PROIECT SGBD

IANUARIE 2022

BOBOC ȘTEFAN
Grupa 241

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

Modelul de date realizat este aplicat pentru o platformă online care se ocupă de rezervare biletelor la teatru. Există spectacole care se desfășoară în orașe diferite, cu actori în roluri principale diferiți și cu echipe de producători diferite.

Baza de date memorează date despre clienți și ține evidența fiecărui client la ce spectacol merge. De asemenea se reține și fiecare spectacol ce echipă de producători are, la ce teatru se joacă și cine joacă în rol principal.

EXERCIȚIUL 2: Realizați diagrama entitate-relație (ERD).

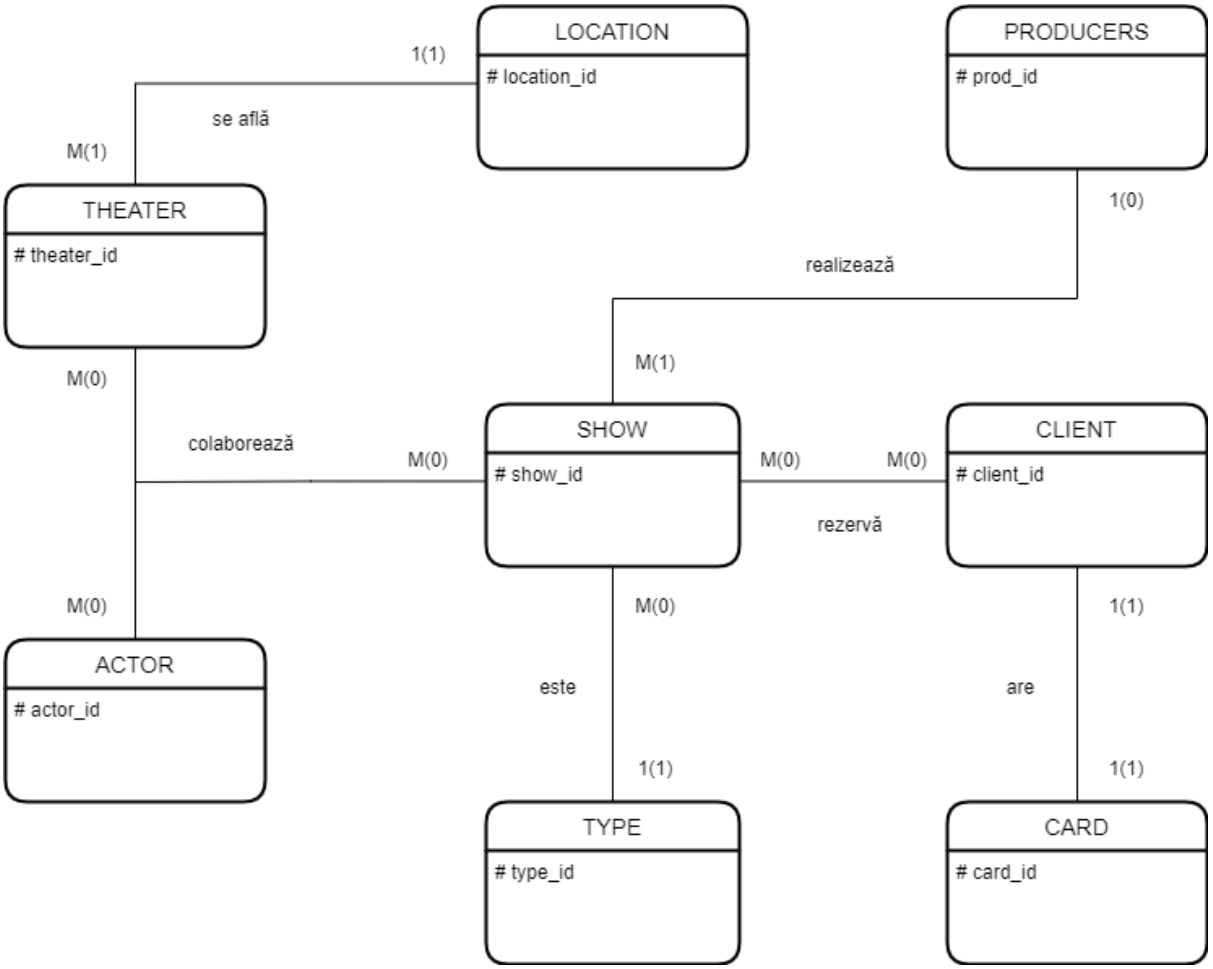


Diagrama Entitate-Relație

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

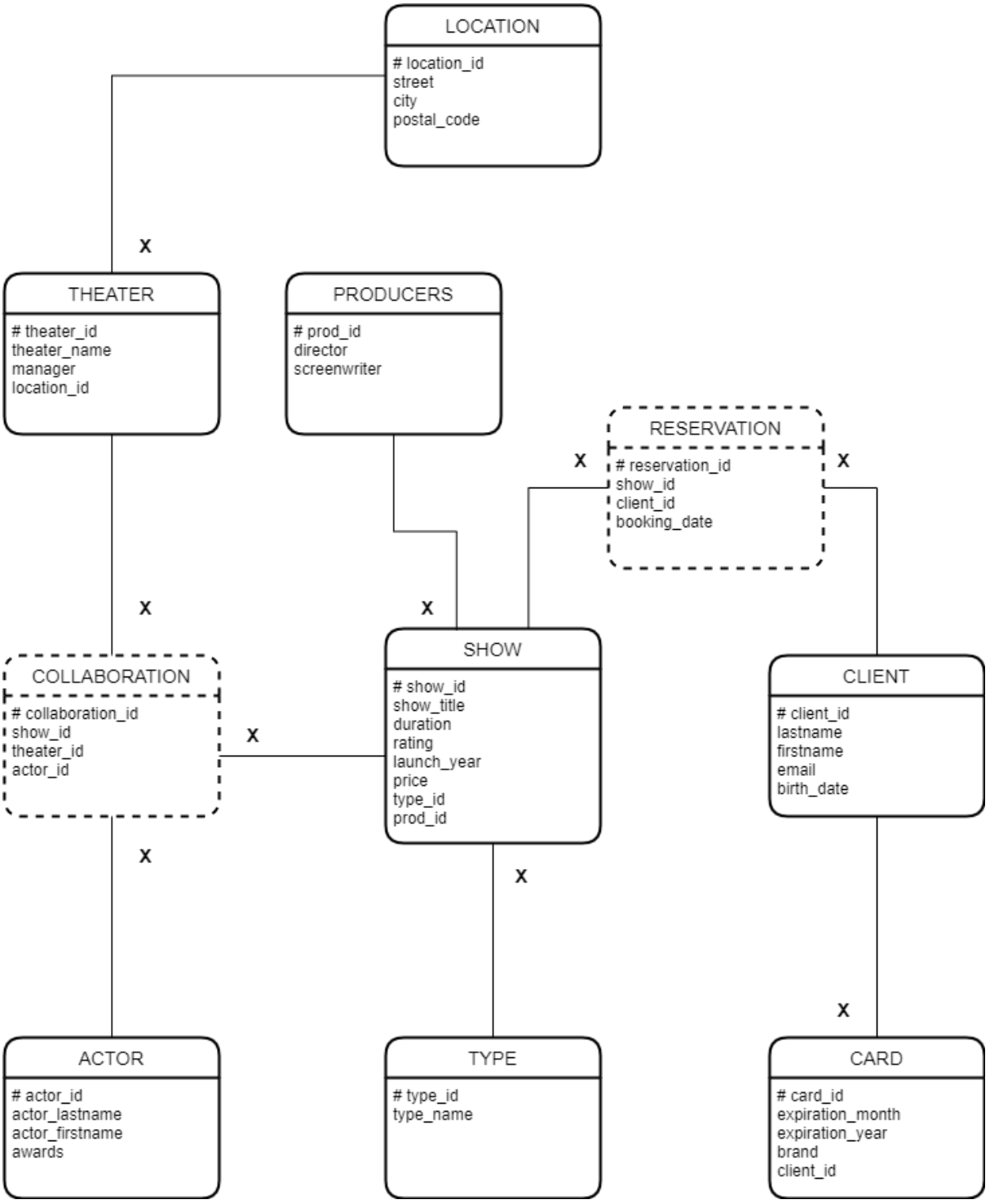


Diagrama Conceptuală

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

```
create table CLIENT(
    client_id number(6) primary key,
    lastname varchar2(30) not null,
    firstname varchar2(30) not null,
    email varchar2(25) not null,
    birth_date date,
    unique (email)
);

create table CARD(
    card_id number(6) primary key,
    expiration_month number(2) not null,
    expiration_year number(2) not null,
    brand varchar2(15) not null,
    client_id number(6) unique,
      constraint FK_client_card foreign key(client_id) references
CLIENT(client_id)
);

create table TYPE(
    type_id number(6) primary key,
    type_name varchar2(10) not null
);
```

Worksheet | Query Builder

```
insert into CLIENT values(70,'Mincu','Daria','mincu.d@yahoo.com',to_date('28-09-20

select * from client;

create table CARD(
    card_id number(6) primary key,
    expiration_month number(2) not null,
    expiration_year number(2) not null,
    brand varchar2(15) not null,
    client_id number(6) unique,
    constraint FK_client_card foreign key(client_id) references CLIENT(client_id)
);

insert into CARD values(1,'09','23','Visa',30);
insert into CARD values(2,'04','21','Mastercard',50);
insert into CARD values(3,'07','22','Mastercard',70);
insert into CARD values(4,'11','21','Citibank',60);
insert into CARD values(5,'04','23','Visa',40);
insert into CARD values(6,'01','24','AmericanExpress',20);

select * from card;

create table TYPE(
    type_id number(6) primary key,
```

Script Output | Query Result

Task completed in 0.033 seconds

```
Table CLIENT created.


Table CARD created.


Table TYPE created.
```

```
create table PRODUCERS(
    prod_id number(6) primary key,
    director varchar(25) not null,
    screenwriter varchar(25) not null
);

create table SHOW(
    show_id number(6) primary key,
    show_title varchar2(35) not null unique,
    duration number(1) not null,
    rating number(2,1),
    launch_year varchar2(4) not null,
    price number(5,1) not null,
    type_id number(6),
    prod_id number(6),
        constraint  FK_type_show  foreign  key(type_id)  references
TYPE(type_id),
     constraint FK_producers_show foreign key(prod_id) references
PRODUCERS(prod_id),
     constraint CONS_duration check (duration <= 5 and duration >=
1),
    constraint CONS_rating check (rating <= 5 and rating >= 1)
);

create table LOCATION(
    location_id number(6) primary key,
    street varchar2(20) not null,
    city varchar2(25) not null,
    postal_code number(6)
);
```

Worksheet | Query Builder | proiect

```
    type_id number(6),
    prod_id number(6),
    constraint FK_type_show foreign key(type_id) references TYPE(type_id),
    constraint FK_producers_show foreign key(prod_id) references PRODUCERS(prod_id
    constraint CONS_duration check (duration <= 5 and duration >= 1),
    constraint CONS_rating check (rating <= 5 and rating >= 1)
);

insert into SHOW values(10,'Divort in ziua nuntii', 2, 3, 2002, 50, 4, 3);
insert into SHOW values(11,'Ursul', 3, 3.5, 2010, 20, 2, 3);
insert into SHOW values(12,'Necasatoria', 2, 4, 2010, 15, 5, 2);
insert into SHOW values(13,'Cursa de soareci', 2, 2, 1999, 33, 6, 3);
insert into SHOW values(14,'O noapte furtunoasa', 3, 5, 1999, 45, 4, 1);
insert into SHOW values(15,'Doi pe o banca', 2, null, 2019, 27, 4, 4);
insert into SHOW values(16,'Colectionarul', 2, null, 2021, 16, 4, 2);
insert into SHOW values(17,'Meciul de comedie', 2, 4, 2011, 17, 2, 2);
insert into SHOW values(18,'Avarul', 2, 4, 2021, 28, 04, 04);
insert into SHOW values(19,'Salut', 2, 3, 2018, 20, 04, 02);
insert into SHOW values(20,'Inca una si ma duc', 2, 3, 2018, 20, 04, 02);

select * from show;

create table LOCATION(
    location_id number(6) primary key,
    street varchar2(20) not null,
```

Script Output | Query Result

Task completed in 0.036 seconds

```
Table PRODUCERS created.


Table SHOW created.


Table LOCATION created.
```

```
create table THEATER(
    theater_id number(6) primary key,
    theater_name varchar2(25) not null,
    manager varchar2(20),
    location_id number(6),
        constraint FK_location_theater foreign key(location_id)
references LOCATION(location_id)
);

create table ACTOR(
    actor_id number(6) primary key,
    actor_lastname varchar(20) not null,
    actor_firstname varchar(20) not null,
    awards number(2)
);
```

```
create table RESERVATION(
    reservation_id number(6) primary key,
    show_id number(6) not null,
    client_id number(6) not null,
    booking_date date not null,
        constraint FK_client_reservation foreign key(client_id)
references CLIENT(client_id),
     constraint FK_show_reservation foreign key(show_id) references
SHOW(show_id)
);
```

Worksheet | Query Builder | proiect

```
insert into THEATER values(50,'Teatrul Nottara','Nechifor Lipan',1);
insert into THEATER values(60,'Teatrul Guguta','Lica Samadau',2);
insert into THEATER values(70,'Teatrul Guguta','Vitoria Lipan',3);
insert into THEATER values(80,'Verde','Ghita Pristanda',3);

select * from theater;

create table ACTOR(
    actor_id number(6) primary key,
    actor_lastname varchar(20) not null,
    actor_firstname varchar(20) not null,
    awards number(2)
);

insert into ACTOR values(10,'Morgenstern','Maia',2);
insert into ACTOR values(20,'Malaele','Horatiu',3);
insert into ACTOR values(30,'Iures','Marcel',1);
insert into ACTOR values(40,'Alexandru','Alex', null);
insert into ACTOR values(50,'Arsinel','Alexandru',2);

select * from actor;

create table RESERVATION(
    reservation_id number(6) primary key,
    show_id number(6) not null,
```

Script Output | Query Result

Task completed in 0.036 seconds

```
Table THEATER created.

Table ACTOR created.

Table RESERVATION created.
```

```
create table COLLABORATION(
    collaboration_id number(6) primary key,
    show_id number(6) not null,
    theater_id number(6) not null,
    actor_id number(6) not null,
         constraint FK_show_collaboration foreign key(show_id)
references SHOW(show_id),
        constraint FK_actor_collaboration foreign key(actor_id)
references ACTOR(actor_id),
```

```
    constraint FK_theater_collaboration foreign key(theater_id)
references THEATER(theater_id)
);
```

```
select * from reservation;

create table COLLABORATION(
    collaboration_id number(6) primary key,
    show_id number(6) not null,
    theater_id number(6) not null,
    actor_id number(6) not null,
    constraint FK_show_collaboration foreign key(show_id) references SHOW(show_id)
    constraint FK_actor_collaboration foreign key(actor_id) references ACTOR(actor
    constraint FK_theater_collaboration foreign key(theater_id) references THEATER
);

insert into COLLABORATION values(101,10,10,20);
insert into COLLABORATION values(102,10,20,20);
insert into COLLABORATION values(103,10,40,20);
insert into COLLABORATION values(104,11,30,30);
insert into COLLABORATION values(105,12,40,30);
insert into COLLABORATION values(106,12,40,30);
insert into COLLABORATION values(107,12,60,40);
insert into COLLABORATION values(108,12,50,40);
insert into COLLABORATION values(109,16,10,40);
insert into COLLABORATION values(110,17,10,20);

select * from collaboration;
```

Script Output × | Query Result ×

Task completed in 0.039 seconds

```
Table ACTOR created.


Table RESERVATION created.


Table COLLABORATION created.
```

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

```
insert into CLIENT values(10,'Pop','Alex','popa@gmail.com',null);
insert into CLIENT values(20,'Ionescu','Ana','ionescu.a@gmail.com',to_date('18-02-1999','DD-MM-YYYY'));
insert into CLIENT values(30,'Gabor','Mara','gabor.m@gmail.com',to_date('06-12-1987','DD-MM-YYYY'));
insert into CLIENT values(40,'Suciu','Matei','suciu.m@gmail.com',to_date('18-05-2001','DD-MM-YYYY'));
insert into CLIENT values(50,'Avram','George','avram.g@gmail.com',null);
insert into CLIENT values(60,'Pop','Alex','popa.a@gmail.com',null);
insert into CLIENT values(70,'Mincu','Daria','mincu.d@yahoo.com',to_date('28-09-2000','DD-MM-YYYY'));
```

	CLIENT_ID	LASTNAME	FIRSTNAME	EMAIL	BIRTH_DATE
1	10	Pop	Alex	popa@gmail.com	(null)
2	20	Ionescu	Ana	ionescu.a@gmail.com	18-FEB-99
3	30	Gabor	Mara	gabor.m@gmail.com	06-DEC-87
4	40	Suciu	Matei	suciu.m@gmail.com	18-MAY-01
5	50	Avram	George	avram.g@gmail.com	(null)
6	60	Pop	Alex	popa.a@gmail.com	(null)
7	70	Mincu	Daria	mincu.d@yahoo.com	28-SEP-00

```
insert into CARD values(1,'09','23','Visa',30);
insert into CARD values(2,'04','21','Mastercard',50);
insert into CARD values(3,'07','22','Mastercard',70);
insert into CARD values(4,'11','21','Citibank',60);
insert into CARD values(5,'04','23','Visa',40);
insert into CARD values(6,'01','24','AmericanExpress',20);
```

```
select * from card;

create table TYPE(
```

Script Output | Query Result

SQL | All Rows Fetched: 6 in 0.004 seconds

	CARD_ID	EXPIRATION_MONTH	EXPIRATION_YEAR	BRAND	CLIENT_ID
1	1	9	23	Visa	30
2	2	4	21	Mastercard	50
3	3	7	22	Mastercard	70
4	4	11	21	Citibank	60
5	5	4	23	Visa	40
6	6	1	24	AmericanExpress	20

```
insert into TYPE values(1,'absurd');
insert into TYPE values(2,'grotesque');
insert into TYPE values(3,'satire');
insert into TYPE values(4,'comedy');
insert into TYPE values(5,'tragic');
insert into TYPE values(6,'drama');
```

```
insert into TYPE values(1,'absurd');
insert into TYPE values(2,'grotesque');
insert into TYPE values(3,'satire');
insert into TYPE values(4,'comedy');
insert into TYPE values(5,'tragic');
insert into TYPE values(6,'drama');

select * from type;

create table PRODUCERS(
    prod_id number(6) primary key,
```

Script Output | Query Result

SQL | All Rows Fetched: 6 in 0.003 seconds

	TYPE_ID	TYPE_NAME
1	1	absurd
2	2	grotesque
3	3	satire
4	4	comedy
5	5	tragic
6	6	drama

```
insert into PRODUCERS values(1,'Felix Alexa', 'Alex Ifrim');
insert into PRODUCERS values(2,'Sica Alexandrescu', 'Vlad Musteriu');
insert into PRODUCERS values(3,'Mara Barbu', 'Mara Barbu');
insert into PRODUCERS values(4,'Mihai Bendeac', 'Alexandra Matei');
insert into PRODUCERS values(5,'Sergiu Nicolaescu', 'Dan Nasta');
```

```
);

insert into PRODUCERS values(1,'Felix Alexa', 'Alex Ifrim');
insert into PRODUCERS values(2,'Sica Alexandrescu', 'Vlad Musteriu');
insert into PRODUCERS values(3,'Mara Barbu', 'Mara Barbu');
insert into PRODUCERS values(4,'Mihai Bendeac', 'Alexandra Matei');
insert into PRODUCERS values(5,'Sergiu Nicolaescu', 'Dan Nasta');

select * from producers;

create table SHOW(
    show_id number(6) primary key,
    show_title varchar2(35) not null unique,
```

cript Output × | Query Result ×

SQL | All Rows Fetched: 5 in 0.013 seconds

	PROD_ID	DIRECTOR	SCREENWRITER
1	1	Felix Alexa	Alex Ifrim
2	2	Sica Alexandrescu	Vlad Musteriu
3	3	Mara Barbu	Mara Barbu
4	4	Mihai Bendeac	Alexandra Matei
5	5	Sergiu Nicolaescu	Dan Nasta

```
insert into SHOW values(10,'Divort in ziua nuntii', 2, 3, 2002,
50, 4, 3);
insert into SHOW values(11,'Ursul', 3, 3.5, 2010, 20, 2, 3);
insert into SHOW values(12,'Necasatoria', 2, 4, 2010, 15, 5, 2);
insert into SHOW values(13,'Cursa de soareci', 2, 2, 1999, 33, 6,
3);
insert into SHOW values(14,'O noapte furtunoasa', 3, 5, 1999, 45,
4, 1);
insert into SHOW values(15,'Doi pe o banca', 2, null, 2019, 27, 4,
4);
insert into SHOW values(16,'Colectionarul', 2, null, 2021, 16, 4,
2);
insert into SHOW values(17,'Meciul de comedie', 2, 4, 2011, 17, 2,
2);
insert into SHOW values(18,'Avarul', 2, 4, 2021, 28, 04, 04);
insert into SHOW values(19,'Salut', 2, 3, 2018, 20, 04, 02);
insert into SHOW values(20,'Inca una si ma duc', 2, 3, 2018, 20,
04, 02);
```

```
insert into SHOW values(11,'Ursul', 3, 3.5, 2010, 20, 2, 3);
insert into SHOW values(12,'Necasatoria', 2, 4, 2010, 15, 5, 2);
insert into SHOW values(13,'Cursa de soareci', 2, 2, 1999, 33, 6, 3);
insert into SHOW values(14,'O noapte furtunoasa', 3, 5, 1999, 45, 4, 1);
insert into SHOW values(15,'Doi pe o banca', 2, null, 2019, 27, 4, 4);
insert into SHOW values(16,'Colectionarul', 2, null, 2021, 16, 4, 2);
insert into SHOW values(17,'Meciul de comedie', 2, 4, 2011, 17, 2, 2);
insert into SHOW values(18,'Avarul', 2, 4, 2021, 28, 04, 04);
insert into SHOW values(19,'Salut', 2, 3, 2018, 20, 04, 02);
insert into SHOW values(20,'Inca una si ma duc', 2, 3, 2018, 20, 04, 02);

select * from show;

create table LOCATION(
    location_id number(6) primary key,
    street varchar2(20) not null,
    city varchar2(25) not null,
```

Script Output | Query Result

SQL | All Rows Fetched: 11 in 0.005 seconds

	SHOW_ID	SHOW_TITLE	DURATION	RATING	LAUNCH_YEAR	PRICE	TYPE_ID	PROD_ID
1	10	Divort in ziua nuntii	2	3	2002	50	4	3
2	11	Ursul	3	3.5	2010	20	2	3
3	12	Necasatoria	2	4	2010	15	5	2
4	13	Cursa de soareci	2	2	1999	33	6	3
5	14	O noapte furtunoasa	3	5	1999	45	4	1
6	15	Doi pe o banca	2	(null)	2019	27	4	4
7	16	Colectionarul	2	(null)	2021	16	4	2
8	17	Meciul de comedie	2	4	2011	17	2	2
9	18	Avarul	2	4	2021	28	4	4
10	19	Salut	2	3	2018	20	4	2

```
insert into LOCATION values(1,'Lalelelor','Bucuresti',null);
insert into LOCATION values(2,'Florilor','Bucuresti',null);
insert into LOCATION values(3,'Ghioceilor','Bucuresti',null);
```

```
insert into LOCATION values(4,'Bujorilor','Cluj',null);
insert into LOCATION values(5,'Frunzelor','Brasov',null);
```

```
insert into LOCATION values(1,'Lalelelor','Bucuresti',null);
insert into LOCATION values(2,'Florilor','Bucuresti',null);
insert into LOCATION values(3,'Ghioceilor','Bucuresti',null);
insert into LOCATION values(4,'Bujorilor','Cluj',null);
insert into LOCATION values(5,'Frunzelor','Brasov',null);

select * from location;

create table THEATER(
```

Script Output × Query Result ×

SQL | All Rows Fetched: 5 in 0.006 seconds

	LOCATION_ID	STREET	CITY	POSTAL_CODE
1	1	Lalelelor	Bucuresti	(null)
2	2	Florilor	Bucuresti	(null)
3	3	Ghioceilor	Bucuresti	(null)
4	4	Bujorilor	Cluj	(null)
5	5	Frunzelor	Brasov	(null)

```
insert into THEATER values(10,'Teatrul Judetean Brasov','Zaharia Herdelea',5);
insert into THEATER values(20,'Teatrul Judetean Cluj','Vasile Baciu',4);
insert into THEATER values(30,'Teatrul de Comedie Brasov','Stefan Gheorghidiu',5);
insert into THEATER values(40,'Teatrul Artcub','Otilia Marculescu',3);
insert into THEATER values(50,'Teatrul Nottara','Nechifor Lipan',1);
insert into THEATER values(60,'Teatrul Guguta','Lica Samadau',2);
insert into THEATER values(70,'Teatrul Guguta','Vitoria Lipan',3);
```

```
insert   into   THEATER   values(80,'Verde','Ghita   Pristanda',3);
```

```
insert into THEATER values(10,'Teatrul Judetean Brasov','Zaharia Herdelea',
insert into THEATER values(20,'Teatrul Judetean Cluj','Vasile Baciu',4);
insert into THEATER values(30,'Teatrul de Comedie Brasov','Stefan Gheorghid
insert into THEATER values(40,'Teatrul Artcub','Otilia Marculescu',3);
insert into THEATER values(50,'Teatrul Nottara','Nechifor Lipan',1);
insert into THEATER values(60,'Teatrul Guguta','Lica Samadau',2);
insert into THEATER values(70,'Teatrul Guguta','Vitoria Lipan',3);
insert into THEATER values(80,'Verde','Ghita Pristanda',3);

select * from theater;

create table ACTOR(
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 8 in 0.006 seconds

	THEATER_ID	THEATER_NAME	MANAGER	LOCATION_ID
1	10	Teatrul Judetean Brasov	Zaharia Herdelea	5
2	20	Teatrul Judetean Cluj	Vasile Baciu	4
3	30	Teatrul de Comedie Brasov	Stefan Gheorghidiu	5
4	40	Teatrul Artcub	Otilia Marculescu	3
5	50	Teatrul Nottara	Nechifor Lipan	1
6	60	Teatrul Guguta	Lica Samadau	2
7	70	Teatrul Guguta	Vitoria Lipan	3
8	80	Verde	Ghita Pristanda	3

```
insert into ACTOR values(10,'Morgenstern','Maia',2);
insert into ACTOR values(20,'Malaele','Horatiu',3);
insert into ACTOR values(30,'Iures','Marcel',1);
insert into ACTOR values(40,'Alexandru','Alex', null);
insert into ACTOR values(50,'Arsinel','Alexandru',2);
```

```
);

insert into ACTOR values(10,'Morgenstern','Maia',2);
insert into ACTOR values(20,'Malaele','Horatiu',3);
insert into ACTOR values(30,'Iures','Marcel',1);
insert into ACTOR values(40,'Alexandru','Alex', null);
insert into ACTOR values(50,'Arsinel','Alexandru',2);

select * from actor;

create table RESERVATION(
    reservation_id number(6) primary key,
    show_id number(6) not null,
    client_id number(6) not null,
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.003 seconds

	ACTOR_ID	ACTOR_LASTNAME	ACTOR_FIRSTNAME	AWARDS
1	10	Morgenstern	Maia	2
2	20	Malaele	Horatiu	3
3	30	Iures	Marcel	1
4	40	Alexandru	Alex	(null)
5	50	Arsinel	Alexandru	2

```
insert into RESERVATION values(101, 10, 10,
to_date('04-11-2021','DD-MM-YYYY'));
insert into RESERVATION values(102, 11, 10,
to_date('13-10-2021','DD-MM-YYYY'));
insert into RESERVATION values(103, 12, 10,
to_date('25-03-2021','DD-MM-YYYY'));
insert into RESERVATION values(104, 10, 70,
to_date('04-12-2021','DD-MM-YYYY'));
insert into RESERVATION values(105, 11, 70,
to_date('28-08-2021','DD-MM-YYYY'));
insert into RESERVATION values(106, 12, 70,
to_date('11-01-2021','DD-MM-YYYY'));
insert into RESERVATION values(107, 13, 70,
to_date('14-08-2021','DD-MM-YYYY'));
insert into RESERVATION values(108, 17, 20,
to_date('07-12-2021','DD-MM-YYYY'));
insert into RESERVATION values(109, 17, 20,
to_date('12-08-2021','DD-MM-YYYY'));
insert into RESERVATION values(110, 10, 40,
to_date('23-10-2021','DD-MM-YYYY'));
insert into RESERVATION values(111, 13, 20,
to_date('14-08-2021','DD-MM-YYYY'));
insert into RESERVATION values(112, 10, 10,
to_date('01-06-2021','DD-MM-YYYY'));
insert into RESERVATION values(113, 10, 70,
to_date('14-10-2021','DD-MM-YYYY'));
insert into RESERVATION values(114, 10, 60,
to_date('30-10-2021','DD-MM-YYYY'));
insert into RESERVATION values(115, 10, 40,
to_date('15-06-2020','DD-MM-YYYY'));
insert into RESERVATION values(116, 10, 30,
to_date('05-06-2020','DD-MM-YYYY'));
```

```
insert into RESERVATION values(101, 10, 10, to_date('04-11-2021','DD-MM-YYYY'));
insert into RESERVATION values(102, 11, 10, to_date('13-10-2021','DD-MM-YYYY'));
insert into RESERVATION values(103, 12, 10, to_date('25-03-2021','DD-MM-YYYY'));
insert into RESERVATION values(104, 10, 70, to_date('04-12-2021','DD-MM-YYYY'));
insert into RESERVATION values(105, 11, 70, to_date('28-08-2021','DD-MM-YYYY'));
insert into RESERVATION values(106, 12, 70, to_date('11-01-2021','DD-MM-YYYY'));
insert into RESERVATION values(107, 13, 70, to_date('14-08-2021','DD-MM-YYYY'));
insert into RESERVATION values(108, 17, 20, to_date('07-12-2021','DD-MM-YYYY'));
insert into RESERVATION values(109, 17, 20, to_date('12-08-2021','DD-MM-YYYY'));
insert into RESERVATION values(110, 10, 40, to_date('23-10-2021','DD-MM-YYYY'));
insert into RESERVATION values(111, 13, 20, to_date('14-08-2021','DD-MM-YYYY'));
insert into RESERVATION values(112, 10, 10, to_date('01-06-2021','DD-MM-YYYY'));
insert into RESERVATION values(113, 10, 70, to_date('14-10-2021','DD-MM-YYYY'));
insert into RESERVATION values(114, 10, 60, to_date('30-10-2021','DD-MM-YYYY'));
insert into RESERVATION values(115, 10, 40, to_date('15-06-2020','DD-MM-YYYY'));
insert into RESERVATION values(116, 10, 30, to_date('05-06-2020','DD-MM-YYYY'));

select * from reservation;

create table COLLABORATION(
```

Script Output × Query Result ×

SQL | All Rows Fetched: 16 in 0.004 seconds

	RESERVATION_ID	SHOW_ID	CLIENT_ID	BOOKING_DATE
1	101	10	10	04-NOV-21
2	102	11	10	13-OCT-21
3	103	12	10	25-MAR-21
4	104	10	70	04-DEC-21
5	105	11	70	28-AUG-21
6	106	12	70	11-JAN-21
7	107	13	70	14-AUG-21
8	108	17	20	07-DEC-21
9	109	17	20	12-AUG-21
10	110	10	40	23-OCT-21

```
insert into COLLABORATION values(101,10,10,20);
insert into COLLABORATION values(102,10,20,20);
insert into COLLABORATION values(103,10,40,20);
insert into COLLABORATION values(104,11,30,30);
insert into COLLABORATION values(105,12,40,30);
insert into COLLABORATION values(106,12,40,30);
insert into COLLABORATION values(107,12,60,40);
insert into COLLABORATION values(108,12,50,40);
insert into COLLABORATION values(109,16,10,40);
insert into COLLABORATION values(110,17,10,20);
```

```sql
insert into COLLABORATION values(101,10,10,20);
insert into COLLABORATION values(102,10,20,20);
insert into COLLABORATION values(103,10,40,20);
insert into COLLABORATION values(104,11,30,30);
insert into COLLABORATION values(105,12,40,30);
insert into COLLABORATION values(106,12,40,30);
insert into COLLABORATION values(107,12,60,40);
insert into COLLABORATION values(108,12,50,40);
insert into COLLABORATION values(109,16,10,40);
insert into COLLABORATION values(110,17,10,20);

select * from collaboration;
```

Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.003 seconds

	COLLABORATION_ID	SHOW_ID	THEATER_ID	ACTOR_ID
1	101	10	10	20
2	102	10	20	20
3	103	10	40	20
4	104	11	30	30
5	105	12	40	30
6	106	12	40	30
7	107	12	60	40
8	108	12	50	40
9	109	16	10	40
10	110	17	10	20

EXERCIȚIUL 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.

Cerință

Realizati un subprogram care primește un parametru un număr
Parametrul poate avea valoarea 1 sau 2:
- pentru 1 va afișa numele fiecărui brand de carduri și de câte ori apare el în tabelă;
- pentru 2 va afișa numele fiecărui oraș și de câte ori apare el în tabelă;

Rezolvare

```
CREATE OR REPLACE PROCEDURE ex_6 (
    comanda NUMBER  --parametrul care trebuie sa primeasca valoari
de la 1 la 2
)
 IS

    --cursor dinamic
      --il vom folosi pentru a lua date ori din 'card' ori din
'location'
    --in functie de valoarea parametrului 'comanda'
    TYPE cursor_date IS REF CURSOR;
    c_date       cursor_date;

    TYPE t_indexat IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
    TYPE t_imbricat IS TABLE OF VARCHAR2(50);

       --tablou indexat pentru numarul de aparitii al datelor
(brand/city)
    v_aparitii   t_indexat;
    --tablou imbricat - salvam datele o singura data
    v_date       t_imbricat := t_imbricat();

     text            VARCHAR2(50);            --variabila va primi
valorile din cursor
     aux           VARCHAR2(50) := 'a';    --vom salva valorile din
'v_date' o singura data
    i            NUMBER := 0;           --contor pentru numarul de
valori din tabele

       parametru_invalid       EXCEPTION;        --exceptie pentru
parametru diferit de 1 si 2
BEGIN
    --pentru valoarea 1 cursorul contine brandurile
    IF comanda = 1 THEN
        OPEN c_date FOR SELECT brand
                        FROM card
                        ORDER BY brand;
```

```
    --pentru valoarea 2 cursorul contine orasele
    ELSIF comanda = 2 THEN
        OPEN c_date FOR SELECT city
                        FROM location
                        ORDER BY city;
    ELSE
        RAISE parametru_invalid;
    END IF;

    LOOP
        FETCH c_date INTO text;
          EXIT WHEN c_date%notfound; --parasim loop-ul cand nu mai
avem date in cursor

           IF aux != text THEN      --daca am mai avut respectiva
informatie (brand/city)
               aux := text;         --vom tine minte valoarea prin
care trecem

            i := i + 1;         --crestem contorul cu 1
            v_date.extend;      --marim tabloul imbricat
            v_date(i) := text;  --adaugam valoarea gasita

            v_aparitii(i) := 1; --valoarea gasita a aparut o data

          ELSIF aux = text THEN   --daca intalnim din noua aceeasi
valoare ii crestem nr de aparitii
            v_aparitii(i) := v_aparitii(i) + 1;
        END IF;
    END LOOP;

    CLOSE c_date;   --inchidem cursorul

    --afisam datele din cele doua tablouri
    FOR j IN 1..i LOOP
          dbms_output.put_line(v_date(j)  ||  '  apare  de  '  ||
v_aparitii(j) || ' ori');
    END LOOP;

EXCEPTION
    WHEN parametru_invalid THEN
               raise_application_error(-20001,  'Parametru  invalid.
Introduceti ca parametru "1" sau "2"');
END ex_6;
/
```

Apelare_1

```
/
BEGIN
    ex_6(1);
END;
/
```

Worksheet | Query Builder

```
            v_date(i) := text;  --adaugam valoarea gasita

            v_aparitii(i) := 1; --valoarea gasita a aparut o data

        ELSIF aux = text THEN   --daca intalnim din noua aceeasi valoare ii crestem nr de aparitii
            v_aparitii(i) := v_aparitii(i) + 1;
        END IF;
    END LOOP;

    CLOSE c_date;   --inchidem cursorul

    --afisam datele din cele doua tablouri
    FOR j IN 1..i LOOP
    dbms_output.put_line(v_date(j) || ' apare de ' || v_aparitii(j) || ' ori');
    END LOOP;

EXCEPTION
    WHEN parametru_invalid THEN
        raise_application_error(-20001, 'Parametru invalid. Introduceti ca parametru "1" sau "2"');
END ex_6;
/

--APELARE
/
BEGIN
    ex_6(1);
--     ex_6(2);
--     ex_6(3);
END;
/
```

Script Output | Task completed in 0.029 seconds

```
PL/SQL procedure successfully completed.
```

proiect

```
AmericanExpress apare de 1 ori
Citibank apare de 1 ori
Mastercard apare de 2 ori
Visa apare de 2 ori
```

Apelare_2

```
/
BEGIN
    ex_6(2);
END;
/
```

Worksheet | Query Builder

```
            v_date(i) := text;  --adaugam valoarea gasita

            v_aparitii(i) := 1; --valoarea gasita a aparut o data

        ELSIF aux = text THEN   --daca intalnim din noua aceeasi valoare ii crestem nr de aparitii
            v_aparitii(i) := v_aparitii(i) + 1;
        END IF;
    END LOOP;

    CLOSE c_date;   --inchidem cursorul

    --afisam datele din cele doua tablouri
    FOR j IN 1..i LOOP
    dbms_output.put_line(v_date(j) || ' apare de ' || v_aparitii(j) || ' ori');
    END LOOP;

EXCEPTION
    WHEN parametru_invalid THEN
        raise_application_error(-20001, 'Parametru invalid. Introduceti ca parametru "1" sau "2"');
END ex_6;
/

--APELARE
/
BEGIN
--      ex_6(1);
    ex_6(2);
--      ex_6(3);
END;
/
```

Script Output | Task completed in 0.027 seconds

```
PL/SQL procedure successfully completed.
```

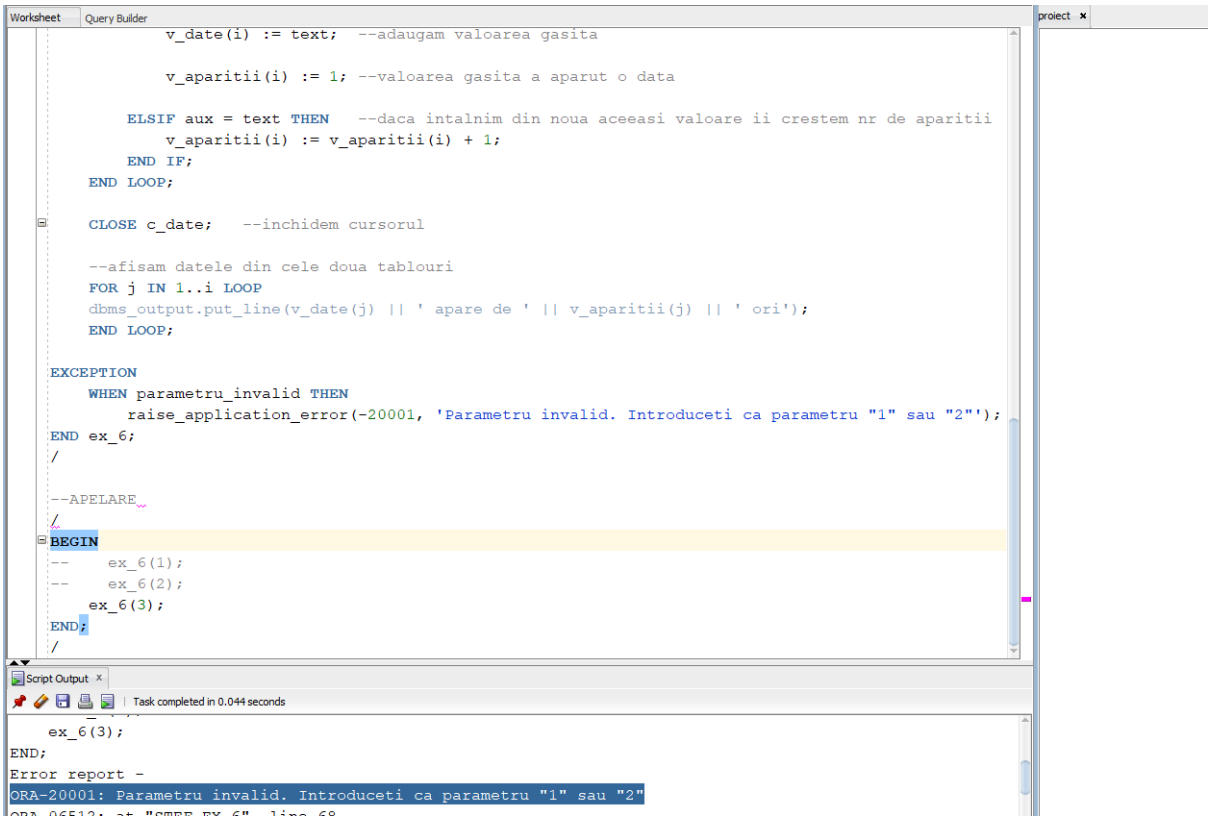
proiect

```
Brasov apare de 1 ori
Bucuresti apare de 3 ori
Cluj apare de 1 ori
```

Apelare_3

```
/
BEGIN
    ex_6(3);
END;
/
```

Pentru apelarea subprogramului ex_6 cu un parametru diferit de 1 si 2 va apărea eroarea:
`ORA-20001: Parametru invalid. Introduceti ca parametru “1” sau “2”`



EXERCIȚIUL 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.

Cerință

Afișarea spectacolelor în ordine descrescătoare al încasărilor lor realizate până în acest moment. Acestea vor fi organizate pe genurile de spectacol existente. La finalul fiecărui top se va afișa numărul total de încasări.

Rezolvare

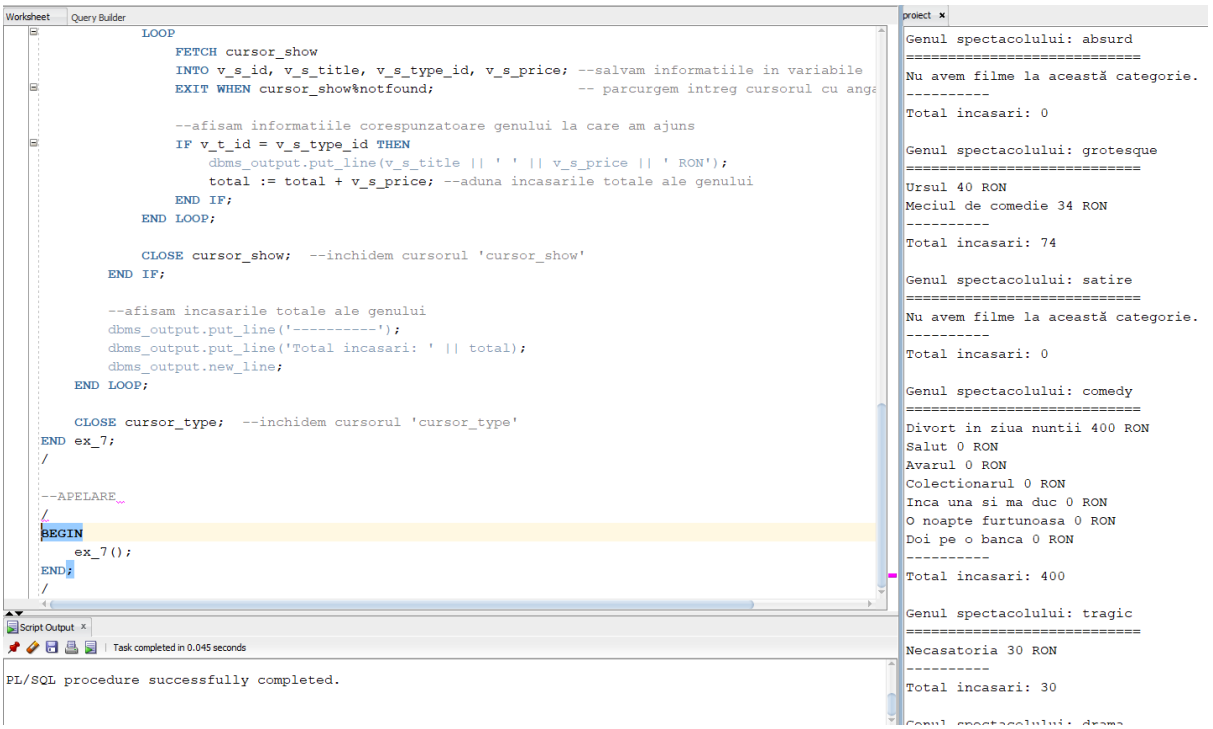
```
CREATE OR REPLACE PROCEDURE ex_7 IS
    --cursor ce contine cate spectacole fac parte dintr-un gen
    CURSOR cursor_type IS(
        SELECT t.type_id, MAX(t.type_name), COUNT(s.show_id)
        FROM type t, show s
        WHERE t.type_id = s.type_id (+)
        GROUP BY t.type_id
    );

      --cursor cu numele spectacolelor, genul lor si pretul unui
bilet
    CURSOR cursor_show IS
     SELECT ss.show_id, ss.show_title, ss.type_id, ss.price * m.n
total_price --calcularea incasarilor per spectacol
    FROM show ss,
              --calculam de cate ori s-a rezervat bilet pt fiecare
spectacol
        (SELECT s.show_id idul, (SELECT COUNT(r.show_id)
                                 FROM reservation r
                                 WHERE r.show_id = s.show_id) n
        FROM show s) m
    WHERE ss.show_id = m.idul
    ORDER BY total_price DESC;  --ordonam dupa incasari

    --variabile pentru datele din cursorul 'cursor_type'
    v_t_id        type.type_id%TYPE;
    v_t_name      type.type_name%TYPE;
    v_t_nr        show.show_id%TYPE;

    --variabile pentru datele din cursorul 'cursor_show'
    v_s_id        show.show_id%TYPE;
    v_s_title     show.show_title%TYPE;
    v_s_type_id   show.type_id%TYPE;
    v_s_price     NUMBER;
      total              NUMBER;      --variabila pentru calculul
incasarilor per gen
BEGIN
    --cursoare imbricate
    OPEN cursor_type;   --deschidem cursorul 'cursor_type'
```

```
    LOOP
        FETCH cursor_type
          INTO v_t_id, v_t_name, v_t_nr;  --salvam informatiile in
variabile

              EXIT  WHEN  cursor_type%notfound;  --parcurgem  intreg
cursorul de genuri
                                                --si pentru fiecare gen
afisam informatiile stocate in
                                            --cursorul 'cursor_show'

        dbms_output.put_line('Genul spectacolului: ' || v_t_name);
        dbms_output.put_line('============================');

        total := 0; --suma este initial zero

        IF v_t_nr = 0 THEN
             --daca nu avem spectacole dintr-un gen afisam un mesaj
corespunzator
                   dbms_output.put_line('Nu  avem  filme  la  această
categorie.');
        ELSE
            OPEN cursor_show;   --deschidem cursorul 'cursor_show'

            LOOP
                FETCH cursor_show
                    INTO v_s_id, v_s_title, v_s_type_id, v_s_price;
--salvam informatiile in variabile
                 EXIT WHEN cursor_show%notfound;                 --
parcurgem intreg cursorul cu angajati

                  --afisam informatiile corespunzatoare genului la
care am ajuns
                IF v_t_id = v_s_type_id THEN
                          dbms_output.put_line(v_s_title || ' ' ||
v_s_price || ' RON');
                     total := total + v_s_price; --aduna incasarile
totale ale genului
                END IF;
            END LOOP;

            CLOSE cursor_show;  --inchidem cursorul 'cursor_show'
        END IF;

        --afisam incasarile totale ale genului
        dbms_output.put_line('----------');
        dbms_output.put_line('Total incasari: ' || total);
```

```
        dbms_output.new_line;
    END LOOP;

    CLOSE cursor_type;  --inchidem cursorul 'cursor_type'
END ex_7;
/
```

Apelare

```
/
BEGIN
    ex_7();
END;
/
```



EXERCIȚIUL 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.

Cerință

Pentru un spectacol citit să se returneze o lista cu lunile în care s-au vandut număr maxim de bilete și o variabilă cu media vârstei spectatorilor (vârsta lor de la momentul achiziționării biletului).

Rezultatul va fi de tipul 'rezultat_ex8':

```
/
CREATE OR REPLACE TYPE lista_luni IS
    TABLE OF NUMBER(4);
/
CREATE OR REPLACE TYPE rezultat_ex8 AS OBJECT (
    luna           lista_luni,
    medie_varsta   NUMBER
);
/
```

Dacă la un spectacol nu s-au cumpărat bilete, lista va avea ca singur element un zero, iar variabila va fi tot zero.

Rezolvare

```
CREATE OR REPLACE FUNCTION ex_8 (
    v_title show.show_title%TYPE    --primeste ca parametru numele
spectacolului
)
  RETURN  rezultat_ex8  IS    --returneaza  un  rezultat  de  tipul
'rezultat_ex8'

    --cursor ce contine data rezervarii biletelor la spectacol si
varsta spectatorilor
    CURSOR c IS
        SELECT r.booking_date, round((booking_date - birth_date) /
365.25, 0)
        FROM show s, reservation r, client c
        WHERE s.show_id = r.show_id
            AND c.client_id = r.client_id
            AND s.show_id = (SELECT show_id
                             FROM show
                                          WHERE  upper(v_title)  =
upper(show_title))       --gasim  id-ul  spectacolului  primit  ca
parametru
           ORDER BY booking_date;   --ordonam in functie de data
rezervarii

    --vector ce va pastra nr de bilete pentru fiecare luna
    TYPE t_aparitii IS VARRAY(12) OF NUMBER(6);
```

```
    --variabile pentru accesarea datelor din cursor
     v_booking     reservation.booking_date%TYPE;  --memoreaza data
achizitionarii unui bilet
    v_age        NUMBER; --memoreaza varsta unui participant

    --vector initializat cu 0 bilete pentru fiecare luna
     v_aparitii   t_aparitii := t_aparitii(0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0);

     --lista unde se vor pastra lunile cu acelasi numar maxim de
bilete vandute
    f_luna       lista_luni := lista_luni();

    maxim        NUMBER := -1; --maximul de bilete vandute
    c_luna       NUMBER := 0;  --numarul de luni cu acelasi maxim
    media        NUMBER := 0;  --suma varstelor
    cc           NUMBER := 0;  --cate persoane avem
    i            NUMBER;
    k            NUMBER;

      --variabila ce va verifica daca spectacolul a avut sau nu
bilete vandute
    if_record    BOOLEAN := false;
BEGIN
    --tratare exceptie NO_DATA_FOUND
    SELECT COUNT(1)
    INTO k
    FROM show
    WHERE upper(v_title) = upper(show_title);

    IF k <= 0 THEN
        RAISE no_data_found;
    END IF;

    OPEN c; --deschidem cursorul

    LOOP
         FETCH c INTO v_booking, v_age;  --pastram datele de pe o
linie in doua variabile
        EXIT WHEN c%notfound;           --ne oprim cand parcurgem
tot cursorul

         if_record := true;  --nu s-a inchis cursorul => inseamna
ca avem date (s-au vandut bilete)

          i := to_number(to_char(extract(MONTH FROM v_booking)));
--extragem luna din data rezervarii
```

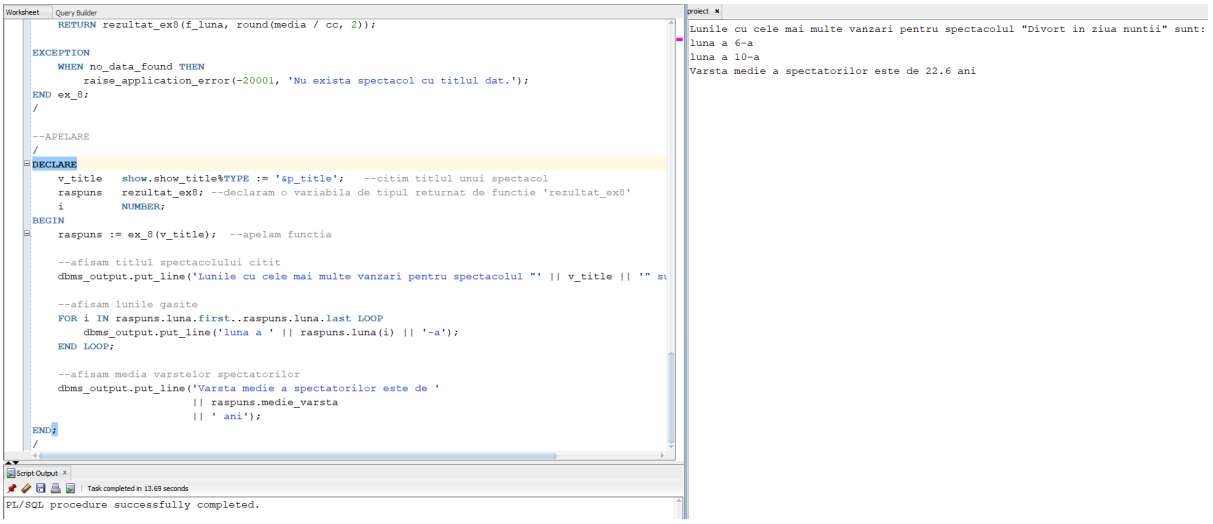
```
                        v_aparitii(i)   :=   v_aparitii(i)   +   1;
--crestem nr de bilete vandute in luna respectiva

        --cautam maximul de bilete vandu-te intr-o luna
        IF ( maxim < v_aparitii(i) ) THEN
            maxim := v_aparitii(i);
        END IF;

          --calculam nr de persoane care au fost la spectacol si
suma varstelor
        IF ( v_age IS NOT NULL ) THEN
            cc := cc + 1;
            media := media + to_number(v_age);
        END IF;
    END LOOP;

    CLOSE c;    --inchidem cursorul

    --adaugam fiecare luna care are nr maxim de bilete vandute
    --in lista 'f_luna'
    FOR j IN v_aparitii.first..v_aparitii.last LOOP
        IF ( v_aparitii(j) = maxim ) THEN
            f_luna.extend;
            c_luna := c_luna + 1;
            f_luna(c_luna) := j;
        END IF;
    END LOOP;

    --returnam lista cu lunile cu nr maxim de bilete vandute si
    --media varstei spectatorilor
    --(asta daca au existat bilete vandute la spectacol)
    IF if_record = false THEN
        RETURN rezultat_ex8(lista_luni(0), 0);
    END IF;

    RETURN rezultat_ex8(f_luna, round(media / cc, 2));

EXCEPTION
    WHEN no_data_found THEN
          raise_application_error(-20001, 'Nu exista spectacol cu
titlul dat.');
END ex_8;
/
```

Apelare

```
/
DECLARE
    v_title   show.show_title%TYPE := '&p_title';   --citim titlul
unui spectacol
      raspuns     rezultat_ex8;  --declaram  o  variabila  de  tipul
returnat de functie 'rezultat_ex8'
    i         NUMBER;
BEGIN
    raspuns := ex_8(v_title);  --apelam functia

    --afisam titlul spectacolului citit
     dbms_output.put_line('Lunile cu cele mai multe vanzari pentru
spectacolul "' || v_title || '" sunt:');

    --afisam lunile gasite
    FOR i IN raspuns.luna.first..raspuns.luna.last LOOP
            dbms_output.put_line('luna  a  '  ||  raspuns.luna(i)  ||
'-a');
    END LOOP;

    --afisam media varstelor spectatorilor
    dbms_output.put_line('Varsta medie a spectatorilor este de '
                         || raspuns.medie_varsta
                         || ' ani');
END;
/
```

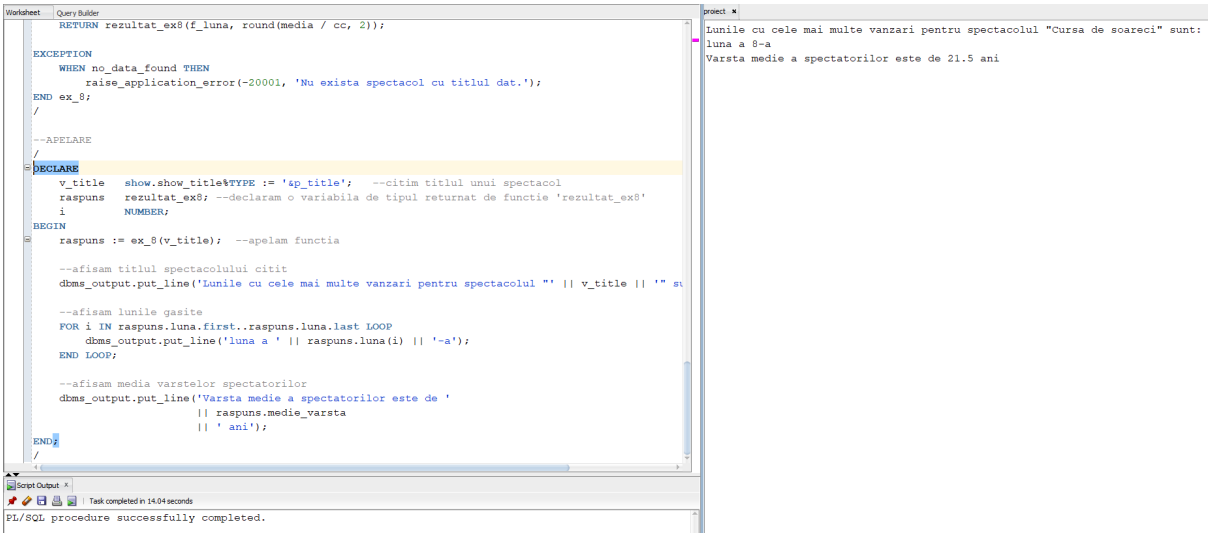
Pentru citirea titlului de spectacol ***Divort in ziua nuntii*** se afișează:

```
Lunile cu cele mai multe vanzari pentru spectacolul "Divort in
ziua nuntii" sunt:
luna a 6-a
luna a 10-a
Varsta medie a spectatorilor este de 22.6 ani
```



Pentru citirea titlului de spectacol ***Cursa de soareci*** se afișează:

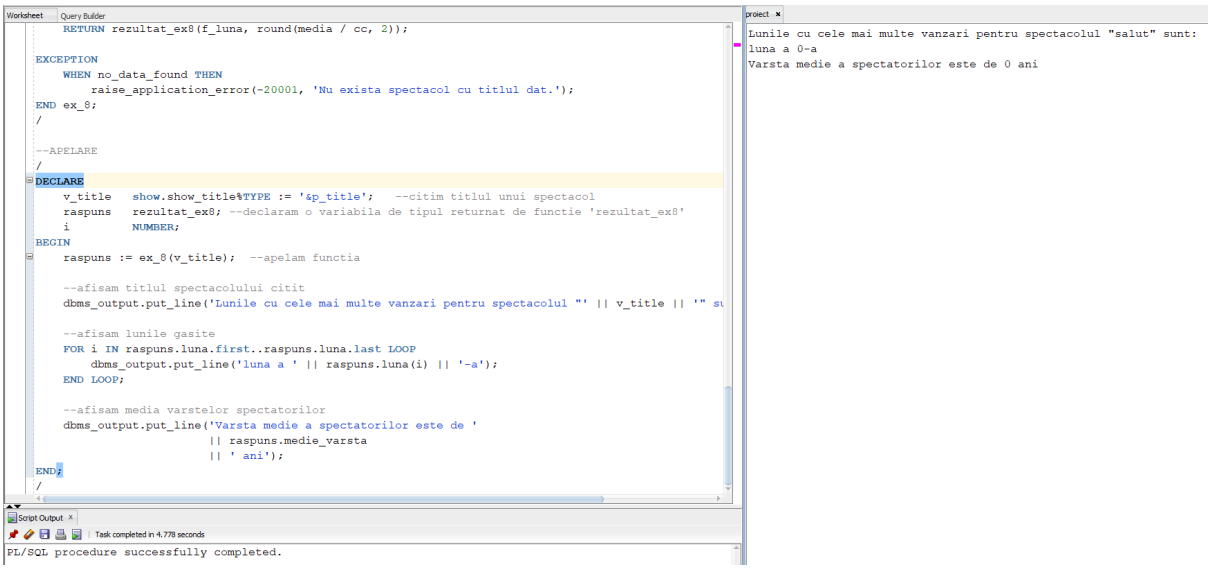
```
Lunile cu cele mai multe vanzari pentru spectacolul "Cursa de
soareci" sunt:
luna a 8-a
Varsta medie a spectatorilor este de 21.5 ani
```



Pentru citirea titlului de spectacol ***Salut*** se afișează:

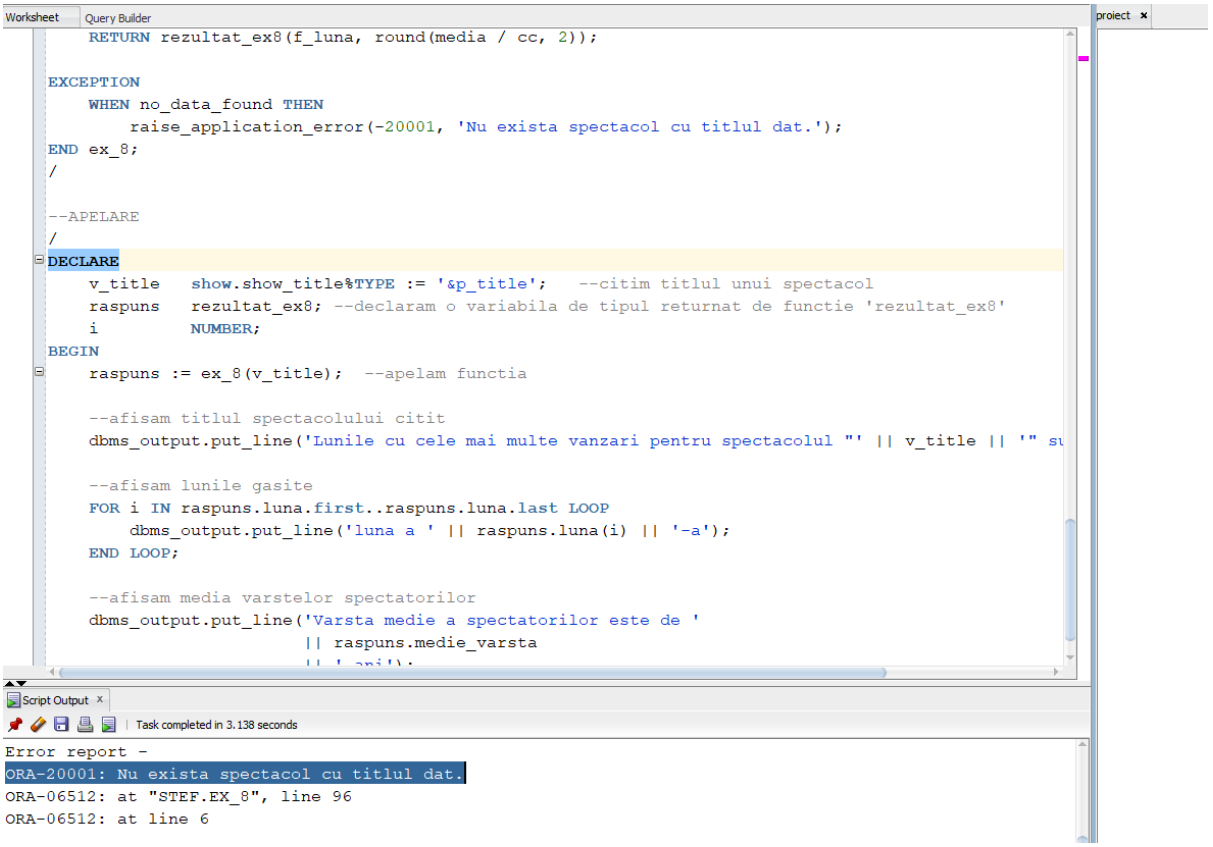
```
Lunile cu cele mai multe vanzari pentru spectacolul "salut" sunt:
luna a 0-a
Varsta medie a spectatorilor este de 0 ani
```

deoarece spectacolul *Salut* nu are bilete vândute. Prin urmare lista conține valoarea zero, iar variabila este tot zero.



Pentru citirea unui titlu de spectacol care nu se regăsește în baza de date va apărea eroare:

```
ORA-20001: Nu exista spectacol cu titlul dat.
```



EXERCIȚIUL 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.

Cerință

Realizați o procedură care pentru numele unui teatru dat ca parametru va determina valorile:

- ***specific*** = specificul teatrului (i.e. ce tipuri de spectacole are cel mai mult; dacă sunt mai multe spectacole cu același specific se va alege ultimul găsit);
- ***maxim*** = numărul maxim de spectacole de la aceeași echipă de producători din cadrul teatrului citit;

În final obțineți lista producătorilor care au mai mult de ***maxim*** spectacole și toate de genul ***specific***.

Rezolvare

```
CREATE OR REPLACE PROCEDURE ex_9 (
          v_nume_teatru  theater.theater_name%TYPE  --primeste  ca
parametru numele teatrului
)
 IS

    --cursor ce contine id-ul producatorilor si id-ul tipului
    --spectacolelor ce se joaca la teatrul citit
    CURSOR c_1 IS
        SELECT p.prod_id   prod, t.type_id   typess
         FROM theater t, collaboration c, show s, producers p, type
t
        WHERE c.theater_id = t.theater_id
            AND c.show_id = s.show_id
            AND s.prod_id = p.prod_id
            AND t.theater_id = (SELECT theater_id
                                FROM theater
                                       WHERE upper(theater_name) =
upper(v_nume_teatru)
                                  )   --obtinem id-ul spectacolului
citit
            AND s.type_id = t.type_id
        ORDER BY
                       s.show_id;  --ordonam  credcator  dupa  id-ul
spectacolului

      --cursor  ce  obtine  id-ul,  directorul  si  scenaristul  unei
echipe de producatori
     --impreuna cu numarul de spectacole pe care le desfasoara de
tipul 'c_specific'
```

```
    --tipul de specatcole pe care trebuie sa le numere le primeste
prin intermediul
    --unui parametru
    CURSOR c_2 (c_specific NUMBER) IS
                SELECT  p.prod_id  id2,  MAX(p.director)  director,
MAX(p.screenwriter) screenw, COUNT(type_id) nr
        FROM producers   p, show        s
        WHERE p.prod_id = s.prod_id AND s.type_id = c_specific
        GROUP BY p.prod_id;

    TYPE t_producers IS TABLE OF NUMBER(4);
    TYPE t_types IS TABLE OF NUMBER(4);
    TYPE t_apar_prod IS TABLE OF NUMBER(4);
    TYPE t_apar_type IS TABLE OF NUMBER(4);

    --tablou cu id-urile producatorilor
    t_prod          t_producers := t_producers();
    --tablou cu id-urile genurilor de spectacole
    t_type          t_types := t_types();
     --tablou pentru a salva numarul de aparitii al valorilor din
't_prod'
    apar_p          t_apar_prod := t_apar_prod();
     --tablou pentru a salva numarul de aparitii al valorilor din
't_type'
    apar_t          t_apar_type := t_apar_type();

     max_prod           NUMBER := -1;    --salvam id-ul maxim de
producatori pentru a extinde lista 'apar_p'
    max_type        NUMBER := -1;   --salvam id-ul maxim de tipuri
pentru a extinde lista 'apar_t'
     maxim              NUMBER := -1;    --salvam numarul maxim de
spectacole la care avem aceeasi producatori in teatrul citit
     specific           NUMBER := -1;    --salvam id-ul tipului de
spectacole care se gasesc cel mai mult la teatrul citit
    specific_m      NUMBER := -1;   --
     specific_nume    VARCHAR2(50);    --salvam numele tipului in
functie de id-ul gasit
    k               NUMBER := 0;

    nimic            BOOLEAN := FALSE;   --verifica daca avem sau
nu echipe de profucatori

     if_found_rec     BOOLEAN := FALSE;    --verifica daca teatrul
introdus are sau nu spectacole
     not_found_rec    EXCEPTION;            --exceptie in cazul in
care teatrul nu are spectacole
BEGIN
```

```
     --vedem de cate ori apare numele teatrului citit in baza de
date
    SELECT COUNT(1)
    INTO k
    FROM theater
    WHERE upper(theater_name) = upper(v_nume_teatru);

    IF k <= 0 THEN
        RAISE no_data_found;    --tratare exceptie NO_DATA_FOUND
    ELSIF k > 1 THEN
        RAISE too_many_rows;    --tratare exceptie TOO_MANY_ROWS
    END IF;

    k := 0;  --numaram cate elemente avem in cursorul c_1
    FOR i IN c_1
    LOOP
        if_found_rec := TRUE;   --daca avem spectacole := TRUE
        k := k + 1;
        t_prod.extend;          --extindem tabloul
        t_prod(k) := i.prod;    --adaugam elementul din cursor

        --salvam id-ul maxim de producator
        IF ( max_prod < i.prod ) THEN
            max_prod := i.prod;
        END IF;

        t_type.extend;          --extindem tabloul
        t_type(k) := i.typess;  --adaugam elementul din cursor

        --salvam id-ul maxim de tip
        IF ( max_type < i.typess ) THEN
            max_type := i.typess;
        END IF;

    END LOOP;

    --daca nu am avut date/spectacole atunci exceptie
    IF ( if_found_rec = FALSE ) THEN
        RAISE not_found_rec;
    END IF;

    --initializam tabloul de aparitii 'apar_p' cu zero
    FOR i IN 1..max_prod LOOP
        apar_p.extend;
        apar_p(i) := 0;
    END LOOP;

    --initializam tabloul de aparitii 'apar_t' cu zero
```

```
    FOR i IN 1..max_type LOOP
        apar_t.extend;
        apar_t(i) := 0;
    END LOOP;

     --populam tabloul de aparitii 'apar_p' in functie de datele
din 't_prod'
    FOR i IN t_prod.first..t_prod.last LOOP
        apar_p(t_prod(i)) := apar_p(t_prod(i)) + 1;

        IF ( maxim < apar_p(t_prod(i)) ) THEN
            maxim := apar_p(t_prod(i));
        END IF;
    END LOOP;

     --populam tabloul de aparitii 'apar_t' in functie de datele
din 't_type'
    FOR i IN t_type.first..t_type.last LOOP
        apar_t(t_type(i)) := apar_t(t_type(i)) + 1;

        IF ( specific_m < apar_t(t_type(i)) ) THEN
            specific_m := apar_t(t_type(i));
            specific := t_type(i);
        END IF;
    END LOOP;

    --salvam in 'specific_nume' denumirea tipului gasit
    SELECT type_name
    INTO specific_nume
    FROM type
    WHERE type_id = specific;

    --afisam ce valoare are 'specific' si ce valoare are 'maxim'
       dbms_output.put_line('Teatrul  "'  ||  v_nume_teatru  ||  '"
are:');
      dbms_output.put_line('-  specificul:  '  ||  specific_nume  ||
';');
     dbms_output.put_line('- maximul de spectacole ale acelorasi
producatori ' || maxim || ';');
     dbms_output.put_line('Lista producatorilor ' || 'cu mai mult
de '
                          || maxim || ' spectacole '
                          || specific_nume || ':');

         --parcurgem  cursorul  'c_2'  pentru  a  obtine  tabloul
producatorilor care
    --au mai mult de 'maxim' spectacole de tipul 'specific' gasit
    FOR i IN c_2(specific) LOOP
```

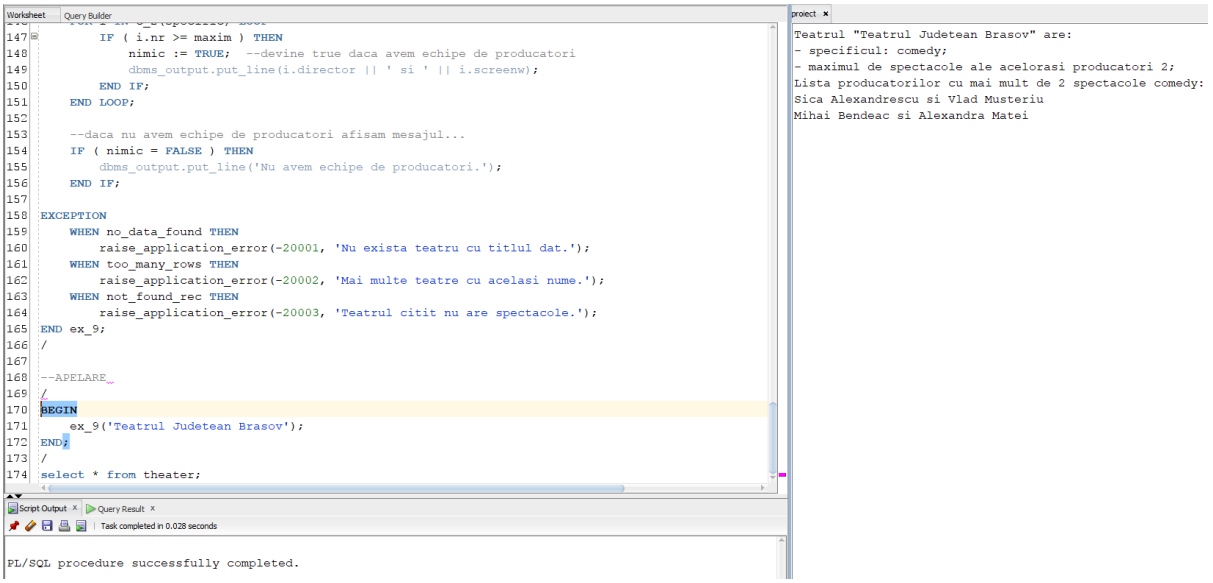
```
        IF ( i.nr >= maxim ) THEN
                nimic := TRUE;  --devine true daca avem echipe de
producatori
                    dbms_output.put_line(i.director || ' si ' ||
i.screenw);
        END IF;
    END LOOP;

    --daca nu avem echipe de producatori afisam mesajul...
    IF ( nimic = FALSE ) THEN
        dbms_output.put_line('Nu avem echipe de producatori.');
    END IF;

EXCEPTION
    WHEN no_data_found THEN
            raise_application_error(-20001, 'Nu exista teatru cu
titlul dat.');
    WHEN too_many_rows THEN
            raise_application_error(-20002, 'Mai multe teatre cu
acelasi nume.');
    WHEN not_found_rec THEN
            raise_application_error(-20003, 'Teatrul citit nu are
spectacole.');
END ex_9;
/
```

Apelare

```
DECLARE
    nume theater.theater_name%TYPE:='&p_nume';
BEGIN
    ex_9(nume);
--     Teatrul Judetean Brasov
--     Teatrul Nottara
--     Teatrul Artcub
--     asd
--     Teatrul Guguta
--     Verde
END;
/
```

Apelare_1

Citirea numelui de teatru **Teatrul Judetean Brasov.**



Apelare_2

Citirea numelui de teatru **Nottara.**

```
        IF ( i.nr >= maxim ) THEN
            nimic := TRUE;  --devine true daca avem echipe de producatori
            dbms_output.put_line(i.director || ' si ' || i.screenw);
        END IF;
    END LOOP;

    --daca nu avem echipe de producatori afisam mesajul...
    IF ( nimic = FALSE ) THEN
        dbms_output.put_line('Nu avem echipe de producatori.');
    END IF;

EXCEPTION
    WHEN no_data_found THEN
        raise_application_error(-20001, 'Nu exista teatru cu titlul dat.');
    WHEN too_many_rows THEN
        raise_application_error(-20002, 'Mai multe teatre cu acelasi nume.');
    WHEN not_found_rec THEN
        raise_application_error(-20003, 'Teatrul citit nu are spectacole.');
END ex_9;
/

--APELARE
/
BEGIN
    ex_9('Teatrul Nottara');
END;
/
select * from theater;
```

```
Teatrul "Teatrul Nottara" are:
- specificul: tragic;
- maximul de spectacole ale acelorasi producatori 1;
Lista producatorilor cu mai mult de 1 spectacole tragic:
Sica Alexandrescu si Vlad Musteriu
```

```
PL/SQL procedure successfully completed.
```

Apelare_3

În urma apelării subprogramului de valoarea ***Teatrul Artcub*** se afișează mesajul
`Nu avem echipe de producatori.`
deoarece nu există echipe de producători cu criteriile găsite.

```
Worksheet   Query Builder
        IF ( i.nr >= maxim ) THEN
            nimic := TRUE;  --devine true daca avem echipe de producatori
            dbms_output.put_line(i.director || ' si ' || i.screenw);
        END IF;
    END LOOP;

    --daca nu avem echipe de producatori afisam mesajul...
    IF ( nimic = FALSE ) THEN
        dbms_output.put_line('Nu avem echipe de producatori.');
    END IF;

EXCEPTION
    WHEN no_data_found THEN
        raise_application_error(-20001, 'Nu exista teatru cu titlul dat.');
    WHEN too_many_rows THEN
        raise_application_error(-20002, 'Mai multe teatre cu acelasi nume.');
    WHEN not_found_rec THEN
        raise_application_error(-20003, 'Teatrul citit nu are spectacole.');
END ex_9;
/

--APELARE
/
BEGIN
    ex_9('Teatrul Artcub');
END;
/
select * from theater;
```

```
Script Output   Query Result
Task completed in 0.032 seconds

PL/SQL procedure successfully completed.
```

```
proiect
Teatrul "Teatrul Artcub" are:
- specificul: tragic;
- maximul de spectacole ale acelorasi producatori 2;
Lista producatorilor cu mai mult de 2 spectacole tragic:
Nu avem echipe de producatori.
```

Apelare_4

În urma apelării subprogramului cu un titlu inexistent în baza de date apare eroarea
`ORA-20001: Nu exista teatru cu titlul dat.`

```
147         IF ( i.nr >= maxim ) THEN
148             nimic := TRUE;  --devine true daca avem echipe de producatori
149             dbms_output.put_line(i.director || ' si ' || i.screenw);
150         END IF;
151     END LOOP;
152
153     --daca nu avem echipe de producatori afisam mesajul...
154     IF ( nimic = FALSE ) THEN
155         dbms_output.put_line('Nu avem echipe de producatori.');
156     END IF;
157
158 EXCEPTION
159     WHEN no_data_found THEN
160         raise_application_error(-20001, 'Nu exista teatru cu titlul dat.');
161     WHEN too_many_rows THEN
162         raise_application_error(-20002, 'Mai multe teatre cu acelasi nume.');
163     WHEN not_found_rec THEN
164         raise_application_error(-20003, 'Teatrul citit nu are spectacole.');
165 END ex_9;
166 /
167
168 --APELARE
169 /
170 BEGIN
171     ex_9('asd');
172 END;
173 /
174 select * from theater;
```

```
Error report -
ORA-20001: Nu exista teatru cu titlul dat.
ORA-06512: at "STEF.EX_9", line 159
ORA-06512: at line 2
```

Apelare_5

În urma apelării subprogramului de valoarea ***Teatrul Guguta*** apare eroarea
`ORA-20002: Mai multe teatre cu acelasi nume.`
deoarece există mai multe teatre cu același nume.

Worksheet | Query Builder | proiect

```
        IF ( i.nr >= maxim ) THEN
            nimic := TRUE;  --devine true daca avem echipe de producatori
            dbms_output.put_line(i.director || ' si ' || i.screenw);
        END IF;
    END LOOP;

    --daca nu avem echipe de producatori afisam mesajul...
    IF ( nimic = FALSE ) THEN
        dbms_output.put_line('Nu avem echipe de producatori.');
    END IF;

EXCEPTION
    WHEN no_data_found THEN
        raise_application_error(-20001, 'Nu exista teatru cu titlul dat.');
    WHEN too_many_rows THEN
        raise_application_error(-20002, 'Mai multe teatre cu acelasi nume.');
    WHEN not_found_rec THEN
        raise_application_error(-20003, 'Teatrul citit nu are spectacole.');
END ex_9;
/

--APELARE
/
BEGIN
    ex_9('Teatrul Guguta');
END;
/
select * from theater;
```

Script Output | Query Result | Task completed in 0.034 seconds

```
Error report -
ORA-20002: Mai multe teatre cu acelasi nume.
ORA-06512: at "STEF.EX_9", line 161
ORA-06512: at line 2
```

Apelare_6

În urma apelării subprogramului de valoarea ***Verde*** apare eroarea
`ORA-20003: Teatrul citit nu are spectacole.`
deoarece teatrul *Verde* nu are spectacole.

Worksheet | Query Builder | proiect

```
147         IF ( i.nr >= maxim ) THEN
148             nimic := TRUE;  --devine true daca avem echipe de producatori
149             dbms_output.put_line(i.director || ' si ' || i.screenw);
150         END IF;
151     END LOOP;
152
153     --daca nu avem echipe de producatori afisam mesajul...
154     IF ( nimic = FALSE ) THEN
155         dbms_output.put_line('Nu avem echipe de producatori.');
156     END IF;
157
158 EXCEPTION
159     WHEN no_data_found THEN
160         raise_application_error(-20001, 'Nu exista teatru cu titlul dat.');
161     WHEN too_many_rows THEN
162         raise_application_error(-20002, 'Mai multe teatre cu acelasi nume.');
163     WHEN not_found_rec THEN
164         raise_application_error(-20003, 'Teatrul citit nu are spectacole.');
165 END ex_9;
166 /
167
168 --APELARE
169 /
170 BEGIN
171     ex_9('Verde');
172 END;
173 /
174 select * from theater;
```

Script Output | Query Result | Task completed in 0.045 seconds

```
Error report -
ORA-20003: Teatrul citit nu are spectacole.
ORA-06512: at "STEF.EX_9", line 163
ORA-06512: at line 2
```

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

Trigger pentru 'client' care permite:

- inserarea oricand si afiseaza mesajul 'Inserare...'
- actualizarea:
  - in intervalul 08:00-20:00 pentru orice coloana si afiseaza mesajul 'Actualizare...';
- stergerea de luni pana vineri;

```
CREATE OR REPLACE TRIGGER trigger_ex_10
  BEFORE INSERT OR
         UPDATE OR
         DELETE
         ON client
BEGIN
  CASE
    WHEN INSERTING THEN
      DBMS_OUTPUT.PUT_LINE('Inserare...');

    WHEN UPDATING THEN
        if(TO_CHAR(SYSDATE,'HH24') BETWEEN 8 AND 20)then
            DBMS_OUTPUT.PUT_LINE('Actualizare...');
        else
                RAISE_APPLICATION_ERROR(-20001,'Actualizarile sunt
permise doar intre 8 si 20');
        end if;

    WHEN DELETING THEN
      if(TO_CHAR(SYSDATE,'D') between 2 and 6) then
        DBMS_OUTPUT.PUT_LINE('Stergere...');
      else
         RAISE_APPLICATION_ERROR(-20002,'Stergerile sunt permise de
luni pana vineri');
      end if;
  END CASE;
END;
/
```

```
--TESTARE
--INSERARE
insert into CLIENT values(80,'A','a','A.a@yahoo.com',to_date('01-01-1999','DD-MM-YYYY'));
select * from client;
--Inserare…
```

Worksheet | Query Builder

```
23          end if;
24
25      WHEN DELETING THEN
26        if(TO_CHAR(SYSDATE,'D') between 2 and 6) then
27          DBMS_OUTPUT.PUT_LINE('Stergere...');
28        else
29          RAISE_APPLICATION_ERROR(-20002,'Stergerile sunt permise de luni pana vineri');
30        end if;
31    END CASE;
32  END;
33  /
34  --INSERARE
35  insert into CLIENT values(80,'A','a','A.a@yahoo.com',to_date('01-01-1999','DD-MM-YYYY'));
36  select * from client;
37  --Inserare...
38
39  --ACTUALIZARE
40  --intre 8-20
41  update client
42  set firstname = 'b'
43  where client_id = 80;
44  --Actualizare...
```

proiect

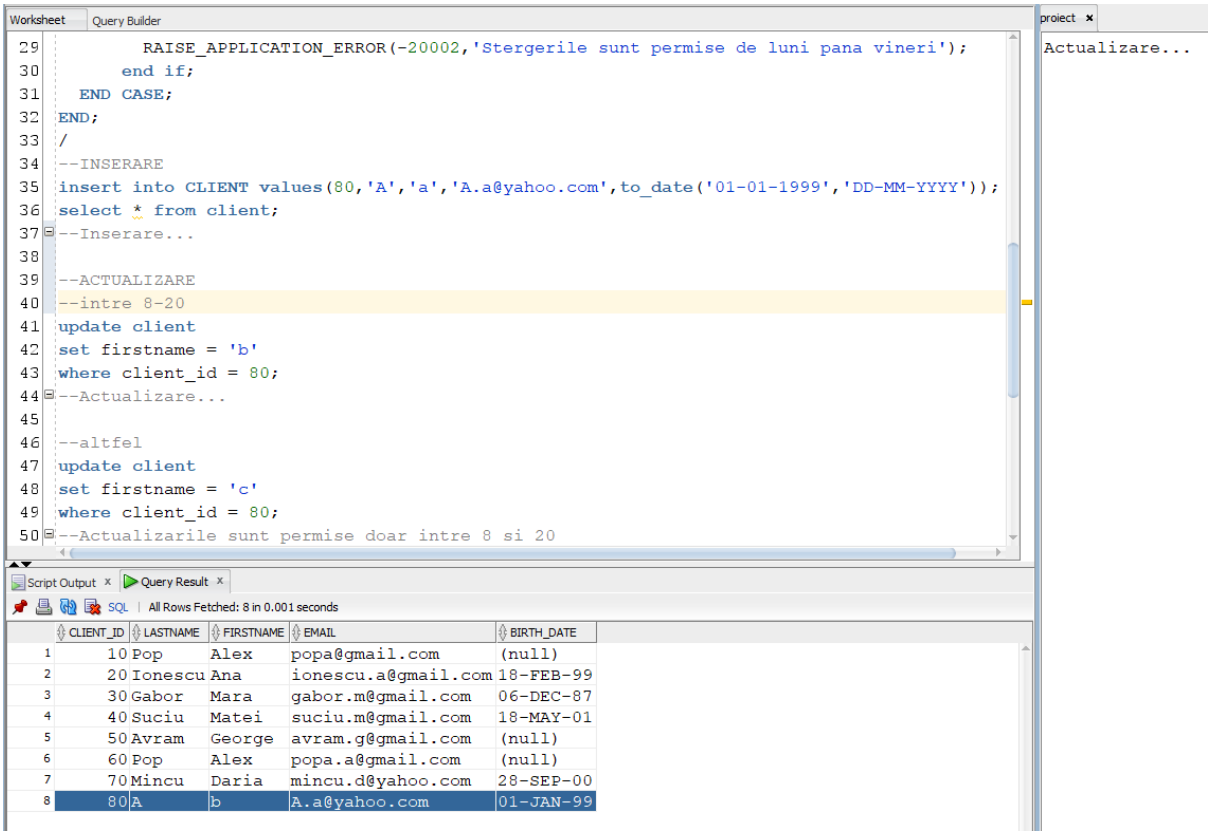
```
Inserare...
```

Script Output | Query Result

SQL | All Rows Fetched: 8 in 0.003 seconds

	CLIENT_ID	LASTNAME	FIRSTNAME	EMAIL	BIRTH_DATE
1	10	Pop	Alex	popa@gmail.com	(null)
2	20	Ionescu	Ana	ionescu.a@gmail.com	18-FEB-99
3	30	Gabor	Mara	gabor.m@gmail.com	06-DEC-87
4	40	Suciu	Matei	suciu.m@gmail.com	18-MAY-01
5	50	Avram	George	avram.g@gmail.com	(null)
6	60	Pop	Alex	popa.a@gmail.com	(null)
7	70	Mincu	Daria	mincu.d@yahoo.com	28-SEP-00
8	80	A	a	A.a@yahoo.com	01-JAN-99

```
--ACTUALIZARE
--intre 8-20
update client
set firstname = 'b'
where client_id = 80;
--Actualizare…
```



```
--altfel
update client
set firstname = 'c'
where client_id = 80;
--Actualizarile sunt permise doar intre 8 si 20
```

Worksheet | Query Builder

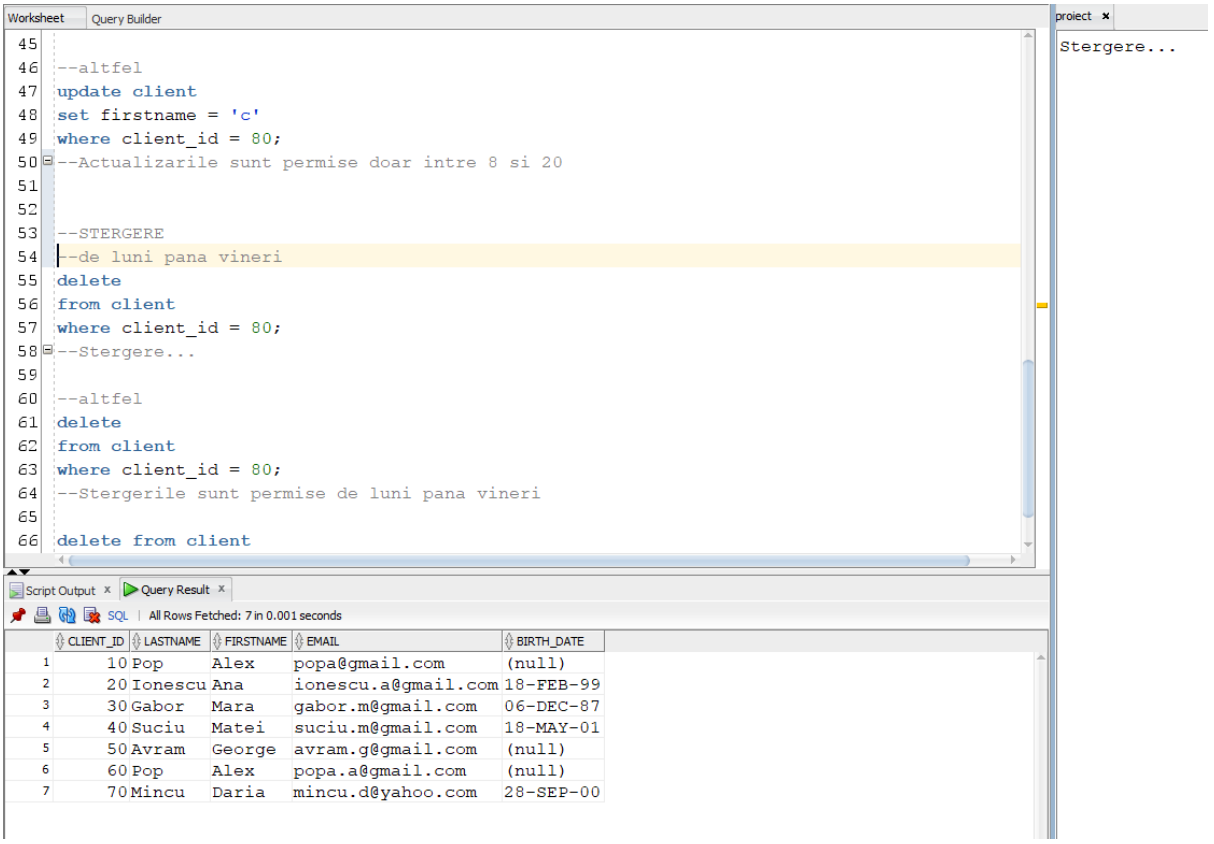
```
30        end if;
31    END CASE;
32  END;
33  /
34  --INSERARE
35  insert into CLIENT values(80,'A','a','A.a@yahoo.com',to_date('01-01-1999','DD-MM-YYYY'));
36  select * from client;
37  --Inserare...
38
39  --ACTUALIZARE
40  --intre 8-20
41  update client
42  set firstname = 'b'
43  where client_id = 80;
44  --Actualizare...
45
46  --altfel
47  update client
48  set firstname = 'c'
49  where client_id = 80;
50  --Actualizarile sunt permise doar intre 8 si 20
51
```

Script Output | Query Result

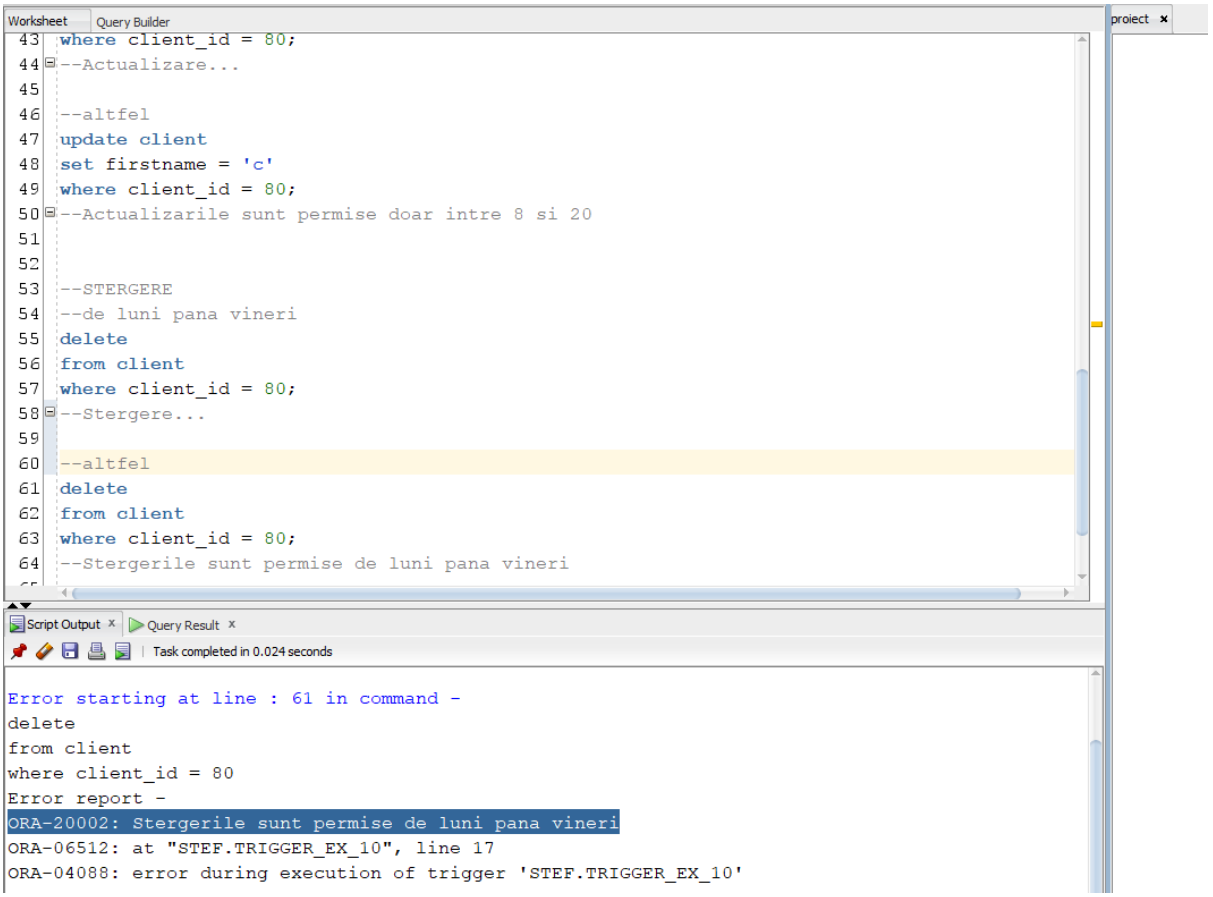
Task completed in 0.029 seconds

```
Error starting at line : 47 in command -
update client
set firstname = 'c'
where client_id = 80
Error report -
ORA-20001: Actualizarile sunt permise doar intre 8 si 20
ORA-06512: at "STEF.TRIGGER_EX_10", line 10
ORA-04088: error during execution of trigger 'STEF.TRIGGER_EX_10'
```

```
--STERGERE
--de luni pana vineri
delete
from client
where client_id = 80;
--Stergere…
```



```
--altfel
delete
from client
where client_id = 80;
--Stergerile sunt permise de luni pana vineri
```




```
rollback;
drop trigger trigger_ex_10;
```

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

Trigger ce realizează ștergeri în cascadă. Mai exact ștergerea unui tip din tabela 'type' (tabela părinte) determină ștergerea spectacolelor și implicit a colaborarilor și rezervarilor - ce conțin spectacolul respectiv.

```
create or replace trigger trigger_ex_11
    before delete on type
     for each row     --pentru fiecare stergere de linie din tabela
'type'

begin
    --ciclu cursor cu subcereri ce contine
    --toate spectacolele de tipul ce urmeaza a fi sters
    for i in (select s.show_id sh
                from show s
                where s.type_id = :old.type_id) loop

        --stergem colaborarile ce contineau spectacole
        --de tipul sters
        delete
        from collaboration c
        where c.show_id = i.sh;

        --stergem rezervarile ce contineau spectacole
        --de tipul sters
        delete
        from reservation r
        where r.show_id = i.sh;

        --stergem spectacolele de tipul sters
        delete
        from show s
        where s.show_id = i.sh;
    end loop;
end;
/
```

Vom șterge categoria cu id-ul 4.

```
--TESTARE
select * from type;             --6 inregistrari
select * from show;             --11 inregistrari
select * from collaboration;    --10 inregistrari
select * from reservation;      --16 inregistrari
```

```sql
delete from type
where type_id = 4;

select * from type;                --5 i
select * from show;                --4 i
select * from collaboration;       --6 i
select * from reservation;         --8 i
```

Script Output x | Query Result x

SQL | All Rows Fetched: 6 in 0.029 seconds

	TYPE_ID	TYPE_NAME
1	1	absurd
2	2	grotesque
3	3	satire
4	4	comedy
5	5	tragic
6	6	drama

```sql
--TESTARE
select * from type;                --6 inregistrari
select * from show;                --11 inregistrari
select * from collaboration;       --10 inregistrari
select * from reservation;         --16 inregistrari

delete from type
where type_id = 4;
```

Script Output x | Query Result x

SQL | All Rows Fetched: 11 in 0.003 seconds

	SHOW_ID	SHOW_TITLE	DURATION	RATING	LAUNCH_YEAR	PRICE	TYPE_ID	PROD_ID
1	10	Divort in ziua nuntii	2	3	2002	50	4	3
2	11	Ursul	3	3.5	2010	20	2	3
3	12	Necasatoria	2	4	2010	15	5	2
4	13	Cursa de soareci	2	2	1999	33	6	3
5	14	O noapte furtunoasa	3	5	1999	45	4	1
6	15	Doi pe o banca	2	(null)	2019	27	4	4
7	16	Colectionarul	2	(null)	2021	16	4	2
8	17	Meciul de comedie	2	4	2011	17	2	2
9	18	Avarul	2	4	2021	28	4	4
10	19	Salut	2	3	2018	20	4	2
11	20	Inca una si ma duc	2	3	2018	20	4	2

```
select * from collaboration;     --10 inregistra
select * from reservation;       --16 inregistra

delete from type
where type_id = 4;
```

Script Output x | Query Result x

SQL | All Rows Fetched: 10 in 0.003 seconds

	COLLABORATION_ID	SHOW_ID	THEATER_ID	ACTOR_ID
1	101	10	10	20
2	102	10	20	20
3	103	10	40	20
4	104	11	30	30
5	105	12	40	30
6	106	12	40	30
7	107	12	60	40
8	108	12	50	40
9	109	16	10	40
10	110	17	10	20

```
    end loop;
end;
/

--TESTARE
select * from type;                --6 inregistrari
select * from show;                --11 inregistrari
```

Script Output x | Query Result x

SQL | All Rows Fetched: 16 in 0.007 seconds

	RESERVATION_ID	SHOW_ID	CLIENT_ID	BOOKING_DATE
1	101	10	10	04-NOV-21
2	102	11	10	13-OCT-21
3	103	12	10	25-MAR-21
4	104	10	70	04-DEC-21
5	105	11	70	28-AUG-21
6	106	12	70	11-JAN-21
7	107	13	70	14-AUG-21
8	108	17	20	07-DEC-21
9	109	17	20	12-AUG-21
10	110	10	40	23-OCT-21
11	111	13	20	14-AUG-21
12	112	10	10	01-JUN-21
13	113	10	70	14-OCT-21
14	114	10	60	30-OCT-21
15	115	10	40	15-JUN-20
16	116	10	30	05-JUN-20

```
delete from type
where type_id = 4;

select * from type;             --5 inregistrari
select * from show;             --4 inregistrari
select * from collaboration;    --6 inregistrari
select * from reservation;      --8 inregistrari
```

```
--TESTARE
select * from type;             --6 inregistrari
select * from show;             --11 inregistrari
select * from collaboration;    --10 inregistrari
select * from reservation;      --16 inregistrari

delete from type
where type_id = 4;

select * from type;             --5 inregistrari
select * from show;             --4 inregistrari
select * from collaboration;    --6 inregistrari
```

Script Output x | Query Result x

SQL | All Rows Fetched: 5 in 0.001 seconds

	TYPE_ID	TYPE_NAME
1	1	absurd
2	2	grotesque
3	3	satire
4	5	tragic
5	6	drama

```sql
--TESTARE
select * from type;                --6 inregistrari
select * from show;                --11 inregistrari
select * from collaboration;       --10 inregistrari
select * from reservation;         --16 inregistrari

delete from type
where type_id = 4;

select * from type;                --5 inregistrari
select * from show;                --4 inregistrari
select * from collaboration;       --6 inregistrari
select * from reservation;         --8 inregistrari
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 4 in 0.003 seconds

	SHOW_ID	SHOW_TITLE	DURATION	RATING	LAUNCH_YEAR	PRICE	TYPE_ID	PROD_ID
1	11	Ursul	3	3.5	2010	20	2	3
2	12	Necasatoria	2	4	2010	15	5	2
3	13	Cursa de soareci	2	2	1999	33	6	3
4	17	Meciul de comedie	2	4	2011	17	2	2

```sql
--TESTARE
select * from type;                --6 inregistrari
select * from show;                --11 inregistrari
select * from collaboration;       --10 inregistrari
select * from reservation;         --16 inregistrari

delete from type
where type_id = 4;

select * from type;                --5 inregistrari
select * from show;                --4 inregistrari
select * from collaboration;       --6 inregistrari
select * from reservation;         --8 inregistrari
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 6 in 0.003 seconds

	COLLABORATION_ID	SHOW_ID	THEATER_ID	ACTOR_ID
1	104	11	30	30
2	105	12	40	30
3	106	12	40	30
4	107	12	60	40
5	108	12	50	40
6	110	17	10	20

```
/

--TESTARE
select * from type;                --6 inregistrari
select * from show;                --11 inregistrari
select * from collaboration;       --10 inregistrari
select * from reservation;         --16 inregistrari

delete from type
where type_id = 4;

select * from type;                --5 inregistrari
select * from show;                --4 inregistrari
select * from collaboration;       --6 inregistrari
select * from reservation;         --8 inregistrari
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 8 in 0.002 seconds

	RESERVATION_ID	SHOW_ID	CLIENT_ID	BOOKING_DATE
1	102	11	10	13-OCT-21
2	103	12	10	25-MAR-21
3	105	11	70	28-AUG-21
4	106	12	70	11-JAN-21
5	107	13	70	14-AUG-21
6	108	17	20	07-DEC-21
7	109	17	20	12-AUG-21
8	111	13	20	14-AUG-21

```
rollback;
drop trigger trigger_ex_11;
```

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

Trigger care introduce în tabelul 'database_history' date legate de acțiunile realizate în baza de date (create/drop/alter).

```
create table database_history(
    user_name   varchar2(50),
    db_name     varchar2(50),
    event       varchar2(25),
    object_name varchar2(30),
    data        date,
    ora         varchar2(30)
);

--REZOLVARE
/
create or replace trigger trigger_ex_12
    AFTER create or drop or alter on schema
begin
    insert into database_history
         values  (SYS.LOGIN_USER,SYS.DATABASE_NAME,  SYS.SYSEVENT,
SYS.DICTIONARY_OBJ_NAME,    SYSDATE,    TO_CHAR(SYSDATE,    'HH24')
||':'||to_char(sysdate, 'MI'));
end;
/
```

```oracle
--TESTARE
--cream un tabel pentru a testa trigger-ul
create table test_table(
    numar   number
);

--adaugam o coloana in plus
alter table test_table
add cifra number(1);

--stergem tabelul
drop table test_table;

--verificam datele stranse in 'database_history'
select * from database_history;

--stergem trigger-ul 'trig_test' si tabela 'database_history'
drop trigger trigger_ex_12;
drop table database_history;
```

Worksheet | Query Builder

```oracle
begin
    insert into database_history
    values (SYS.LOGIN_USER,SYS.DATABASE_NAME, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME,
end;
/

--TESTARE
--cream un tabel pentru a testa trigger-ul
create table test_table(
    numar   number
);

--adaugam o coloana in plus
alter table test_table
add cifra number(1);

--stergem tabelul
drop table test_table;

--verificam datele stranse in 'database_history'
select * from database_history;

--stergem trigger-ul 'trig_test' si tabela 'database_history'
drop trigger trigger_ex_12;
drop table database_history;
```

Script Output | Query Result

SQL | All Rows Fetched: 3 in 0.004 seconds

	USER_NAME	DB_NAME	EVENT	OBJECT_NAME	DATA	ORA
1	STEF	XE	CREATE	TEST_TABLE	07-JAN-22	23:20
2	STEF	XE	ALTER	TEST_TABLE	07-JAN-22	23:20
3	STEF	XE	DROP	TEST_TABLE	07-JAN-22	23:20

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

```
/
CREATE OR REPLACE PACKAGE pachet_ex_13 AS
    PROCEDURE ex_6 (comanda NUMBER);

    PROCEDURE ex_7;

    TYPE rezultat_ex8 IS TABLE OF NUMBER(4);
       FUNCTION  ex_8  (v_title  show.show_title%TYPE)  RETURN
rezultat_ex8;

    PROCEDURE ex_9 (v_nume_teatru theater.theater_name%TYPE);
END pachet_ex_13;
/
CREATE OR REPLACE PACKAGE BODY pachet_ex_13 AS
    PROCEDURE ex_6 (
             comanda  NUMBER   --parametrul  care  trebuie  sa
primeasca valoari de la 1 la 2
    )
     IS

        --cursor dinamic
         --il vom folosi pentru a lua date ori din 'card' ori
din 'location'
        --in functie de valoarea parametrului 'comanda'
        TYPE cursor_date IS REF CURSOR;
        c_date       cursor_date;

              TYPE  t_indexat  IS  TABLE  OF  NUMBER  INDEX  BY
PLS_INTEGER;
        TYPE t_imbricat IS TABLE OF VARCHAR2(50);

            --tablou  indexat  pentru  numarul  de  aparitii  al
datelor (brand/city)
        v_aparitii   t_indexat;
        --tablou imbricat - salvam datele o singura data
        v_date       t_imbricat := t_imbricat();

         text          VARCHAR2(50);           --variabila va
primi valorile din cursor
           aux            VARCHAR2(50)  := 'a';    --vom salva
valorile din 'v_date' o singura data
         i             NUMBER := 0;           --contor pentru
numarul de valori din tabele
```

```
              parametru_invalid    EXCEPTION;     --exceptie pentru
parametru diferit de 1 si 2
         BEGIN
             --pentru valoarea 1 cursorul contine brandurile
             IF comanda = 1 THEN
                 OPEN c_date FOR SELECT brand
                                 FROM card
                                 ORDER BY brand;

             --pentru valoarea 2 cursorul contine orasele
             ELSIF comanda = 2 THEN
                 OPEN c_date FOR SELECT city
                                 FROM location
                                 ORDER BY city;
             ELSE
                 RAISE parametru_invalid;
             END IF;

             LOOP
                 FETCH c_date INTO text;
                  EXIT WHEN c_date%notfound; --parasim loop-ul cand
nu mai avem date in cursor

                     IF aux != text THEN      --daca am mai avut
respectiva informatie (brand/city)
                      aux := text;        --vom tine minte valoarea
prin care trecem

                     i := i + 1;         --crestem contorul cu 1
                     v_date.extend;      --marim tabloul imbricat
                     v_date(i) := text;  --adaugam valoarea gasita

                          v_aparitii(i) := 1; --valoarea gasita a
aparut o data

                   ELSIF aux = text THEN   --daca intalnim din noua
aceeasi valoare ii crestem nr de aparitii
                     v_aparitii(i) := v_aparitii(i) + 1;
                 END IF;
             END LOOP;

             CLOSE c_date;   --inchidem cursorul

             --afisam datele din cele doua tablouri
             FOR j IN 1..i LOOP
                dbms_output.put_line(v_date(j) || ' apare de ' ||
v_aparitii(j) || ' ori');
             END LOOP;
```

```
        EXCEPTION
            WHEN parametru_invalid THEN
                      raise_application_error(-20001,  'Parametru
invalid. Introduceti ca parametru "1" sau "2"');
        END ex_6;

        PROCEDURE ex_7 IS
                 --cursor  ce  contine  cate  spectacole  fac  parte
dintr-un gen
            CURSOR cursor_type IS(
                            SELECT  t.type_id,  MAX(t.type_name),
COUNT(s.show_id)
                FROM type t, show s
                WHERE t.type_id = s.type_id (+)
                GROUP BY t.type_id
            );

             --cursor cu numele spectacolelor, genul lor si pretul
unui bilet
            CURSOR cursor_show IS
                  SELECT  ss.show_id,  ss.show_title,  ss.type_id,
ss.price * m.n total_price --calcularea incasarilor per spectacol
            FROM show ss,
                     --calculam de cate ori s-a rezervat bilet pt
fiecare spectacol
                (SELECT s.show_id idul, (SELECT COUNT(r.show_id)
                                         FROM reservation r
                                                 WHERE  r.show_id =
s.show_id) n
                FROM show s) m
            WHERE ss.show_id = m.idul
            ORDER BY total_price DESC;  --ordonam dupa incasari

            --variabile pentru datele din cursorul 'cursor_type'
            v_t_id        type.type_id%TYPE;
            v_t_name      type.type_name%TYPE;
            v_t_nr        show.show_id%TYPE;

            --variabile pentru datele din cursorul 'cursor_show'
            v_s_id        show.show_id%TYPE;
            v_s_title     show.show_title%TYPE;
            v_s_type_id   show.type_id%TYPE;
            v_s_price     NUMBER;
             total          NUMBER;    --variabila pentru calculul
incasarilor per gen
        BEGIN
            --cursoare imbricate
```

```
                    OPEN  cursor_type;      --deschidem  cursorul
'cursor_type'

             LOOP
                 FETCH cursor_type
                       INTO  v_t_id,  v_t_name,  v_t_nr;   --salvam
informatiile in variabile

                      EXIT  WHEN  cursor_type%notfound; --parcurgem
intreg cursorul de genuri
                                                          --si pentru
fiecare gen afisam informatiile stocate in
                                                           --cursorul
'cursor_show'

                   dbms_output.put_line('Genul spectacolului: ' ||
v_t_name);

dbms_output.put_line('============================');

                 total := 0; --suma este initial zero

                 IF v_t_nr = 0 THEN
                     --daca nu avem spectacole dintr-un gen afisam
un mesaj corespunzator
                           dbms_output.put_line('Nu  avem  filme  la
această categorie.');
                 ELSE
                         OPEN cursor_show;    --deschidem cursorul
'cursor_show'

                     LOOP
                         FETCH cursor_show
                              INTO v_s_id, v_s_title, v_s_type_id,
v_s_price; --salvam informatiile in variabile
                                   EXIT  WHEN  cursor_show%notfound;
-- parcurgem intreg cursorul cu angajati

                              --afisam informatiile corespunzatoare
genului la care am ajuns
                         IF v_t_id = v_s_type_id THEN
                              dbms_output.put_line(v_s_title || ' '
|| v_s_price || ' RON');
                                total := total + v_s_price; --aduna
incasarile totale ale genului
                         END IF;
                     END LOOP;
```

```
                        CLOSE cursor_show;   --inchidem cursorul
'cursor_show'
                END IF;

                --afisam incasarile totale ale genului
                dbms_output.put_line('----------');
                      dbms_output.put_line('Total incasari: ' ||
total);
                dbms_output.new_line;
            END LOOP;

            CLOSE cursor_type;  --inchidem cursorul 'cursor_type'
        END ex_7;

        FUNCTION ex_8 (
                 v_title  show.show_title%TYPE      --primeste ca
parametru numele spectacolului
        )
           RETURN rezultat_ex8 --returneaza un rezultat de tipul
'rezultat_ex8'
         IS
            v_rezultat rezultat_ex8:= rezultat_ex8();

               --cursor ce contine data rezervarii biletelor la
spectacol si varsta spectatorilor
            CURSOR c IS
                    SELECT r.booking_date, round((booking_date -
birth_date) / 365.25, 0)
                FROM show s, reservation r, client c
                WHERE s.show_id = r.show_id
                    AND c.client_id = r.client_id
                    AND s.show_id = (SELECT show_id
                                     FROM show
                                            WHERE upper(v_title) =
upper(show_title))       --gasim  id-ul  spectacolului  primit  ca
parametru
                 ORDER BY booking_date;  --ordonam in functie de
data rezervarii

               --vector ce va pastra nr de bilete pentru fiecare
luna
            TYPE t_aparitii IS VARRAY(12) OF NUMBER(6);

            --variabile pentru accesarea datelor din cursor
                   v_booking       reservation.booking_date%TYPE;
--memoreaza data achizitionarii unui bilet
                 v_age           NUMBER; --memoreaza varsta unui
participant
```

```
            --vector initializat cu 0 bilete pentru fiecare luna
             v_aparitii   t_aparitii := t_aparitii(0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0);

               maxim          NUMBER := -1; --maximul de bilete
vandute
              c_luna         NUMBER := 0;  --numarul de luni cu
acelasi maxim
            media        NUMBER := 0;  --suma varstelor
            cc           NUMBER := 0;  --cate persoane avem
            i            NUMBER;
            k            NUMBER;

              --variabila ce va verifica daca spectacolul a avut
sau nu bilete vandute
            if_record    BOOLEAN := false;
        BEGIN
            --tratare exceptie NO_DATA_FOUND
            SELECT COUNT(1)
            INTO k
            FROM show
            WHERE upper(v_title) = upper(show_title);

            IF k <= 0 THEN
                RAISE no_data_found;
            END IF;

            OPEN c; --deschidem cursorul

            LOOP
                 FETCH c INTO v_booking, v_age;  --pastram datele
de pe o linie in doua variabile
                 EXIT WHEN c%notfound;           --ne oprim cand
parcurgem tot cursorul

                 if_record := true;  --nu s-a inchis cursorul =>
inseamna ca avem date (s-au vandut bilete)

                      i  :=  to_number(to_char(extract(MONTH  FROM
v_booking))); --extragem luna din data rezervarii
                          v_aparitii(i)  :=  v_aparitii(i)  +  1;
--crestem nr de bilete vandute in luna respectiva

                --cautam maximul de bilete vandu-te intr-o luna
                IF ( maxim < v_aparitii(i) ) THEN
                    maxim := v_aparitii(i);
                END IF;
```

```
                    --calculam nr de persoane care au fost la
spectacol si suma varstelor
                IF ( v_age IS NOT NULL ) THEN
                    cc := cc + 1;
                    media := media + to_number(v_age);
                END IF;
            END LOOP;

            CLOSE c;    --inchidem cursorul

              --adaugam fiecare luna care are nr maxim de bilete
vandute
            --in lista 'f_luna'
            FOR j IN v_aparitii.first..v_aparitii.last LOOP
                IF ( v_aparitii(j) = maxim ) THEN
                    v_rezultat.extend;
                    c_luna := c_luna + 1;
                    v_rezultat(c_luna) := j;
                END IF;
            END LOOP;

               --returnam lista cu lunile cu nr maxim de bilete
vandute si
            --media varstei spectatorilor
            --(asta daca au existat bilete vandute la spectacol)
            IF if_record = false THEN
                RETURN rezultat_ex8(0, 0);
            END IF;

            v_rezultat.extend;
            c_luna := c_luna + 1;
            v_rezultat(c_luna) := round(media / cc, 2);

            RETURN v_rezultat;

        EXCEPTION
            WHEN no_data_found THEN
                      raise_application_error(-20001, 'Nu exista
spectacol cu titlul dat.');
            WHEN OTHERS THEN
                raise_application_error(-20002, 'Alta eroare!');
        END ex_8;

        PROCEDURE ex_9 (
             v_nume_teatru theater.theater_name%TYPE --primeste ca
parametru numele teatrului
        )
```

```
          IS

                --cursor ce contine id-ul producatorilor si id-ul
tipului
             --spectacolelor ce se joaca la teatrul citit
             CURSOR c_1 IS
                 SELECT p.prod_id   prod, t.type_id   typess
                        FROM theater t, collaboration c, show s,
producers p, type t
                 WHERE c.theater_id = t.theater_id
                     AND c.show_id = s.show_id
                     AND s.prod_id = p.prod_id
                     AND t.theater_id = (SELECT theater_id
                                         FROM theater
                                          WHERE upper(theater_name)
= upper(v_nume_teatru)
                                                )   --obtinem id-ul
spectacolului citit
                     AND s.type_id = t.type_id
                 ORDER BY
                         s.show_id; --ordonam credcator dupa id-ul
spectacolului

               --cursor ce obtine id-ul, directorul si scenaristul
unei echipe de producatori
                 --impreuna cu numarul de spectacole pe care le
desfasoara de tipul 'c_specific'
              --tipul de specatcole pe care trebuie sa le numere le
primeste prin intermediul
             --unui parametru
             CURSOR c_2 (c_specific NUMBER) IS
                   SELECT p.prod_id id2, MAX(p.director) director,
MAX(p.screenwriter) screenw, COUNT(type_id) nr
                 FROM producers   p, show        s
                     WHERE p.prod_id = s.prod_id AND s.type_id =
c_specific
                 GROUP BY p.prod_id;

             TYPE tablou_imbricat IS TABLE OF NUMBER(4);

             --tablou cu id-urile producatorilor
             t_prod          tablou_imbricat := tablou_imbricat();
             --tablou cu id-urile genurilor de spectacole
             t_type          tablou_imbricat := tablou_imbricat();
                 --tablou pentru a salva numarul de aparitii al
valorilor din 't_prod'
             apar_p          tablou_imbricat := tablou_imbricat();
```

```
                --tablou pentru a salva numarul de aparitii al
valorilor din 't_type'
            apar_t          tablou_imbricat := tablou_imbricat();

             max_prod        NUMBER := -1;   --salvam id-ul maxim
de producatori pentru a extinde lista 'apar_p'
             max_type        NUMBER := -1;   --salvam id-ul maxim
de tipuri pentru a extinde lista 'apar_t'
              maxim             NUMBER := -1;    --salvam numarul
maxim de spectacole la care avem aceeasi producatori in teatrul
citit
               specific          NUMBER := -1;    --salvam id-ul
tipului de spectacole care se gasesc cel mai mult la teatrul citit
            specific_m      NUMBER := -1;   --
                specific_nume    VARCHAR2(50);    --salvam numele
tipului in functie de id-ul gasit
            k               NUMBER := 0;

             nimic            BOOLEAN := FALSE;   --verifica daca
avem sau nu echipe de profucatori

             if_found_rec    BOOLEAN := FALSE;   --verifica daca
teatrul introdus are sau nu spectacole
              not_found_rec   EXCEPTION;           --exceptie in
cazul in care teatrul nu are spectacole
        BEGIN
              --vedem de cate ori apare numele teatrului citit in
baza de date
            SELECT COUNT(1)
            INTO k
            FROM theater
            WHERE upper(theater_name) = upper(v_nume_teatru);

            IF k <= 0 THEN
                     RAISE  no_data_found;     --tratare exceptie
NO_DATA_FOUND
            ELSIF k > 1 THEN
                     RAISE  too_many_rows;     --tratare exceptie
TOO_MANY_ROWS
            END IF;

            k := 0;  --numaram cate elemente avem in cursorul c_1
            FOR i IN c_1
            LOOP
                if_found_rec := TRUE;   --daca avem spectacole :=
TRUE
                k := k + 1;
                t_prod.extend;          --extindem tabloul
```

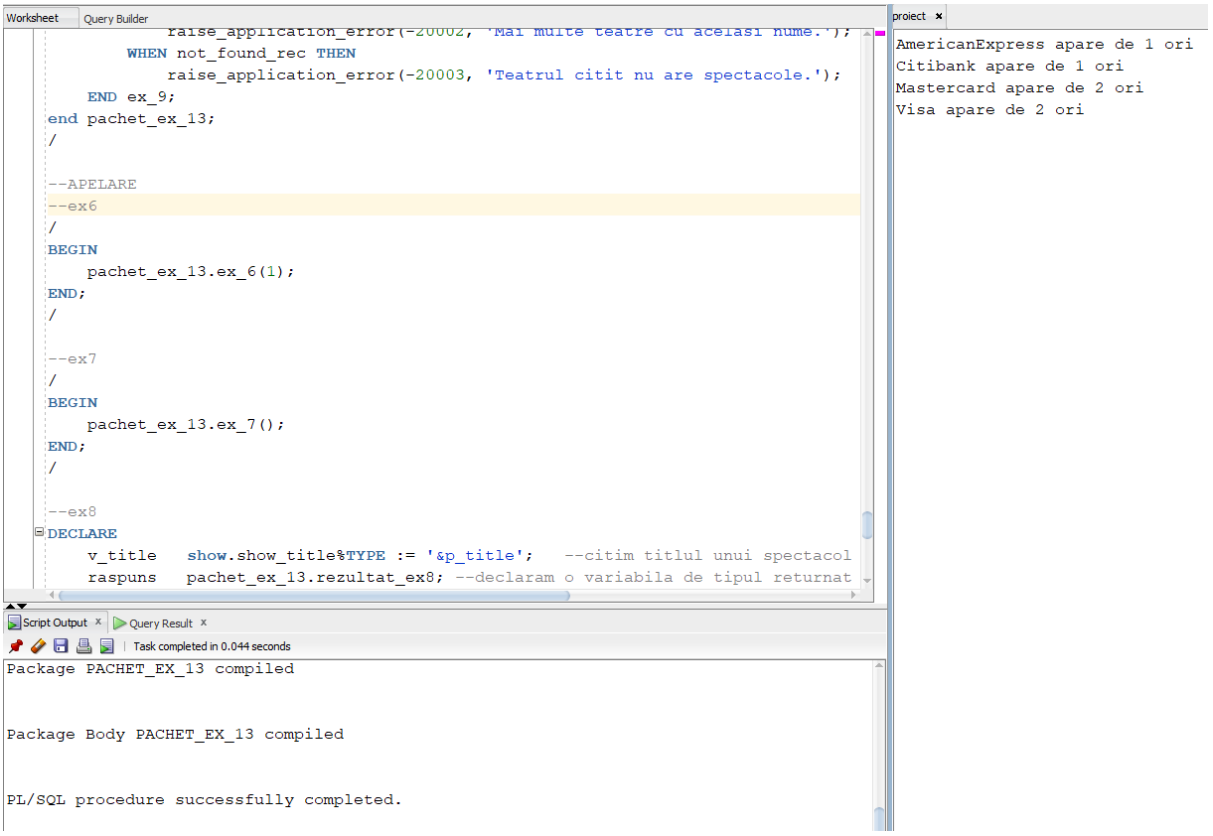
```
                  t_prod(k) := i.prod;     --adaugam elementul din
cursor

                --salvam id-ul maxim de producator
                IF ( max_prod < i.prod ) THEN
                    max_prod := i.prod;
                END IF;

                t_type.extend;          --extindem tabloul
                  t_type(k) := i.typess;  --adaugam elementul din
cursor

                --salvam id-ul maxim de tip
                IF ( max_type < i.typess ) THEN
                    max_type := i.typess;
                END IF;

            END LOOP;

            --daca nu am avut date/spectacole atunci exceptie
            IF ( if_found_rec = FALSE ) THEN
                RAISE not_found_rec;
            END IF;

            --initializam tabloul de aparitii 'apar_p' cu zero
            FOR i IN 1..max_prod LOOP
                apar_p.extend;
                apar_p(i) := 0;
            END LOOP;

            --initializam tabloul de aparitii 'apar_t' cu zero
            FOR i IN 1..max_type LOOP
                apar_t.extend;
                apar_t(i) := 0;
            END LOOP;

             --populam tabloul de aparitii 'apar_p' in functie de
datele din 't_prod'
            FOR i IN t_prod.first..t_prod.last LOOP
                apar_p(t_prod(i)) := apar_p(t_prod(i)) + 1;

                IF ( maxim < apar_p(t_prod(i)) ) THEN
                    maxim := apar_p(t_prod(i));
                END IF;
            END LOOP;

             --populam tabloul de aparitii 'apar_t' in functie de
datele din 't_type'
```

```
              FOR i IN t_type.first..t_type.last LOOP
                  apar_t(t_type(i)) := apar_t(t_type(i)) + 1;

                  IF ( specific_m < apar_t(t_type(i)) ) THEN
                      specific_m := apar_t(t_type(i));
                      specific := t_type(i);
                  END IF;
              END LOOP;

              --salvam in 'specific_nume' denumirea tipului gasit
              SELECT type_name
              INTO specific_nume
              FROM type
              WHERE type_id = specific;

               --afisam ce valoare are 'specific' si ce valoare are
'maxim'
               dbms_output.put_line('Teatrul "' || v_nume_teatru ||
'" are:');
                       dbms_output.put_line('-  specificul:  '  ||
specific_nume || ';');
                dbms_output.put_line('- maximul de spectacole ale
acelorasi producatori ' || maxim || ';');
               dbms_output.put_line('Lista producatorilor ' || 'cu
mai mult de '
                                    || maxim || ' spectacole '
                                    || specific_nume || ':');

               --parcurgem cursorul 'c_2' pentru a obtine tabloul
producatorilor care
                  --au mai mult de 'maxim' spectacole de tipul
'specific' gasit
              FOR i IN c_2(specific) LOOP
                  IF ( i.nr >= maxim ) THEN
                          nimic := TRUE;  --devine true daca avem
echipe de producatori
                       dbms_output.put_line(i.director || ' si ' ||
i.screenw);
                  END IF;
              END LOOP;

                    --daca nu avem echipe de producatori afisam
mesajul...
              IF ( nimic = FALSE ) THEN
                          dbms_output.put_line('Nu avem echipe de
producatori.');
              END IF;
```

```
        EXCEPTION
            WHEN no_data_found THEN
                raise_application_error(-20001, 'Nu exista teatru
cu titlul dat.');
            WHEN too_many_rows THEN
                raise_application_error(-20002, 'Mai multe teatre
cu acelasi nume.');
            WHEN not_found_rec THEN
                raise_application_error(-20003, 'Teatrul citit nu
are spectacole.');
        END ex_9;
    end pachet_ex_13;
    /
```

Apelare

```
--ex6
/
BEGIN
    pachet_ex_13.ex_6(1);
END;
/
```



```
--ex7
/
BEGIN
    pachet_ex_13.ex_7();
END;
/
```

Worksheet | Query Builder

```
            raise_application_error(-20002, 'Mai multe teatre cu acelasi nume.');
        WHEN not_found_rec THEN
            raise_application_error(-20003, 'Teatrul citit nu are spectacole.');
    END ex_9;
end pachet_ex_13;
/

--APELARE
--ex6
/
BEGIN
    pachet_ex_13.ex_6(1);
END;
/

--ex7
/
BEGIN
    pachet_ex_13.ex_7();
END;
/

--ex8
DECLARE
    v_title    show.show_title%TYPE := '&p_title';    --citim titlul unui spectacol
    raspuns    pachet_ex_13.rezultat_ex8; --declaram o variabila de tipul returnat
```

Script Output | Query Result

Task completed in 0.055 seconds

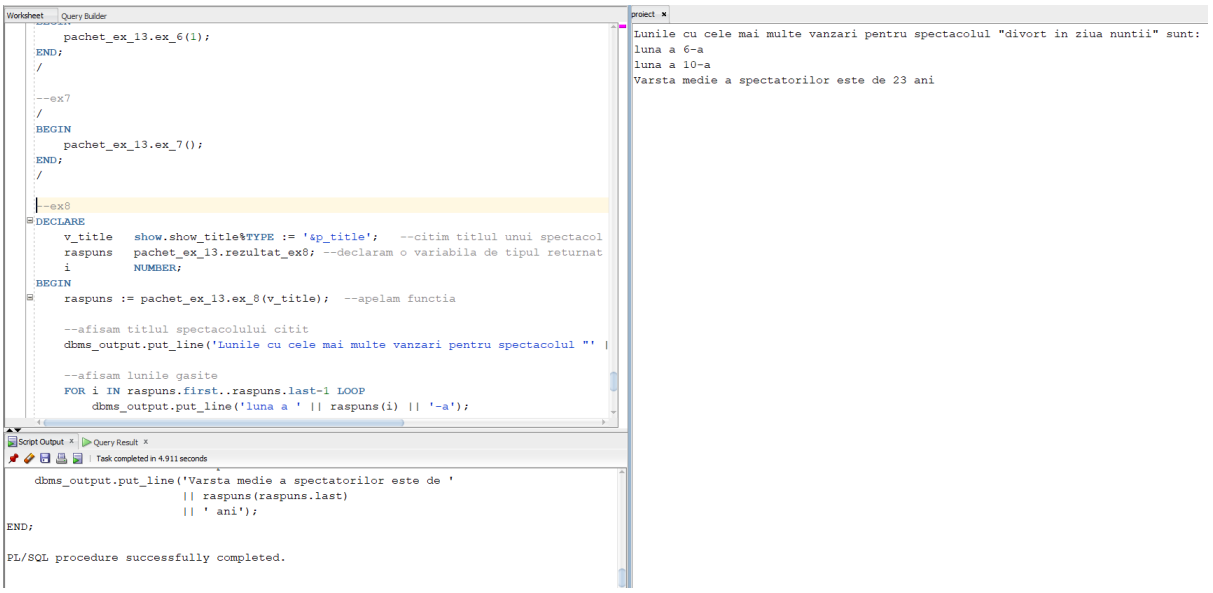
```
PL/SQL procedure successfully completed.
```

proiect

```
Genul spectacolului: absurd
============================
Nu avem filme la această categorie.
----------
Total incasari: 0

Genul spectacolului: grotesque
============================
Ursul 40 RON
Meciul de comedie 34 RON
----------
Total incasari: 74

Genul spectacolului: satire
============================
Nu avem filme la această categorie.
----------
Total incasari: 0

Genul spectacolului: comedy
============================
Divort in ziua nuntii 400 RON
Salut 0 RON
Avarul 0 RON
Colectionarul 0 RON
Inca una si ma duc 0 RON
O noapte furtunoasa 0 RON
Doi pe o banca 0 RON
----------
Total incasari: 400

Genul spectacolului: tragic
============================
Necasatoria 30 RON
----------
Total incasari: 30

Genul spectacolului: drama
```

```
--ex8
DECLARE
     v_title    show.show_title%TYPE := '&p_title';    --citim titlul unui spectacol
          raspuns      pachet_ex_13.rezultat_ex8;  --declaram o variabila de tipul returnat de functie 'rezultat_ex8'
     i          NUMBER;
BEGIN
     raspuns := pachet_ex_13.ex_8(v_title);  --apelam functia

     --afisam titlul spectacolului citit
      dbms_output.put_line('Lunile cu cele mai multe vanzari pentru spectacolul "' || v_title || '" sunt:');

     --afisam lunile gasite
     FOR i IN raspuns.first..raspuns.last-1 LOOP
             dbms_output.put_line('luna a ' || raspuns(i) || '-a');
     END LOOP;

     --afisam media varstelor spectatorilor
      dbms_output.put_line('Varsta medie a spectatorilor este de '
                                || raspuns(raspuns.last)
                                || ' ani');
END;
/
```

Citire **Divort in ziua nuntii**



```
--ex9
/
DECLARE
    nume theater.theater_name%TYPE:='&p_nume';
BEGIN
    pachet_ex_13.ex_9(nume);
--    Teatrul Judetean Brasov
--    Teatrul Nottara
--    Teatrul Artcub
--    asd
--    Teatrul Guguta
--    Verde
END;
/
```

Citire **Teatrul Judetean Brasov**

```
    --afisam lunile gasite
    FOR i IN raspuns.first..raspuns.last-1 LOOP
        dbms_output.put_line('luna a ' || raspuns(i) || '-a');
    END LOOP;

    --afisam media varstelor spectatorilor
    dbms_output.put_line('Varsta medie a spectatorilor este de '
                         || raspuns(raspuns.last)
                         || ' ani');
END;
/

--ex9
/
DECLARE
    nume theater.theater_name%TYPE:='&p_nume';
BEGIN
    pachet_ex_13.ex_9(nume);
--    Teatrul Judetean Brasov
--    Teatrul Nottara
--    Teatrul Artcub
--    asd
--    Teatrul Guguta
--    Verde
END;
/
```

Script Output | Query Result | Task completed in 8.29 seconds

```
--    Teatrul Artcub
--    asd
--    Teatrul Guguta
--    Verde
END;

PL/SQL procedure successfully completed.
```

```
Teatrul "teatrul judetean brasov" are:
- specificul: comedy;
- maximul de spectacole ale acelorasi producatori 2;
Lista producatorilor cu mai mult de 2 spectacole comedy:
Sica Alexandrescu si Vlad Musteriu
Mihai Bendeac si Alexandra Matei
```

EXERCIȚIUL 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).

Pentru apelarea functiei 'procedura_inceput' din pachetul 'pachet_ex_14' cu parametrul unui producator se va afisa numele producatorilor urmat de categoriile de spectacole unde filmele lor se afla in top 3 cel mai scump bilet. Fiecare categorie este urmata de suma, in total, al preturilor unui bilet de la spectacole.

```
/
CREATE OR REPLACE PACKAGE pachet_ex_14 AS
    --tablou imbricat
    TYPE tablou_imbricat IS TABLE OF NUMBER(4);

    --tablou imbricat
    TYPE tablou_nume IS TABLE OF VARCHAR2(50);

    PROCEDURE procedura_inceput (id_producatori number);

    function functie_apare(
        lista tablou_imbricat,  --topul
        producatori number      --id-ul producatorilor
    ) return boolean;

    function functie_suma(
        lista tablou_imbricat
    ) return number;

    procedure procedura_afisare(
        producatori number,
        lista_nume tablou_nume,
        lista_sume tablou_imbricat
    );
END pachet_ex_14;
/
CREATE OR REPLACE PACKAGE BODY pachet_ex_14 AS
    --procedura ce gaseste primele 2 cela mai scumpe spectacole,
    --din fiecare categorie;
        --functia  primeste  ca  parametru  id-ul  unei  echipe  de
producatori
    PROCEDURE procedura_inceput (
        id_producatori number
    )
    IS
        --cursor cu tipurile de spectacole
        CURSOR cursor_type IS(
            SELECT t.type_id, MAX(t.type_name), COUNT(s.show_id)
            FROM type t, show s
```

```
            WHERE t.type_id = s.type_id (+)
            GROUP BY t.type_id
        );

        --cursor cu spectacolele
        CURSOR cursor_show IS
        SELECT show_id, type_id, price
        FROM show
        ORDER BY price desc;

        --variabile pentru datele din cursorul 'cursor_type'
        v_t_id        type.type_id%TYPE;
        v_t_name      type.type_name%TYPE;
        v_t_nr        show.show_id%TYPE;

        --variabile pentru datele din cursorul 'cursor_show'
        v_s_id        show.show_id%TYPE;
        v_s_type_id   show.type_id%TYPE;
        v_s_price     NUMBER;

        --tablou cu specatcolele din fiecare top
        lista       tablou_imbricat := tablou_imbricat();
        --tablou cu sumele fiecarui top
        lista_sume  tablou_imbricat := tablou_imbricat();
        --tablou cu denumirea categoriei fiecarui top
        lista_nume  tablou_nume := tablou_nume();

         i number;        --verifica ca fiecare top sa fie de maxim
2 valori
           k number:=0;     --nr de elemente din 'lista_sume' si
'lista_nume'
        nr number;      --salveaza suma din fiecare categorie

              verificare number:=0;     --variabila pt verificare
no_data_found

    BEGIN
        --verificare no_data_found
        SELECT COUNT(1)
        INTO verificare
        FROM producers
        WHERE prod_id = id_producatori;

        IF verificare <= 0 THEN
            RAISE no_data_found;
        END IF;

        OPEN cursor_type;   --deschidere cursor 'cursor_type'
```

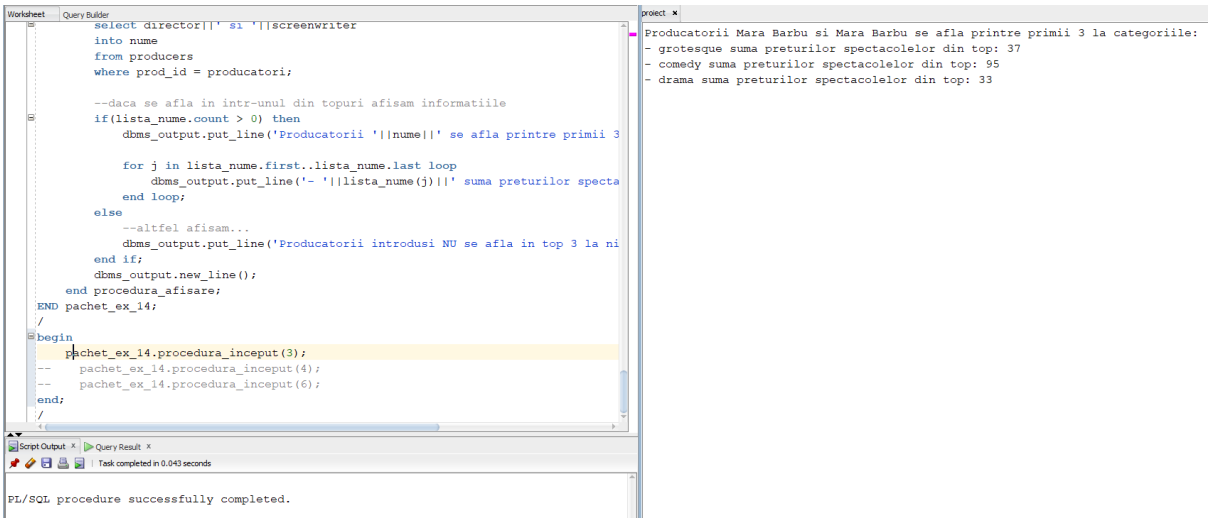
```
        LOOP
            FETCH cursor_type INTO
                v_t_id,
                v_t_name,
                v_t_nr;
            EXIT WHEN cursor_type%notfound;

            IF v_t_nr >= 1 THEN --daca avem topuri cream un top
                i:=0;   --zero elemente in top
                        OPEN cursor_show;    --deschidem cursorul
'cursor_show'

                LOOP
                    FETCH cursor_show
                      INTO v_s_id, v_s_type_id, v_s_price; --salvam
informatiile in variabile
                          EXIT WHEN cursor_show%notfound;       --
parcurgem intreg cursorul cu angajati

                      --salvam informatiile corespunzatoare genului
la care am ajuns
                    --in 'lista'
                    IF v_t_id = v_s_type_id and i<2 THEN
                        i:=i+1;
                        lista.extend();
                        lista(i) := v_s_id;
                    END IF;
                END LOOP;

                          CLOSE cursor_show;   --inchidem cursorul
'cursor_show'

                --daca spectacolele producatorilor sunt in top

if(pachet_ex_14.functie_apare(lista,id_producatori) = true) then
                    --aflam suma si o salvam in 'lista_sume'
                    --salvam numele categoriei de spectacol
                    nr:= pachet_ex_14.functie_suma(lista);
                    k:=k+1;

                    lista_nume.extend;
                    lista_nume(k) := v_t_name;

                    lista_sume.extend;
                    lista_sume(k) := nr;
                end if;
                lista.delete;   --golim tabloul pt urmatorul top
            END IF;
```

```
        END LOOP;
        CLOSE cursor_type;  --inchidem cursorul 'cursor_type'

        --afisam
         pachet_ex_14.procedura_afisare(id_producatori, lista_nume,
lista_sume);

    EXCEPTION
        when no_data_found then
             raise_application_error(-20001, 'Nu exista producatori
cu id-ul dat.');
    END procedura_inceput;


    --functia verifica daca producatorii introdusi se afla in
    --unul dintre topuri
    function functie_apare(
        lista tablou_imbricat,  --topul
        producatori number      --id-ul producatorilor
    )
    return boolean
    is
    begin
          --cautam id-ul producatorilor in cursorul cu spectacole
din top
        for j in lista.first..lista.last loop
            for i in (select show_id sh
                        from show
                        where prod_id = producatori
                        ) loop
                --daca gasim o aparitie returnam 'true'
                if(lista(j) = i.sh) then
                    return true;
                end if;
            end loop;
        end loop;

        --daca nu gasim => 'false'
        return false;
    end functie_apare;

     --functia returneaza suma preturilor biletelor pentru fiecare
spectacol
    function functie_suma(
        lista tablou_imbricat
    )
    return number
    is
```

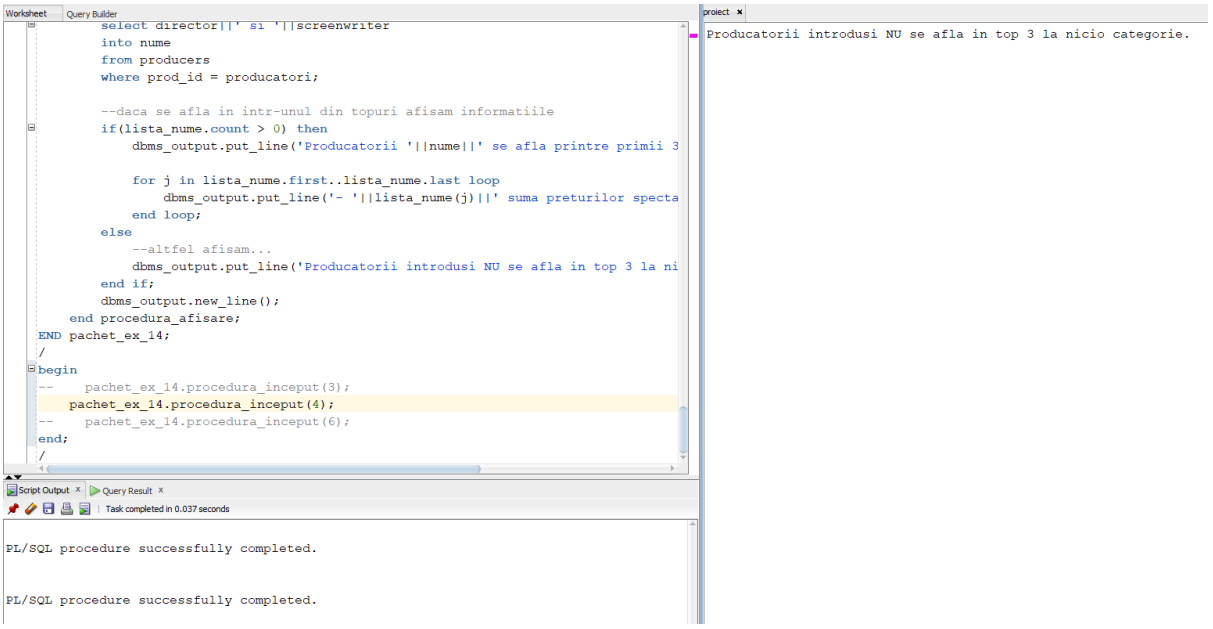
```
        suma number:=0; --se calculeaza suma

        nr number:=0;   --se salveaza pretul
    begin
        for j in lista.first..lista.last loop
            select price
            into nr
            from show
            where show_id = lista(j);

            suma:=suma+nr;
        end loop;
        return suma;--returnam suma

    end functie_suma;

    --procedura pentru afisarea informatiilor
    procedure procedura_afisare(
        producatori number,
        lista_nume tablou_nume,
        lista_sume tablou_imbricat
    )
    is
        nume varchar2(200); --numele producatorilor

    begin
        --gasim nume producatorilor dupa is
        select director||' si '||screenwriter
        into nume
        from producers
        where prod_id = producatori;

        --daca se afla in intr-unul din topuri afisam informatiile
        if(lista_nume.count > 0) then
             dbms_output.put_line('Producatorii '||nume||' se afla
printre primii 3 la categoriile: ' );

            for j in lista_nume.first..lista_nume.last loop
                  dbms_output.put_line('- '||lista_nume(j)||' suma
preturilor spectacolelor din top: '||lista_sume(j));
            end loop;
        else
            --altfel afisam...
                dbms_output.put_line('Producatorii introdusi NU se
afla in top 3 la nicio categorie. ');
        end if;
        dbms_output.new_line();
    end procedura_afisare;
```

```
END pachet_ex_14;
-APELARE
/
begin
      pachet_ex_14.procedura_inceput(3);
--    pachet_ex_14.procedura_inceput(4);
--    pachet_ex_14.procedura_inceput(6);

end;
/
```

Apelare `pachet_ex_14.procedura_inceput(3);`



Apelare `pachet_ex_14.procedura_inceput(4);`



Apelare `pachet_ex_14.procedura_inceput(6);` (id-ul 6 nu exista)

Worksheet | Query Builder | proiect

```
        select director||' si '||screenwriter
        into nume
        from producers
        where prod_id = producatori;

        --daca se afla in intr-unul din topuri afisam informatiile
        if(lista_nume.count > 0) then
            dbms_output.put_line('Producatorii '||nume||' se afla printre primii 3

            for j in lista_nume.first..lista_nume.last loop
                dbms_output.put_line('- '||lista_nume(j)||' suma preturilor specta
            end loop;
        else
            --altfel afisam...
            dbms_output.put_line('Producatorii introdusi NU se afla in top 3 la ni
        end if;
        dbms_output.new_line();
    end procedura_afisare;
END pachet_ex_14;
/
begin
--    pachet_ex_14.procedura_inceput(3);
--    pachet_ex_14.procedura_inceput(4);
    pachet_ex_14.procedura_inceput(6);
end;
/
```

Script Output | Query Result | Task completed in 0.038 seconds

```
--    pachet_ex_14.procedura_inceput(4);
    pachet_ex_14.procedura_inceput(6);
end;
Error report -
ORA-20001: Nu exista producatori cu id-ul dat.
ORA-06512: at "STEF.PACHET_EX_14", line 108
ORA-06512: at line 4
```