Popescu Ioana-Livia, Grupa 252

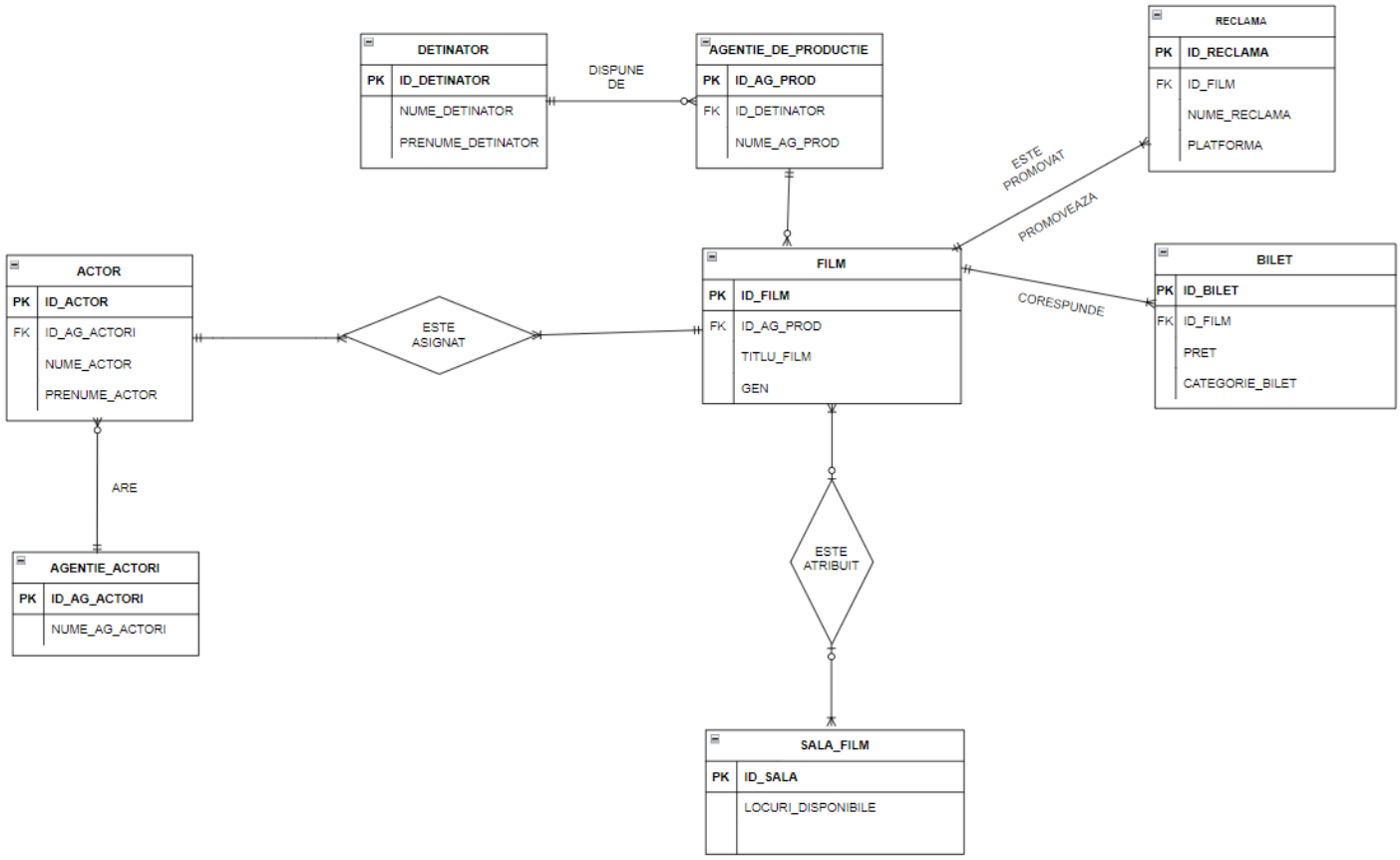
Sisteme de Gestionare a Bazelor de Date

Proiect: Gestionarea bazei de date a unui sediu cinema

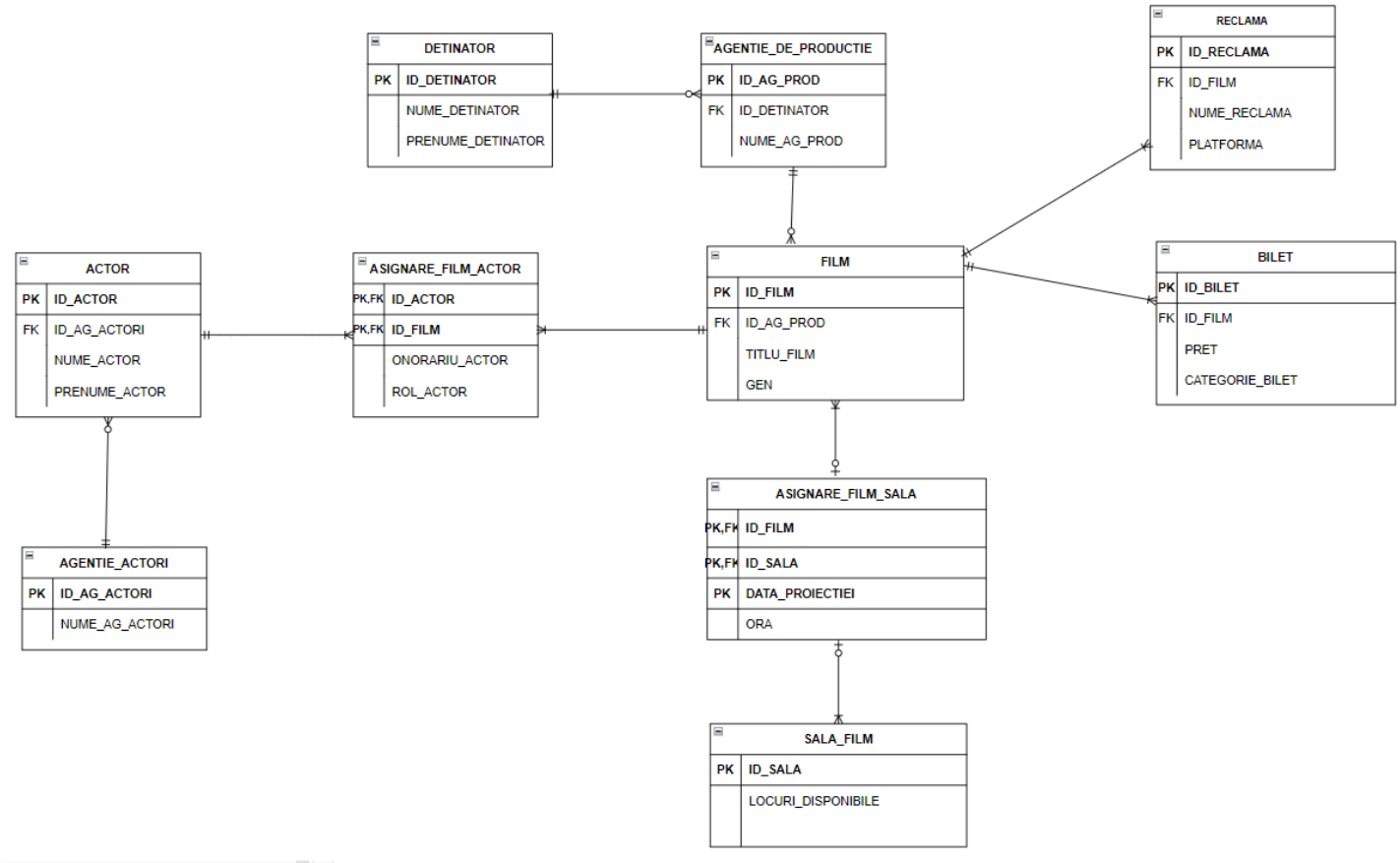
1. Prezentați pe scurt baza de date (utilitatea ei).

Baza de date isi propune sa faciliteze modul in care sunt stocate si gestionate informatiile unui sediu local de cinema. In acest sens, se va tine evidenta filmelor, agentiilor de productie asociate filmelor, detinatorii agentiilor, reclamele prin care este promovat fiecare film, biletele vandute, salile de cinema si asignarea acestora filmelor, asignarea actorilor din filme, respectiv a actorilor si agentiilor cu care colaboreaza acestia. De asemenea, prin intermediul pachetelor sunt puse la dispozitie diverse actiuni asupra bazei de date: adaugarea/stergerea unui film, adaugarea/stergerea unui bilet, obtinerea numarului de bilete vandute pentru un anumit film, afisarea unui liste a titlurilor si genurilor filmelor etc. In plus, se poate controla si accesul la baza de date si nu numai, prin intermediul unor declansatori.

2. Realizați diagrama entitate-relație (ERD).



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



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

```
CREATE TABLE DETINATOR(
   id_detinator number(4) not null,
   nume_detinator varchar2(50) not null,
   prenume_detinator varchar2(50) not null,
   constraint pk_Detinator primary key (id_detinator)
);
```

Welcome Page | bdproiectfinal | DETINATOR

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_DETINATOR	NUMBER(4,0)	No	(null)	1	(null)
2	NUME_DETINATOR	VARCHAR2(50 BYTE)	No	(null)	2	(null)
3	PRENUME_DETINATOR	VARCHAR2(50 BYTE)	No	(null)	3	(null)

```
CREATE TABLE AGENTIE_DE_PRODUCTIE(
   id_ag_prod number(4) not null,
   id_detinator number(4) not null,
   nume_ag_prod varchar2(50) not null,
   constraint pk_Ag_Productie primary key (id_ag_prod),
   constraint detinator_ag_fk foreign key (id_detinator) references DETINATOR(id_detinator)
);
```

Welcome Page | bdproiectfinal | AGENTIE_DE_PRODUCTIE

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_AG_PROD	NUMBER(4,0)	No	(null)	1	(null)
2	ID_DETINATOR	NUMBER(4,0)	No	(null)	2	(null)
3	NUME_AG_PROD	VARCHAR2(50 BYTE)	No	(null)	3	(null)

```
CREATE TABLE FILM(
   id_film number(4) not null,
   id_ag_prod number(4) not null,
   titlu_film varchar2(50) not null,
   gen varchar2(50) not null,
   constraint pk_Film primary key(id_film),
   constraint ag_film foreign key(id_ag_prod) references AGENTIE_DE_PRODUCTIE(id_ag_prod)
);
```

Welcome Page | bdproiectfinal | FILM

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_FILM	NUMBER(4,0)	No	(null)	1	(null)
2	ID_AG_PROD	NUMBER(4,0)	No	(null)	2	(null)
3	TITLU_FILM	VARCHAR2(50 BYTE)	No	(null)	3	(null)
4	GEN	VARCHAR2(50 BYTE)	No	(null)	4	(null)

```
CREATE TABLE BILET(
   id_bilet number(4) not null,
   id_film number(4) not null,
   pret number(4) not null,
   categorie_bilet varchar2(50) not null,
   constraint pk_Bilet primary key (id_bilet),
   constraint bilet_film_fk foreign key (id_film) references FILM(id_film)
);
```

Welcome Page | bdproiectfinal | BILET

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_BILET	NUMBER(4,0)	No	(null)	1	(null)
2	ID_FILM	NUMBER(4,0)	No	(null)	2	(null)
3	PRET	NUMBER(4,0)	No	(null)	3	(null)
4	CATEGORIE_BILET	VARCHAR2(50 BYTE)	No	(null)	4	(null)

```
CREATE TABLE SALA_FILM(
   id_sala number(4) not null,
   locuri_disponibile number(4),
   constraint pk_sala_film primary key (id_sala)
);
```

Welcome Page | bdproiectfinal | SALA_FILM

olumns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_SALA	NUMBER(4,0)	No	(null)	1	(null)
2	LOCURI_DISPONIBILE	NUMBER(4,0)	Yes	(null)	2	(null)

```
CREATE TABLE ASIGNARE_FILM_SALA(
   id_film number(4) not null,
```

```
    id_sala number(4) not null,
    data_proiectiei date not null,
    ora number(4) not null,
    constraint fk_as_sala foreign key (id_sala) references SALA_FILM(id_sala),
    constraint fk_as_film foreign key (id_film) references FILM(id_film),
    constraint pk_as_film_sala primary key (id_film, id_sala, data_proiectiei)
);
```

Welcome Page | bdproiectfinal | ASIGNARE_FILM_SALA

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_FILM	NUMBER(4,0)	No	(null)	1	(null)
2	ID_SALA	NUMBER(4,0)	No	(null)	2	(null)
3	DATA_PROIECTIEI	DATE	No	(null)	3	(null)
4	ORA	NUMBER(4,0)	No	(null)	4	(null)

```
CREATE TABLE AGENTIE_ACTORI(
    id_ag_actori number(4) not null,
    nume_ag_actori varchar2(50) not null,
    constraint pk_ag_actori primary key (id_ag_actori)
);
```

Welcome Page | bdproiectfinal | AGENTIE_ACTORI

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_AG_ACTORI	NUMBER(4,0)	No	(null)	1	(null)
2	NUME_AG_ACTORI	VARCHAR2(50 BYTE)	No	(null)	2	(null)

```
CREATE TABLE ACTOR(
    id_actor number(4) not null,
    id_ag_actori number(4) not null,
    nume_actor varchar2(50) not null,
    prenume_actor varchar2(50) not null,
    constraint pk_actor primary key(id_actor),
```

constraint fk_actor_ag foreign key (id_ag_actori) references AGENTIE_ACTORI(id_ag_actori)

);

Welcome Page | bdproiectfinal | ACTOR

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_ACTOR	NUMBER(4,0)	No	(null)	1	(null)
2	ID_AG_ACTORI	NUMBER(4,0)	No	(null)	2	(null)
3	NUME_ACTOR	VARCHAR2(50 BYTE)	No	(null)	3	(null)
4	PRENUME_ACTOR	VARCHAR2(50 BYTE)	No	(null)	4	(null)

CREATE TABLE ASIGNARE_FILM_ACTOR(

id_film number(4) not null,

id_actor number(4) not null,

onorariu_actor number(6) not null,

rol_actor varchar2(50) not null,

constraint fk_as_actor foreign key (id_actor) references ACTOR(id_actor),

constraint fk_as_ac_film foreign key (id_film) references FILM(id_film),

constraint pk_as_film_actor primary key (id_film, id_actor)

);

Welcome Page | bdproiectfinal | ASIGNARE_FILM_ACTOR

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_FILM	NUMBER(4,0)	No	(null)	1	(null)
2	ID_ACTOR	NUMBER(4,0)	No	(null)	2	(null)
3	ONORARIU_ACTOR	NUMBER(6,0)	No	(null)	3	(null)
4	ROL_ACTOR	VARCHAR2(50 BYTE)	No	(null)	4	(null)

CREATE TABLE RECLAMA(

id_reclama number(4) not null,

id_film number(4) not null,

nume_reclama varchar2(50) not null,

platforma varchar2(50) not null,

constraint pk_reclama primary key(id_reclama),

constraint fk_r_film foreign key (id_film) references FILM(id_film)

);

Welcome Page | bdproiectfinal | RECLAMA

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_RECLAMA	NUMBER(4,0)	No	(null)	1	(null)
2	ID_FILM	NUMBER(4,0)	No	(null)	2	(null)
3	NUME_RECLAMA	VARCHAR2(50 BYTE)	No	(null)	3	(null)
4	PLATFORMA	VARCHAR2(50 BYTE)	No	(null)	4	(null)

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

INSERT INTO DETINATOR

VALUES (1, 'Popescu' , 'Ion');

INSERT INTO DETINATOR

VALUES (2, 'Ionescu' , 'Ioana');

INSERT INTO DETINATOR

VALUES (3, 'Petrescu' , 'Ana');

INSERT INTO DETINATOR

VALUES (4, 'Georgescu' , 'Andrei');

INSERT INTO DETINATOR

VALUES (5, 'Popa' , 'Alex');

```
select * from detinator;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.11 seconds

	ID_DETINATOR	NUME_DETINATOR	PRENUME_DETINATOR
1	1	Popescu	Ion
2	2	Ionescu	Ioana
3	3	Petrescu	Ana
4	4	Georgescu	Andrei
5	5	Popa	Alex

INSERT INTO AGENTIE_DE_PRODUCTIE

VALUES (10, 1, 'Agentia 1');

INSERT INTO AGENTIE_DE_PRODUCTIE

VALUES (20, 2, 'Agentia 2');

INSERT INTO AGENTIE_DE_PRODUCTIE

VALUES (30, 3, 'Agentia 3');

INSERT INTO AGENTIE_DE_PRODUCTIE

VALUES (40, 4, 'Agentia 4');

INSERT INTO AGENTIE_DE_PRODUCTIE

VALUES (50, 5, 'Agentia 5');

```
select * from agentie_de_productie;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.09 seconds

	ID_AG_PROD	ID_DETINATOR	NUME_AG_PROD
1	10	1	Agentia 1
2	20	2	Agentia 2
3	30	3	Agentia 3
4	40	4	Agentia 4
5	50	5	Agentia 5

INSERT INTO FILM

VALUES (100, 10, 'Iron Man', 'Action');

INSERT INTO FILM

VALUES(110, 30, 'HoneyBear', 'Family');

INSERT INTO FILM

VALUES (120, 20, 'Spiderman', 'Action');

INSERT INTO FILM

VALUES (130, 30, 'Bridezilla', 'Drama');

INSERT INTO FILM

VALUES (140, 40, 'The Notebook', 'Romance');

INSERT INTO FILM

VALUES (150, 50, 'Scary', 'Horror');

SQL | All Rows Fetched: 6 in 0.002 seconds

	ID_FILM	ID_AG_PROD	TITLU_FILM	GEN
1	100	10	Iron Man	Action
2	120	20	Spiderman	Action
3	130	30	Bridezilla	Drama
4	140	40	The Notebook	Romance
5	150	50	Scary	Horror
6	110	30	HoneyBear	Family

INSERT INTO BILET

VALUES(11, 100, 20, 'VIP');

INSERT INTO BILET

VALUES(12, 120, 20, 'VIP');

INSERT INTO BILET

VALUES(13, 130, 10, 'NORMAL');

INSERT INTO BILET

VALUES(14, 140, 20, 'VIP');

INSERT INTO BILET

VALUES(15, 150, 10, 'NORMAL');

INSERT INTO BILET

VALUES(16, 100, 20, 'VIP');

INSERT INTO BILET

VALUES(17, 120, 20, 'VIP');

INSERT INTO BILET

VALUES(18, 130, 10, 'NORMAL');

INSERT INTO BILET

VALUES(19, 140, 20, 'VIP');

INSERT INTO BILET

VALUES(20, 150, 10, 'NORMAL');

```
select * from bilet;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 10 in 0.003 seconds

	ID_BILET	ID_FILM	PRET	CATEGORIE_BILET
1	11	100	20	VIP
2	12	120	20	VIP
3	13	130	10	NORMAL
4	14	140	20	VIP
5	15	150	10	NORMAL
6	16	100	20	VIP
7	17	120	20	VIP
8	18	130	10	NORMAL
9	19	140	20	VIP
10	20	150	10	NORMAL

INSERT INTO SALA_FILM

VALUES(91, 30);

INSERT INTO SALA_FILM

VALUES(92, 25);

INSERT INTO SALA_FILM

VALUES(93, 20);

INSERT INTO SALA_FILM

VALUES(94, 50);

INSERT INTO SALA_FILM

VALUES(95, 80);

ipt Output × | Query Result ×

SQL | All Rows Fetched: 5 in 0.003 seconds

	ID_SALA	LOCURI_DISPONIBILE
1	91	30
2	92	25
3	93	20
4	94	50
5	95	80

INSERT INTO AGENTIE_ACTORI

VALUES(71, 'Actori 1');

INSERT INTO AGENTIE_ACTORI

VALUES(72, 'Actori 2');

INSERT INTO AGENTIE_ACTORI

VALUES(73, 'Actori 3');

INSERT INTO AGENTIE_ACTORI

VALUES(74, 'Actori 4');

INSERT INTO AGENTIE_ACTORI

VALUES(75, 'Actori 5');

Script Output × | Query Result ×

SQL | All Rows Fetched: 5 in 0.112 sec

	ID_AG_ACTORI	NUME_AG_ACTORI
1	71	Actori 1
2	72	Actori 2
3	73	Actori 3
4	74	Actori 4
5	75	Actori 5

INSERT INTO ACTOR

VALUES(1000, 71, 'Ionescu', 'John');

INSERT INTO ACTOR

VALUES(1001, 72, 'Kent', 'Mariah');

INSERT INTO ACTOR

VALUES(1002, 73, 'Winston', 'Elle');

INSERT INTO ACTOR

VALUES (1003, 75, 'Lesco', 'Ana');

INSERT INTO ACTOR

VALUES(1004, 74, 'Jonas', 'Nick');

INSERT INTO ACTOR

VALUES(1005, 75, 'West', 'Kendrick');

All Rows Fetched: 6 in 0.094 seconds

	ID_ACTOR	ID_AG_ACTORI	NUME_ACTOR	PRENUME_ACTOR
1	1000	71	Ionescu	John
2	1001	72	Kent	Mariah
3	1002	73	Winston	Elle
4	1003	75	Lesco	Ana
5	1004	74	Jonas	Nick
6	1005	75	West	Kendrick

INSERT INTO ASIGNARE_FILM_ACTOR

VALUES(100, 1000, 67892, 'Principal');

INSERT INTO ASIGNARE_FILM_ACTOR

VALUES(120, 1001, 678, 'Figuratie');

INSERT INTO ASIGNARE_FILM_ACTOR

VALUES(130, 1002, 7000, 'Secundar');

INSERT INTO ASIGNARE_FILM_ACTOR

VALUES(140, 1003, 5689, 'Secundar');

INSERT INTO ASIGNARE_FILM_ACTOR

VALUES(150, 1004, 1890, 'Figuratie');

INSERT INTO ASIGNARE_FILM_ACTOR

VALUES(120, 1005, 8999, 'Principal');

INSERT INTO ASIGNARE_FILM_ACTOR

VALUES(140, 1002, 789, 'Figuratie');

INSERT INTO ASIGNARE_FILM_ACTOR

VALUES(100, 1001, 4321, 'Episodic');

INSERT INTO ASIGNARE_FILM_ACTOR

VALUES(100, 1003, 3456, 'Episodic');

INSERT INTO ASIGNARE_FILM_ACTOR

VALUES(100, 1005, 3456, 'Episodic');

	ID_FILM	ID_ACTOR	ONORARIU_ACTOR	ROL_ACTOR
1	100	1000	67892	Principal
2	120	1001	678	Figuratie
3	130	1002	7000	Secundar
4	140	1003	5689	Secundar
5	150	1004	1890	Figuratie
6	120	1005	8999	Principal
7	140	1002	789	Figuratie
8	100	1001	4321	Episodic
9	100	1003	3456	Episodic
10	100	1005	3456	Episodic

INSERT INTO ASIGNARE_FILM_SALA

```
VALUES(100, 91, '25-JUN-22', 16);

INSERT INTO ASIGNARE_FILM_SALA
VALUES(110, 95, '10-MAY-22', 9);

INSERT INTO ASIGNARE_FILM_SALA
VALUES(120, 93, '02-JUN-22', 11);

INSERT INTO ASIGNARE_FILM_SALA
VALUES(130, 91, '25-JUN-22', 22);

INSERT INTO ASIGNARE_FILM_SALA
VALUES(140, 93, '12-JAN-23', 10);

INSERT INTO ASIGNARE_FILM_SALA
VALUES(150, 94, '21-JUN-22', 8);

INSERT INTO ASIGNARE_FILM_SALA
VALUES(140, 92, '25-JUL-22', 20);

INSERT INTO ASIGNARE_FILM_SALA
VALUES(120, 91, '11-JUN-22', 11);

INSERT INTO ASIGNARE_FILM_SALA
VALUES(130, 91, '30-JAN-23', 9);

INSERT INTO ASIGNARE_FILM_SALA
VALUES(110, 92, '06-DEC-22', 8);
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 10 in 0.002 seconds

	ID_FILM	ID_SALA	DATA_PROIECTIEI	ORA
1	100	91	25-JUN-22	16
2	120	93	02-JUN-22	11
3	130	91	25-JUN-22	22
4	140	93	12-JAN-23	10
5	150	94	21-JUN-22	8
6	140	92	25-JUL-22	20
7	120	91	11-JUN-22	11
8	130	91	30-JAN-23	9
9	110	92	06-DEC-22	8
10	110	95	10-MAY-22	9

```
INSERT INTO RECLAMA
VALUES(900, 100, 'Coming soon..', 'Youtube');

INSERT INTO RECLAMA
VALUES(901, 100, 'Next week', 'Twitch');

INSERT INTO RECLAMA
VALUES(902, 100, 'Can't wait for', 'Instagram');

INSERT INTO RECLAMA
VALUES(903, 120, 'Are you ready for...', 'TV');

INSERT INTO RECLAMA
VALUES(904, 130, 'You can't miss this', 'TV');

INSERT INTO RECLAMA
VALUES(905, 140, 'Stay tuned', 'Twitter');

INSERT INTO RECLAMA
VALUES(906, 150, 'Coming next', 'Facebook');

INSERT INTO RECLAMA
```

VALUES(907, 110, 'Enjoy the next movie', 'Youtube');

SQL | All Rows Fetched: 5 in 0.002 seconds

	ID_RECLAMA	ID_FILM	NUME_RECLAMA	PLATFORMA
1	900	100	Coming soon..	Youtube
2	901	100	Next week	Twitch
3	905	140	Stay tuned	Twitter
4	906	150	Coming next	Facebook
5	907	110	Enjoy the next movie	Youtube

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

--Creati o procedura prin care pentru un film dat ca parametru afisez biletele care au fost cumparate si --reclamele prin care a fost promovat.

```
SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE ex6
(v_id_film film.id_film%TYPE)
IS
TYPE lista_id_bilet IS VARRAY(150) OF bilet.id_bilet%TYPE;
TYPE tablou_imbricat IS TABLE OF reclama.nume_reclama%TYPE;
t tablou_imbricat := tablou_imbricat();
v_lista lista_id_bilet := lista_id_bilet();
BEGIN
   select id_bilet
   bulk collect into v_lista
   from bilet
   where id_film = v_id_film;

   IF v_lista.COUNT() > 0 THEN
      DBMS_OUTPUT.PUT_LINE('Biletele inregistrate sunt: ');
      FOR i IN v_lista.FIRST..v_lista.LAST LOOP
```

```
            DBMS_OUTPUT.PUT(v_lista(i) || ' ');
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('');
     ELSE
        DBMS_OUTPUT.PUT_LINE('Nu sunt inregistrate bilete');
    END IF;

    select nume_reclama
    bulk collect into t
    from reclama
    where id_film = v_id_film;

    IF t.COUNT() > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Iar reclamele sunt: ');
        FOR i IN t.FIRST..t.LAST LOOP
            DBMS_OUTPUT.PUT(t(i) || ' ');
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Nu sunt inregistrate reclame');
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20001, 'Nu exista destule date inregistrate.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END ex6;
/
```

--apel

BEGIN

ex6(100);

END;

--rezultat

Biletele inregistrate sunt:

11 16

Iar reclamele sunt:

Coming soon.. Next week

```
Procedure EX6 compiled

Biletele inregistrate sunt:
11 16
Iar reclamele sunt:
Coming soon.. Next week

PL/SQL procedure successfully completed.
```

--verificare

select id_bilet

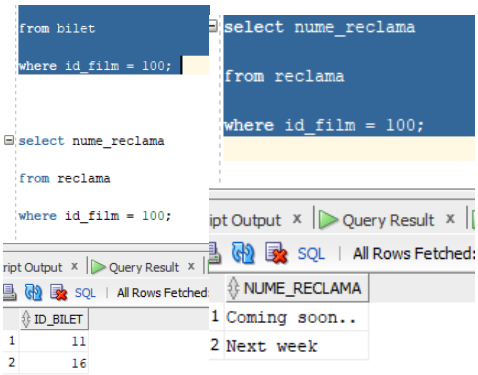
from bilet

where id_film = 100;

select nume_reclama

from reclama

where id_film = 100;



--apel

```
BEGIN
ex6(101);
END;
```

--rezultat

```
Procedure EX6 compiled

Nu sunt inregistrate bilete
Nu sunt inregistrate reclame


PL/SQL procedure successfully completed.
```

7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

--Obtineti pentru fiecare gen de film numarul de bilete inregistrate.

```
CREATE OR REPLACE PROCEDURE ex7
IS
CURSOR c IS
    SELECT gen genul, COUNT(id_bilet) nr_bilete
    FROM film f, bilet b
    WHERE f.id_film = b.id_film(+)
    GROUP BY gen;
v_gen_film film.gen%TYPE;
v_nr number(4);
BEGIN
   OPEN c;
```

```
   LOOP
      FETCH c INTO v_gen_film,v_nr;
      EXIT WHEN c%NOTFOUND;
      IF v_nr=0 THEN
         DBMS_OUTPUT.PUT_LINE('Pentru genul '|| v_gen_film ||
         ' nu exista bilete inregistrate.');
         ELSIF v_nr=1 THEN
         DBMS_OUTPUT.PUT_LINE('Pentru genul '|| v_gen_film ||
         ' exista un bilet inregistrat.');
         ELSE
         DBMS_OUTPUT.PUT_LINE('Pentru genul '|| v_gen_film ||
         ' exista '|| v_nr||' bilete inregistrate.');
      END IF;
   END LOOP;
   CLOSE c;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RAISE_APPLICATION_ERROR(-20001, 'Nu exista destule date inregistrate.');
   WHEN OTHERS THEN
      RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END ex7;
/
--apel
BEGIN
ex7();
END;
--rezultat
```

```
Procedure EX7 compiled

Pentru genul Family nu exista bilete inregistrate.
Pentru genul Drama exista 2 bilete inregistrate.
Pentru genul Romance exista 2 bilete inregistrate.
Pentru genul Horror exista 2 bilete inregistrate.
Pentru genul Action exista 4 bilete inregistrate.


PL/SQL procedure successfully completed.
```

--verificare

SELECT gen genul, COUNT(id_bilet) nr_bilete

FROM film f, bilet b

WHERE f.id_film = b.id_film(+)

GROUP BY gen;

SQL | All Rows Fetched: 5 in 0

	GENUL	NR_BILETE
1	Family	0
2	Drama	2
3	Romance	2
4	Horror	2
5	Action	4

8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

--Obtineti pentru id-ul unui film dat de la tastatura, id-ul agentiei de actori la care este personajul principal daca onorariul sau este mai mare de 1000.

CREATE OR REPLACE FUNCTION ex8

(v_id_film film.id_film%TYPE)

RETURN NUMBER IS

rezultat actor.id_ag_actori%TYPE;

BEGIN

SELECT aa.id_ag_actori

INTO rezultat

FROM asignare_film_actor a, film f, actor aa

```
WHERE f.id_film = 100 and f.id_film = a.id_film and a.id_actor = aa.id_actor and a.rol_actor = 'Principal'
and a.onorariu_actor > 1000;
RETURN rezultat;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20000,
'Nu exista, pentru filmul dat, actor care sa indeplineasca conditiile.');
WHEN TOO_MANY_ROWS THEN
RAISE_APPLICATION_ERROR(-20001,
'Exista mai multi actori care indeplinesc conditiile.');
WHEN OTHERS THEN
RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END ex8;
/
--apel
BEGIN
DBMS_OUTPUT.PUT_LINE('Id-ul agentiei de actori este '|| ex8(100));
END;
/
--rezultat
```

```
BEGIN
DBMS_OUTPUT.PUT_LINE('Id-ul agentiei de actori este '|| ex8(100));
END;
/
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result

Task completed in 0.07 seconds

```
ntru genul Family nu exista bilete inregistrate.
ntru genul Drama exista 2 bilete inregistrate.
ntru genul Romance exista 2 bilete inregistrate.
ntru genul Horror exista 2 bilete inregistrate.
ntru genul Action exista 4 bilete inregistrate.

/SQL procedure successfully completed.

nction EX8 compiled

l-ul agentiei de actori este 71

/SQL procedure successfully completed.
```

--verificare

SELECT aa.id_ag_actori

FROM asignare_film_actor a, film f, actor aa

WHERE f.id_film = 100 and f.id_film = a.id_film and a.id_actor = aa.id_actor and a.rol_actor = 'Principal'

and a.onorariu_actor > 1000;

```
SELECT aa.id_ag_actori
FROM asignare_film_actor a, film f, actor aa
WHERE f.id_film = 100 and f.id_film = a.id_film and a.id_actor = aa.id_actor and a.rol_actor = 'Principal'
and a.onorariu_actor > 1000;
```

ript Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | All Rows Fetched: 1 in 0.006 seconds

	ID_AG_ACTORI
1	71

9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat 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.

--Pentru un id dat al unui film, sa obtinem numele agentiei de actori la care se afla actorul cu rol principal si

--numele agentiei de productie a filmului.

```
CREATE OR REPLACE PROCEDURE ex9

(v_id_film film.id_film%TYPE)

IS

v_nume_agentie_actori agentie_actori.nume_ag_actori%TYPE;

v_nume_agentie_productie agentie_de_productie.nume_ag_prod%TYPE;

BEGIN

   SELECT aaa.nume_ag_actori nume_agentie, b.nume_ag_prod

   INTO v_nume_agentie_actori, v_nume_agentie_productie

   FROM film f, asignare_film_actor a, actor aa, agentie_actori aaa, agentie_de_productie b

   WHERE a.rol_actor = 'Principal' and b.id_ag_prod = f.id_ag_prod and a.id_film = f.id_film and
aa.id_actor = a.id_actor

   and aa.id_ag_actori = aaa.id_ag_actori and f.id_film = v_id_film;


   DBMS_OUTPUT.PUT_LINE('Agentia de productie se numeste ' || v_nume_agentie_productie || ' iar
cea de actori ' ||

   v_nume_agentie_actori || '.');


EXCEPTION

   WHEN NO_DATA_FOUND THEN

      RAISE_APPLICATION_ERROR(-20001, 'Nu exista destule date inregistrate.');

WHEN TOO_MANY_ROWS THEN

      RAISE_APPLICATION_ERROR(-20002, 'Sunt mai multe date inregistrate.');

   WHEN OTHERS THEN

      RAISE_APPLICATION_ERROR(-20003,'Alta eroare!');

END ex9;

/

--apel

BEGIN
```

ex9(100);

END;

--rezultat

```
END ex9;
/
--apel
BEGIN
ex9(100);
END;
```

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

Task completed in 0.079 seconds

```
PL/SQL procedure successfully completed.

Function EX8 compiled

Id-ul agentiei de actori este 71

PL/SQL procedure successfully completed.

Procedure EX9 compiled

Agentia de productie se numeste Agentia 1 iar cea de actori Actori 1.

PL/SQL procedure successfully completed.
```

--apel

BEGIN

ex9(110);

END;

--rezultat

```
BEGIN
ex9(110);
END;
```

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

Task completed in 0.193 seconds

```
gentia de productie se numeste Agentia 1 iar cea de actori Actori 1.

L/SQL procedure successfully completed.

rror starting at line : 477 in command -
EGIN
x9(110);
ND;
rror report -
RA-20001: Nu exista destule date inregistrate.
RA-06512: at "USER_1.EX9", line 35
RA-06512: at line 3
```

Intr-adevar, nu sunt inregistrari.

```
select *
from asignare_film_actor
where id_film = 110;
```

bdproiectfinal : select nume_r

ript Output | Query Result | Query Result 1 | Query Result 2 | Quer

SQL | All Rows Fetched: 0 in 0.004 seconds

ID_FILM	ID_ACTOR	ONORARI...	ROL_ACTOR

Aceeasi eroare se obtine si pentru apelul ex9(101), 101 nefiind inregistrat ca id film.

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

-- Definiti un trigger de tip LMD la nivel de comanda astfel incat tabelul film poate fi editat doar in timpul saptamanii,

-- nu si in weekend, respectiv doar in intervalul 5-23, intrucat in intervalul 24-4 se fac revizii tehnice ale sistemului.

```
CREATE OR REPLACE TRIGGER ex10
BEFORE INSERT OR UPDATE OR DELETE ON film
BEGIN
IF (TO_CHAR(SYSDATE,'D') = 1 OR TO_CHAR(SYSDATE,'D') = 7)
OR (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 5 AND 23)
THEN
```

RAISE_APPLICATION_ERROR(-20001,'ex10 : Tabelul nu poate fi actualizat');

END IF;

END;

/

--declansare

INSERT INTO FILM

VALUES(160, 40, 'Honey' , 'Drama');

--rezultat

```
Trigger EX10 compiled


Error starting at line : 501 in command -
INSERT INTO FILM

VALUES(160, 40, 'Honey' , 'Drama')
Error report -
ORA-20001: ex10 : Tabelul nu poate fi actualizat
ORA-06512: at "USER_1.EX10", line 9
ORA-04088: error during execution of trigger 'USER_1.EX10'
```

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

--Declansator care sa nu permita micsorarea pretului biletor.

CREATE OR REPLACE TRIGGER ex11

BEFORE UPDATE OF pret ON bilet

FOR EACH ROW

BEGIN

IF (:NEW.pret < :OLD.pret) THEN

RAISE_APPLICATION_ERROR(-20002,'Pretul biletului nu poate fi micsorat.');

END IF;

END;

/

--declansare

UPDATE bilet

SET pret = pret-10

where id_bilet = 11;

--rezultat

```
UPDATE bilet

SET pret = pret-10

where id_bilet = 11;
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query R

Task completed in 0.243 seconds

```
rigger EX10 dropped.


rigger EX11 compiled


rror starting at line : 524 in command -
PDATE bilet

ET pret = pret-10

here id_bilet = 11
rror report -
RA-20002: Pretul biletului nu poate fi micsorat.
RA-06512: at "USER_1.EX11", line 5
RA-04088: error during execution of trigger 'USER_1.EX11'
```

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

--Definiti un trigger de tip LDD care restrictioneaza operatiile alter/drop/create

--in afara intervalului 5-9 si in afara zilelor de lucru saptamanale.

CREATE OR REPLACE TRIGGER ex12

BEFORE ALTER OR CREATE OR DROP ON DATABASE

BEGIN

IF (TO_CHAR(SYSDATE,'D') = 1 OR TO_CHAR(SYSDATE, 'D') = 7)

THEN

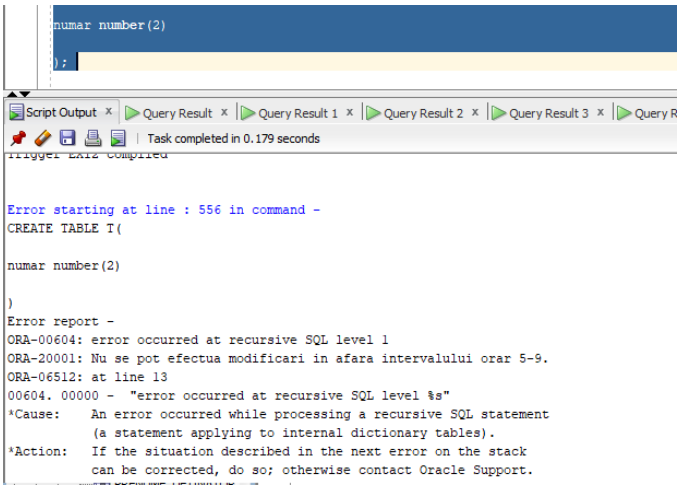
RAISE_APPLICATION_ERROR(-20001,'Nu se pot efectua modificari in weekend.');

ELSIF(TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 5 AND 9)

THEN

RAISE_APPLICATION_ERROR(-20001,'Nu se pot efectua modificari in afara intervalului orar 5-9.');

```
END IF;
END;
/
--declansare
CREATE TABLE T(
numar number(2)
);
--rezultat
```



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

```
CREATE OR REPLACE PACKAGE ex13
IS
   PROCEDURE ex6(
      v_id_film film.id_film%TYPE
   );
   PROCEDURE ex7;
   PROCEDURE ex9(
      v_id_film film.id_film%TYPE
   );
```

```
   FUNCTION ex8
   (v_id_film film.id_film%TYPE)
   RETURN NUMBER;
END;
/

CREATE OR REPLACE PACKAGE BODY ex13
IS
   PROCEDURE ex6
   (v_id_film film.id_film%TYPE)
   IS
   TYPE lista_id_bilet IS VARRAY(150) OF bilet.id_bilet%TYPE;
   TYPE tablou_imbricat IS TABLE OF reclama.nume_reclama%TYPE;
   t tablou_imbricat := tablou_imbricat();
   v_lista lista_id_bilet := lista_id_bilet();
   BEGIN
   select id_bilet
   bulk collect into v_lista
   from bilet
   where id_film = v_id_film;

   IF v_lista.COUNT() > 0 THEN
      DBMS_OUTPUT.PUT_LINE('Biletele inregistrate sunt: ');
      FOR i IN v_lista.FIRST..v_lista.LAST LOOP
         DBMS_OUTPUT.PUT(v_lista(i) || ' ');
      END LOOP;
      DBMS_OUTPUT.PUT_LINE('');
   ELSE
```

```
    DBMS_OUTPUT.PUT_LINE('Nu sunt inregistrate bilete');
  END IF;

  select nume_reclama
  bulk collect into t
  from reclama
  where id_film = v_id_film;

  IF t.COUNT() > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Iar reclamele sunt: ');
    FOR i IN t.FIRST..t.LAST LOOP
      DBMS_OUTPUT.PUT(t(i) || ' ');
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('');
 ELSE
    DBMS_OUTPUT.PUT_LINE('Nu sunt inregistrate reclame');
  END IF;

  EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20001, 'Nu exista destule date inregistrate.');
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
  END ex6;

  PROCEDURE ex7
  IS
  CURSOR c IS
     SELECT gen genul, COUNT(id_bilet) nr_bilete
```

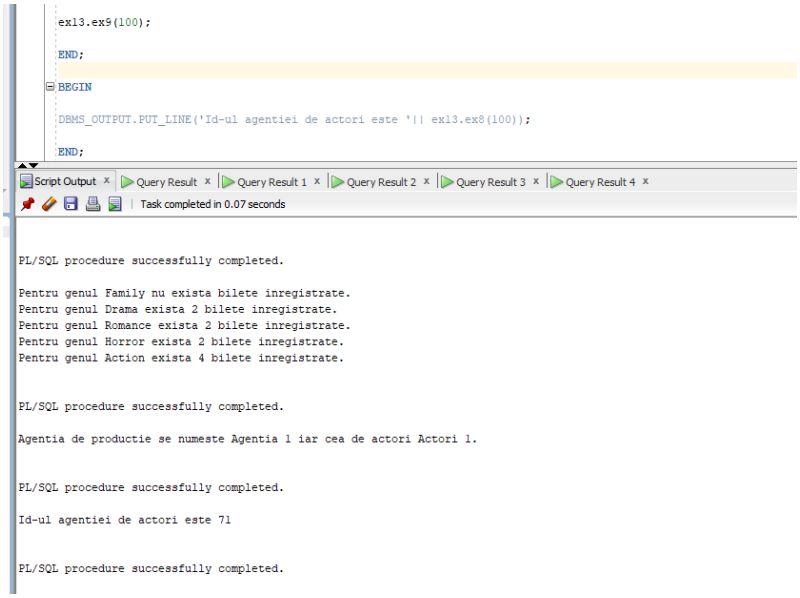
```
    FROM film f, bilet b
    WHERE f.id_film = b.id_film(+)
    GROUP BY gen;
v_gen_film film.gen%TYPE;
v_nr number(4);
BEGIN
   OPEN c;
   LOOP
      FETCH c INTO v_gen_film,v_nr;
      EXIT WHEN c%NOTFOUND;
      IF v_nr=0 THEN
         DBMS_OUTPUT.PUT_LINE('Pentru genul '|| v_gen_film ||
         ' nu exista bilete inregistrate.');
         ELSIF v_nr=1 THEN
         DBMS_OUTPUT.PUT_LINE('Pentru genul '|| v_gen_film ||
         ' exista un bilet inregistrat.');
         ELSE
         DBMS_OUTPUT.PUT_LINE('Pentru genul '|| v_gen_film ||
         ' exista '|| v_nr||' bilete inregistrate.');
      END IF;
   END LOOP;
   CLOSE c;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RAISE_APPLICATION_ERROR(-20001, 'Nu exista destule date inregistrate.');
   WHEN OTHERS THEN
      RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END ex7;
```

```
PROCEDURE ex9
(v_id_film film.id_film%TYPE)
IS
v_nume_agentie_actori agentie_actori.nume_ag_actori%TYPE;
v_nume_agentie_productie agentie_de_productie.nume_ag_prod%TYPE;
BEGIN
   SELECT aaa.nume_ag_actori nume_agentie, b.nume_ag_prod
   INTO v_nume_agentie_actori, v_nume_agentie_productie
   FROM film f, asignare_film_actor a, actor aa, agentie_actori aaa, agentie_de_productie b
   WHERE a.rol_actor = 'Principal' and b.id_ag_prod = f.id_ag_prod and a.id_film = f.id_film and
aa.id_actor = a.id_actor
   and aa.id_ag_actori = aaa.id_ag_actori and f.id_film = v_id_film;

   DBMS_OUTPUT.PUT_LINE('Agentia de productie se numeste ' || v_nume_agentie_productie || ' iar
cea de actori ' ||
   v_nume_agentie_actori || '.');

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RAISE_APPLICATION_ERROR(-20001, 'Nu exista destule date inregistrate.');
   WHEN OTHERS THEN
      RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END ex9;

FUNCTION ex8
(v_id_film film.id_film%TYPE)
RETURN NUMBER IS rezultat actor.id_ag_actori%TYPE;
BEGIN
```

```
    SELECT aa.id_ag_actori
    INTO rezultat
    FROM asignare_film_actor a, film f, actor aa
    WHERE f.id_film = 100 and f.id_film = a.id_film and a.id_actor = aa.id_actor and a.rol_actor = 'Principal'
    and a.onorariu_actor > 1000;
  RETURN rezultat;
  EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20000, 'Nu exista, pentru filmul dat, actor care sa indeplineasca conditiile.');
  WHEN TOO_MANY_ROWS THEN
    RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi actori care indeplinesc conditiile.');
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
  END ex8;


END ex13;
/


--rulare
--teste
BEGIN
ex13.ex6(100);
END;
```

```
Package EX13 compiled


Package Body EX13 compiled

Biletele inregistrate sunt:
11 16
Iar reclamele sunt:
Coming soon.. Next week
```

```
BEGIN
ex13.ex7();
END;

BEGIN
ex13.ex9(100);
END;

BEGIN
DBMS_OUTPUT.PUT_LINE('Id-ul agentiei de actori este '|| ex13.ex8(100));
END;
```



14. 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).

```
--secventa pentru functia adauga_film
CREATE SEQUENCE filme_seq
```

```
START WITH 180
INCREMENT BY 10
NOCACHE NOCYCLE;
/

CREATE OR REPLACE PACKAGE ex14 IS
   TYPE lista_ag_prod IS VARRAY(150) OF agentie_de_productie.nume_ag_prod%TYPE;

   TYPE detinator_record IS RECORD
   (nume_d detinator.nume_detinator%TYPE,
   prenume_d detinator.prenume_detinator%TYPE,
   lista_d lista_ag_prod);

   TYPE tablou_dati_sala IS TABLE OF asignare_film_sala.data_proiectiei%TYPE;

   TYPE sala_record IS RECORD
   (id_s sala_film.id_sala%TYPE,
    tablou tablou_dati_sala);


   FUNCTION gaseste_film(v_titlu_film film.titlu_film%TYPE)
   RETURN NUMBER;

   FUNCTION gaseste_actor(
      prenume actor.prenume_actor%TYPE,
      nume actor.nume_actor%TYPE
   ) RETURN actor.id_actor%TYPE;

   FUNCTION bilete_vandute(
```

```
   v_film film.id_film%TYPE
) RETURN NUMBER;

FUNCTION gaseste_id_ag_prod(
   nume_ag agentie_de_productie.nume_ag_prod%TYPE
) RETURN agentie_de_productie.id_ag_prod%TYPE;

CURSOR lista_toate_filmele
RETURN film%ROWTYPE
IS
SELECT * FROM film;

CURSOR lista_toate_salile
RETURN sala_film%ROWTYPE
IS
SELECT * FROM sala_film;

PROCEDURE afiseaza_filme;

PROCEDURE adauga_film(
   nume_ag_prod agentie_de_productie.nume_ag_prod%TYPE,
   titlu film.titlu_film%TYPE,
   genre film.gen%TYPE
);

PROCEDURE sterge_film(
   titlu film.titlu_film%TYPE
);
```

```
    PROCEDURE adauga_bilet(
        titlu film.titlu_film%TYPE,
        c bilet.categorie_bilet%TYPE
    );

    PROCEDURE sterge_bilet(
        id   bilet.id_bilet%TYPE
    );

    PROCEDURE afis_det_info(
        id detinator.id_detinator%TYPE
    );

    PROCEDURE afis_dati_sala(
        id sala_film.id_sala%TYPE
    );

END ex14;
/
CREATE OR REPLACE PACKAGE BODY ex14 IS
    FUNCTION gaseste_film(v_titlu_film film.titlu_film%TYPE)
    RETURN NUMBER IS rezultat film.id_film%TYPE;
    BEGIN
        SELECT id_film
        INTO rezultat
        FROM FILM
        WHERE lower(titlu_film) = lower(v_titlu_film);
    RETURN rezultat;
     EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu este inregistrat un film cu acest nume.');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Exista mai multe filme cu aceasta denumire!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END gaseste_film;

FUNCTION gaseste_actor(
    prenume actor.prenume_actor%TYPE,
    nume actor.nume_actor%TYPE
)
RETURN actor.id_actor%TYPE
IS
    v_id actor.id_actor%TYPE;
BEGIN
    SELECT id_actor INTO v_id
    FROM actor
    WHERE lower(nume_actor) = lower(nume) AND lower(prenume) = lower(prenume_actor);

    RETURN v_id;

EXCEPTION
    WHEN no_data_found THEN
        raise_application_error(-20000, 'Nu am gasit actorul  '
                || prenume || ' ' || nume);
END gaseste_actor;

FUNCTION gaseste_id_ag_prod(
```

```
   nume_ag agentie_de_productie.nume_ag_prod%TYPE
) RETURN agentie_de_productie.id_ag_prod%TYPE
IS
   v_id agentie_de_productie.id_ag_prod%TYPE;
BEGIN
   SELECT id_ag_prod INTO v_id
   FROM agentie_de_productie
   WHERE lower(nume_ag_prod) = lower(nume_ag);

   RETURN v_id;

EXCEPTION
   WHEN no_data_found THEN
      raise_application_error(-20000, 'Nu am gasit agentia cu numele ' || nume_ag || '.');
END gaseste_id_ag_prod;

FUNCTION bilete_vandute(
   v_film film.id_film%TYPE
) RETURN NUMBER
IS
   v_rezultat number(5);
BEGIN
   SELECT COUNT(*)
   INTO v_rezultat
   FROM BILET
   WHERE id_film = v_film;
RETURN v_rezultat;
EXCEPTION
   WHEN no_data_found THEN
```

```
        raise_application_error(-20000, 'Nu au fost vandute bilete pentru acest film.');
END bilete_vandute;

PROCEDURE afiseaza_filme
IS
BEGIN
    FOR filmm IN lista_toate_filmele LOOP
        DBMS_OUTPUT.PUT_LINE(filmm.titlu_film|| ' ; ' || filmm.gen);
    END LOOP;
END afiseaza_filme;

PROCEDURE adauga_film(
    nume_ag_prod agentie_de_productie.nume_ag_prod%TYPE,
    titlu film.titlu_film%TYPE,
    genre film.gen%TYPE
) IS
    v_id film.id_film%TYPE;
    v_id_ag film.id_ag_prod%TYPE;
BEGIN
     v_id := filme_seq.NEXTVAL;
     v_id_ag := gaseste_id_ag_prod(nume_ag_prod);

    INSERT INTO FILM (
        id_film,
        id_ag_prod,
        titlu_film,
        gen
    )
    VALUES(
```

```
        v_id,
        v_id_ag,
        titlu,
        genre
    );
END adauga_film;

PROCEDURE sterge_film(
    titlu film.titlu_film%TYPE
) IS
    v_id film.id_film%TYPE;
BEGIN
     v_id := gaseste_film(titlu);

     DELETE FROM FILM
     WHERE id_film = v_id;

     DELETE FROM BILET
     WHERE id_film = v_id;

     DELETE FROM ASIGNARE_FILM_ACTOR
     WHERE id_film = v_id;

     DELETE FROM ASIGNARE_FILM_SALA
     WHERE id_film = v_id;

EXCEPTION
 WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu este inregistrat un film cu acest nume.');
```

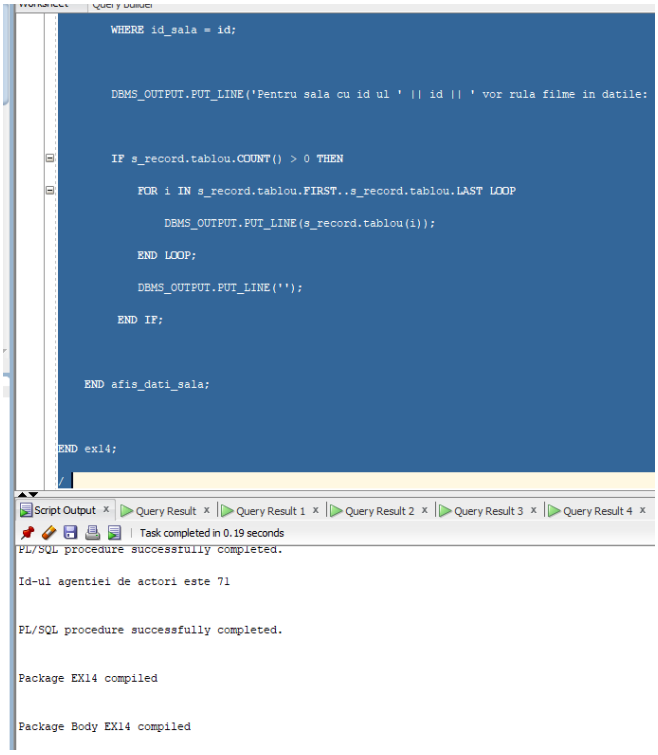
```
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Exista mai multe filme cu aceasta denumire!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

END sterge_film;

PROCEDURE adauga_bilet(
    titlu film.titlu_film%TYPE,
    c bilet.categorie_bilet%TYPE
) IS
    v_id_film film.id_film%TYPE;
    v_id_bilet bilet.id_bilet%TYPE;
    v_pret bilet.pret%TYPE;
    v_pret_vip bilet.pret%TYPE;
    v_pret_normal bilet.pret%TYPE;
BEGIN
    v_id_film := gaseste_film(titlu);

    SELECT MAX(id_bilet)
    INTO v_id_bilet
    FROM BILET;

    v_id_bilet := v_id_bilet + 1;

    SELECT MAX(pret)
    INTO v_pret_vip
    FROM bilet
    WHERE v_id_film = id_film;
```

```
    SELECT MIN(pret)
    INTO v_pret_normal
    FROM bilet
    WHERE v_id_film = id_film;

    IF c LIKE 'VIP' THEN
        v_pret := v_pret_vip;
    ELSE
        v_pret := v_pret_normal;
    END IF;

    INSERT INTO BILET (
        id_bilet,
        id_film,
        pret,
        categorie_bilet)
    VALUES(
        v_id_bilet,
        v_id_film,
        v_pret,
        c
    );
END adauga_bilet;

PROCEDURE sterge_bilet(
    id   bilet.id_bilet%TYPE
)IS
BEGIN
```

```
    DELETE FROM bilet
    WHERE id_bilet = id;
EXCEPTION
     WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu este inregistrat un bilet pentru acest id.');
END sterge_bilet;

PROCEDURE afis_det_info(
    id detinator.id_detinator%TYPE
)
IS
    d_record detinator_record;
BEGIN
    SELECT nume_ag_prod
    BULK COLLECT INTO d_record.lista_d
    FROM agentie_de_productie
    WHERE id_detinator = id;

    SELECT nume_detinator
    INTO d_record.nume_d
    FROM detinator
    WHERE id_detinator = id;

    SELECT prenume_detinator
    INTO d_record.prenume_d
    FROM detinator
    WHERE id_detinator = id;
```

```
      DBMS_OUTPUT.PUT_LINE('Pentru detinatorul ' || d_record.nume_d || ' ' ||  d_record.prenume_d
);
       DBMS_OUTPUT.PUT_LINE(' ');

      IF d_record.lista_d.COUNT() > 0 THEN
         DBMS_OUTPUT.PUT_LINE('Agentiile inregistrate sunt: ');

         FOR i IN d_record.lista_d.FIRST..d_record.lista_d.LAST LOOP
            DBMS_OUTPUT.PUT_LINE(d_record.lista_d(i));
         END LOOP;
      ELSE
          DBMS_OUTPUT.PUT_LINE('Nu sunt inregistrate agentii.');
      END IF;

   END afis_det_info;

   PROCEDURE afis_dati_sala(
      id sala_film.id_sala%TYPE
   )
   IS
      s_record sala_record;
   BEGIN
      SELECT data_proiectiei
      BULK COLLECT INTO s_record.tablou
      FROM asignare_film_sala
      WHERE id_sala = id;

      DBMS_OUTPUT.PUT_LINE('Pentru sala cu id ul ' || id || ' vor rula filme in datile: ');
```

```
      IF s_record.tablou.COUNT() > 0 THEN
         FOR i IN s_record.tablou.FIRST..s_record.tablou.LAST LOOP
            DBMS_OUTPUT.PUT_LINE(s_record.tablou(i));
         END LOOP;
         DBMS_OUTPUT.PUT_LINE('');
       END IF;

   END afis_dati_sala;

END ex14;
/
```




```
--teste
BEGIN
DBMS_OUTPUT.PUT_LINE('Id-ul filmului cautat este '|| ex14.gaseste_film('Iron Man'));
END;
/
```

```
BEGIN
DBMS_OUTPUT.PUT_LINE('Id-ul actorului cautat este ' || ex14.gaseste_actor('John' , 'Ionescu'));
END;
/

BEGIN
DBMS_OUTPUT.PUT_LINE('Pentru id-ul filmului oferit s-au vandut bilete in numar de ' || ex14.bilete_vandute(100) || '.');
END;
/
```

```
Package EX14 compiled


Package Body EX14 compiled

Id-ul filmului cautat este 100


PL/SQL procedure successfully completed.

Id-ul actorului cautat este 1000


PL/SQL procedure successfully completed.

Pentru id-ul filmului oferit s-au vandut bilete in numar de 2.


PL/SQL procedure successfully completed.
```

```
BEGIN
   ex14.afiseaza_filme;
END;
/
```

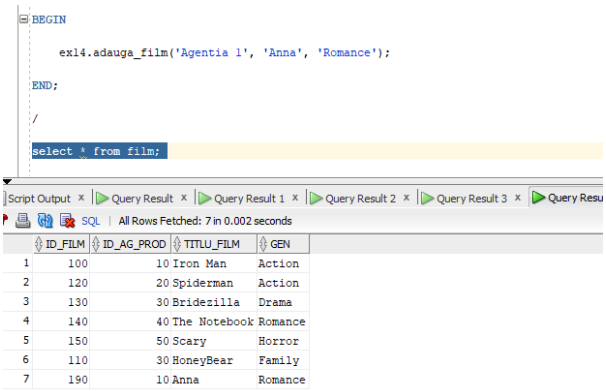
```
PL/SQL procedure successfully completed.

Iron Man ; Action
Spiderman ; Action
Bridezilla ; Drama
The Notebook ; Romance
Scary ; Horror
HoneyBear ; Family


PL/SQL procedure successfully completed.
```

```
--select * from film;
BEGIN
   ex14.adauga_film(170, 50, 'Anna', 'Romance');
```

END;

/

--select * from film;

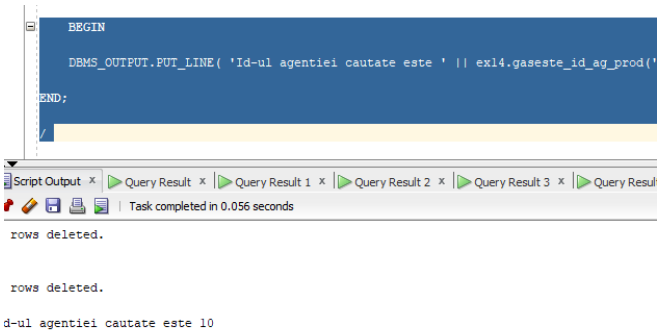


BEGIN

DBMS_OUTPUT.PUT_LINE('Id-ul agentiei cautate este ' || ex14.gaseste_id_ag_prod('Agentia 1'));

END;

/



--test adauga/sterge bilet

BEGIN

--select * from bilet;

ex14.adauga_bilet('Scary', 'VIP');

-- select * from bilet;

--ex14.sterge_bilet(21);

-- select * from bilet;

--rollback;

```
  -- select * from bilet;

END;

/
```

SQL | All Rows Fetched: 11 in 0.003 seconds

	ID_BILET	ID_FILM	PRET	CATEGORIE_BILET
1	11	100	20	VIP
2	12	120	20	VIP
3	13	130	10	NORMAL
4	14	140	20	VIP
5	15	150	10	NORMAL
6	16	100	20	VIP
7	17	120	20	VIP
8	18	130	10	NORMAL
9	19	140	20	VIP
10	20	150	10	NORMAL
11	21	150	10	VIP

A fost adaugat biletul cu id-ul 21 ex14.adauga_bilet('Scary', 'VIP'); .

```
BEGIN

--   select * from bilet;

--   ex14.adauga_bilet('Scary', 'VIP');

--   select * from bilet;
--
     ex14.sterge_bilet(21);
--
--   select * from bilet;
--
--   rollback;
--
--   select * from bilet;

END;

/
     select * from bilet;
```

Script Output × | Query Result × | Query Result 1 × | Query

SQL | All Rows Fetched: 10 in 0.003 seconds

	ID_BILET	ID_FILM	PRET	CATEGORIE_BILET
1	11	100	20	VIP
2	12	120	20	VIP
3	13	130	10	NORMAL
4	14	140	20	VIP
5	15	150	10	NORMAL
6	16	100	20	VIP
7	17	120	20	VIP
8	18	130	10	NORMAL
9	19	140	20	VIP
10	20	150	10	NORMAL

A fost sters biletul cu id-ul 21 prin apelul ex14.sterge_bilet(21); .

```
BEGIN

  ex14.afis_det_info(3);

END;

/
```

```
BEGIN
  ex14.afis_det_info(3);
END;
/
```

```
Script Output | Query Result | Query Result 1 | Query Result 2 | Query
Task completed in 0.086 seconds

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

Pentru detinatorul Petrescu Ana

Agentiile inregistrate sunt:
Agentia 3


PL/SQL procedure successfully completed.
```

BEGIN

  ex14.afis_dati_sala(91);

END;

/

```
BEGIN
  ex14.afis_dati_sala(91);
END;
/
```

```
Script Output | Query Result | Query Result 1 | Query Result 2
Task completed in 0.043 seconds

PL/SQL procedure successfully completed.

Pentru sala cu id ul 91 vor rula filme in datile:
25-JUN-22
25-JUN-22
11-JUN-22
30-JAN-23



PL/SQL procedure successfully completed.
```

In acest pachet sunt functii pentru gasirea unui film dupa titlu, a unui actor dupa nume si prenume, id-ului unei agentii de productie dupa nume, a numarului de bilete vandute pentru un film al carui id este trimis ca parametru, proceduri pentru afisarea titlurilor si genurilor filmelor, adaugarea unui film, stergerea unui film, adaugarea unui bilet, stergerea unui bilet, afisarea agentiilor pentru un detinator al carui id este dat ca parametru, afisarea datilor in care se ruleaza filme pentru o sala a carui id este dat ca parametru.