex9_rs('2000', 'Ryan');`. Below this, there are three tabs: "Script Output", "PL/SQL procedure successfully completed.", and "P_SGBD". The "P_SGBD" tab displays the results of the query, which include three movie records and their average premiere scores.

```
EXECUTE pachet_proiect_ex13_rs.proc_proiect_ex9_rs('2000', 'Ryan');

PL/SQL procedure successfully completed.

P_SGBD

[...]
[Cinematograf: CINEMA CITY API COTROCENI] [Zi deschidere: SAMBATA] [Tip sala difuzare: VIP] [Numar sala difuzare: 28] [Film: THE BATMAN] [
Despre film se stie ca a avut ca a avut 2 membrii in productie, iar acestia au castigat in total 31 premii
----- Media de premiere a filmului -----> 15.5

[Cinematograf: CINEMA CITY API COTROCENI] [Zi deschidere: SAMBATA] [Tip sala difuzare: VIP] [Numar sala difuzare: 31] [Film: DEATH ON THE
Despre film se stie ca a avut ca a avut 2 membrii in productie, iar acestia au castigat in total 25 premii
----- Media de premiere a filmului -----> 12.5

[Cinematograf: CINEMA CITY API COTROCENI] [Zi deschidere: SAMBATA] [Tip sala difuzare: VIP] [Numar sala difuzare: 28] [Film: RED NOTICE] [
Despre film se stie ca a avut ca a avut un singur membru in productie, iar acesta a castigat in total 3 premii
```

Aducerea tăblei FILM la structura inițială, fără coloana de tip vector, și ștergerea pachetului auxiliar și al celui definit pentru exercițiul dat.

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a script editor window containing the commands: `EXECUTE pach_aux.dropping;`, `DROP PACKAGE pach_aux;`, and `DROP PACKAGE pachet_proiect_ex13_rs;`. Below this, there are three tabs: "Script Output", "PL/SQL procedure successfully completed.", and "P_SGBD". The "P_SGBD" tab displays the results of the query, which include the dropping of two packages.

```
EXECUTE pach_aux.dropping;
DROP PACKAGE pach_aux;
DROP PACKAGE pachet_proiect_ex13_rs;

PL/SQL procedure successfully completed.

P_SGBD

Package PACH_AUX dropped.

Package PACHET_PROIECT_EX13_RS dropped.
```

14. Pachet complex

Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

Procedura "colectare_date" adună în tabelul de tip record, definit în cadrul pachetului informații coerente, memorând date despre fiecare cinematograf din baza de date, colectând id-ul și numele acestuia, numele unui film care rulează în acel cinematograf, numărul de rulări ce au avut loc în cinematograful actual pentru filmul actual, numărul de difuzări și numărul de bilete cumpărate. Numărul de difuzări și de bilete se vor afla cu ajutorul funcțiilor "nr_difuzări" și "nr_bilete".

Procedura "afisare" va afișa datele memorate în tabelul de tip record cu ajutorul procedurii "colectare_date". Afișarea se va face pe categorii, listându-se prima dată cazul în care au fost memorate date complete despre cinematograf, filme, rulări, difuzări și bilete (cel puțin o informație completă), sau cazul în care nu avem nicio dată completă memorată. De asemenea se va trata, cazul în care nu avem deloc date despre filme care să fi rulat în acel cinematograf, sau ne lipsesc date despre rulări, difuzări sau bilete (cel puțin una dintre ele are valoarea 0) pentru fiecare film care a rulat în acel cinematograf, iar atunci vom afișa conținutul tabelului imbricat ce retine numele cinematografelor care nu înscriează nici măcar o informație completă.

Procedura "modificare" mărește salariul tuturor angajaților care lucrează în unul din cinematografele care au vândut maximul de bilete vândute (calculat de noi din baza de date). Se va verifica dacă noul salar se încadrează în limitele salariilor pentru jobul pe care se află acel angajat, cu ajutorul unei funcții ("verif_sal"). Salarul se va mări cu numărul maxim de bilete vândute * 0.5. De asemenea, la final se va afișa numărul de linii afectate de comanda UPDATE.

```
CREATE OR REPLACE PACKAGE pachet_proiect_ex14_rs AS
```

```
--tip de date -> RECORD
```

```
TYPE rec_pak IS RECORD
```

```
(id_cinema cinematograf.cod_cinema%TYPE,  
n_cinema cinematograf.nume_cinema%TYPE,  
n_film film.nume_film%TYPE,  
cnt_rulat NUMBER,  
cnt_dif NUMBER,  
cnt_bilet NUMBER);
```

```
--tip de date -> tabel imbricat ce retine obiecte de tipul RECORD definit mai sus
```

```
TYPE tabel_auxiliar IS TABLE OF rec_pak;
```

```
info_compl tabel_auxiliar := tabel_auxiliar();
```

```
--tip de date -> tabel imbricat ce nume de cinematografe
```

```
TYPE tabel_auxiliar_2 IS TABLE OF cinematograf.nume_cinema%TYPE;
```

```
info_err tabel_auxiliar_2 := tabel_auxiliar_2();
```

```
--tip de date -> cursor explicit
```

```
CURSOR c_cinema RETURN cinematograf%ROWTYPE;
```

```
--tip de date -> cursor parametrizat
```

```
CURSOR c_film (p_cinema cinematograf.cod_cinema%TYPE) IS
```

```

(SELECT f.cod_film, nume_film, COUNT(*)
FROM film f, ruleaza r
WHERE f.cod_film = r.cod_film AND p_cinema = r.cod_cinema
GROUP BY f.cod_film, nume_film);

FUNCTION nr_bilete_cump(c_cin cinematograf.cod_cinema%TYPE, c_film
film.cod_film%TYPE)
RETURN NUMBER;

FUNCTION nr_difuzari(c_cin cinematograf.cod_cinema%TYPE, c_film
film.cod_film%TYPE)
RETURN NUMBER;

FUNCTION max_bilet RETURN NUMBER;

FUNCTION verif_sal(c_job job.cod_job%TYPE, new_sal NUMBER) RETURN NUMBER;

PROCEDURE colectare_date;

PROCEDURE afisare;

PROCEDURE modificare;
END packet_proiect_ex14_rs;
/

```



The screenshot shows the Oracle SQL Developer interface with the 'Query Builder' tab selected. The code area displays the PL/SQL package definition:

```

CREATE OR REPLACE PACKAGE packet_proiect_ex14_rs AS
    --tip de date -> RECORD
    TYPE rec_pak IS RECORD
        (id_cinema cinematograf.cod_cinema%TYPE,
         n_cinema cinematograf.nume_cinema%TYPE,
         n_film film.nume_film%TYPE,
         cnt_rulat NUMBER,
         cnt_dif NUMBER,
         cnt_bilet NUMBER);
    --tip de date -> tabel imbricat ce retine obiecte de tipul RECORD definit mai sus
    TYPE tabel_auxiliar IS TABLE OF rec_pak;
    info_compl tabel_auxiliar := tabel_auxiliar();
    --tip de date -> tabel imbricat ce nume de cinematografe
    TYPE tabel_auxiliar_2 IS TABLE OF cinematograf.nume_cinema%TYPE;
    info_err tabel_auxiliar_2 := tabel_auxiliar_2();
    --tip de date -> cursor explicit
    CURSOR c_cinema RETURN cinematograf%ROWTYPE;
    --tip de date -> cursor parametrizat
    CURSOR c_film (p_cinema cinematograf.cod_cinema%TYPE) IS
        (SELECT f.cod_film, nume_film, COUNT(*)
        FROM film f, ruleaza r
        WHERE f.cod_film = r.cod_film AND p_cinema = r.cod_cinema
        GROUP BY f.cod_film, nume_film);

```

```

FUNCTION nr_bilete_cump(c_cin cinematograf.cod_cinema%TYPE, c_film film.cod_film%TYPE)
RETURN NUMBER;

FUNCTION nr_difuzari(c_cin cinematograf.cod_cinema%TYPE, c_film film.cod_film%TYPE)
RETURN NUMBER;

FUNCTION max_bilet RETURN NUMBER;

FUNCTION verif_sal(c_job job.cod_job%TYPE, new_sal NUMBER) RETURN NUMBER;

PROCEDURE colectare_date;

PROCEDURE afisare;

PROCEDURE modificare;
END pachet_proiect_ex14_rs;
/

```

Script Output: Task completed in 0.279 seconds

Package PACHET_PROJECT_EX14_RS compiled

```

CREATE OR REPLACE PACKAGE BODY pachet_proiect_ex14_rs AS
  CURSOR c_cinema RETURN cinematograf%ROWTYPE IS
    SELECT *
      FROM cinematograf;

    FUNCTION nr_bilete_cump(c_cin cinematograf.cod_cinema%TYPE, c_film
                           film.cod_film%TYPE)
    RETURN NUMBER
    IS
      cnt NUMBER;
    BEGIN
      SELECT COUNT(*) INTO cnt
        FROM bilet_cumparat b, difuzare d
       WHERE b.cod_difuzare = d.cod_difuzare AND d.cod_cinema = c_cin AND d.cod_film =
             c_film;
      RETURN cnt;
    END nr_bilete_cump;

    FUNCTION nr_difuzari(c_cin cinematograf.cod_cinema%TYPE, c_film
                           film.cod_film%TYPE)
    RETURN NUMBER
    IS
      cnt NUMBER;
    BEGIN
      SELECT COUNT(*) INTO cnt
        FROM difuzare d
       WHERE d.cod_cinema = c_cin AND d.cod_film = c_film;
      RETURN cnt;
    END nr_difuzari;

    FUNCTION max_bilet
    RETURN NUMBER
  
```

```

IS
    maxim NUMBER := -1;
BEGIN
    IF info_compl.COUNT > 0 THEN
        FOR i IN info_compl.FIRST..info_compl.LAST LOOP
            IF maxim < info_compl(i).cnt_bilet THEN
                maxim := info_compl(i).cnt_bilet;
            END IF;
        END LOOP;
    END IF;
    RETURN maxim;
END max_bilet;

FUNCTION verif_sal(c_job job.cod_job%TYPE, new_sal NUMBER)
    RETURN NUMBER
IS
    sal_max NUMBER;
BEGIN
    SELECT salariu_max INTO sal_max
    FROM job
    WHERE cod_job = c_job;

    IF sal_max > new_sal THEN
        RETURN 1;
    ELSE
        RETURN -1;
    END IF;
END verif_sal;

PROCEDURE colectare_date
IS
    cod_f film.cod_film%TYPE;
    nume_f film.nume_film%TYPE;
    cnt_rul NUMBER;
    cnt_bil NUMBER;
    cnt_dif NUMBER;
    contor NUMBER;
BEGIN
    IF info_compl.COUNT = 0 AND info_err.COUNT = 0 THEN
        FOR i IN c_cinema LOOP
            OPEN c_film(i.cod_cinema);
            contor := 0;
            LOOP

```

```

        FETCH c_film INTO cod_f, nume_f, cnt_rul;
        EXIT WHEN c_film%NOTFOUND;
        cnt_dif := nr_difuzari(i.cod_cinema, cod_f);
        cnt_bil := nr_bilet_cump(i.cod_cinema, cod_f);
        IF cnt_rul != 0 AND cnt_dif != 0 AND cnt_bil != 0 THEN
            info_compl.EXTEND;
            info_compl(info_compl.COUNT).id_cinema := i.cod_cinema;
            info_compl(info_compl.COUNT).n_cinema := i.nume_cinema;
            info_compl(info_compl.COUNT).n_film := nume_f;
            info_compl(info_compl.COUNT).cnt_rulat := cnt_rul;
            info_compl(info_compl.COUNT).cnt_bilet := cnt_bil;
            info_compl(info_compl.COUNT).cnt_dif := cnt_dif;
        ELSE
            contor := contor + 1;
        END IF;
    END LOOP;
    IF c_film%ROWCOUNT = 0 OR c_film%ROWCOUNT = contor THEN
        info_err.EXTEND;
        info_err(info_err.COUNT) := i.nume_cinema;
    END IF;
    CLOSE c_film;
    END LOOP;
    END IF;
END colectare_date;

```

```

PROCEDURE afisare
IS
    v_prec cinematograf.cod_cinema%TYPE := -100;
BEGIN
    IF info_compl.COUNT = 0 AND info_err.COUNT = 0 THEN
        pachet_proiect_ex14_rs.colectare_date;
    END IF;
    IF info_compl.COUNT > 0 THEN
        dbms_output.put_line('-----DATE COMPLETE (FILM, RULARE, DIFUZARE,
BILETE)-----');
        FOR i IN info_compl.FIRST..info_compl.LAST LOOP
            IF v_prec != info_compl(i).id_cinema THEN
                dbms_output.new_line();
                dbms_output.put_line('-----');
                dbms_output.put_line(' ' || UPPER(info_compl(i).n_cinema));
                dbms_output.new_line();
                v_prec := info_compl(i).id_cinema;
            END IF;
        END LOOP;
    END IF;

```

```

        dbms_output.put_line(' [Filmul ' || UPPER(info_compl(i).n_film) || '] [Nr. Rulari ' ||
info_compl(i).cnt_rulat || '] [Nr. Difuzari '|| info_compl(i).cnt_dif || '] [Nr. Bilet ' ||
info_compl(i).cnt_bilet || ']');
    END LOOP;
    dbms_output.new_line();
    dbms_output.put_line('-----');
    dbms_output.new_line();
    dbms_output.new_line();
    dbms_output.new_line();
ELSE
    dbms_output.put_line('-----NU EXISTA DATE COMPLETE (FILM, RULARE,
DIFUZARE, BILETE)-----');
    dbms_output.put_line('-----');
    dbms_output.new_line();
    dbms_output.new_line();
END IF;
IF info_err.COUNT > 0 THEN
    dbms_output.put_line('-----DATE LIPSA-----');
);
    dbms_output.new_line();
    FOR i IN info_err.FIRST..info_err.LAST LOOP
        dbms_output.put_line('      ' || UPPER(info_err(i)));
    END LOOP;
    ELSE
        dbms_output.put_line('-----NU EXISTA DATE LIPSA-----');
    );
    dbms_output.new_line();
END IF;
END afisare;

```

```

PROCEDURE modificare
IS
    maxim NUMBER;
    v_prec cinematograf.cod_cinema%TYPE := -100;
    v_flag NUMBER := 0;
    ok_update NUMBER;
    contor NUMBER := 0;
BEGIN
    IF info_compl.COUNT = 0 AND info_err.COUNT = 0 THEN
        colectare_date;
    END IF;
    maxim := max_bilet;
    IF maxim != -1 THEN
        FOR i IN info_compl.FIRST..info_compl.LAST LOOP
            IF v_prec != info_compl(i).id_cinema THEN

```

```

    v_prec := info_compl(i).id_cinema;
    v_flag := 0;
END IF;
IF v_prec = info_compl(i).id_cinema AND v_flag = 0 AND info_compl(i).cnt_bilet =
maxim THEN
    FOR k IN (SELECT cod_angajat, cod_job, cod_cinema, salariu
              FROM angajat
              WHERE cod_cinema = info_compl(i).id_cinema) LOOP
        ok_update := verif_sal(k.cod_job, k.salariu + (maxim * 0.5));
        IF ok_update = 1 THEN
            UPDATE angajat
            SET salariu = k.salariu + (maxim * 0.5)
            WHERE cod_angajat = k.cod_angajat;
            contor := contor + 1;
        END IF;
    END LOOP;
    v_flag := 1;
END IF;
END LOOP;
END IF;
dbms_output.new_line();
dbms_output.new_line();
dbms_output.put_line('-----Linii updateate: ' || contor || ' -----');
END modificare;
END packet_proiect_ex14_rs;
/

```



The screenshot shows the Oracle SQL Developer interface with the 'Query Builder' tab selected. The code is displayed in the main pane:

```

CREATE OR REPLACE PACKAGE BODY packet_proiect_ex14_rs AS
  CURSOR c_cinema RETURN cinematograf%ROWTYPE IS
    SELECT *
      FROM cinematograf;

  FUNCTION nr_bilete_cump(c_cin cinematograf.cod_cinema%TYPE, c_film film.cod_film%TYPE)
    RETURN NUMBER
  IS
    cnt NUMBER;
  BEGIN
    SELECT COUNT(*) INTO cnt
      FROM bilet_cumparator b, difuzare d
     WHERE b.cod_difuzare = d.cod_difuzare AND d.cod_cinema = c_cin AND d.cod_film = c_film;
    RETURN cnt;
  END nr_bilete_cump;

  FUNCTION nr_difuzari(c_cin cinematograf.cod_cinema%TYPE, c_film film.cod_film%TYPE)
    RETURN NUMBER
  IS
    cnt NUMBER;
  BEGIN
    SELECT COUNT(*) INTO cnt
      FROM difuzare d
     WHERE d.cod_cinema = c_cin AND d.cod_film = c_film;
    RETURN cnt;
  END nr_difuzari;

```

```

END nr_difuzari;

FUNCTION max_bilet
    RETURN NUMBER
IS
    maxim NUMBER := -1;
BEGIN
    IF info_compl.COUNT > 0 THEN
        FOR i IN info_compl.FIRST..info_compl.LAST LOOP
            IF maxim < info_compl(i).cnt_bilet THEN
                maxim := info_compl(i).cnt_bilet;
            END IF;
        END LOOP;
    END IF;
    RETURN maxim;
END max_bilet;

FUNCTION verif_sal(c_job job.cod_job%TYPE, new_sal NUMBER)
    RETURN NUMBER
IS
    sal_max NUMBER;
BEGIN
    SELECT salariu_max INTO sal_max
    FROM job
    WHERE cod_job = c_job;
    IF sal_max > new_sal THEN
        RETURN 1;
    ELSE
        RETURN -1;
    END IF;
END verif_sal;

PROCEDURE colectare_date
IS
    cod_f film.cod_film%TYPE;
    nume_f film.nume_film%TYPE;
    cnt_rul NUMBER;
    cnt_bil NUMBER;
    cnt_dif NUMBER;
    contor NUMBER;
BEGIN
    IF info_compl.COUNT = 0 AND info_err.COUNT = 0 THEN
        FOR i IN c_cinema LOOP
            OPEN c_film(i.cod_cinema);
            contor := 0;
            LOOP
                FETCH c_film INTO cod_f, nume_f, cnt_rul;
                EXIT WHEN c_film%NOTFOUND;
                cnt_dif := nr_difuzari(i.cod_cinema, cod_f);
                cnt_bil := nr_bilete_cump(i.cod_cinema, cod_f);
                IF cnt_rul != 0 AND cnt_dif != 0 AND cnt_bil != 0 THEN
                    info_compl.EXTEND;
                    info_compl(info_compl.COUNT).id_cinema := i.cod_cinema;
                    info_compl(info_compl.COUNT).n_cinema := i.nume_cinema;
                    info_compl(info_compl.COUNT).n_film := nume_f;
                    info_compl(info_compl.COUNT).cnt_rulat := cnt_rul;
                    info_compl(info_compl.COUNT).cnt_bilet := cnt_bil;
                    info_compl(info_compl.COUNT).cnt_dif := cnt_dif;
                ELSE
                    contor := contor + 1;
                END IF;
            END LOOP;
            IF c_film%ROWCOUNT = 0 OR c_film%ROWCOUNT = contor THEN
                info_err.EXTEND;
                info_err(info_err.COUNT) := i.nume_cinema;
            END IF;
            CLOSE c_film;
        END LOOP;
    END IF;
END colectare_date;

PROCEDURE afisare
IS
    v_prec cinematograf.cod_cinema%TYPE := -100;
BEGIN
    IF info_compl.COUNT = 0 AND info_err.COUNT = 0 THEN
        pachet_proiect_ex14_rs.colectare_date;
    END IF;

```

```

        IF info_compl.COUNT > 0 THEN
            dbms_output.put_line('-----DATE COMPLETE (FILM, RULARE, DIFUZARE, BILETE)-----');
            FOR i IN info_compl.FIRST..info_compl.LAST LOOP
                IF v_prec != info_compl(i).id_cinema THEN
                    dbms_output.new_line();
                    dbms_output.put_line('-----' || UPPER(info_compl(i).n_cinema));
                    dbms_output.new_line();
                    v_prec := info_compl(i).id_cinema;
                END IF;
                dbms_output.put_line(' [Filmul ' || UPPER(info_compl(i).n_film) || '] [Nr. Rulari ' || info_compl(i).cnt_rulat || ');
            END LOOP;
            dbms_output.new_line();
            dbms_output.put_line('-----');
            dbms_output.new_line();
            dbms_output.new_line();
        ELSE
            dbms_output.put_line('-----NU EXISTA DATE COMPLETE (FILM, RULARE, DIFUZARE, BILETE)-----');
            dbms_output.put_line('-----');
            dbms_output.new_line();
            dbms_output.new_line();
        END IF;
        IF info_err.COUNT > 0 THEN
            dbms_output.put_line('-----DATE LIPSA-----');
            dbms_output.new_line();

```

```

            FOR i IN info_err.FIRST..info_err.LAST LOOP
                dbms_output.put_line('-----' || UPPER(info_err(i)));
            END LOOP;
            ELSE
                dbms_output.put_line('-----NU EXISTA DATE LIPSA-----');
                dbms_output.new_line();
            END IF;
        END afisare;

        PROCEDURE modificare
        IS
            maxim NUMBER;
            v_prec cinematograf.cod_cinema$TYPE := -100;
            v_flag NUMBER := 0;
            ok_update NUMBER;
            contor NUMBER := 0;
        BEGIN
            IF info_compl.COUNT = 0 AND info_err.COUNT = 0 THEN
                colectare_date;
            END IF;
            maxim := max_bilet;
            IF maxim != -1 THEN
                FOR i IN info_compl.FIRST..info_compl.LAST LOOP
                    IF v_prec != info_compl(i).id_cinema THEN
                        v_prec := info_compl(i).id_cinema;
                        v_flag := 0;
                    END IF;
                    IF v_prec = info_compl(i).id_cinema AND v_flag = 0 AND info_compl(i).cnt_bilet = maxim THEN
                        FOR k IN (SELECT cod_angajat, cod_job, cod_cinema, salariu
                                  FROM angajat
                                 WHERE cod_cinema = info_compl(i).id_cinema) LOOP
                            ok_update := verif_sal(k.cod_job, k.salariu + (maxim * 0.5));
                            IF ok_update = 1 THEN
                                UPDATE angajat
                                SET salariu = k.salariu + (maxim * 0.5)
                                WHERE cod_angajat = k.cod_angajat;
                                contor := contor + 1;
                            END IF;
                        END LOOP;
                        v_flag := 1;
                    END IF;
                END LOOP;
            END IF;
            dbms_output.new_line();
            dbms_output.new_line();
            dbms_output.put_line('-----Linii updateate: ' || contor || ' -----');
        END modificare;
    END pachet_proiect_ex14_rs;
/

```

Exemple de apelare a procedurilor și funcțiilor din interiorul pachetului:

```
EXECUTE pachet_proiect_ex14_rs.colectare_date;
```

```
EXECUTE pachet_proiect_ex14_rs.colectare_date;
PL/SQL procedure successfully completed.
```

Apel care arată faptul că putem apela procedura individual, însă nu va afișa nimic

```
EXECUTE pachet_proiect_ex14_rs.afisare;
```

```
EXECUTE pachet_proiect_ex14_rs.afisare;
PL/SQL procedure successfully completed.

-----DATE COMPLETE (FILM, RULARE, DIFUZARE, BILETE)-----

-----CINEMA CITY PLAKE
[Filmul UNCHARTED] [Nr. Rulari 1 ] [Nr. Difuzari 1 ] [Nr. Bilete 1]

-----CINEMA CITY SIBIU
[Filmul BLACK WIDOW] [Nr. Rulari 1 ] [Nr. Difuzari 1 ] [Nr. Bilete 1]

-----CINEMA CITY CLUJ
[Filmul IT CHAPTER 2] [Nr. Rulari 1 ] [Nr. Difuzari 1 ] [Nr. Bilete 1]

-----CINEMA CITY CONSTANTA
[Filmul CRIMSON PEAK] [Nr. Rulari 1 ] [Nr. Difuzari 1 ] [Nr. Bilete 2]

-----CINEMA CITY BAIA MARE
[Filmul BLACK WIDOW] [Nr. Rulari 1 ] [Nr. Difuzari 1 ] [Nr. Bilete 1]

-----CINEMA CITY API COTROCENI
[Filmul RED NOTICE] [Nr. Rulari 1 ] [Nr. Difuzari 1 ] [Nr. Bilete 1]

-----CINEMA CITY IASI
[Filmul CRIMSON PEAK] [Nr. Rulari 1 ] [Nr. Difuzari 1 ] [Nr. Bilete 1]

-----CINEMA CITY PITESTI
[Filmul DOCTOR STRANGE] [Nr. Rulari 1 ] [Nr. Difuzari 2 ] [Nr. Bilete 2]

-----DATE LIPSA
-----CINEMA CITY DROBETATS
```

Observăm faptul că procedura "colectare_date" este apelată și în interiorul procedurii de afișare, însă nu ne apar duplicate, deoarece colectarea va fi efectuată doar în cazul în care tabelul imbricat nu este deja populat.

EXECUTE pachet_proiect_ex14_rs.modificare;

```
EXECUTE pachet_proiect_ex14_rs.modificare;
PL/SQL procedure successfully completed.
```

Observăm faptul că procedura "colectare_date" este apelată și în interiorul procedurii de modificare, însă nu ne apar duplicate, deoarece colectarea va fi efectuată doar în cazul în care tabelul imbricat nu este deja populat.

SELECT * FROM angajat;

| COD_ANGAJAT | NUME | PRENUME | EMAIL | TELEFON | DATA_ANG | SALARIU | NR_ORE | ID_SUPERIOR | COD_CINEMA | COD_JOB |
|-------------|------|----------|-----------|-------------------------|--------------|-----------|--------|-------------|------------|-----------|
| 4 | 1051 | Matei | Andrei | andrei.matei@gmail.com | 0799.532.678 | 27-JAN-20 | 4672.5 | 8 | 1000 | 120 CAS |
| 5 | 1085 | Iana | Gabriela | gabriela.iana@gmail.com | 0712.623.881 | 12-APR-21 | 3972.5 | 8 | 1051 | 120 CAS |
| 6 | 1102 | Miron | Costin | costin.miron@gmail.com | 0771.229.345 | 19-NOV-19 | 5789.1 | 9 | (null) | 260 MAN |
| 7 | 1119 | Banu | Codrin | codrin.banu@gmail.com | 0741.721.921 | 19-NOV-19 | 1231.1 | 4 | 1102 | 260 ING_S |
| 8 | 1136 | Crivat | Gheorghe | gheo.crivat@gmail.com | (null) | 21-DEC-18 | 5772.6 | 7 | (null) | 180 MAN |
| 9 | 1153 | Stan | Elena | elena.stan@gmail.com | 0789.123.433 | 13-FEB-19 | 4129.7 | 10 | 1136 | 180 ING_S |
| 10 | 1170 | Pitco | Natanael | nate.pitco@gmail.com | (null) | 02-MAY-18 | 6191 | 6 | (null) | 200 MAN |
| 11 | 1187 | Florea | Ana | ana.florea@gmail.com | (null) | 09-JUL-20 | 5972.5 | 8 | (null) | 240 MAN |
| 12 | 1204 | Mistor | Viviana | viv.nistor@gmail.com | 0712.385.763 | 12-APR-22 | 3472.5 | 6 | 1187 | 240 ING_T |
| 13 | 1221 | Leona | Antonia | anto.leona@gmail.com | 0711.332.454 | 19-AUG-20 | 5789.1 | 8 | (null) | 260 MAN |
| 14 | 1238 | Nils | Corina | corina.nils@gmail.com | (null) | 01-MAR-19 | 6100.5 | 9 | (null) | 160 MAN |
| 15 | 1255 | Petre | Cosmin | cosmin.petre@gmail.com | 0718.223.344 | 26-JUN-21 | 5972.5 | 7 | 1255 | 160 ING_S |
| 16 | 1272 | Popa | Maria | maria.popa@yahoo.com | 0792.233.211 | 15-MAY-17 | 6132.1 | 8 | (null) | 220 MAN |
| 17 | 1289 | Dumitran | George | george.dumi@gmail.com | 0793.557.876 | 22-JUN-19 | 4432.1 | 4 | (null) | 160 ING_T |
| 18 | 1306 | Stanciu | Ecaterina | eca_stnc@gmail.com | 0732.323.233 | 12-DEC-21 | 3693.6 | 6 | 1170 | 200 ING_S |
| 19 | 1323 | Chesaru | Ana | ana.chesaru@yahoo.com | 0723.324.443 | 06-OCT-22 | 1922 | 8 | 1272 | 220 BAR |

Înainte de modificări

| COD_ANGAJAT | NUME | PRENUME | EMAIL | TELEFON | DATA_ANG | SALARIU | NR_ORE | ID_SUPERIOR | COD_CINEMA | COD_JOB |
|-------------|------|----------|-----------|-------------------------|--------------|-----------|--------|-------------|------------|-----------|
| 5 | 1085 | Iana | Gabriela | gabriela.iana@gmail.com | 0712.623.881 | 12-APR-21 | 3972.5 | 8 | 1051 | 120 CAS |
| 6 | 1102 | Miron | Costin | costin.miron@gmail.com | 0771.229.345 | 19-NOV-19 | 5790.1 | 9 | (null) | 260 MAN |
| 7 | 1119 | Banu | Codrin | codrin.banu@gmail.com | 0741.721.921 | 19-NOV-19 | 1232.1 | 4 | 1102 | 260 ING_S |
| 8 | 1136 | Crivat | Gheorghe | gheo.crivat@gmail.com | (null) | 21-DEC-18 | 5772.6 | 7 | (null) | 180 MAN |
| 9 | 1153 | Stan | Elena | elena.stan@gmail.com | 0789.123.433 | 13-FEB-19 | 4129.7 | 10 | 1136 | 180 ING_S |
| 10 | 1170 | Pitco | Natanael | nate.pitco@gmail.com | (null) | 02-MAY-18 | 6192 | 6 | (null) | 200 MAN |
| 11 | 1187 | Florea | Ana | ana.florea@gmail.com | (null) | 09-JUL-20 | 5972.5 | 8 | (null) | 240 MAN |
| 12 | 1204 | Mistor | Viviana | viv.nistor@gmail.com | 0712.385.763 | 12-APR-22 | 3472.5 | 6 | 1187 | 240 ING_T |
| 13 | 1221 | Leona | Antonia | anto.leona@gmail.com | 0711.332.454 | 19-AUG-20 | 5790.1 | 8 | (null) | 260 MAN |
| 14 | 1238 | Nils | Corina | corina.nils@gmail.com | (null) | 01-MAR-19 | 6100.5 | 9 | (null) | 160 MAN |
| 15 | 1255 | Petre | Cosmin | cosmin.petre@gmail.com | 0718.223.344 | 26-JUN-21 | 5972.5 | 7 | 1255 | 160 ING_S |
| 16 | 1272 | Popa | Maria | maria.popa@yahoo.com | 0792.233.211 | 15-MAY-17 | 6132.1 | 8 | (null) | 220 MAN |
| 17 | 1289 | Dumitran | George | george.dumi@gmail.com | 0793.557.876 | 22-JUN-19 | 4432.1 | 4 | (null) | 160 ING_T |
| 18 | 1306 | Stanciu | Ecaterina | eca_stnc@gmail.com | 0732.323.233 | 12-DEC-21 | 3693.6 | 6 | 1170 | 200 ING_S |
| 19 | 1323 | Chesaru | Ana | ana.chesaru@yahoo.com | 0723.324.443 | 06-OCT-22 | 1922 | 8 | 1272 | 220 BAR |
| 20 | 1340 | Toth | Raluca | raluca_toth@gmail.com | 0799.112.321 | 15-FEB-22 | 1666.5 | 4 | 1272 | 220 BAR |

După modificări

Ștergerea pachetului:

```
DROP PACKAGE pachet_proiect_ex14_rs;
```

Package PACHET_PROIECT_EX14_RS dropped.