Bază de date pentru Netflix

Prof. Laborator: AnaMaria Ștefania Zănoagă

Prof. Curs: Gabriela Mihai

Mincu Adrian-Lucian

grupa 232

Cuprins

Contents

[Descriere model 3](#_Toc155809487)

[Constrângeri 3](#_Toc155809488)

[Descriere entități 3](#_Toc155809489)

[Descriere relații 4](#_Toc155809490)

[Atribute (\* ⬄ obligatoriu) 7](#_Toc155809491)

[Diagrama entitate-relație 10](#_Toc155809492)

[Diagrama conceptuală 11](#_Toc155809493)

[Schemele relaționale 12](#_Toc155809494)

[Secvență 13](#_Toc155809495)

[Creare și inserare 14](#_Toc155809496)

[creare de tabele: 14](#_Toc155809497)

[inserari: 16](#_Toc155809498)

[Subprogram stocat independent cu 3 tipuri de colecții 27](#_Toc155809499)

[Subprogram stocat independent cu 2 tipuri de cursoare, unul parametrizat 29](#_Toc155809500)

[Subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă sql 3 tabele + 2 excepții propprii 30](#_Toc155809501)

[Subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă sql 5 tabele + 2 tratare excepții 33](#_Toc155809502)

[Definiți un trigger de tip LMD la nivel de comandă 36](#_Toc155809503)

[Definiți un trigger de tip LMD la nivel de linie 36](#_Toc155809504)

[Definiți un trigger de tip LDD 38](#_Toc155809505)

[Definiți un pachet care să conțină toate obiectele definite 39](#_Toc155809506)

[Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite 45](#_Toc155809507)

[Motivație: 45](#_Toc155809508)

[Funcționalități: 45](#_Toc155809509)

[Cod: 46](#_Toc155809510)

[Screenshots: 53](#_Toc155809511)

# 

# Descriere model

Un utilizator își poate alege o singură subscripție, și se poate uita doar la filmele pe care aceasta le conține. Lunar, el va trebui să plătească costul subscripției respective. Există mai multe tipuri de subscripție, cea de nivel cel mai mic având un anumit număr de filme/seriale, cea de nivel următor are filmele/serialele subscripție de nivel anterior plus încă câteva.

Fiecare film este obligatoriu să fie regizat de un Director, iar în el pot juca mai mulți actori, fiecare actor având posibilitatea de a juca mai multe roluri.

Pentru seriale, ele pot avea mai multe episoade, fiecare de o durație diferită.

# Constrângeri

* Un utilizator trebuie să iși aleagă o singură subscripție
* Se poate plăti doar cu cardul
* O subscripție are mai multe filme și seriale, dar filmele și serialele pot aparține tuturor subscripțiilor, sau doar la câteva
* Un serial poate să aibă mai multe episoade, dar un episod are un singur serial
* Un film trebuie să aibă un director, dar un director poate să regizeze mai multe filme
* Un actor poate să nu aibă niciun rol, sau mai multe, și poate juca in mai multe filme/seriale, sau niciunul.

# Descriere entități

* **UTILIZATOR**
* persoana fizică care iși poate crea cont pe platforma Netflix
* entitate dependentă de **SUBSCRIPȚIE** și **PLATĂ**
* cheie primară: Utilizator\_ID
* **SUBSCRIPȚIE**
* un pachet care contine filme/seriale cu un pret specific
* entitate independentă
* cheie primară: Subscripție\_ID
* **PLATĂ**
* informațiile cardului cu care plătește utilizatorul
* entitate independentă
* cheie primară: Plată\_ID
* **FILM**
* film, la care se poate uita un utilizator
* entitate dependentă de **DIRECTOR**
* cheie primară: Film\_ID
* **SERIAL**
* serial, la care se poate uita un utilizator
* entitate independentă
* cheie primară: Serial\_ID
* **EPISOD**
* episod, care aparține unui serial
* entitate dependentă de **SERIAL**
* cheieprimară: Episod\_ID
* **ACTOR**
* persoană care joacă într-un film/serial
* entitate independentă
* cheie primară: Actor\_ID
* **ROL**
* caracter dintr-un film care poate fi jucat de un/mai mulți actor/i
* entitate independentă
* cheie primară: Rol\_ID
* **DIRECTOR**
* persoană care regizează un film
* entitate independentă
* cheie primară: Director\_ID

# Descriere relații

UTILIZATOR\_alege\_o\_SUBSCRIPȚIE = relație între UTILIZATOR și SUBSCRIPȚIE

Un UTILIZATOR poate să aleagă decât o singură SUBSCRIPȚIE

O SUBSCRIPȚIE poate să fie aleasă de zero sau mai mulți UTILIZATORI

* Cardinalitate maximă: M:1
* Cardinalitate minimă: 0:1

UTILIZATOR\_adaugă \_o\_PLATĂ = relație între UTILIZATOR și PLATĂ

Un UTILIZATOR poate să adauge 1 sau mai multe PLATĂ

O PLATĂ poate să fie adăugată de 1 sau mai mulți UTILIZATORI

* Cardinalitate maximă: M:M
* Cardinalitate minimă: 1:1

SUBSCRIȚIE\_are \_un\_SERIAL = relație între SUBSCRIPȚIE și SERIAL

O SUBSCRIPȚIE poate să aibă zero sau mai multe SERIALE

Un SERIAL poate să aparțină de zero sau mai multe SUBSCRIPȚII

* Cardinalitate maximă: M:M
* Cardinalitate minimă: 0:0

SUBSCRIȚIE\_are \_un\_FILM = relație între SUBSCRIPȚIE și FILM

O SUBSCRIPȚIE poate să aibă zero sau mai multe FILME

Un FILM poate să aparțină de zero sau mai multe SUBSCRIPȚII

* Cardinalitate maximă: M:M
* Cardinalitate minimă: 0:0

ACTOR\_joacă\_în \_FILM = relație între ACTOR și FILM

Un ACTOR poate să joace în zero sau mai multe FILME

Într-un FILM poate să joace zero sau mai mulți ACTORI

* Cardinalitate maximă: M:M
* Cardinalitate minimă: 0:0

ACTOR\_joacă\_în \_SERIAL = relație între ACTOR și SERIAL

Un ACTOR poate să joace în zero sau mai multe SERIALE

Într-un SERIAL poate să joace zero sau mai mulți ACTORI

* Cardinalitate maximă: M:M
* Cardinalitate minimă: 0:0

ACTOR\_are \_ROL = relație între ACTOR și ROL

Un ACTOR poate să aibă zero sau mai multe ROLURI

Un ROL poate să aibă zero sau mai mulți ACTORI

* Cardinalitate maximă: M:M
* Cardinalitate minimă: 0:0

FILM\_are \_ROL = relație între FILM și ROL

Un FILM poate să aibă zero sau mai multe ROLURI

Un ROL poate să aibă zero sau mai multe FILME

* Cardinalitate maximă: M:M
* Cardinalitate minimă: 0:0

DIRECTOR\_regizează\_un\_FILM = relație între DIRECTOR și FILM

Un DIRECTOR poate să regizeze în unul sau mai multe FILME

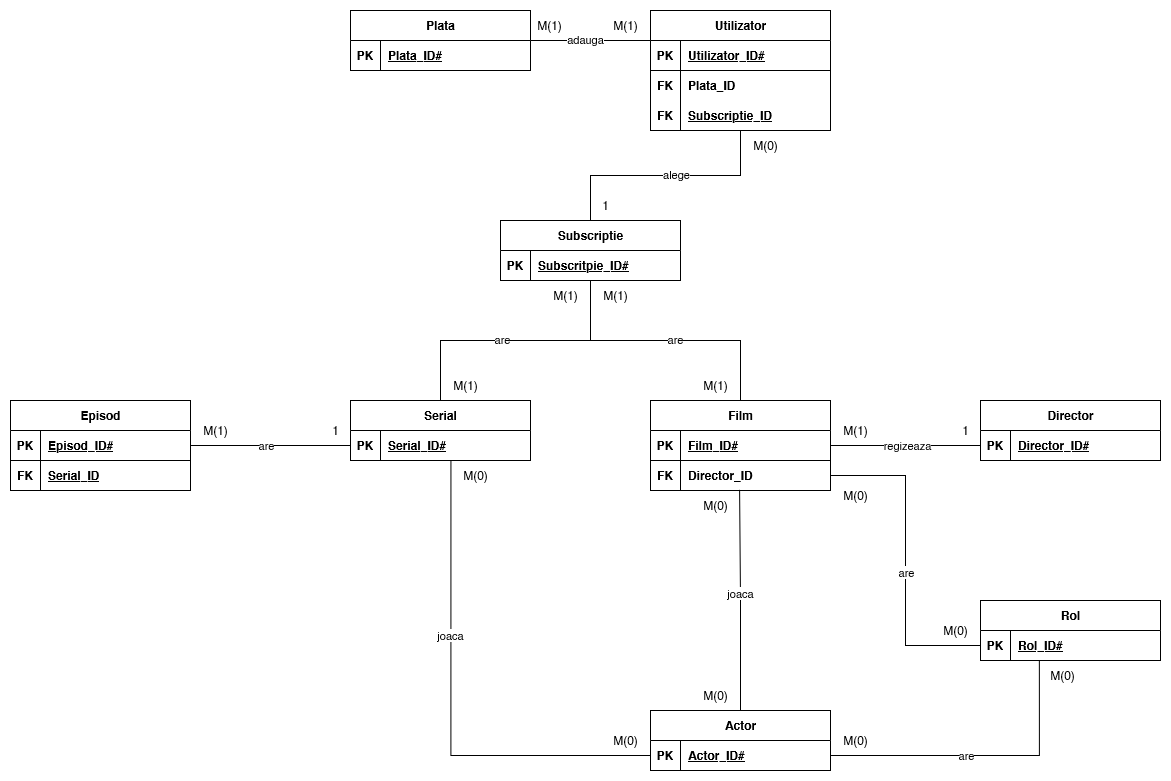
FILM trebuie să fie regizat de un singur DIRECTOR

* Cardinalitate maximă: 1:M
* Cardinalitate minimă: 1:1

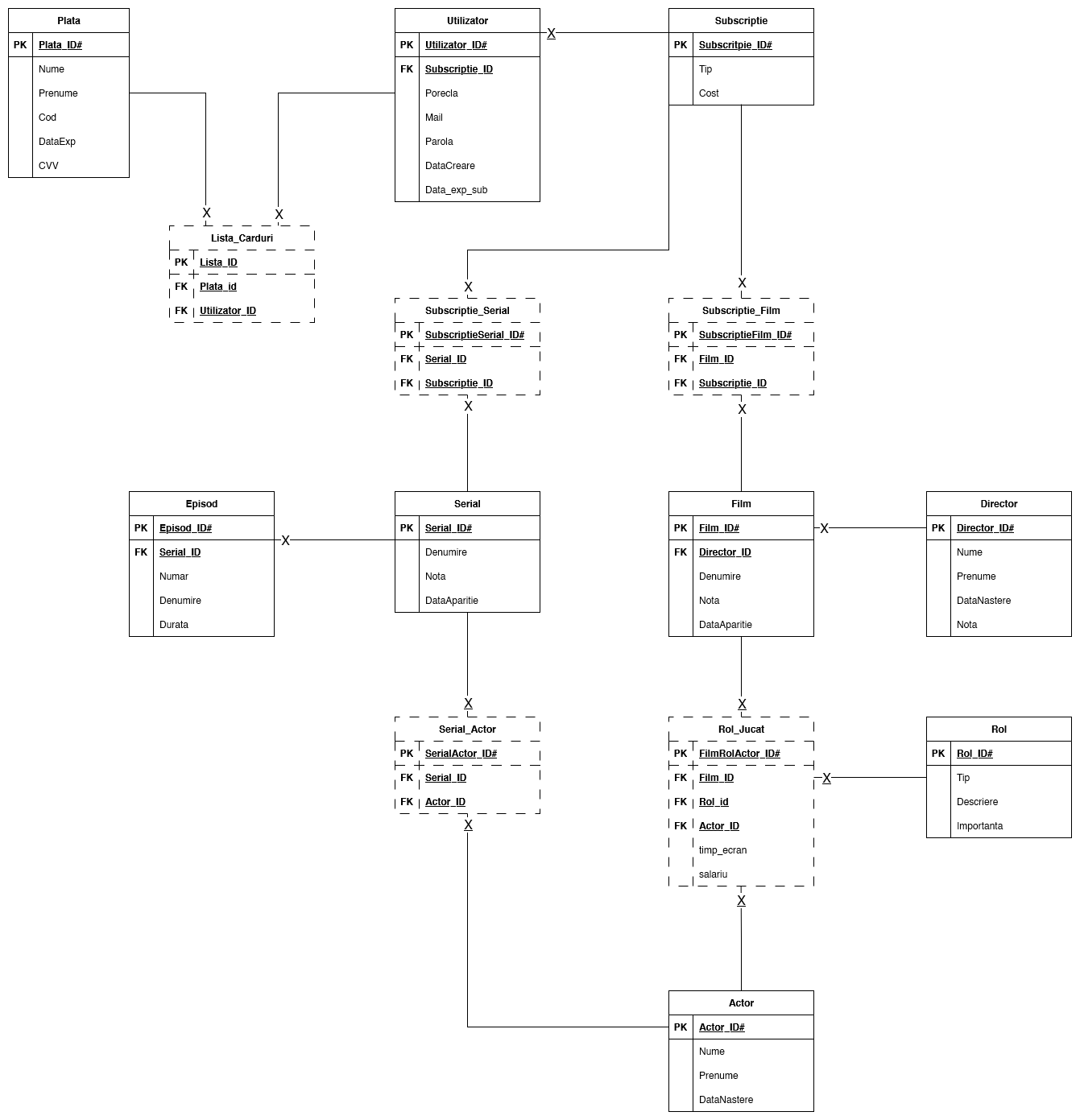
# Atribute (\* ⬄ obligatoriu)

* **UTILIZATOR**
* Utilizator\_ID\* = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea contului unui utilizator
* Subscripție \_ID\* = cheie străină din tabela Subscripție
* Poreclă\* = variabilă de tip varchar2, de lungime 50
* Mail\* = variabilă de tip varchar2 de lungime 255
* Parolă\* = variabilă de tip varchar2 de lungime 255
* DatăCreare = variabilă de tip date (valoare implicită = dată curentă)
* Data\_Exp\_Sub\* = date, cand utilizatorul trebuie sa plătească iar
* **SUBSCRIPȚIE**
* Subscriptie\_ID\* = cod numeric unic, de tip number, de 6 cifre pentru pachetele cu filme/seriale
* Tip\* = varchar2(20) dintre valorile: basic, simplu, normal, premium, ultimate, basic e implicită
* Cost = variabilă de tip number, de 2 cifre
* **PLATĂ**
* Plată\_ID\* = cod numeric unic, de tip number, de 6 cifre pentru evidența plăților
* Nume\* = varchar2(50)
* Prenume\* = varchar2(50)
* Cod\* = number(16)
* DatăExp\* = date
* CVV\* = number(3), obligatoriu de 3 cifre
* **FILM**
* Film\_ID\* = cod numeric unic, de tip number, de 6 cifre pentru evidența filmelor
* Director\_ID\* = cheie străină din tabela Director
* Denumire\* = varchar2(50)
* Notă = number(4,2), număr între 1-10
* DatăApariție = date
* **SERIAL**
* Serial\_ID\* = cod numeric unic, de tip number, de 6 cifre pentru evidența serialelor
* Denumire\* = varchar2(50)
* Notă = number(4,2), număr între 1-10
* DatăApariție = date
* **EPISOD**
* Episod\_ID\* = cod numeric unic, de tip number, de 6 cifre pentru evidența episoadelor
* Serial\_ID\* = cheie străină din tabela Serial
* Număr = number(2), numărul episodului în serial
* Denumire\* = varchar2(50)
* Durată\* = number(3), unitate de măsură = minut
* **ACTOR**
* Actor\_ID\* = cod numeric unic, de tip number, de 6 cifre pentru fiecare actor
* Nume\* = varchar2(50)
* Prenume\* = varchar2(50)
* DatăNaștere = date
* **ROL**
* Rol\_ID\* = cod numeric unic, de tip number, de 6 cifre pentru evidența rolurilor
* NumeCaracter\* = varchar2(50)
* Descriere = varchar2(255)
* Importanță = number(4,2), număr între 1-10
* **DIRECTOR**
* Director\_ID\* = cod numeric unic, de tip number, de 6 cifre pentru fiecare director
* Nume\* = varchar2(50)
* Prenume = varchar2(50)
* DatăNaștere = date
* Notă = number(4,2), număr între 1-10
* **Subscripție\_Serial**
* SubscripțieSerial\_ID = cod numeric unic, de tip number, de 6 cifre
* Serial\_ID = cheie străină din tabela Serial
* Subscripție\_ID = cheie străină din tabela Subsripție
* **Subscripție\_Film**
* SubscripțieFilm\_ID = cod numeric unic, de tip number, de 6 cifre
* Film\_ID = cheie străină din tabela Film
* Subscripție\_ID = cheie străină din tabela Subsripție
* **Serial\_Actor**
* SerialActor\_ID = cod numeric unic, de tip number, de 6 cifre
* Serial\_ID = cheie străină din tabela Serial
* Actor\_ID = cheie străină din tabela Actor
* **Rol\_Jucat**
* FilmRolActor\_ID = cod numeric unic, de tip number, de 6 cifre
* Film\_ID = cheie străină din tabela FILM
* Rol\_ID = cheie străină din tabela ROL
* Actor\_ID = cheie străină din tabela ACTOR
* Timp\_Ecran = number(3), unitate de măsură = minut
* Salariu = number(7), valoare implicită = 3000
* **Listă\_Carduri**
* Listă\_ID = cod numeric unic, de tip number, de 6 cifre
* Plată\_ID = cheie străină din tabela PLATĂ
* Utilizator\_ID = cheie străină din tabela UTILIZATOR

# Diagrama entitate-relație

****

# Diagrama conceptuală



# Schemele relaționale

PLATĂ(#Plată\_id, Nume, Prenume, Cod, DataExp, CVV)

UTILIZATOR(#Utilizator\_id, #Subscripție\_id, #Plată\_id, Poreclă, Mail,

Parolă, DatăCreare, Dată\_Exp\_Sub)

SUBSCRIPȚIE(#Subscipție\_id, Tip, Cost)

SubscripțieSerial(#SubscripțieSerial\_id, #Serial\_id, #Subscripție\_id)

SubscripțieFilm(#SubscripțieFilm\_id, #Film \_id, #Subscripție\_id)

SERIAL(#Serial\_id, Denumire, Notă, DatăApariție)

EPISOD(#Episod\_id, #Serial\_id, Număr, Denumire, Durată)

SerialActor(#SerialActor\_id, #Serial\_id, #Actor\_id)

FILM(#Film\_id, #Director\_id, Denumire, Notă, DatăApariție)

DIRECTOR(#Director\_id, Nume, Prenume, DatăNaștere, Notă)

ROL(#Rol\_id, NumeCaracter, Descriere, Importanță)

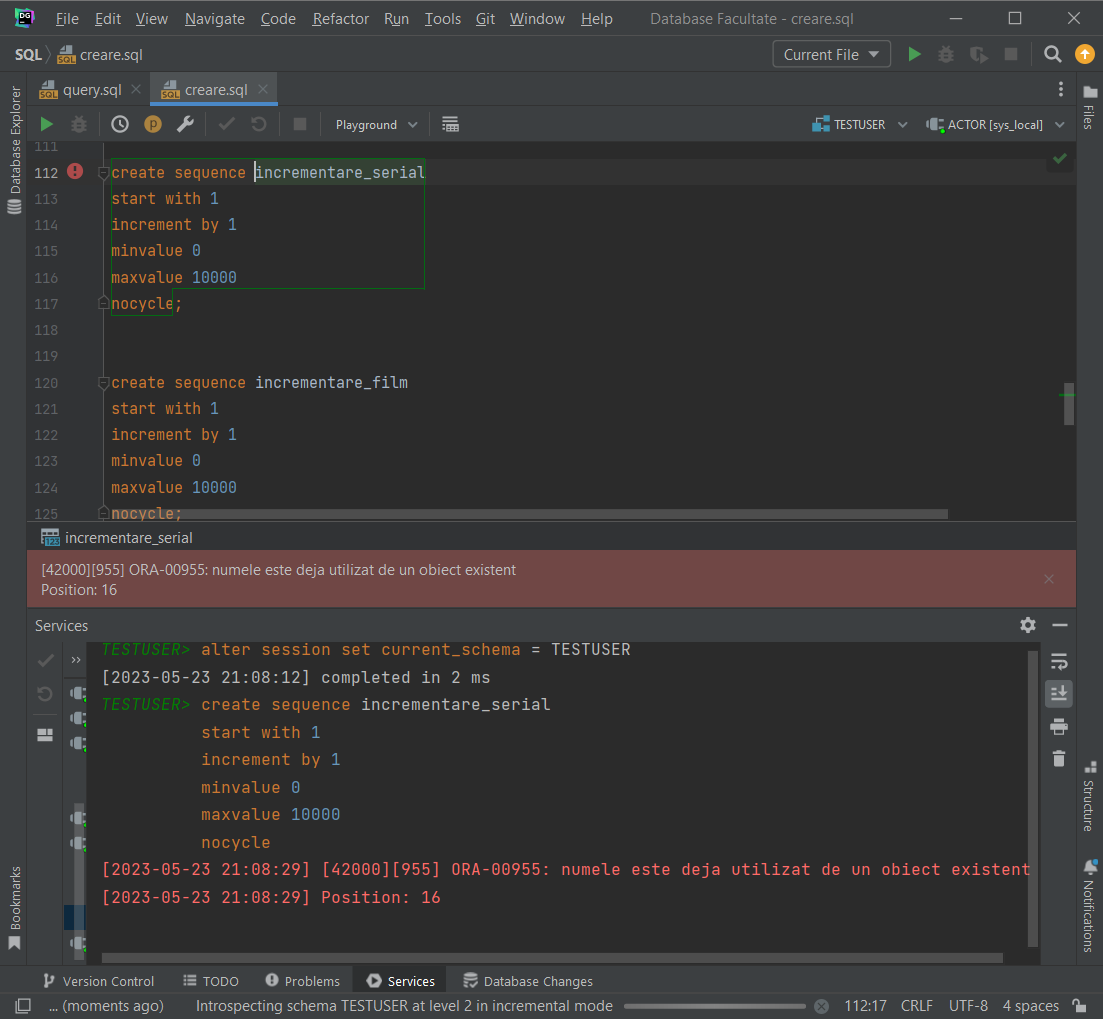
ACTOR(#Actor\_id, Nume, Prenume, DatăNaștere, Salariu)

Rol\_Jucat(#FilmRolActor\_id, #Film\_id, #Rol\_id, #Actor\_id, Timp\_Ecran)

Listă\_Carduri (#Listă\_id, #Plată \_id, #Utilizator \_id)

# Secvență

create sequence incrementare\_serial  
start with 1  
increment by 1  
minvalue 0  
maxvalue 10000  
nocycle;  
  
create sequence incrementare\_film  
start with 1  
increment by 1  
minvalue 0  
maxvalue 10000  
nocycle;  
  
create sequence incrementare\_actor  
start with 1  
increment by 1  
minvalue 0  
maxvalue 10000  
nocycle;  
  
create sequence incrementare\_rol\_jucat  
start with 1  
increment by 1  
minvalue 0  
maxvalue 10000  
nocycle;  
  
create sequence incrementare\_lista\_carduri  
start with 1  
increment by 1  
minvalue 0  
maxvalue 10000  
nocycle;

****

# Creare și inserare

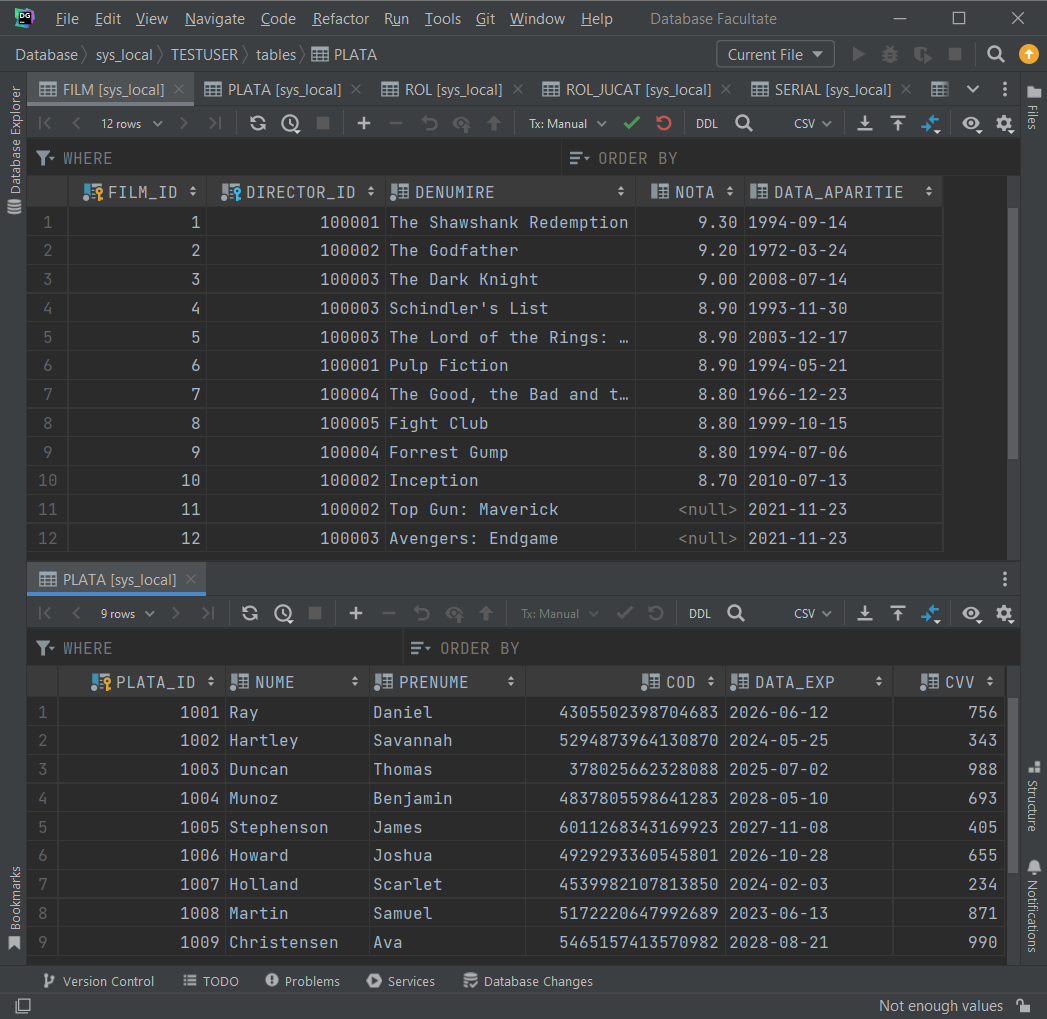
## creare de tabele:

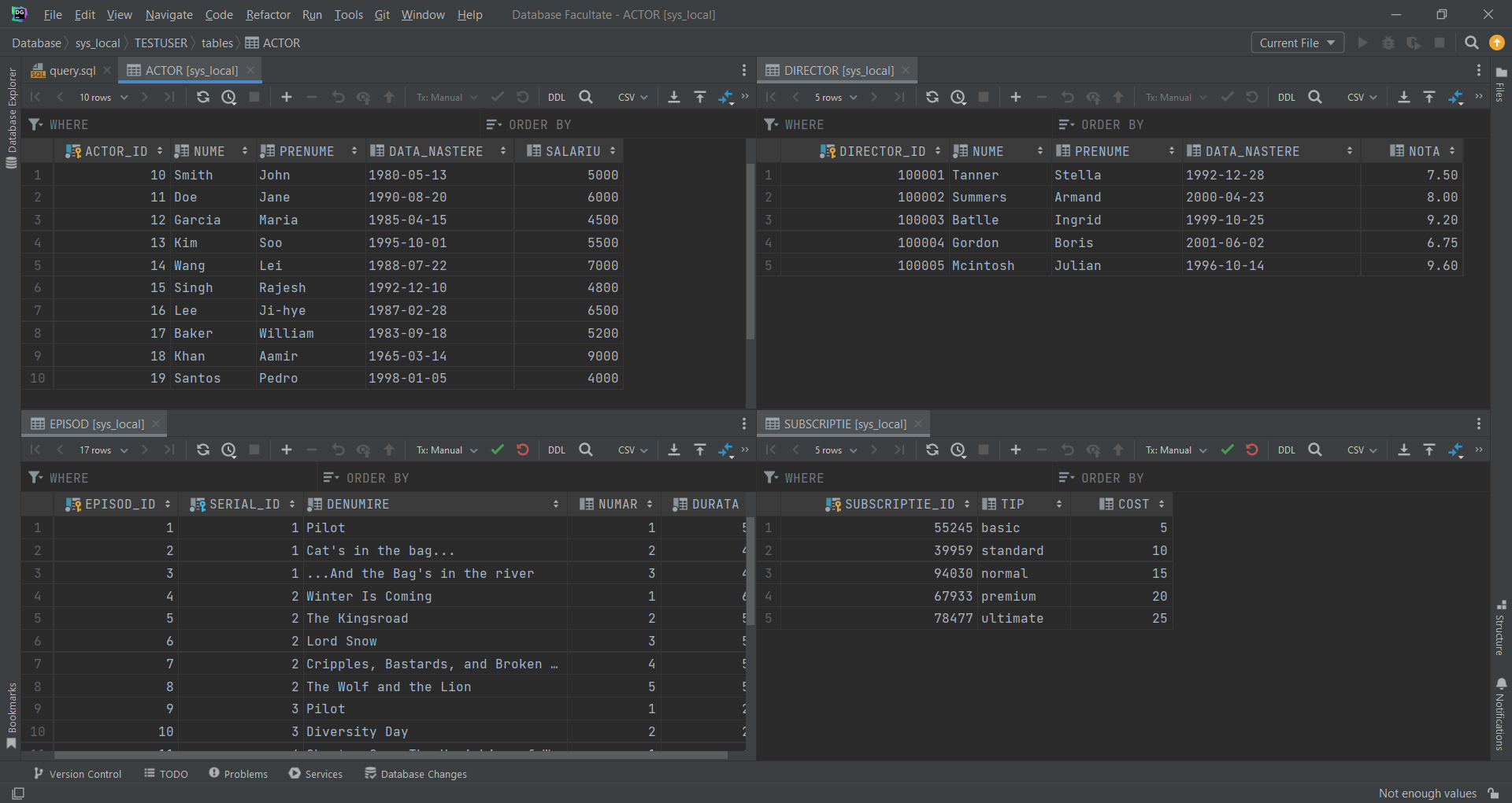
create table PLATA (  
 plata\_id number(6) constraint pk\_plata primary key,  
 nume varchar2(50) constraint nume\_null not null ,  
 prenume varchar2(50) constraint prenume\_null not null ,  
 cod number(16) constraint cod\_null not null ,  
 data\_exp date constraint date\_null not null ,  
 cvv number(3) constraint cvv\_null not null  
);  
  
create table SUBSCRIPTIE (  
 subscriptie\_id number(6) constraint pk\_subscritpie primary key ,  
 tip varchar2(20) default 'basic' check ( *lower*(tip) in ('basic','standard', 'normal', 'premium','ultimate') ) ,  
 cost number(2)  
);  
  
create table UTILIZATOR (  
 utilizator\_id number(6) constraint pk\_utilizator primary key ,  
 subscriptie\_id number(6) not null ,  
 porecla varchar2(50) constraint porecla\_null not null ,  
 mail varchar2(255) constraint mail\_null not null ,  
 parola varchar2(255) constraint parola\_null not null,  
 data\_creare date default *current\_date* ,  
 data\_exp\_sub date not null ,  
 constraint fk\_subscriptie foreign key (subscriptie\_id) references SUBSCRIPTIE(subscriptie\_id)  
);  
  
create table LISTA\_CARDURI (  
 lista\_carduri\_id number(6) constraint pk\_list\_card primary key ,  
 plata\_id number(6) not null ,  
 utilizator\_id number(6) not null ,  
 constraint fk\_plata\_card foreign key (plata\_id) references PLATA(plata\_id) ,  
 constraint fk\_util\_card foreign key (utilizator\_id) references UTILIZATOR(utilizator\_id) ,  
 CONSTRAINT uk\_plata\_util UNIQUE (utilizator\_id, plata\_id)  
);  
  
create table SERIAL (  
 serial\_id number(6) constraint pk\_serial primary key ,  
 denumire varchar2(50) constraint denumire\_null not null ,  
 nota number(4,2) constraint nota\_check check ( nota >= 1 and nota <= 10 ),  
 data\_aparitie date  
);  
  
create table EPISOD (  
 episod\_id number(6) constraint pk\_episod primary key ,  
 serial\_id number(6) not null ,  
 denumire varchar2(50)constraint denumire\_episod\_null not null ,  
 numar number(2),  
 durata number(3) constraint durata\_null not null ,  
 constraint fk\_serial foreign key (serial\_id) references SERIAL(serial\_id)  
);

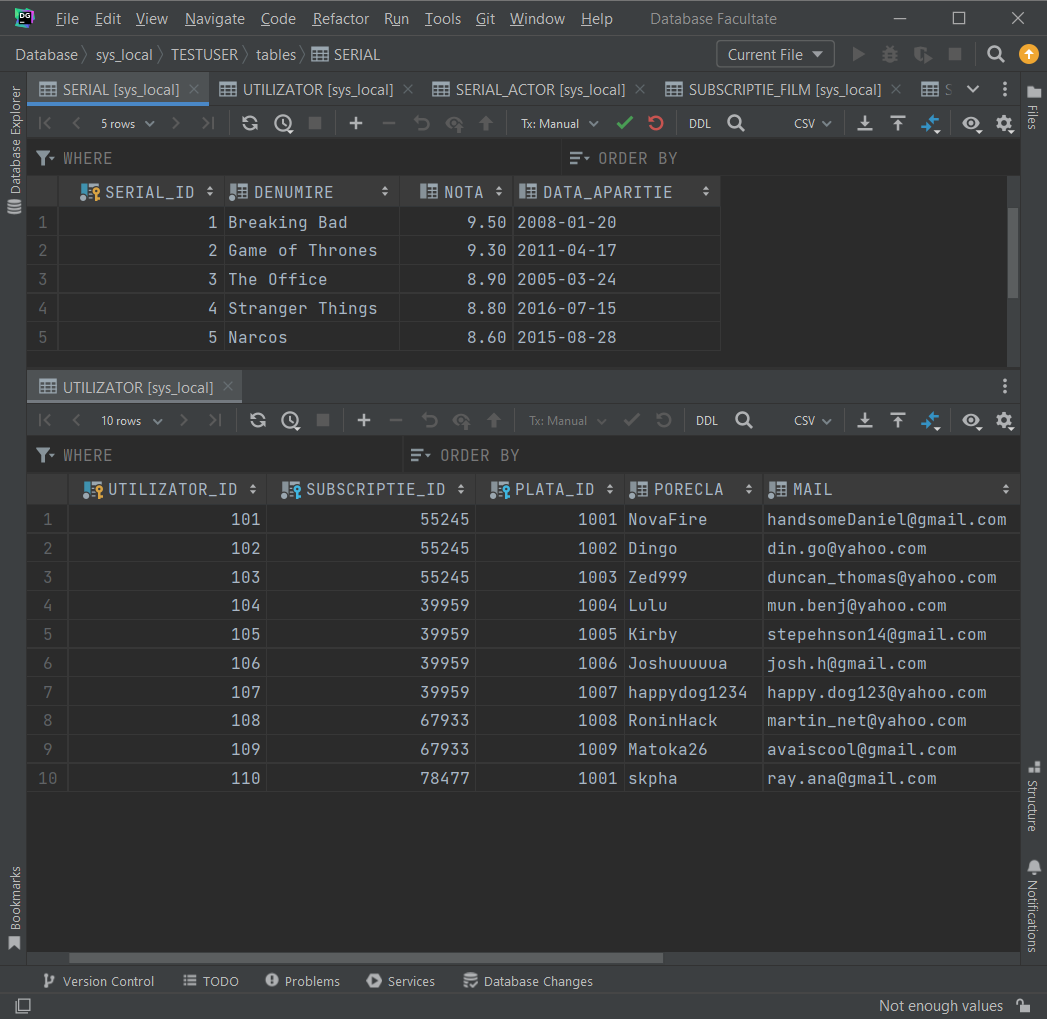
create table SUBSCRIPTIE\_SERIAL (  
 subscriptie\_serial\_id number(6) constraint pk\_sub\_ser primary key ,  
 serial\_id number(6) not null ,  
 subscriptie\_id number(6) not null ,  
 constraint fk\_subscriptie\_asoc foreign key (subscriptie\_id) references SUBSCRIPTIE(subscriptie\_id) ,  
 constraint fk\_serial\_asoc foreign key (serial\_id) references SERIAL(serial\_id) ,  
 CONSTRAINT uk\_serial\_subscriptie UNIQUE (serial\_id, subscriptie\_id)  
);  
  
create table DIRECTOR (  
 director\_id number(6) constraint pk\_director primary key ,  
 nume varchar2(50) constraint nume\_dir\_null not null ,  
 prenume varchar2(50) ,  
 data\_nastere date ,  
 nota number(4,2) constraint nota\_director\_check check ( nota >= 1 and nota <= 10 )  
);  
  
create table FILM (  
 film\_id number(6) constraint pk\_film primary key ,  
 director\_id number(6) not null ,  
 denumire varchar2(50) constraint denumire\_film\_null not null ,  
 nota number(4,2) constraint nota\_film\_check check ( nota >= 1 and nota <= 10 ) ,  
 data\_aparitie date ,  
 constraint fk\_director foreign key (director\_id) references DIRECTOR(director\_id)  
);  
  
create table SUBSCRIPTIE\_FILM (  
 subscriptie\_film\_id number(6) constraint pk\_sub\_film primary key ,  
 film\_id number(6) not null ,  
 subscriptie\_id number(6) not null ,  
 constraint fk\_subscriptie\_asocf foreign key (subscriptie\_id) references SUBSCRIPTIE(subscriptie\_id) ,  
 constraint fk\_serial\_asocf foreign key (film\_id) references FILM(film\_id) ,  
 CONSTRAINT uk\_film\_subscriptie UNIQUE (film\_id, subscriptie\_id)  
);  
  
create table ROL (  
 rol\_id number(6) constraint pk\_rol primary key ,  
 tip varchar2(50) constraint caracter\_null not null ,  
 descriere varchar2(255) ,  
 importanta number(4,2) constraint imp\_check check ( importanta >= 1 and importanta <= 10 )  
);  
  
create table ACTOR (  
 actor\_id number(6) constraint pk\_actor primary key ,  
 nume varchar2(50) constraint nume\_act\_null not null ,  
 prenume varchar2(50) constraint prenume\_act\_null not null ,  
 data\_nastere date  
);  
  
create table SERIAL\_ACTOR (  
 serial\_actor\_id number(6) constraint pk\_ser\_act primary key ,  
 serial\_id number(6) not null ,  
 actor\_id number(6) not null ,  
 constraint fk\_actor\_asocA foreign key (actor\_id) references ACTOR(actor\_id) ,  
 constraint fk\_serial\_asocA foreign key (serial\_id) references SERIAL(serial\_id) ,  
 CONSTRAINT uk\_serial\_actor UNIQUE (serial\_id, actor\_id)  
);  
  
create table ROL\_JUCAT (  
 film\_rol\_actor\_id number(6) constraint pk\_rol\_jucat primary key ,  
 film\_id number(6) not null ,  
 actor\_id number(6) not null ,  
 rol\_id number(6) not null ,  
 timp\_ecran number(3) ,  
 salariu number(7) default 3000 ,  
 constraint fk\_film\_asocRJ foreign key (film\_id) references FILM(film\_id) ,  
 constraint fk\_actor\_asocRJ foreign key (actor\_id) references ACTOR(actor\_id) ,  
 constraint fk\_rol foreign key (rol\_id) references ROL(rol\_id) ,  
 CONSTRAINT uk\_film\_actor\_rol UNIQUE (film\_id, actor\_id, rol\_id)  
);

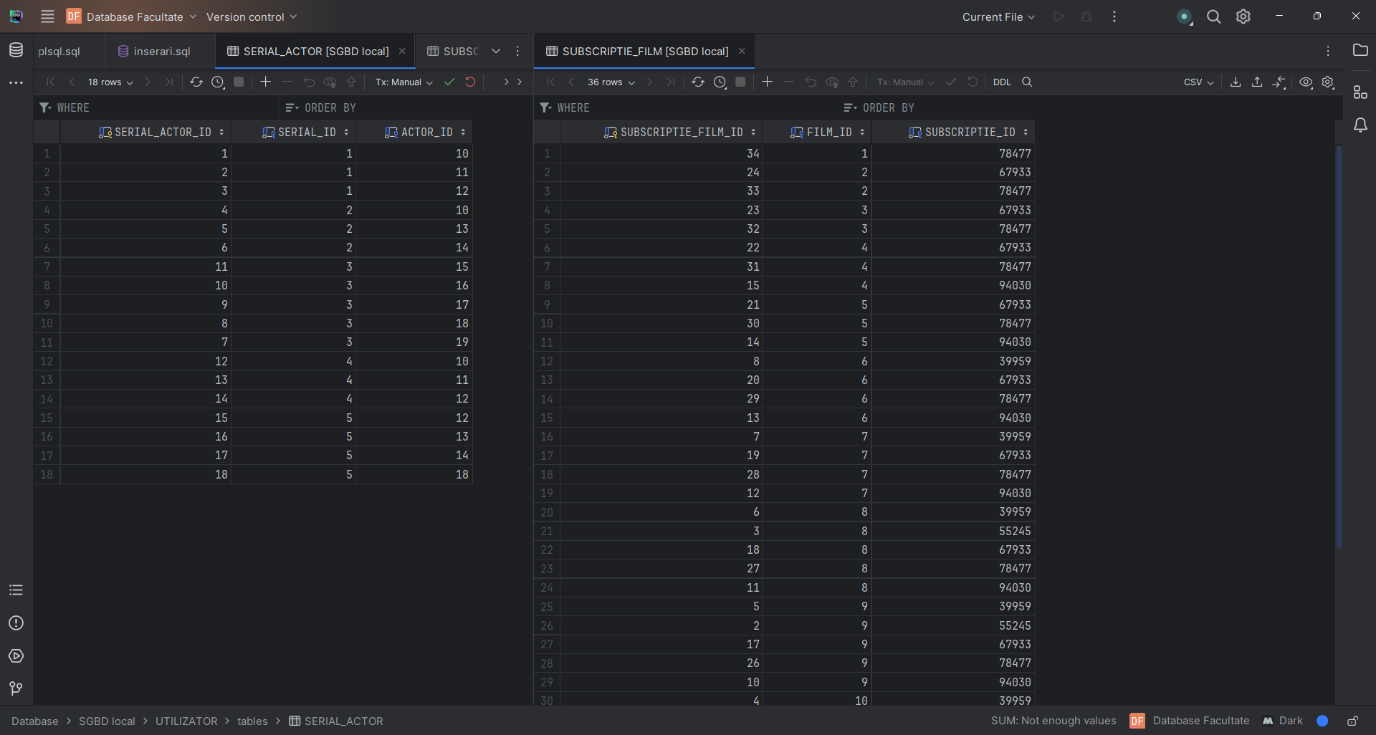
## inserari:

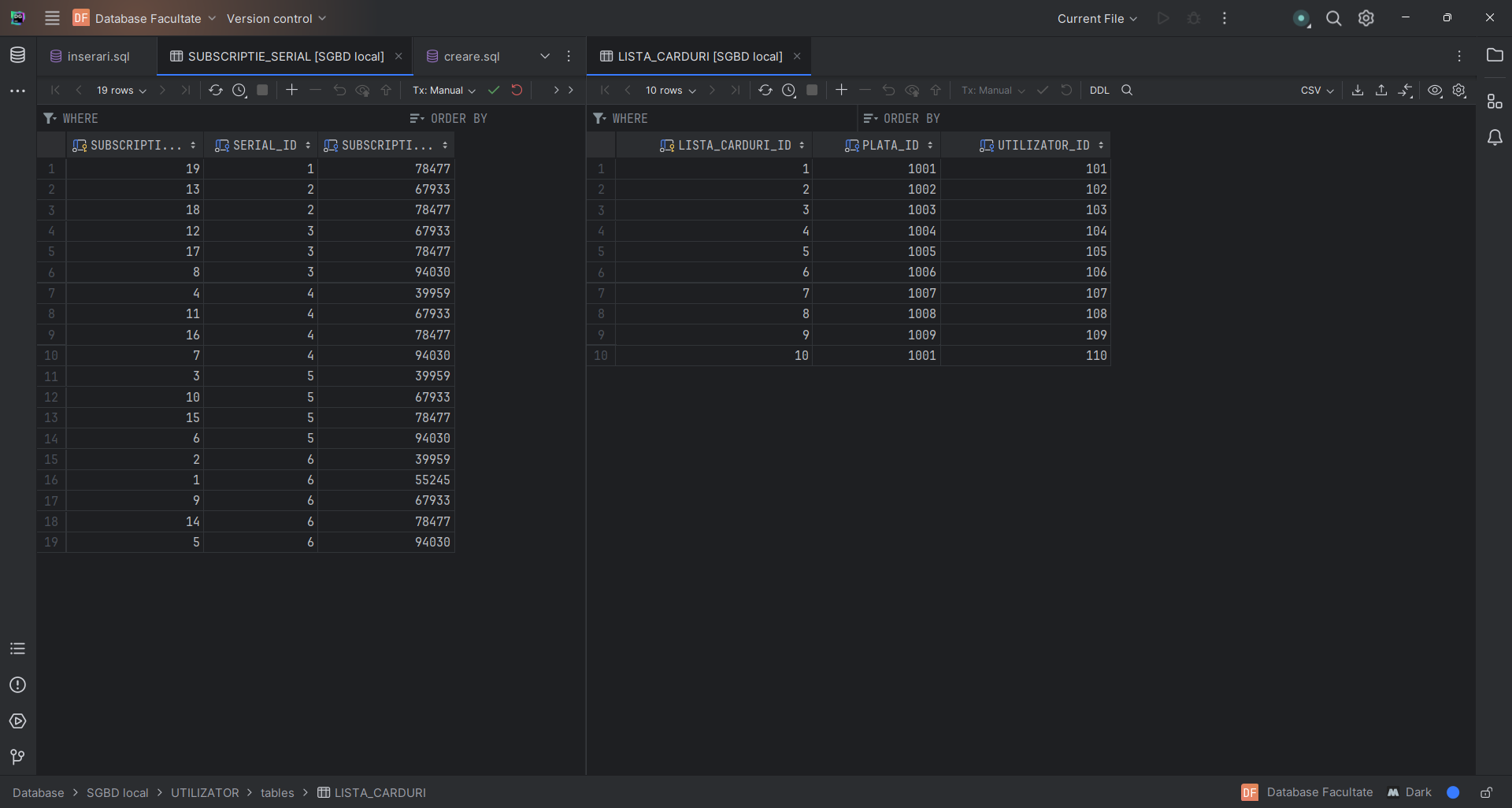
insert all  
 into PLATA(plata\_id, nume, prenume, cod, data\_exp, cvv) values (1001,'Ray','Daniel',4305502398704683,'12/06/2026',756)  
 into PLATA(plata\_id, nume, prenume, cod, data\_exp, cvv) values (1002,'Hartley','Savannah',5294873964130870,'25/05/2024',343)  
 into PLATA(plata\_id, nume, prenume, cod, data\_exp, cvv) values (1003,'Duncan','Thomas',378025662328088,'02/07/2025',988)  
 into PLATA(plata\_id, nume, prenume, cod, data\_exp, cvv) values (1004,'Munoz','Benjamin',4837805598641283,'10/05/2028',693)  
 into PLATA(plata\_id, nume, prenume, cod, data\_exp, cvv) values (1005,'Stephenson','James',6011268343169923,'08/11/2027',405)  
 into PLATA(plata\_id, nume, prenume, cod, data\_exp, cvv) values (1006,'Howard','Joshua',4929293360545801,'28/10/2026',655)  
 into PLATA(plata\_id, nume, prenume, cod, data\_exp, cvv) values (1007,'Holland','Scarlet',4539982107813850,'03/02/2024',234)  
 into PLATA(plata\_id, nume, prenume, cod, data\_exp, cvv) values (1008,'Martin','Samuel',5172220647992689,'13/06/2023',871)  
 into PLATA(plata\_id, nume, prenume, cod, data\_exp, cvv) values (1009,'Christensen','Ava',5465157413570982,'21/08/2028',990)  
select 'worked' from dual;  
  
insert all  
 into SUBSCRIPTIE(subscriptie\_id, tip, cost) values (55245,'basic',5)  
 into SUBSCRIPTIE(subscriptie\_id, tip, cost) values (39959,'standard',10)  
 into SUBSCRIPTIE(subscriptie\_id, tip, cost) values (94030,'normal',15)  
 into SUBSCRIPTIE(subscriptie\_id, tip, cost) values (67933,'premium',20)  
 into SUBSCRIPTIE(subscriptie\_id, tip, cost) values (78477,'ultimate',25)  
select 'worked' from dual;  
  
insert all  
 into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (101,55245,'NovaFire','handsomeDaniel@gmail.com','RD\_2003','14/06/2023')  
 into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (102,55245,'Dingo','din.go@yahoo.com','39hsrda9','05/05/2023')  
 into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (103,55245,'Zed999','duncan\_thomas@yahoo.com','Zed\_MY\_life','27/05/2023')  
 into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (104,39959,'Lulu','mun.benj@yahoo.com','Aida\_2022','22/05/2023')  
 into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (105,39959,'Kirby','stepehnson14@gmail.com','kirby1306','14/05/2023')  
 into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (106,39959,'Joshuuuuua','josh.h@gmail.com','netflixPassword','18/05/2023')  
 into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (107,39959,'happydog1234','happy.dog123@yahoo.com','dogggsAREsweet','10/06/2023')  
 into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (108,67933,'RoninHack','martin\_net@yahoo.com','genshin001','01/06/2023')  
 into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (109,67933,'Matoka26','avaiscool@gmail.com','Mihai1234568','08/06/2023')  
 into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (110,78477,'skpha','ray.ana@gmail.com','VulturViteaz','15/06/2023')  
select 'worked' from dual;  
  
-- secventa nu merge cu insert all, da aceeasi valoare  
insert into LISTA\_CARDURI(lista\_carduri\_id, plata\_id, utilizator\_id) values (INCREMENTARE\_LISTA\_CARDURI.nextval,1001,101);  
insert into LISTA\_CARDURI(lista\_carduri\_id, plata\_id, utilizator\_id) values (INCREMENTARE\_LISTA\_CARDURI.nextval,1002,102);  
insert into LISTA\_CARDURI(lista\_carduri\_id, plata\_id, utilizator\_id) values (INCREMENTARE\_LISTA\_CARDURI.nextval,1003,103);  
insert into LISTA\_CARDURI(lista\_carduri\_id, plata\_id, utilizator\_id) values (INCREMENTARE\_LISTA\_CARDURI.nextval,1004,104);  
insert into LISTA\_CARDURI(lista\_carduri\_id, plata\_id, utilizator\_id) values (INCREMENTARE\_LISTA\_CARDURI.nextval,1005,105);  
insert into LISTA\_CARDURI(lista\_carduri\_id, plata\_id, utilizator\_id) values (INCREMENTARE\_LISTA\_CARDURI.nextval,1006,106);  
insert into LISTA\_CARDURI(lista\_carduri\_id, plata\_id, utilizator\_id) values (INCREMENTARE\_LISTA\_CARDURI.nextval,1007,107);  
insert into LISTA\_CARDURI(lista\_carduri\_id, plata\_id, utilizator\_id) values (INCREMENTARE\_LISTA\_CARDURI.nextval,1008,108);  
insert into LISTA\_CARDURI(lista\_carduri\_id, plata\_id, utilizator\_id) values (INCREMENTARE\_LISTA\_CARDURI.nextval,1009,109);  
insert into LISTA\_CARDURI(lista\_carduri\_id, plata\_id, utilizator\_id) values (INCREMENTARE\_LISTA\_CARDURI.nextval,1001,110);  
  
insert all  
 into DIRECTOR(director\_id, nume, prenume, data\_nastere, nota) VALUES (100001,'Tanner','Stella','28/12/1992',7.5)  
 into DIRECTOR(director\_id, nume, prenume, data\_nastere, nota) VALUES (100002,'Summers','Armand','23/04/2000',8)  
 into DIRECTOR(director\_id, nume, prenume, data\_nastere, nota) VALUES (100003,'Batlle','Ingrid','25/10/1999',9.2)  
 into DIRECTOR(director\_id, nume, prenume, data\_nastere, nota) VALUES (100004,'Gordon','Boris','02/06/2001',6.75)  
 into DIRECTOR(director\_id, nume, prenume, data\_nastere, nota) VALUES (100005,'Mcintosh','Julian','14/10/1996',9.6)  
select 'worked' from dual;  
  
insert all  
 into ROL(rol\_id, tip, descriere, importanta) VALUES (111,'Principal','Va juca cel mai mult',10)  
 into ROL(rol\_id, tip, descriere, importanta) VALUES (112,'Secundar','Va participa destul de des',9.5)  
 into ROL(rol\_id, tip, descriere, importanta) VALUES (113,'Erou','Cel care ii va salva pe toti',9.75)  
 into ROL(rol\_id, tip, descriere, importanta) VALUES (114,'Raufacator','Cel care va incerca sa ii distruga pe toti',8.8)  
 into ROL(rol\_id, tip, descriere, importanta) VALUES (115,'Fundal','Doar va aparea pe fundalul filmului pentru a popula scena cu oameni',4.5)  
select 'worked' from dual;  
  
insert all  
 into ACTOR(actor\_id, nume, prenume, data\_nastere) VALUES (10, 'Smith', 'John', '13/05/1980')  
 into ACTOR(actor\_id, nume, prenume, data\_nastere) VALUES (11, 'Doe', 'Jane', '20/08/1990')  
 into ACTOR(actor\_id, nume, prenume, data\_nastere) VALUES (12, 'Garcia', 'Maria', '15/04/1985')  
 into ACTOR(actor\_id, nume, prenume, data\_nastere) VALUES (13, 'Kim', 'Soo', '01/10/1995')  
 into ACTOR(actor\_id, nume, prenume, data\_nastere) VALUES (14, 'Wang', 'Lei', '22/07/1988')  
 into ACTOR(actor\_id, nume, prenume, data\_nastere) VALUES (15, 'Singh', 'Rajesh', '10/12/1992')  
 into ACTOR(actor\_id, nume, prenume, data\_nastere) VALUES (16, 'Lee', 'Ji-hye', '28/02/1987')  
 into ACTOR(actor\_id, nume, prenume, data\_nastere) VALUES (17, 'Baker', 'William', '18/09/1983')  
 into ACTOR(actor\_id, nume, prenume, data\_nastere) VALUES (18, 'Khan', 'Aamir', '14/03/1965')  
 into ACTOR(actor\_id, nume, prenume, data\_nastere) VALUES (19, 'Santos', 'Pedro', '05/01/1998')  
select 'worked' from dual;  
  
insert all  
 into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (1, 100001, 'The Shawshank Redemption', 9.3, '14/09/1994')  
 into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (2, 100002, 'The Godfather', 9.2, '24/03/1972')  
 into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (3, 100003, 'The Dark Knight', 9.0, '14/07/2008')  
 into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (4, 100003, 'Schindler''s List', 8.9, '30/11/1993') -- 2 '' pentru a avea doar unul  
 into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (5, 100003, 'The Lord of the Rings: The Return of the King', 8.9, '17/12/2003')  
 into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (6, 100001, 'Pulp Fiction', 8.9, '21/05/1994')  
 into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (7, 100004, 'The Good, the Bad and the Ugly', 8.8, '23/12/1966')  
 into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (8, 100005, 'Fight Club', 8.8, '15/10/1999')  
 into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (9, 100004, 'Forrest Gump', 8.8, '06/07/1994')  
 into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (10, 100002, 'Inception', 8.7, '13/07/2010')  
select 'worked' from dual;  
  
insert all  
 into SERIAL(serial\_id, denumire, nota, data\_aparitie) VALUES (1, 'Breaking Bad', 9.5, '20/01/2008')  
 into SERIAL(serial\_id, denumire, nota, data\_aparitie) VALUES (2, 'Game of Thrones', 9.3, '17/04/2011')  
 into SERIAL(serial\_id, denumire, nota, data\_aparitie) VALUES (3, 'The Office', 8.9, '24/03/2005')  
 into SERIAL(serial\_id, denumire, nota, data\_aparitie) VALUES (4, 'Stranger Things', 8.8, '15/07/2016')  
 into SERIAL(serial\_id, denumire, nota, data\_aparitie) VALUES (5, 'Narcos', 8.6, '28/08/2015')  
select 'worked' from dual;  
  
insert all  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (1,1,'Pilot',1,58)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (2,1,'Cat''s in the bag...',2,48)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (3,1,'...And the Bag''s in the river',3,47)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (4,2,'Winter Is Coming',1,60)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (5,2,'The Kingsroad',2,56)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (6,2,'Lord Snow',3,58)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (7,2,'Cripples, Bastards, and Broken Things',4,54)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (8,2,'The Wolf and the Lion',5,55)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (9,3,'Pilot',1,22)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (10,3,'Diversity Day',2,23)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (11,4,'Chapter One: The Vanishing of Will Byers',1,48)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (12,4,'Chapter Two: The Weirdo on Maple Street',2,55)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (13,4,'Chapter Three: Holly, Jolly',3,51)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (14,4,'Chapter Four: The Body',4,58)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (15,5,'Descenso',1,57)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (16,5,'The Sword of Simón Bolivar',2,50)  
 into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (17,5,'The Men of Always',3,46)  
select 'worked' from dual;  
  
-- secventa nu merge cu insert all, da aceeasi valoare  
  
 -- SUBSCRIPTIE\_SERIAL --  
  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,6,55245);  
  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,6,39959);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,5,39959);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,4,39959);  
  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,6,94030);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,5,94030);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,4,94030);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,3,94030);  
  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,6,67933);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,5,67933);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,4,67933);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,3,67933);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,2,67933);  
  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,6,78477);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,5,78477);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,4,78477);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,3,78477);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,2,78477);  
insert into SUBSCRIPTIE\_SERIAL(subscriptie\_serial\_id, serial\_id, subscriptie\_id) VALUES (INCREMENTARE\_SERIAL.nextval,1,78477);  
-- ==========================================================================================================================  
  
 -- SUBSCRIPTIE\_FILM --  
  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,10,55245);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,9,55245);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,8,55245);  
  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,10,39959);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,9,39959);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,8,39959);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,7,39959);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,6,39959);  
  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,10,94030);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,9,94030);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,8,94030);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,7,94030);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,6,94030);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,5,94030);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,4,94030);  
  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,10,67933);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,9,67933);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,8,67933);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,7,67933);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,6,67933);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,5,67933);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,4,67933);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,3,67933);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,2,67933);  
  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,10,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,9,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,8,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,7,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,6,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,5,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,4,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,3,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,2,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,1,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,11,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,12,78477);  
-- =====================================================================================================================  
  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,1,10);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,1,11);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,1,12);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,2,10);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,2,13);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,2,14);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,3,19);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,3,18);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,3,17);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,3,16);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,3,15);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,4,10);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,4,11);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,4,12);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,5,12);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,5,13);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,5,14);  
insert into SERIAL\_ACTOR(serial\_actor\_id, serial\_id, actor\_id) VALUES (INCREMENTARE\_ACTOR.nextval,5,18);  
  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,1,111,10,80,10000);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,1,112,11,35,3500);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,1,115,12,10);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,1,115,13,5, 4000);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,2,113,14,67,8000);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,2,114,15,55,4570);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,3,111,16,72,9800);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,3,113,17,48,4300);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,3,114,18,37,3500);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,4,111,18,57,6000);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,4,113,18,43,6500);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,4,114,19,23,3400);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,5,111,15,66,7500);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,5,112,13,43,4600);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,5,113,12,52,6350);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,5,115,10,12,4000);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,6,111,11,12,4000);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,6,112,16,12,3000);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,6,115,17,12,5500);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,6,115,15,22,5750);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,6,115,13,10,3200);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,7,111,14,57,9000);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,7,114,14,63,8700);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,7,112,13,44,6570);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,8,113,19,77,10000);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,8,114,18,82,12000);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,9,111,12,93,30000);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,9,115,17,23,5300);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,9,115,13,44,3100);  
insert into ROL\_JUCAT(film\_rol\_actor\_id, film\_id, rol\_id, actor\_id, timp\_ecran, salariu) VALUES (INCREMENTARE\_ROL\_JUCAT.nextval,10,111,19,123,7000);  
  
insert into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (11, 100002, 'Top Gun: Maverick', NULL, '23/11/2021');  
insert into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (12, 100003, 'Avengers: Endgame', NULL, '23/11/2021');  
  
insert into DIRECTOR(director\_id, nume, prenume, data\_nastere, nota) VALUES (100006,'William','Anderson','10/05/1990',5);  
  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,11,78477);  
insert into SUBSCRIPTIE\_film(subscriptie\_film\_id, film\_id, subscriptie\_id) VALUES (INCREMENTARE\_film.nextval,12,78477);  
  
insert into ACTOR(actor\_id, nume, prenume, data\_nastere) VALUES (20, 'James', 'Bond', '13/02/2000');  
  
insert into SERIAL(serial\_id, denumire, nota, data\_aparitie) VALUES (6, 'Rick and Morty', 7.4, '12/11/2016');  
insert into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (18,6,'Pilot',1,21);  
insert into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (19,6,'Lawnmower Dog',2,19);  
insert into EPISOD(episod\_id, serial\_id, denumire, numar, durata) VALUES (20,6,'Anatomy Park',3,23);  
  
-- inserari pentru ex 9  
insert into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (111,78477,'skpha','skpha@gmail.com','VulturViteaz','15/06/2023');  
insert into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) VALUES (13,100006,'Talk with me',9,'28/07/2023');  
insert into SUBSCRIPTIE(subscriptie\_id, tip, cost) VALUES (12345,'ULTIMATE',0);  
insert into UTILIZATOR(utilizator\_id, subscriptie\_id, porecla, mail, parola,data\_exp\_sub) values (112,12345,'OnePiece','anime@gmail.com','animelover03','21/11/2023');  
  
-- inserare film nou pentru ex 11  
insert into FILM(film\_id, director\_id, denumire, nota, data\_aparitie) values (999,100006,'Test',2,'15/11/2023');

****

****

****

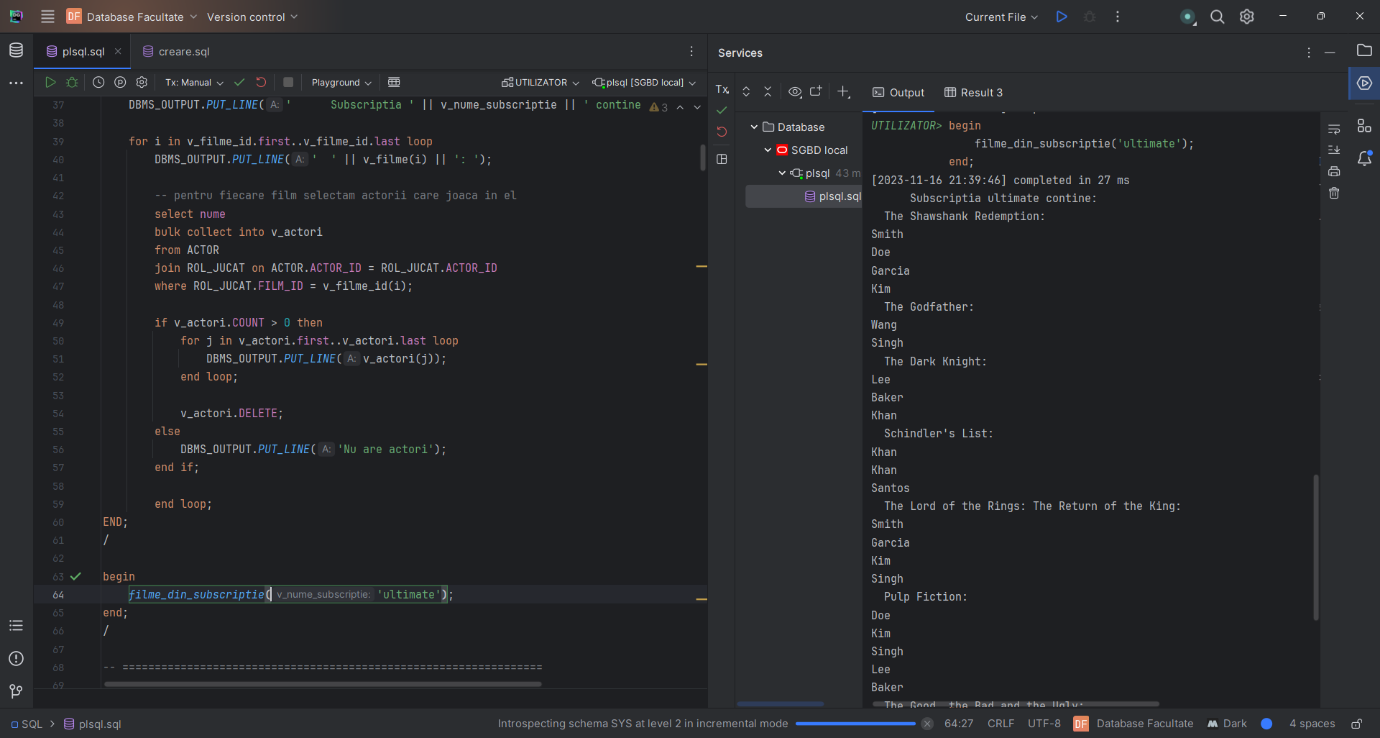
****

****

# Subprogram stocat independent cu 3 tipuri de colecții

Cerință: Pentru o subscripție afișați toate filmele ei, și pentru fiecare film, actorii lui.

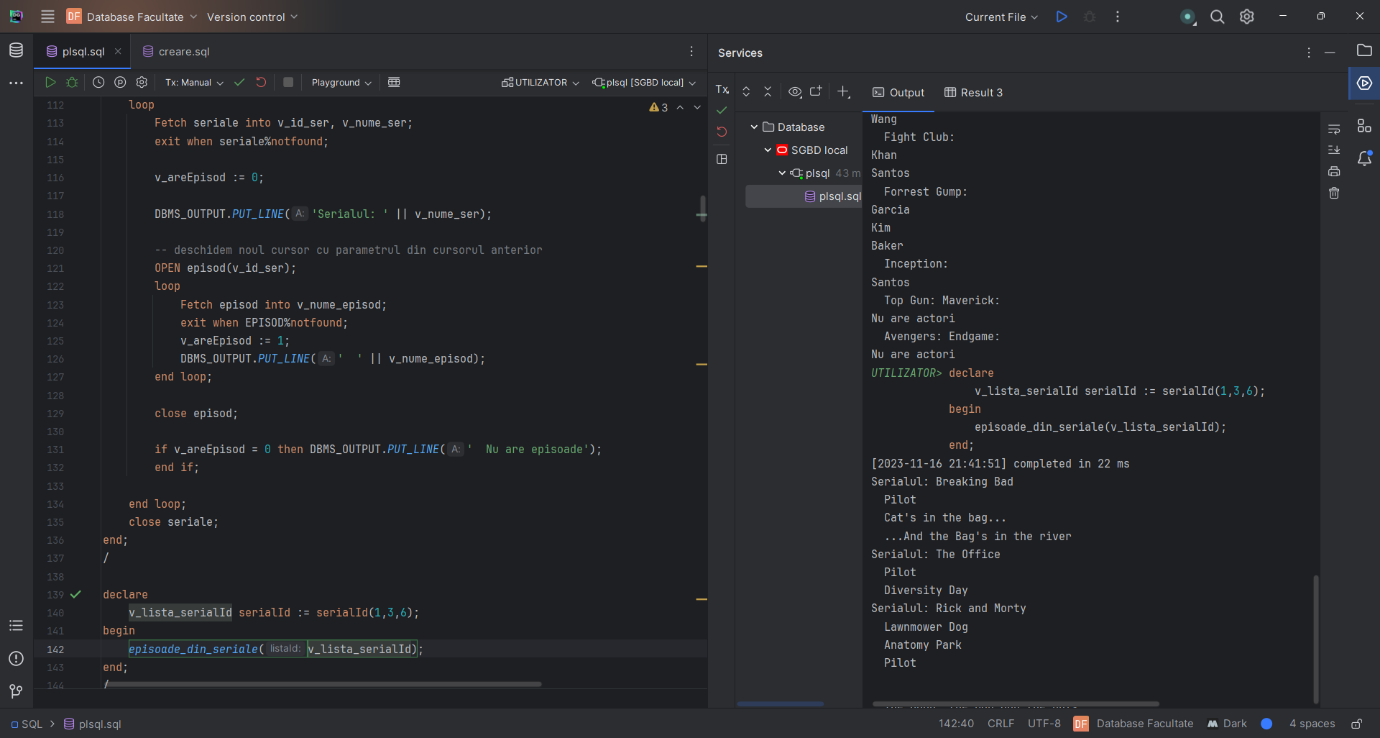
-- ====== EX6 ======  
  
-- pentru o subscriptie afisati toate filmele ei  
-- + toti actorii fiecarui film  
  
-- varray  
create or replace type actori as varray(20) of varchar2(50);  
  
CREATE OR REPLACE PROCEDURE *filme\_din\_subscriptie*(v\_nume\_subscriptie SUBSCRIPTIE.TIP%TYPE)  
AS  
 -- tablou indexat care retine numele filmelor  
 type filme is table of FILM.denumire%type index by pls\_integer;  
 v\_filme filme;  
  
 -- tablou imbricat care retine id-ul filmelor  
 type filme\_id is table of ACTOR.nume%type;  
 v\_filme\_id filme\_id;  
  
 -- varray care contine numele actorilor  
 v\_actori actori;  
  
 v\_subscriptie\_id SUBSCRIPTIE.subscriptie\_id%type;  
BEGIN  
 -- obtine id-ul subscriptiei  
 select unique SUBSCRIPTIE\_ID  
 into v\_subscriptie\_id  
 from SUBSCRIPTIE  
 where TIP = v\_nume\_subscriptie;  
  
 -- obtin toate numele filmelor si id-urile lor in tabloul indexat v\_filme/v\_filme\_id  
 select DENUMIRE, FILM.FILM\_ID  
 bulk collect into v\_filme, v\_filme\_id  
 from FILM  
 join SUBSCRIPTIE\_FILM on FILM.FILM\_ID = SUBSCRIPTIE\_FILM.FILM\_ID  
 where SUBSCRIPTIE\_FILM.SUBSCRIPTIE\_ID = v\_subscriptie\_id;  
  
 DBMS\_OUTPUT.*PUT\_LINE*(' Subscriptia ' || v\_nume\_subscriptie || ' contine:');  
  
 for i in v\_filme\_id.first..v\_filme\_id.last loop  
 DBMS\_OUTPUT.*PUT\_LINE*(' ' || v\_filme(i) || ': ');  
  
 -- pentru fiecare film selectam actorii care joaca in el  
 select nume  
 bulk collect into v\_actori  
 from ACTOR  
 join ROL\_JUCAT on ACTOR.ACTOR\_ID = ROL\_JUCAT.ACTOR\_ID  
 where ROL\_JUCAT.FILM\_ID = v\_filme\_id(i);  
  
 if v\_actori.COUNT > 0 then  
 for j in v\_actori.first..v\_actori.last loop  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_actori(j));  
 end loop;  
  
 v\_actori.DELETE;  
 else  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu are actori');  
 end if;  
  
 end loop;  
END;  
/  
  
begin  
 *filme\_din\_subscriptie*('ultimate');  
end;  
/  
-- =================================================================



# Subprogram stocat independent cu 2 tipuri de cursoare, unul parametrizat

Cerință: pentru fiecare serial cu id într-o listă, afișați numele tutuor episoadelor care apațin serialului respectiv

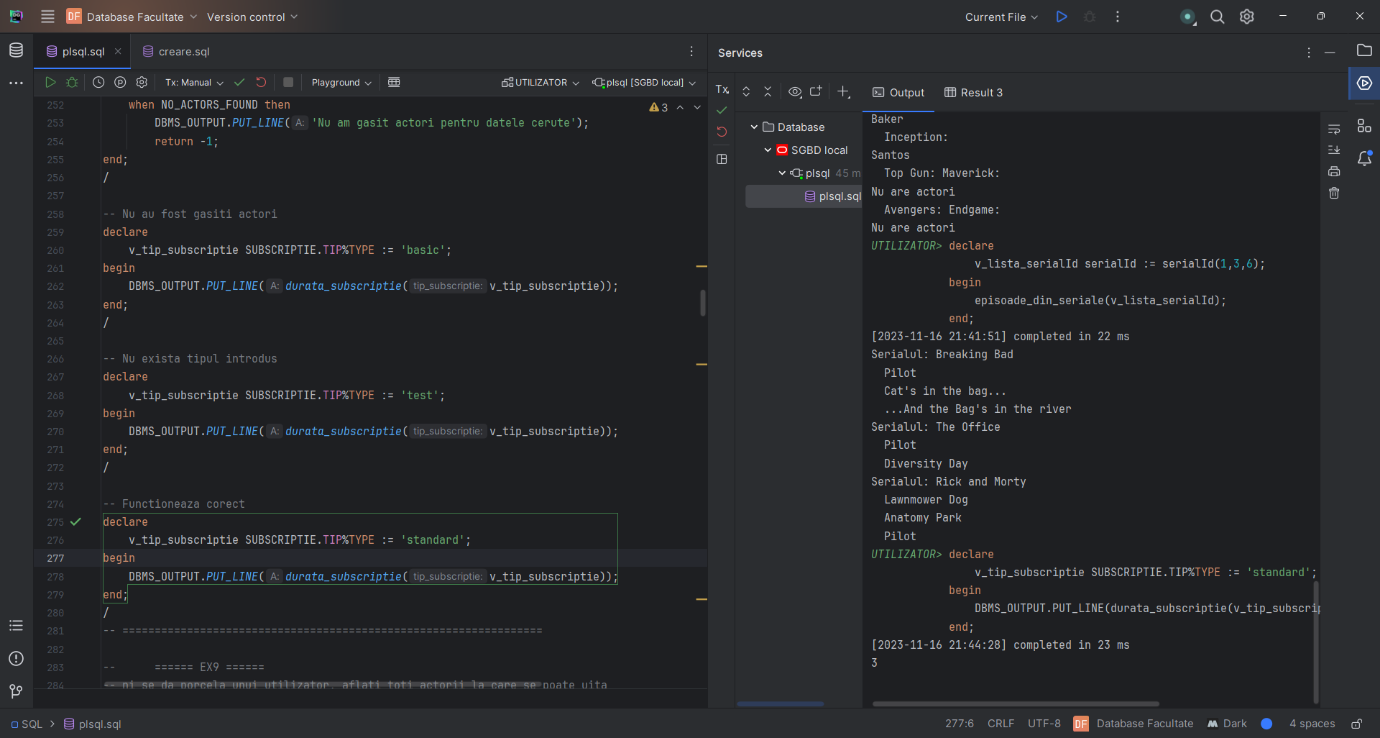
-- ====== EX7 ======  
-- pentru fiecare Serial cu id-ul intr-o lista afisati numele tuturor  
-- episoadelor care apartin  
create or replace type serialId as varray(10) of number(6);  
  
-- functie ajutatoare pentru a verifica daca un serial apartine unui varray de seriale  
-- am facut asta ca primeam o eroare ciudata si asta mi s-a parut un workaound destul de bun  
create or replace function *verifica\_serial*(v\_serialId SERIAL.SERIAL\_ID%TYPE, listaId serialId) RETURN NUMBER AS  
 v\_found number(1) := 0;  
begin  
 for i in 1..listaId.COUNT loop  
 if v\_serialId = listaId(i) then  
 v\_found := 1;  
 return v\_found;  
 end if;  
 end loop;  
  
 return v\_found;  
end;  
/  
  
CREATE OR REPLACE PROCEDURE *episoade\_din\_seriale*(listaId serialId) AS  
 v\_id\_ser SERIAL.serial\_id%type;  
  
 v\_nume\_ser SERIAL.denumire%type;  
 v\_nume\_episod varchar2(50);  
 v\_areEpisod number(1) := 0;  
  
 -- cursor clasic  
 CURSOR seriale IS  
 select SERIAL\_ID, DENUMIRE  
 from SERIAL  
 WHERE *verifica\_serial*(SERIAL\_ID, listaId) = 1;  
  
 -- cursor parametrizat dependent de cel anterior  
 CURSOR episod(id number) IS  
 select DENUMIRE  
 from EPISOD  
 where SERIAL\_ID = id;  
  
begin  
 OPEN seriale;  
 loop  
 Fetch seriale into v\_id\_ser, v\_nume\_ser;  
 exit when seriale%notfound;  
  
 v\_areEpisod := 0;  
  
 DBMS\_OUTPUT.*PUT\_LINE*('Serialul: ' || v\_nume\_ser);  
  
 -- deschidem noul cursor cu parametrul din cursorul anterior  
 OPEN episod(v\_id\_ser);  
 loop  
 Fetch episod into v\_nume\_episod;  
 exit when EPISOD%notfound;  
 v\_areEpisod := 1;  
 DBMS\_OUTPUT.*PUT\_LINE*(' ' || v\_nume\_episod);  
 end loop;  
  
 close episod;  
  
 if v\_areEpisod = 0 then DBMS\_OUTPUT.*PUT\_LINE*(' Nu are episoade');  
 end if;  
  
 end loop;  
 close seriale;  
end;  
/  
  
declare  
 v\_lista\_serialId serialId := serialId(1,3,6);  
begin  
 *episoade\_din\_seriale*(v\_lista\_serialId);  
end;  
/  
-- =================================================================



# Subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă sql 3 tabele + 2 excepții propprii

Cerință: pentru o subscripție dată să se ia serialul cu durata cea mai mare și să se afișeze câți actori are

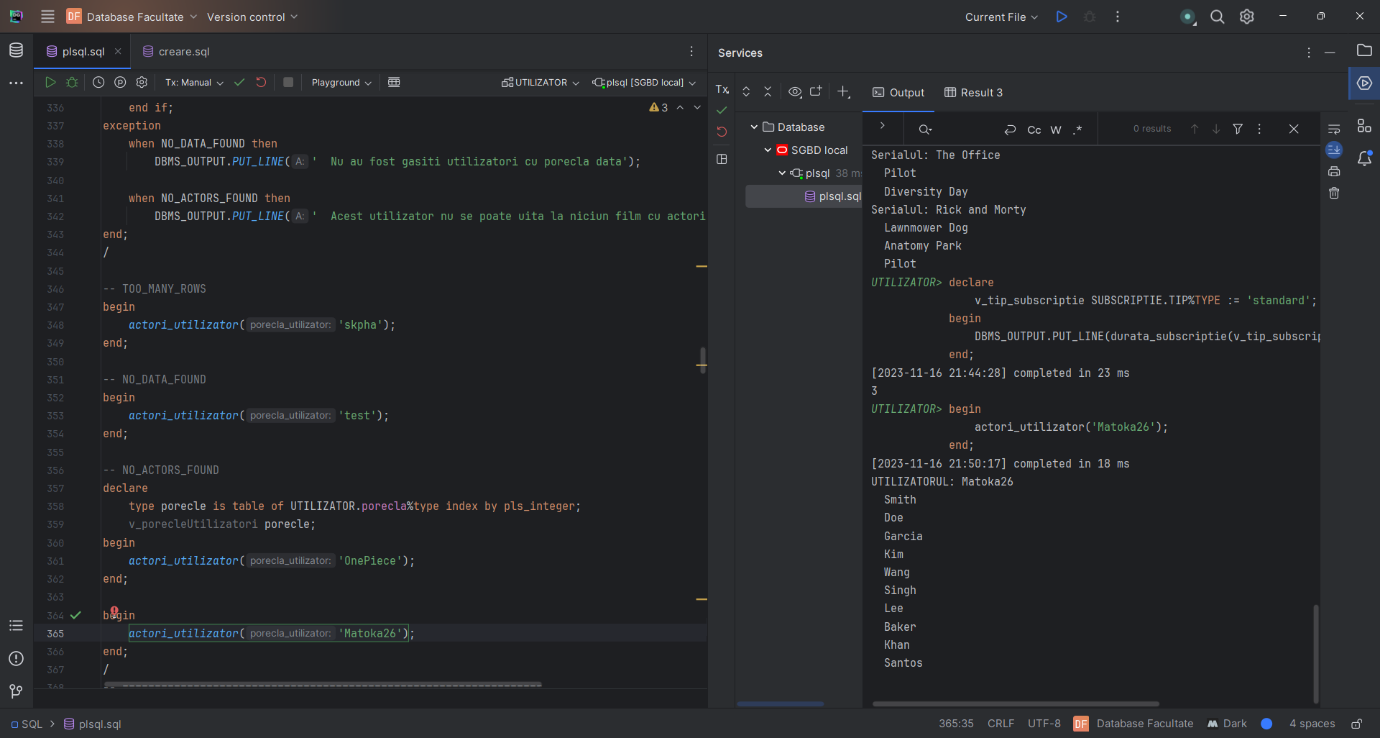
-- ====== EX8 ======  
-- pentru o subscriptie data sa se ia serialul cu durata cea mai mare si  
-- sa se afiseze cati actori are  
CREATE OR REPLACE FUNCTION *durata\_subscriptie*(tip\_subscriptie subscriptie.tip%type) return Number  
AS  
 type tipSubscriptie is table of SUBSCRIPTIE.TIP%type index by pls\_integer;  
  
 v\_subscriptie\_id SUBSCRIPTIE.subscriptie\_id%type;  
 v\_tipuri\_subscriptii tipSubscriptie;  
 v\_hasBeenFound boolean := false;  
 v\_maxSerialDuration number(4) := 0;  
 v\_idSerialMaxDuration SERIAL.SERIAL\_ID%type;  
 v\_countActors number(3) := 0;  
 v\_foundActors boolean := false;  
  
 -- cursor care ne ofera id-ul serialelor din subscriptia data ca parametru si duratat totala  
 -- aflata adunand durata fiecarui episod din acel serial  
 CURSOR durataEpisoade(id\_subscriptie SUBSCRIPTIE.SUBSCRIPTIE\_ID%type) IS  
 SELECT ss.SERIAL\_ID, *SUM*(EP.DURATA) AS suma  
 FROM SUBSCRIPTIE\_SERIAL ss  
 JOIN EPISOD EP ON ss.SERIAL\_ID = EP.SERIAL\_ID  
 JOIN SERIAL S ON ss.SERIAL\_ID = S.SERIAL\_ID  
 WHERE ss.SUBSCRIPTIE\_ID = v\_subscriptie\_id  
 GROUP BY ss.SERIAL\_ID  
 ORDER BY suma DESC;  
  
 -- cursor care ne da pentru fiecare serial cati actori are  
 CURSOR actors IS  
 select sa.SERIAL\_ID id, *count*(\*) nr  
 from ACTOR a  
 Join SERIAL\_ACTOR sa on a.ACTOR\_ID = sa.ACTOR\_ID  
 group by sa.serial\_id;  
  
 type serial\_info is table of durataEpisoade%rowtype index by pls\_integer;  
 v\_infoSeriale serial\_info;  
  
 type actors\_info is table of actors%rowtype index by pls\_integer;  
 v\_infoActors actors\_info;  
  
 -- exceptii  
 NAME\_NOT\_FOUND EXCEPTION;  
 NO\_ACTORS\_FOUND EXCEPTION;  
BEGIN  
 -- selectarea tututor tipurilor si verificarea ca tipul dat ca parametru sa existe  
 -- in aceasta lista  
 select *lower*(tip)  
 bulk collect into v\_tipuri\_subscriptii  
 from SUBSCRIPTIE;  
  
 for i in v\_tipuri\_subscriptii.first..v\_tipuri\_subscriptii.last loop  
 if *lower*(tip\_subscriptie) = v\_tipuri\_subscriptii(i) then  
 v\_hasBeenFound := true;  
 end if;  
 end loop;  
  
 if v\_hasBeenFound = false then  
 RAISE NAME\_NOT\_FOUND;  
 end if;  
 -- ========================  
  
 -- daca avem un tip corect ii aflam id-ul  
 select SUBSCRIPTIE\_ID  
 into v\_subscriptie\_id  
 from SUBSCRIPTIE  
 where *lower*(TIP) = *lower*(tip\_subscriptie);  
 -- ===============  
  
 -- colectam informatia pentru seriale, adica (id, durata totala)  
 OPEN durataEpisoade(v\_subscriptie\_id);  
 FETCH durataEpisoade bulk collect into v\_infoSeriale;  
 CLOSE durataEpisoade;  
  
 -- parcurgem serialele si luam id-ul celui cu cea mai mare durata  
 FOR i IN v\_infoSeriale.FIRST..v\_infoSeriale.LAST LOOP  
 if v\_maxSerialDuration < v\_infoSeriale(i).suma then  
 v\_maxSerialDuration := v\_infoSeriale(i).suma;  
 v\_idSerialMaxDuration := v\_infoSeriale(i).SERIAL\_ID;  
 end if;  
 END LOOP;  
  
 -- colectam informatia pentru actori  
 OPEN actors;  
 FETCH actors bulk collect into v\_infoActors;  
 CLOSE actors;  
  
 -- pentru fiecare serial vedem daca e egal cu cel aflat anterior  
 -- daca da ii dam actualizam countActors  
 for i in v\_infoActors.first..v\_infoActors.last loop  
 if v\_infoActors(i).id = v\_idSerialMaxDuration then  
 v\_foundActors := true;  
 v\_countActors := v\_infoActors(i).nr;  
 end if;  
 end loop;  
  
 if v\_foundActors = false then  
 RAISE NO\_ACTORS\_FOUND;  
 end if;  
  
 return v\_countActors;  
  
exception  
 when NAME\_NOT\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista tipul introdus');  
 return -1;  
  
 when NO\_ACTORS\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu am gasit actori pentru datele cerute');  
 return -1;  
end;  
/  
  
-- Nu au fost gasiti actori  
declare  
 v\_tip\_subscriptie SUBSCRIPTIE.TIP%TYPE := 'basic';  
begin  
 DBMS\_OUTPUT.*PUT\_LINE*(*durata\_subscriptie*(v\_tip\_subscriptie));  
end;  
/  
  
-- Nu exista tipul introdus  
declare  
 v\_tip\_subscriptie SUBSCRIPTIE.TIP%TYPE := 'test';  
begin  
 DBMS\_OUTPUT.*PUT\_LINE*(*durata\_subscriptie*(v\_tip\_subscriptie));  
end;  
/  
  
-- Functioneaza corect  
declare  
 v\_tip\_subscriptie SUBSCRIPTIE.TIP%TYPE := 'standard';  
begin  
 DBMS\_OUTPUT.*PUT\_LINE*(*durata\_subscriptie*(v\_tip\_subscriptie));  
end;  
/  
-- =================================================================

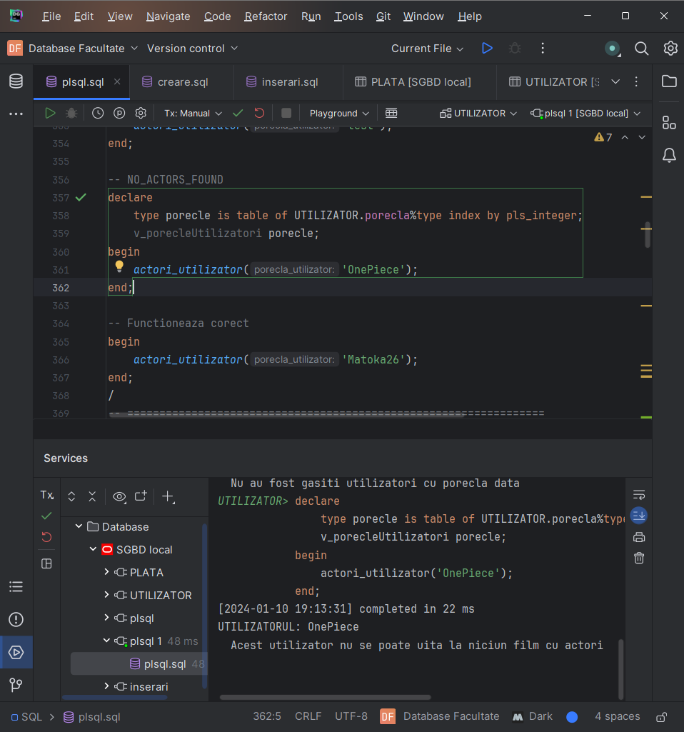
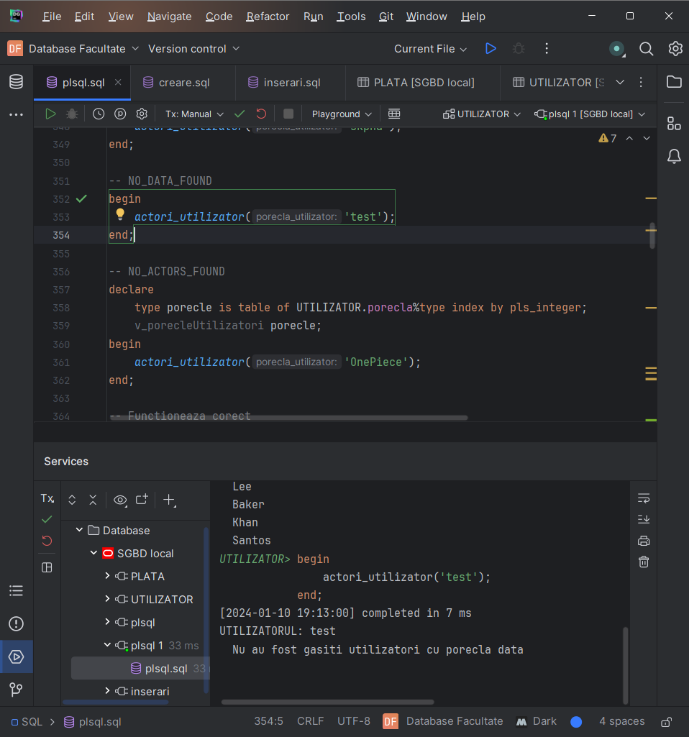


# Subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă sql 5 tabele + 2 tratare excepții

Cerință: se dă porecla unui utilizator, aflați toți actorii la care se poate uita acesta

-- ====== EX9 ======  
-- ni se da porcela unui utilizator, aflati toti actorii la care se poate uita  
CREATE OR REPLACE PROCEDURE *actori\_utilizator*(porecla\_utilizator UTILIZATOR.PORECLA%TYPE)  
AS  
 v\_idUtilizator UTILIZATOR.UTILIZATOR\_ID%TYPE;  
 v\_firstIdUtilizator UTILIZATOR.UTILIZATOR\_ID%TYPE;  
 v\_actorFound boolean := false;  
  
 -- distinct pentru ca un actor poate juca in mai multe filme  
 CURSOR getActori(idUtilizator UTILIZATOR.UTILIZATOR\_ID%TYPE) IS  
 select DISTINCT NUME  
 from ACTOR a  
 JOIN ROL\_JUCAT rl on a.ACTOR\_ID = rl.ACTOR\_ID  
 JOIN FILM f on rl.FILM\_ID = f.FILM\_ID  
 JOIN SUBSCRIPTIE\_FILM sf on f.FILM\_ID = sf.FILM\_ID  
 JOIN SUBSCRIPTIE s on sf.SUBSCRIPTIE\_ID = s.SUBSCRIPTIE\_ID  
 JOIN UTILIZATOR u on s.SUBSCRIPTIE\_ID = u.SUBSCRIPTIE\_ID  
 WHERE u.UTILIZATOR\_ID = idUtilizator;  
  
 NO\_ACTORS\_FOUND EXCEPTION;  
  
BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('UTILIZATORUL: ' || porecla\_utilizator);  
  
 -- ia id-ul utilizatorului cu porecla data | TOO\_MANY\_ROWS/NO\_DATA\_FOUND  
 -- alt bloc in caz ca exista doi sau mai multi utilizatori cu aceeasi porecla  
 -- daca se intampla asta il luam pe primul  
 BEGIN  
 select UTILIZATOR\_ID  
 into v\_idUtilizator  
 from UTILIZATOR  
 where *lower*(porecla\_utilizator) = *lower*(porecla);  
 EXCEPTION  
 when TOO\_MANY\_ROWS then  
 SELECT UTILIZATOR\_ID  
 INTO v\_firstIdUtilizator  
 FROM (  
 SELECT UTILIZATOR\_ID  
 FROM UTILIZATOR  
 WHERE *lower*(porecla\_utilizator) = *lower*(porecla)  
 AND ROWNUM = 1  
 );  
 v\_idUtilizator := v\_firstIdUtilizator;  
 end;  
  
 -- daca trecem de toate verificarile afisam toti actorii utilizatorului  
 for i in getActori(v\_idUtilizator) loop  
 v\_actorFound := true;  
 DBMS\_OUTPUT.*PUT\_LINE*(' ' || i.NUME);  
 end loop;  
  
 -- verificam daca am gasit actori in cursor  
 if v\_actorFound = false then RAISE NO\_ACTORS\_FOUND;  
 end if;  
exception  
 when NO\_DATA\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*(' Nu au fost gasiti utilizatori cu porecla data');  
  
 when NO\_ACTORS\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*(' Acest utilizator nu se poate uita la niciun film cu actori');  
end;  
/  
  
-- TOO\_MANY\_ROWS  
begin  
 *actori\_utilizator*('skpha');  
end;  
  
-- NO\_DATA\_FOUND  
begin  
 *actori\_utilizator*('test');  
end;  
  
-- NO\_ACTORS\_FOUND  
declare  
 type porecle is table of UTILIZATOR.porecla%type index by pls\_integer;  
 v\_porecleUtilizatori porecle;  
begin  
 *actori\_utilizator*('OnePiece');  
end;  
-- =================================================================

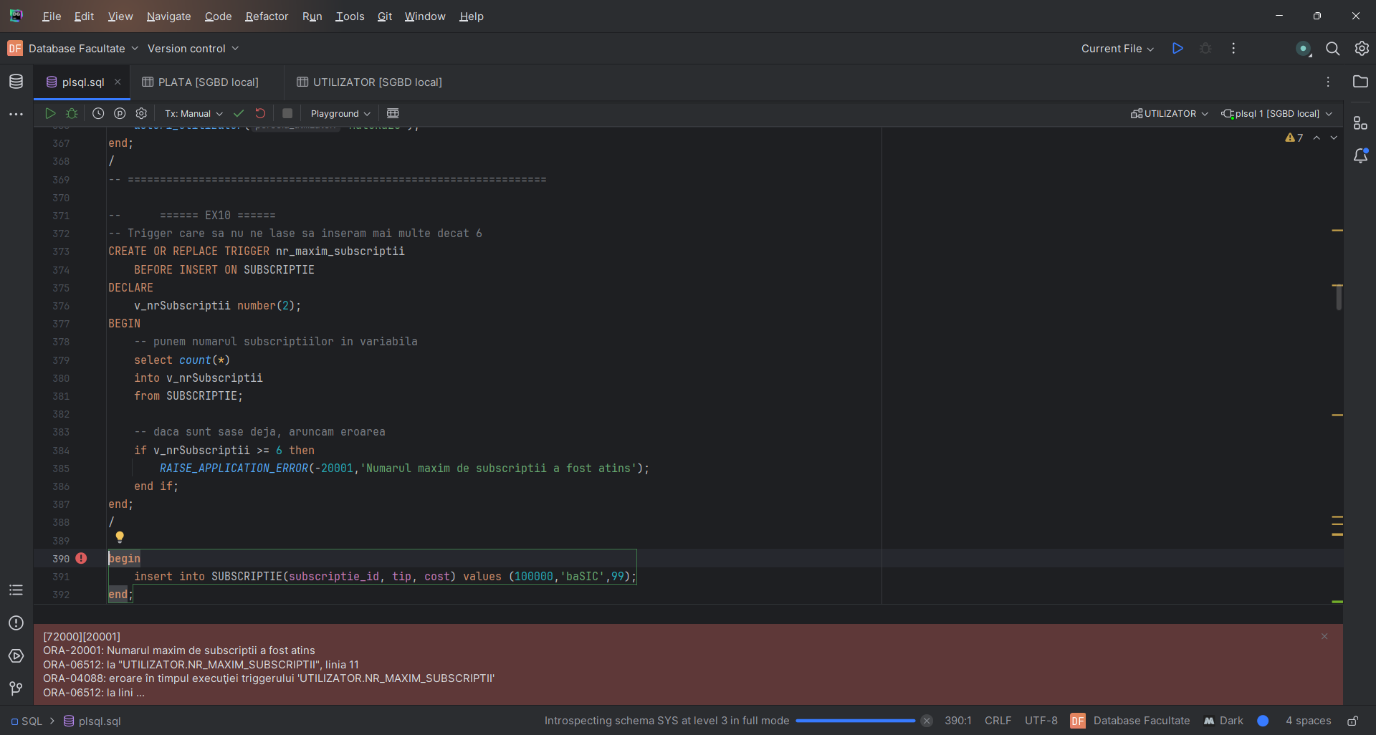




# Definiți un trigger de tip LMD la nivel de comandă

Cerință: un trigger care să nu ne lase să inserăm mai mult de 6 subscripții

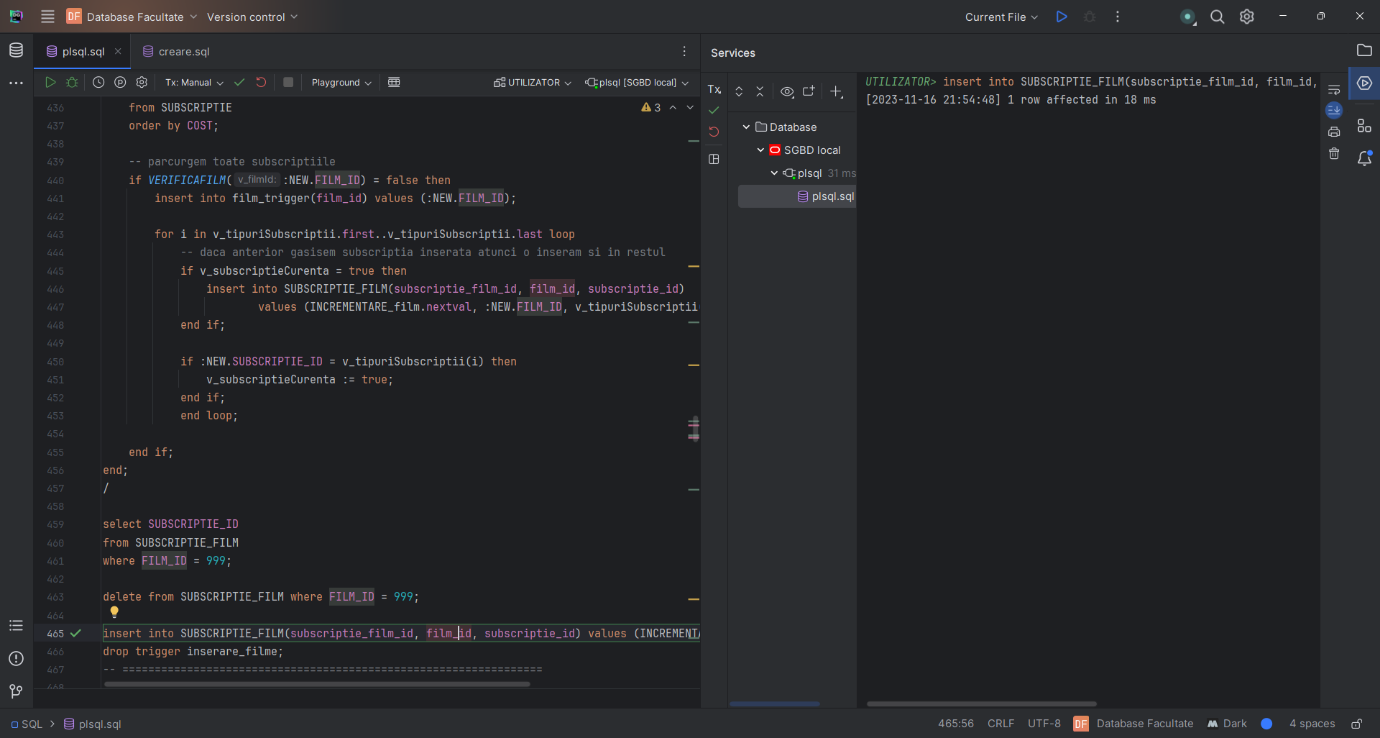
CREATE OR REPLACE TRIGGER nr\_maxim\_subscriptii  
 BEFORE INSERT ON SUBSCRIPTIE  
DECLARE  
 v\_nrSubscriptii number(2);  
BEGIN  
 -- punem numarul subscriptiilor in variabila  
 select *count*(\*)  
 into v\_nrSubscriptii  
 from SUBSCRIPTIE;  
  
 -- daca sunt sase deja, aruncam eroarea  
 if v\_nrSubscriptii >= 6 then  
 *RAISE\_APPLICATION\_ERROR*(-20001,'Numarul maxim de subscriptii a fost atins');  
 end if;  
end;  
/  
  
begin  
 insert into SUBSCRIPTIE(subscriptie\_id, tip, cost) values (100000,'baSIC',99);  
end;  
/  
  
drop trigger nr\_maxim\_subscriptii;  
select *count*(\*) from SUBSCRIPTIE;  
delete from SUBSCRIPTIE where SUBSCRIPTIE\_ID = 99999;  
-- =================================================================



# Definiți un trigger de tip LMD la nivel de linie

Cerință: un trigger care atunci când se inserează un film nou într-o subscripție, acesta să fie inserat și în subscripțiile superioare din punct de vedere ierarhic

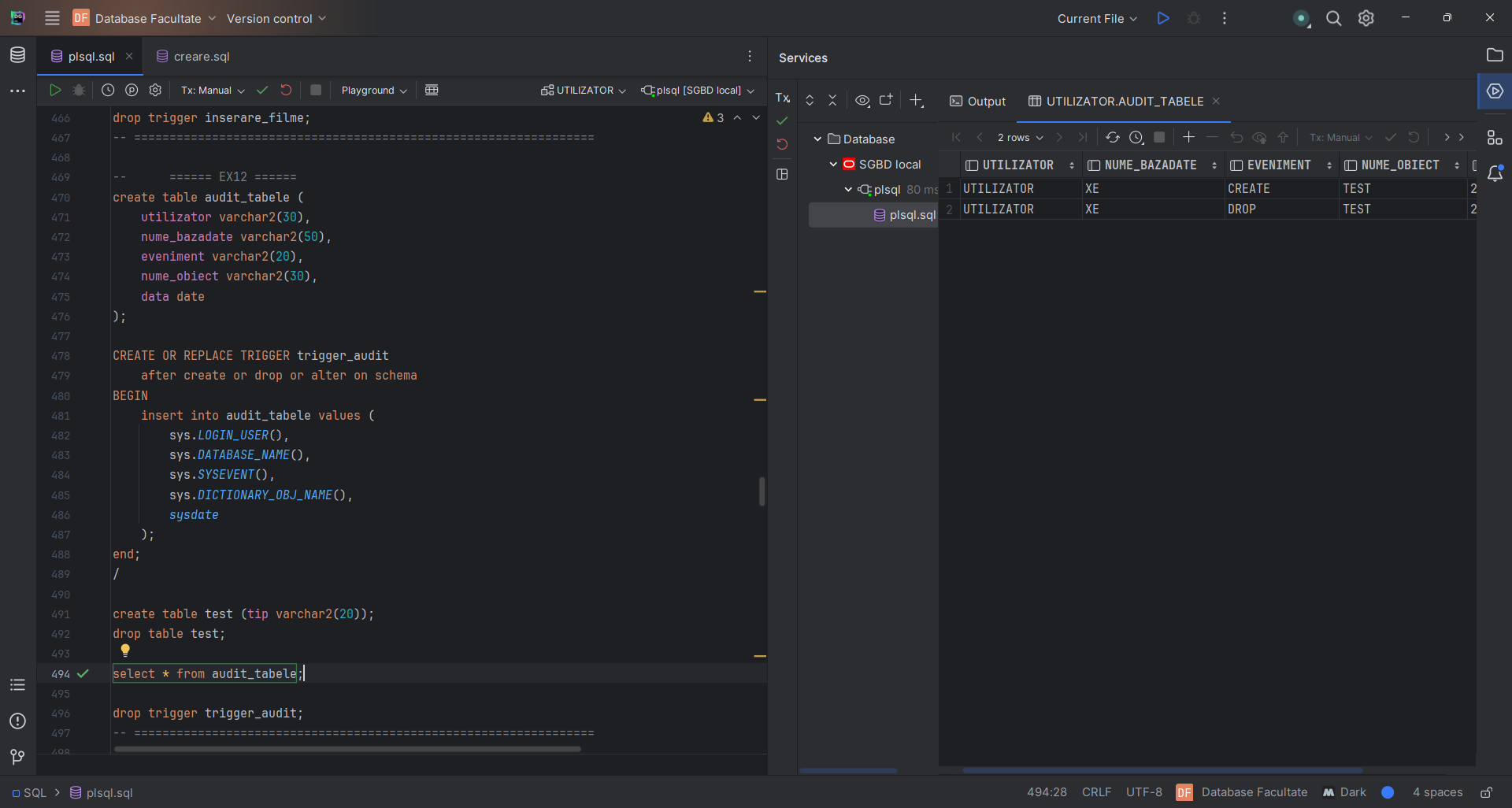
-- ====== EX11 ======  
create or replace type subscriptii as varray(6) of number(6);  
  
-- tabela pentru a retine daca filmul a fost deja inserate de trigger  
create table film\_trigger (  
 film\_id number(6)  
);  
  
-- functie ajutatoare pentru a verifica daca un film se afla in film\_trigger  
CREATE OR REPLACE FUNCTION *verificaFilm*(v\_filmId FILM.FILM\_ID%TYPE) return boolean AS  
 v\_count number(2) := 0;  
BEGIN  
 select *count*(\*)  
 into v\_count  
 from film\_trigger  
 where film\_id = v\_filmId;  
  
 if v\_count > 0 then  
 return TRUE;  
 end if;  
  
 return false;  
END;  
/  
  
CREATE OR REPLACE TRIGGER inserare\_filme  
 BEFORE INSERT  
 ON SUBSCRIPTIE\_FILM  
 FOR EACH ROW  
DECLARE  
 v\_tipuriSubscriptii subscriptii;  
 v\_subscriptieCurenta boolean := false;  
BEGIN  
 -- obtinem id-urile subscriptiilor in functie de cost  
 -- asta pentru a le avea in ordinea ierarhica corecta  
 select SUBSCRIPTIE\_ID  
 bulk collect into v\_tipuriSubscriptii  
 from SUBSCRIPTIE  
 order by COST;  
  
 -- parcurgem toate subscriptiile  
 if *VERIFICAFILM*(:NEW.FILM\_ID) = false then  
 insert into film\_trigger(film\_id) values (:NEW.FILM\_ID);  
  
 for i in v\_tipuriSubscriptii.first..v\_tipuriSubscriptii.last loop  
 -- daca anterior gasisem subscriptia inserata atunci o inseram si in restul  
 if v\_subscriptieCurenta = true then  
 insert into SUBSCRIPTIE\_FILM(subscriptie\_film\_id, film\_id, subscriptie\_id)  
 values (INCREMENTARE\_film.nextval, :NEW.FILM\_ID, v\_tipuriSubscriptii(i));  
 end if;  
  
 if :NEW.SUBSCRIPTIE\_ID = v\_tipuriSubscriptii(i) then  
 v\_subscriptieCurenta := true;  
 end if;  
 end loop;  
  
 end if;  
end;  
/  
  
select SUBSCRIPTIE\_ID  
from SUBSCRIPTIE\_FILM  
where FILM\_ID = 999;  
  
delete from SUBSCRIPTIE\_FILM where FILM\_ID = 999;  
  
insert into SUBSCRIPTIE\_FILM(subscriptie\_film\_id, film\_id, subscriptie\_id) values (INCREMENTARE\_FILM.nextval,999,55245);  
drop trigger inserare\_filme;  
-- =================================================================



# Definiți un trigger de tip LDD

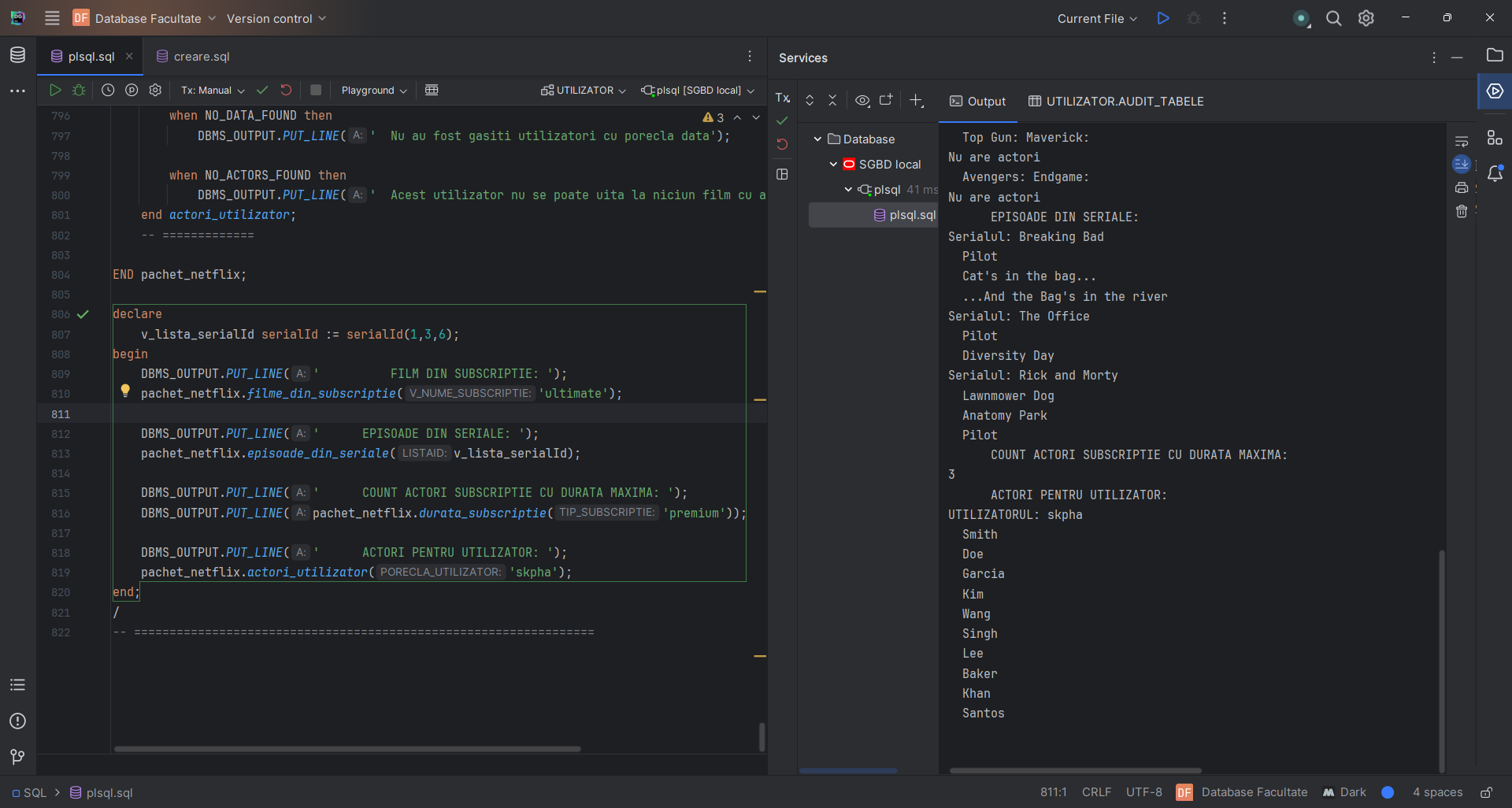
Cerință: un trigger care reține actiunile făcute în baza de date

-- ====== EX12 ======  
create table audit\_tabele (  
 utilizator varchar2(30),  
 nume\_bazadate varchar2(50),  
 eveniment varchar2(20),  
 nume\_obiect varchar2(30),  
 data date  
);  
  
CREATE OR REPLACE TRIGGER trigger\_audit  
 after create or drop or alter on schema  
BEGIN  
 insert into audit\_tabele values (  
 sys.*LOGIN\_USER*(),  
 sys.*DATABASE\_NAME*(),  
 sys.*SYSEVENT*(),  
 sys.*DICTIONARY\_OBJ\_NAME*(),  
 *sysdate* );  
end;  
/  
  
create table test (tip varchar2(20));  
drop table test;  
  
select \* from audit\_tabele;  
  
drop trigger trigger\_audit;  
-- =================================================================



# Definiți un pachet care să conțină toate obiectele definite

-- ====== EX13 ======  
CREATE OR REPLACE PACKAGE pachet\_netflix AS  
 -- 6  
 PROCEDURE *filme\_din\_subscriptie*(v\_nume\_subscriptie SUBSCRIPTIE.TIP%TYPE);  
 -- 7 functie ajutatoare + procedura  
 FUNCTION *verifica\_serial*(v\_serialId SERIAL.SERIAL\_ID%TYPE, listaId serialId) RETURN NUMBER;  
 PROCEDURE *episoade\_din\_seriale*(listaId serialId);  
 -- 8  
 FUNCTION *durata\_subscriptie*(tip\_subscriptie subscriptie.tip%type) RETURN NUMBER;  
 -- 9  
 PROCEDURE *actori\_utilizator*(porecla\_utilizator UTILIZATOR.PORECLA%TYPE);  
END pachet\_netflix;  
  
CREATE OR REPLACE PACKAGE BODY pachet\_netflix AS  
 -- ===== 6 =====  
 PROCEDURE *filme\_din\_subscriptie*(v\_nume\_subscriptie SUBSCRIPTIE.TIP%TYPE)  
 AS  
 -- tablou indexat care retine numele filmelor  
 type filme is table of FILM.denumire%type index by pls\_integer;  
 v\_filme filme;  
  
 -- tablou imbricat care retine id-ul filmelor  
 type filme\_id is table of ACTOR.nume%type;  
 v\_filme\_id filme\_id;  
  
 -- varray care contine numele actorilor  
 v\_actori actori;  
  
 v\_subscriptie\_id SUBSCRIPTIE.subscriptie\_id%type;  
  
 BEGIN  
 -- obtine id-ul subscriptiei  
 select unique SUBSCRIPTIE\_ID  
 into v\_subscriptie\_id  
 from SUBSCRIPTIE  
 where TIP = v\_nume\_subscriptie;  
  
 -- obtin toate numele filmelor si id-urile lor in tabloul indexat v\_filme/v\_filme\_id  
 select DENUMIRE, FILM.FILM\_ID  
 bulk collect into v\_filme, v\_filme\_id  
 from FILM  
 join SUBSCRIPTIE\_FILM on FILM.FILM\_ID = SUBSCRIPTIE\_FILM.FILM\_ID  
 where SUBSCRIPTIE\_FILM.SUBSCRIPTIE\_ID = v\_subscriptie\_id;  
  
 DBMS\_OUTPUT.*PUT\_LINE*(' Subscriptia ' || v\_nume\_subscriptie || ' contine:');  
  
 for i in v\_filme\_id.first..v\_filme\_id.last loop  
 DBMS\_OUTPUT.*PUT\_LINE*(' ' || v\_filme(i) || ': ');  
  
 -- pentru fiecare film selectam actorii care joaca in el  
 select nume  
 bulk collect into v\_actori  
 from ACTOR  
 join ROL\_JUCAT on ACTOR.ACTOR\_ID = ROL\_JUCAT.ACTOR\_ID  
 where ROL\_JUCAT.FILM\_ID = v\_filme\_id(i);  
  
 if v\_actori.COUNT > 0 then  
 for j in v\_actori.first..v\_actori.last loop  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_actori(j));  
 end loop;  
  
 v\_actori.DELETE;  
 else  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu are actori');  
 end if;  
  
 end loop;  
 END *filme\_din\_subscriptie*;  
 -- =============  
  
 -- ===== 7 =====  
 function *verifica\_serial*(v\_serialId SERIAL.SERIAL\_ID%TYPE, listaId serialId) RETURN NUMBER AS  
 v\_found number(1) := 0;  
 begin  
 for i in 1..listaId.COUNT loop  
 if v\_serialId = listaId(i) then  
 v\_found := 1;  
 return v\_found;  
 end if;  
 end loop;  
  
 return v\_found;  
 end *verifica\_serial*;  
  
 PROCEDURE *episoade\_din\_seriale*(listaId serialId) AS  
 v\_id\_ser SERIAL.serial\_id%type;  
  
 v\_nume\_ser SERIAL.denumire%type;  
 v\_nume\_episod varchar2(50);  
 v\_areEpisod number(1) := 0;  
  
 -- cursor clasic  
 CURSOR seriale IS  
 select SERIAL\_ID, DENUMIRE  
 from SERIAL  
 WHERE *verifica\_serial*(SERIAL\_ID, listaId) = 1;  
  
 -- cursor parametrizat dependent de cel anterior  
 CURSOR episod(id number) IS  
 select DENUMIRE  
 from EPISOD  
 where SERIAL\_ID = id;  
  
 begin  
 OPEN seriale;  
 loop  
 Fetch seriale into v\_id\_ser, v\_nume\_ser;  
 exit when seriale%notfound;  
  
 v\_areEpisod := 0;  
  
 DBMS\_OUTPUT.*PUT\_LINE*('Serialul: ' || v\_nume\_ser);  
  
 -- deschidem noul cursor cu parametrul din cursorul anterior  
 OPEN episod(v\_id\_ser);  
 loop  
 Fetch episod into v\_nume\_episod;  
 exit when EPISOD%notfound;  
 v\_areEpisod := 1;  
 DBMS\_OUTPUT.*PUT\_LINE*(' ' || v\_nume\_episod);  
 end loop;  
  
 close episod;  
  
 if v\_areEpisod = 0 then DBMS\_OUTPUT.*PUT\_LINE*(' Nu are episoade');  
 end if;  
  
 end loop;  
 close seriale;  
 end *episoade\_din\_seriale*;  
 -- =============  
  
 -- ===== 8 =====  
 FUNCTION *durata\_subscriptie*(tip\_subscriptie subscriptie.tip%type) return Number  
 AS  
 type tipSubscriptie is table of SUBSCRIPTIE.TIP%type index by pls\_integer;  
  
 v\_subscriptie\_id SUBSCRIPTIE.subscriptie\_id%type;  
 v\_tipuri\_subscriptii tipSubscriptie;  
 v\_hasBeenFound boolean := false;  
 v\_maxSerialDuration number(4) := 0;  
 v\_idSerialMaxDuration SERIAL.SERIAL\_ID%type;  
 v\_countActors number(3) := 0;  
 v\_foundActors boolean := false;  
  
 -- cursor care ne ofera id-ul serialelor din subscriptia data ca parametru si duratat totala  
 -- aflata adunand durata fiecarui episod din acel serial  
 CURSOR durataEpisoade(id\_subscriptie SUBSCRIPTIE.SUBSCRIPTIE\_ID%type) IS  
 SELECT ss.SERIAL\_ID, *SUM*(EP.DURATA) AS suma  
 FROM SUBSCRIPTIE\_SERIAL ss  
 JOIN EPISOD EP ON ss.SERIAL\_ID = EP.SERIAL\_ID  
 JOIN SERIAL S ON ss.SERIAL\_ID = S.SERIAL\_ID  
 WHERE ss.SUBSCRIPTIE\_ID = v\_subscriptie\_id  
 GROUP BY ss.SERIAL\_ID  
 ORDER BY suma DESC;  
  
 -- cursor care ne da pentru fiecare serial cati actori are  
 -- sau e bine si asa?  
 CURSOR actors IS  
 select sa.SERIAL\_ID id, *count*(\*) nr  
 from ACTOR a  
 Join SERIAL\_ACTOR sa on a.ACTOR\_ID = sa.ACTOR\_ID  
 group by sa.serial\_id;  
  
 type serial\_info is table of durataEpisoade%rowtype index by pls\_integer;  
 v\_infoSeriale serial\_info;  
  
 type actors\_info is table of actors%rowtype index by pls\_integer;  
 v\_infoActors actors\_info;  
  
 -- exceptii  
 NAME\_NOT\_FOUND EXCEPTION;  
 NO\_ACTORS\_FOUND EXCEPTION;  
 BEGIN  
 -- selectarea tututor tipurilor si verificarea ca tipul dat ca parametru sa existe  
 -- in aceasta lista  
 select *lower*(tip)  
 bulk collect into v\_tipuri\_subscriptii  
 from SUBSCRIPTIE;  
  
 for i in v\_tipuri\_subscriptii.first..v\_tipuri\_subscriptii.last loop  
 if *lower*(tip\_subscriptie) = v\_tipuri\_subscriptii(i) then  
 v\_hasBeenFound := true;  
 end if;  
 end loop;  
  
 if v\_hasBeenFound = false then  
 RAISE NAME\_NOT\_FOUND;  
 end if;  
 -- ========================  
  
 -- daca avem un tip corect ii aflam id-ul  
 select SUBSCRIPTIE\_ID  
 into v\_subscriptie\_id  
 from SUBSCRIPTIE  
 where *lower*(TIP) = *lower*(tip\_subscriptie);  
 -- ===============  
  
 -- colectam informatia pentru seriale, adica (id, durata totala)  
 OPEN durataEpisoade(v\_subscriptie\_id);  
 FETCH durataEpisoade bulk collect into v\_infoSeriale;  
 CLOSE durataEpisoade;  
  
 -- parcurgem serialele si luam id-ul celui cu cea mai mare durata  
 FOR i IN v\_infoSeriale.FIRST..v\_infoSeriale.LAST LOOP  
 if v\_maxSerialDuration < v\_infoSeriale(i).suma then  
 v\_maxSerialDuration := v\_infoSeriale(i).suma;  
 v\_idSerialMaxDuration := v\_infoSeriale(i).SERIAL\_ID;  
 end if;  
 END LOOP;  
  
 -- colectam informatia pentru actori  
 OPEN actors;  
 FETCH actors bulk collect into v\_infoActors;  
 CLOSE actors;  
  
 -- pentru fiecare serial vedem daca e egal cu cel aflat anterior  
 -- daca da ii dam actualizam countActors  
 for i in v\_infoActors.first..v\_infoActors.last loop  
 if v\_infoActors(i).id = v\_idSerialMaxDuration then  
 v\_foundActors := true;  
 v\_countActors := v\_infoActors(i).nr;  
 end if;  
 end loop;  
  
 if v\_foundActors = false then  
 RAISE NO\_ACTORS\_FOUND;  
 end if;  
  
 return v\_countActors;  
  
 exception  
 when NAME\_NOT\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista tipul introdus');  
 return -1;  
  
 when NO\_ACTORS\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu am gasit actori pentru datele cerute');  
 return -1;  
  
 end *durata\_subscriptie*;  
 -- =============  
  
 -- ===== 9 =====  
 PROCEDURE *actori\_utilizator*(porecla\_utilizator UTILIZATOR.PORECLA%TYPE)  
 AS  
 v\_idUtilizator UTILIZATOR.UTILIZATOR\_ID%TYPE;  
 v\_firstIdUtilizator UTILIZATOR.UTILIZATOR\_ID%TYPE;  
 v\_actorFound boolean := false;  
  
 -- distinct pentru ca un actor poate juca in mai multe filme  
 CURSOR getActori(idUtilizator UTILIZATOR.UTILIZATOR\_ID%TYPE) IS  
 select DISTINCT NUME  
 from ACTOR a  
 JOIN ROL\_JUCAT rl on a.ACTOR\_ID = rl.ACTOR\_ID  
 JOIN FILM f on rl.FILM\_ID = f.FILM\_ID  
 JOIN SUBSCRIPTIE\_FILM sf on f.FILM\_ID = sf.FILM\_ID  
 JOIN SUBSCRIPTIE s on sf.SUBSCRIPTIE\_ID = s.SUBSCRIPTIE\_ID  
 JOIN UTILIZATOR u on s.SUBSCRIPTIE\_ID = u.SUBSCRIPTIE\_ID  
 WHERE u.UTILIZATOR\_ID = idUtilizator;  
  
 NO\_ACTORS\_FOUND EXCEPTION;  
  
 BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*('UTILIZATORUL: ' || porecla\_utilizator);  
  
 -- ia id-ul utilizatorului cu porecla data | TOO\_MANY\_ROWS/NO\_DATA\_FOUND  
 -- alt bloc in caz ca exista doi sau mai multi utilizatori cu aceeasi porecla  
 -- daca se intampla asta il luam pe primul  
 BEGIN  
 select UTILIZATOR\_ID  
 into v\_idUtilizator  
 from UTILIZATOR  
 where *lower*(porecla\_utilizator) = *lower*(porecla);  
 EXCEPTION  
 when TOO\_MANY\_ROWS then  
 SELECT UTILIZATOR\_ID  
 INTO v\_firstIdUtilizator  
 FROM (  
 SELECT UTILIZATOR\_ID  
 FROM UTILIZATOR  
 WHERE *lower*(porecla\_utilizator) = *lower*(porecla)  
 AND ROWNUM = 1  
 );  
 v\_idUtilizator := v\_firstIdUtilizator;  
 end;  
  
 -- daca trecem de toate verificarile afisam toti actorii utilizatorului  
 for i in getActori(v\_idUtilizator) loop  
 v\_actorFound := true;  
 DBMS\_OUTPUT.*PUT\_LINE*(' ' || i.NUME);  
 end loop;  
  
 -- verificam daca am gasit actori in cursor  
 if v\_actorFound = false then RAISE NO\_ACTORS\_FOUND;  
 end if;  
 exception  
 when NO\_DATA\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*(' Nu au fost gasiti utilizatori cu porecla data');  
  
 when NO\_ACTORS\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*(' Acest utilizator nu se poate uita la niciun film cu actori');  
 end *actori\_utilizator*;  
 -- =============  
  
END pachet\_netflix;  
  
declare  
 v\_lista\_serialId serialId := serialId(1,3,6);  
begin  
 DBMS\_OUTPUT.*PUT\_LINE*(' FILM DIN SUBSCRIPTIE: ');  
 pachet\_netflix.*filme\_din\_subscriptie*('ultimate');  
  
 DBMS\_OUTPUT.*PUT\_LINE*(' EPISOADE DIN SERIALE: ');  
 pachet\_netflix.*episoade\_din\_seriale*(v\_lista\_serialId);  
  
 DBMS\_OUTPUT.*PUT\_LINE*(' COUNT ACTORI SUBSCRIPTIE CU DURATA MAXIMA: ');  
 DBMS\_OUTPUT.*PUT\_LINE*(pachet\_netflix.*durata\_subscriptie*('premium'));  
  
 DBMS\_OUTPUT.*PUT\_LINE*(' ACTORI PENTRU UTILIZATOR: ');  
 pachet\_netflix.*actori\_utilizator*('skpha');  
end;  
/  
-- =================================================================



# Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite

## Motivație:

La acest exercitiu eu m-am gandit să creez un pachet care să faciliteze diferite operații pentru un anumit Utilizator.

## Funcționalități:

* verifică\_subscripție = functie care returnează true/false în cazul în care data de expirare a subscripției unui utilizator nu a expirat încă
* inserare\_plată = face inserările necesare pentru a adăuga o plată nouă pentru un utilizator
* verifică\_aniversare = dacă data când contul utilizatorului a fost creat este aceeași cu data din ziua curentă atunci se adaugă o lună gratis la abonamentul utilizatorului
* toate\_filmele = procedură care se utilizează de cursorul filme\_actori pentru a afișa toate filmele la care se poate uita un utilizator
* toate\_serialele = procedură care se utilizează de cursorul seriale\_actori pentru a afișa toate serialele la care se poate uita un utilizator
* toți\_actorii = procedură care se utilizează de cursoarele filme\_actori și seriale\_actori pentru a face a reunii si afișa toți actorii la care se poate uita un utilizator
* film\_notă\_maximă = funcție ce returneaza un record film\_record cu filmul cu notă maximă la care se poate uita un utilizator
* filtrează\_filme = funcție ce returnează un cursor care conține toate filmele la care se poate uita un utilizator, doar că filtrate in funcție de parametrii transmiși.

## Cod:

-- ====== EX14 ======  
-- care rep. toate filmele/serialele si actorii la care se poate uita un user  
 -- [X] 2 proceduri, toate filmele/serialele  
 -- [X] 1 procedura toti actorii  
 -- [X] 1 function cu record pentru a obtine filmul cu nota maxima  
 -- [X] 1 functie filtrare, cu notamin notamax parametrii default restul bool uri (ordonare cresc sau nu)  
CREATE OR REPLACE PACKAGE pachet\_utilizator AS  
 -- verifica daca un utilizator are un abonament valid (data de expirare a subscriptiei nu a expirat inca)  
 function *verifica\_subscriptie*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE) RETURN BOOLEAN;  
 -- face insert-urile necesare pentru o plata a unui utilizator  
 procedure *inserare\_plata*(v\_utilizatorId UTILIZATOR.UTILIZATOR\_ID%TYPE,  
 v\_plataId PLATA.PLATA\_ID%TYPE,  
 v\_nume PLATA.NUME%TYPE,  
 v\_prenume PLATA.PRENUME%TYPE,  
 V\_cod PLATA.COD%TYPE,  
 v\_data\_exp PLATA.DATA\_EXP%TYPE,  
 v\_cvv PLATA.CVV%TYPE);  
 -- daca data curenta este aceeasi cu data cand utilizatorul si a facut contul ii oferim o luna gratis  
 procedure *verifica\_aniversare*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE);  
  
 -- 2 cursoare, unul care ia toate filmele si actorii la care se poate uita un user  
 -- si celalalt toate seriale si actorii respectivi la care se poate uita un user  
 CURSOR filme\_actori(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE) IS  
 SELECT f.DENUMIRE AS FILM, a.NUME AS ACTOR, f.NOTA as NOTA  
 FROM UTILIZATOR u  
 JOIN SUBSCRIPTIE s ON u.SUBSCRIPTIE\_ID = s.SUBSCRIPTIE\_ID  
 JOIN SUBSCRIPTIE\_FILM sf ON s.SUBSCRIPTIE\_ID = sf.SUBSCRIPTIE\_ID  
 JOIN FILM f ON sf.FILM\_ID = f.FILM\_ID  
 LEFT JOIN ROL\_JUCAT rl ON f.FILM\_ID = rl.FILM\_ID  
 LEFT JOIN ACTOR a ON rl.ACTOR\_ID = a.ACTOR\_ID  
 WHERE u.UTILIZATOR\_ID = v\_userId;  
  
 CURSOR seriale\_actori(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE) IS  
 SELECT ser.DENUMIRE AS SERIAL, a.NUME AS ACTOR  
 FROM UTILIZATOR u  
 JOIN SUBSCRIPTIE s ON u.SUBSCRIPTIE\_ID = s.SUBSCRIPTIE\_ID  
 JOIN SUBSCRIPTIE\_SERIAL ss ON s.SUBSCRIPTIE\_ID = ss.SUBSCRIPTIE\_ID  
 JOIN SERIAL ser ON ss.SERIAL\_ID = ser.SERIAL\_ID  
 LEFT JOIN SERIAL\_ACTOR sa ON ser.SERIAL\_ID = sa.SERIAL\_ID  
 LEFT JOIN ACTOR a ON sa.ACTOR\_ID = a.ACTOR\_ID  
 WHERE u.UTILIZATOR\_ID = v\_userId;  
  
 -- ia toate filmele distincte din cursorul filme\_actori  
 procedure *toate\_filmele*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE);  
 -- ia toate serialele distincte din cursorul seriale\_actori  
 procedure *toate\_serialele*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE);  
 -- ia toti actorii distincti din cursorul filme\_actori reunit cu cursorul seriale\_actori  
 procedure *toti\_actorii*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE);  
  
 -- tip de record pentru a retine numele filmului si nota lui maxima  
 type film\_record is record(nume film.denumire%type, nota film.nota%type);  
 -- functie de returnare a filmului cu nota maxima la care se poate uita un utilizator  
 function *film\_nota\_maxima*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE) return film\_record;  
  
 -- filtrare filme pe care le poate vedea un utilizator  
 TYPE cursor\_filtrare IS REF CURSOR RETURN FILM%ROWTYPE;  
 function *filtreaza\_filme*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE,  
 v\_nota\_min FILM.NOTA%TYPE DEFAULT 0,  
 v\_nota\_max FILM.NOTA%TYPE DEFAULT 10,  
 v\_crescator boolean DEFAULT false,  
 v\_descrescator boolean DEFAULT false)  
 return cursor\_filtrare;  
END pachet\_utilizator;  
  
CREATE OR REPLACE PACKAGE BODY pachet\_utilizator AS  
 function *verifica\_subscriptie*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE) RETURN BOOLEAN AS  
 v\_expired boolean := false;  
 v\_expiry\_date UTILIZATOR.DATA\_EXP\_SUB%TYPE;  
 begin  
 select DATA\_EXP\_SUB  
 into v\_expiry\_date  
 from UTILIZATOR  
 where UTILIZATOR.UTILIZATOR\_ID = v\_userId;  
  
 if v\_expiry\_date < *SYSDATE* then  
 v\_expired := true;  
 end if;  
  
 return v\_expired;  
  
 EXCEPTION  
 WHEN NO\_DATA\_FOUND then  
 return false;  
  
 END *verifica\_subscriptie*;  
  
 procedure *inserare\_plata*(v\_utilizatorId UTILIZATOR.UTILIZATOR\_ID%TYPE,  
 v\_plataId PLATA.PLATA\_ID%TYPE,  
 v\_nume PLATA.NUME%TYPE,  
 v\_prenume PLATA.PRENUME%TYPE,  
 V\_cod PLATA.COD%TYPE,  
 v\_data\_exp PLATA.DATA\_EXP%TYPE,  
 v\_cvv PLATA.CVV%TYPE)  
 AS  
  
 BEGIN  
 insert into PLATA values(v\_plataId, v\_nume, v\_prenume, v\_cod, v\_data\_exp, v\_cvv);  
 insert into LISTA\_CARDURI values(INCREMENTARE\_LISTA\_CARDURI.nextval, v\_plataId, v\_utilizatorId);  
 END *inserare\_plata*;  
  
 procedure *verifica\_aniversare*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE) AS  
 v\_expiry\_date UTILIZATOR.DATA\_EXP\_SUB%TYPE;  
 different\_date exception;  
 BEGIN  
 select DATA\_EXP\_SUB  
 into v\_expiry\_date  
 from UTILIZATOR  
 where v\_userId = UTILIZATOR.UTILIZATOR\_ID;  
  
 if *to\_char*(v\_expiry\_date,'DD/MM/YYYY') = *to\_char*(*SYSDATE*,'DD/MM/YYYY') then  
 v\_expiry\_date := *ADD\_MONTHS*(v\_expiry\_date, 1);  
  
 UPDATE UTILIZATOR  
 SET DATA\_EXP\_SUB = v\_expiry\_date  
 WHERE UTILIZATOR\_ID = v\_userId;  
 else  
 RAISE different\_date;  
 end if;  
  
 EXCEPTION  
 WHEN NO\_DATA\_FOUND THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Utilizatorul nu a fost gasit');  
  
 WHEN different\_date then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu este eligibil pentru bonus');  
  
 END *verifica\_aniversare*;  
  
 procedure *toate\_filmele*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE) AS  
 TYPE row\_filme is TABLE OF FILM.DENUMIRE%TYPE;  
 TYPE row\_actori is TABLE OF ACTOR.NUME%TYPE;  
 type row\_note is table of FILM.NOTA%TYPE;  
  
 v\_filme row\_filme;  
 v\_actori row\_actori;  
 v\_note row\_note;  
  
 v\_filme\_distincte row\_filme;  
  
 v\_id number := -1;  
 user\_doesnt\_exist exception;  
 BEGIN  
 select UTILIZATOR\_id  
 into v\_id  
 from UTILIZATOR  
 where UTILIZATOR\_ID = v\_userId;  
  
 if v\_id = -1 then  
 raise user\_doesnt\_exist;  
 end if;  
  
 OPEN filme\_actori(v\_userId);  
 Fetch filme\_actori bulk collect into v\_filme, v\_actori, v\_note;  
 CLOSE filme\_actori;  
  
 -- pentru a obtine filmele distincte  
 v\_filme\_distincte := *SET*(v\_filme);  
  
 for iterator in v\_filme\_distincte.FIRST..v\_filme\_distincte.LAST LOOP  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_filme\_distincte(iterator));  
 end loop;  
 EXCEPTION  
 WHEN NO\_DATA\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*('Userul nu exista');  
  
 WHEN USER\_DOESNT\_EXIST then  
 DBMS\_OUTPUT.*PUT\_LINE*('Userul nu exista');  
 END *toate\_filmele*;  
  
 procedure *toate\_serialele*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE) AS  
 TYPE row\_seriale is TABLE OF SERIAL.DENUMIRE%TYPE;  
 TYPE row\_actori is TABLE OF ACTOR.NUME%TYPE;  
  
 v\_seriale row\_seriale;  
 v\_actori row\_actori;  
  
 v\_seriale\_distincte row\_seriale;  
  
 v\_id number := -1;  
 user\_doesnt\_exist exception;  
 BEGIN  
 select UTILIZATOR\_id  
 into v\_id  
 from UTILIZATOR  
 where UTILIZATOR\_ID = v\_userId;  
  
 if v\_id = -1 then  
 raise user\_doesnt\_exist;  
 end if;  
  
 OPEN seriale\_actori(v\_userId);  
 Fetch seriale\_actori bulk collect into v\_seriale, v\_actori;  
 CLOSE seriale\_actori;  
  
 -- pentru a obtine filmele distincte  
 v\_seriale\_distincte := *SET*(v\_seriale);  
  
 for iterator in v\_seriale\_distincte.FIRST..v\_seriale\_distincte.LAST LOOP  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_seriale\_distincte(iterator));  
 end loop;  
  
 EXCEPTION  
 WHEN NO\_DATA\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*('Userul nu exista');  
  
 WHEN USER\_DOESNT\_EXIST then  
 DBMS\_OUTPUT.*PUT\_LINE*('Userul nu exista');  
 END *toate\_serialele*;  
  
 procedure *toti\_actorii*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE) AS  
 TYPE row\_filme is TABLE OF FILM.DENUMIRE%TYPE;  
 TYPE row\_actori is TABLE OF ACTOR.NUME%TYPE;  
 type row\_note is table of FILM.NOTA%TYPE;  
  
 v\_filme row\_filme;  
 v\_actori\_filme row\_actori;  
 v\_actori\_seriale row\_actori;  
 v\_note row\_note;  
  
 v\_actori\_distincti row\_actori;  
  
 v\_id number := -1;  
 user\_doesnt\_exist exception;  
 BEGIN  
 select UTILIZATOR\_id  
 into v\_id  
 from UTILIZATOR  
 where UTILIZATOR\_ID = v\_userId;  
  
 if v\_id = -1 then  
 raise user\_doesnt\_exist;  
 end if;  
  
 OPEN filme\_actori(v\_userId);  
 Fetch filme\_actori bulk collect into v\_filme, v\_actori\_filme, v\_note;  
 CLOSE filme\_actori;  
  
 OPEN seriale\_actori(v\_userId);  
 Fetch seriale\_actori bulk collect into v\_filme, v\_actori\_seriale;  
 CLOSE seriale\_actori;  
  
 -- pentru a face o reuniune distincta intre actori  
 v\_actori\_distincti := v\_actori\_filme MULTISET UNION DISTINCT v\_actori\_seriale;  
  
 for iterator in v\_actori\_distincti.FIRST..v\_actori\_distincti.LAST LOOP  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_actori\_distincti(iterator));  
 end loop;  
 EXCEPTION  
 WHEN NO\_DATA\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*('Userul nu exista');  
  
 WHEN USER\_DOESNT\_EXIST then  
 DBMS\_OUTPUT.*PUT\_LINE*('Userul nu exista');  
 END *toti\_actorii*;  
  
 function *film\_nota\_maxima*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE) return film\_record AS  
 v\_film\_max film\_record;  
 BEGIN  
 v\_film\_max.nume := 'Nu exista film peste aceasta nota';  
 v\_film\_max.nota := 0;  
 for iterator in filme\_actori(v\_userId) LOOP  
 if v\_film\_max.nota < iterator.NOTA then  
 v\_film\_max.nota := iterator.NOTA;  
 v\_film\_max.nume := iterator.FILM;  
 end if;  
 end loop;  
  
 return v\_film\_max;  
 EXCEPTION  
 WHEN NO\_DATA\_FOUND THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Userul nu exista');  
 return NULL;  
 END *film\_nota\_maxima*;  
  
 function *filtreaza\_filme*(v\_userId UTILIZATOR.UTILIZATOR\_ID%TYPE,  
 v\_nota\_min FILM.NOTA%TYPE DEFAULT 0,  
 v\_nota\_max FILM.NOTA%TYPE DEFAULT 10,  
 v\_crescator boolean DEFAULT false,  
 v\_descrescator boolean DEFAULT false)  
 return cursor\_filtrare AS  
 nota\_min\_invalida exception;  
 nota\_max\_invalida exception;  
 nota\_relatie\_invalida exception;  
 sortare\_invalida exception;  
  
 v\_cursor cursor\_filtrare;  
 v\_order\_by varchar2(50);  
 BEGIN  
 if v\_nota\_min < 0 or v\_nota\_min > 10 then  
 raise nota\_min\_invalida;  
 end if;  
  
 if v\_nota\_max < 0 or v\_nota\_max > 10 then  
 raise nota\_max\_invalida;  
 end if;  
  
 if v\_nota\_min > v\_nota\_max then  
 raise nota\_relatie\_invalida;  
 end if;  
  
 if v\_crescator = true and v\_descrescator = true then  
 raise sortare\_invalida;  
 end if;  
  
 if v\_crescator = true then  
 v\_order\_by := 'ORDER BY FILM.NOTA ASC';  
 ELSIF v\_descrescator = true then  
 v\_order\_by := 'ORDER BY FILM.NOTA DESC';  
 end if;  
  
 if v\_crescator = false and v\_descrescator = false then  
 OPEN v\_cursor FOR  
 SELECT film.\*  
 FROM film  
 JOIN SUBSCRIPTIE\_FILM on film.FILM\_ID = SUBSCRIPTIE\_FILM.FILM\_ID  
 JOIN SUBSCRIPTIE on SUBSCRIPTIE\_FILM.SUBSCRIPTIE\_ID = SUBSCRIPTIE.SUBSCRIPTIE\_ID  
 JOIN UTILIZATOR on SUBSCRIPTIE.SUBSCRIPTIE\_ID = UTILIZATOR.SUBSCRIPTIE\_ID  
 WHERE UTILIZATOR\_ID = v\_userId and  
 film.nota between v\_nota\_min and v\_nota\_max  
 || v\_order\_by;  
 end if;  
  
 return v\_cursor;  
  
 EXCEPTION  
 WHEN nota\_min\_invalida then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nota minima invalida');  
 return null;  
  
 WHEN nota\_max\_invalida then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nota maxima invalida');  
 return null;  
  
 WHEN nota\_relatie\_invalida then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nota minima este mai mare decat nota maxima');  
 return null;  
  
 WHEN sortare\_invalida then  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu se poate sorta si crescator si descrescator');  
 return null;  
  
 WHEN NO\_DATA\_FOUND then  
 DBMS\_OUTPUT.*PUT\_LINE*('Userul nu exista');  
 return null;  
  
 END *filtreaza\_filme*;  
END pachet\_utilizator;  
  
-- ======== EXCEPTII ========  
declare  
 v\_result boolean;  
 v\_film\_cursor pachet\_utilizator.cursor\_filtrare;  
 v\_param\_cresc\_desc boolean := true;  
begin  
 -- ===== VERIFICA\_SUBSCRIPTIE =====  
 -- pentru true ex user id: 111  
 -- NO\_DATA\_FOUND pentru fiecare functioneaza la fel, un utilizator\_id care nu exista  
 v\_result := pachet\_utilizator.*verifica\_subscriptie*(111111);  
 if v\_result = true then  
 DBMS\_OUTPUT.*PUT\_LINE*('true');  
 else  
 DBMS\_OUTPUT.*PUT\_LINE*('false');  
 end if;  
 -- ================================  
  
 -- ===== FILTREAZA FILM =====  
 v\_film\_cursor := pachet\_utilizator.*filtreaza\_filme*(111,-1,10);  
 v\_film\_cursor := pachet\_utilizator.*filtreaza\_filme*(111, 2,11);  
 v\_film\_cursor := pachet\_utilizator.*filtreaza\_filme*(111,7,3);  
 v\_film\_cursor := pachet\_utilizator.*filtreaza\_filme*(111,2,5,v\_param\_cresc\_desc,v\_param\_cresc\_desc);  
 -- ==========================  
  
 pachet\_utilizator.*toate\_filmele*(11111);  
 pachet\_utilizator.*toate\_serialele*(11111);  
 pachet\_utilizator.*toti\_actorii*(11111);  
end;  
-- ==========================  
  
begin  
 pachet\_utilizator.*inserare\_plata*(111,  
 1010,  
 'TEST',  
 'TEST',  
 123456789,  
 *SYSDATE*,  
 123);  
end;  
  
select \* from PLATA;  
select \*  
from LISTA\_CARDURI  
where UTILIZATOR\_ID = 111 and PLATA\_ID=1010;  
  
declare  
 v\_film\_cursor pachet\_utilizator.cursor\_filtrare;  
 v\_film\_row FILM%ROWTYPE;  
 v\_film pachet\_utilizator.film\_record;  
begin  
 pachet\_utilizator.*verifica\_aniversare*(111);  
 pachet\_utilizator.*toate\_filmele*(111);  
 pachet\_utilizator.*toate\_serialele*(111);  
 pachet\_utilizator.*toti\_actorii*(111);  
  
 v\_film := pachet\_utilizator.*FILM\_NOTA\_MAXIMA*(111);  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_film.nume || ' ' || v\_film.nota);  
  
 v\_film\_cursor := pachet\_utilizator.*filtreaza\_filme*(111,11);  
  
 if v\_film\_cursor is not null then  
 for it in v\_film\_cursor loop  
 v\_film\_row := it;  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_film\_row.FILM\_ID || ' ' || v\_film\_row.DENUMIRE || ' ' || v\_film\_row.NOTA);  
 end loop;  
 end if;  
end;  
-- =================================================================

## Screenshots:

