Proiect SGBD

-Gestiune unui magazin online-

Radulescu Mihai-Alexandru

Grupa 234

Cuprins

Introducere

Penrtu acest proiect am decis sa creez o baza de date pentru un magazin online. In aceasta retin date despre utilizatori, locatie, produse si stoc disponibil.

Baza de date este alcatuita din 6 tabele: *Utilizator*, *Locatie*, *Comanda*, *Produs*, *Categorie* si *Depozit*.

Au fost utilizate si 2 tabele asociative: *StocDepozit* si *PlasareComanda* in care sunt retinute informatii despre stocul disponibil, respectiv cantitatea de produs dintr-o comanda.

Diagrama Entitate-Relatie

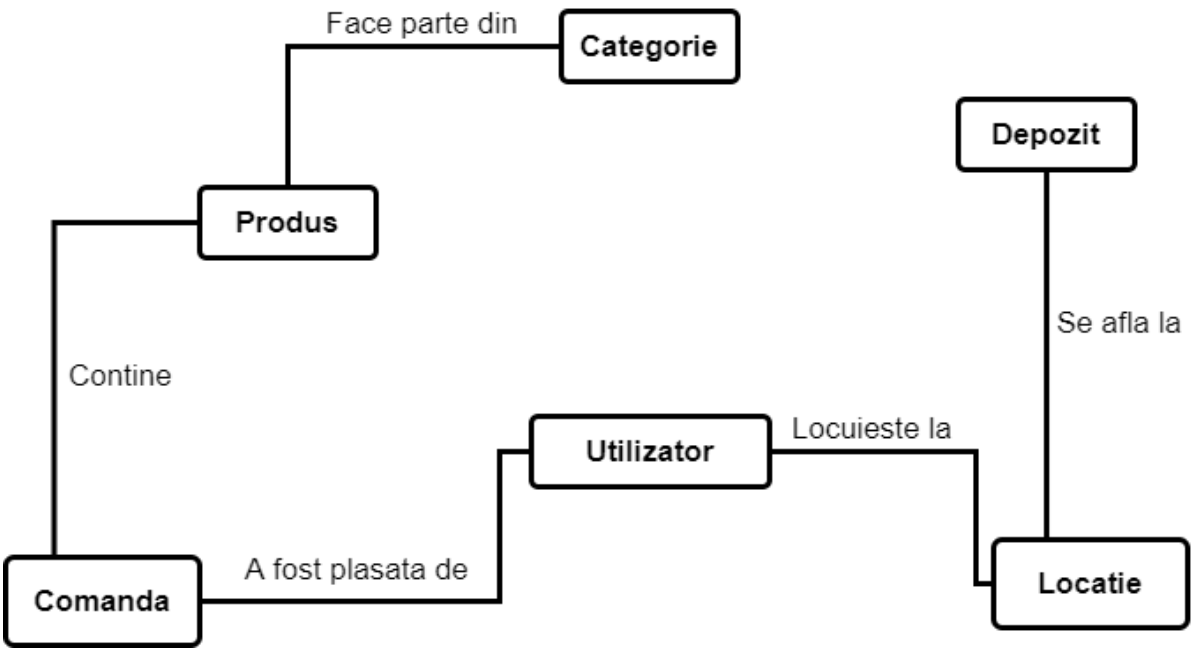
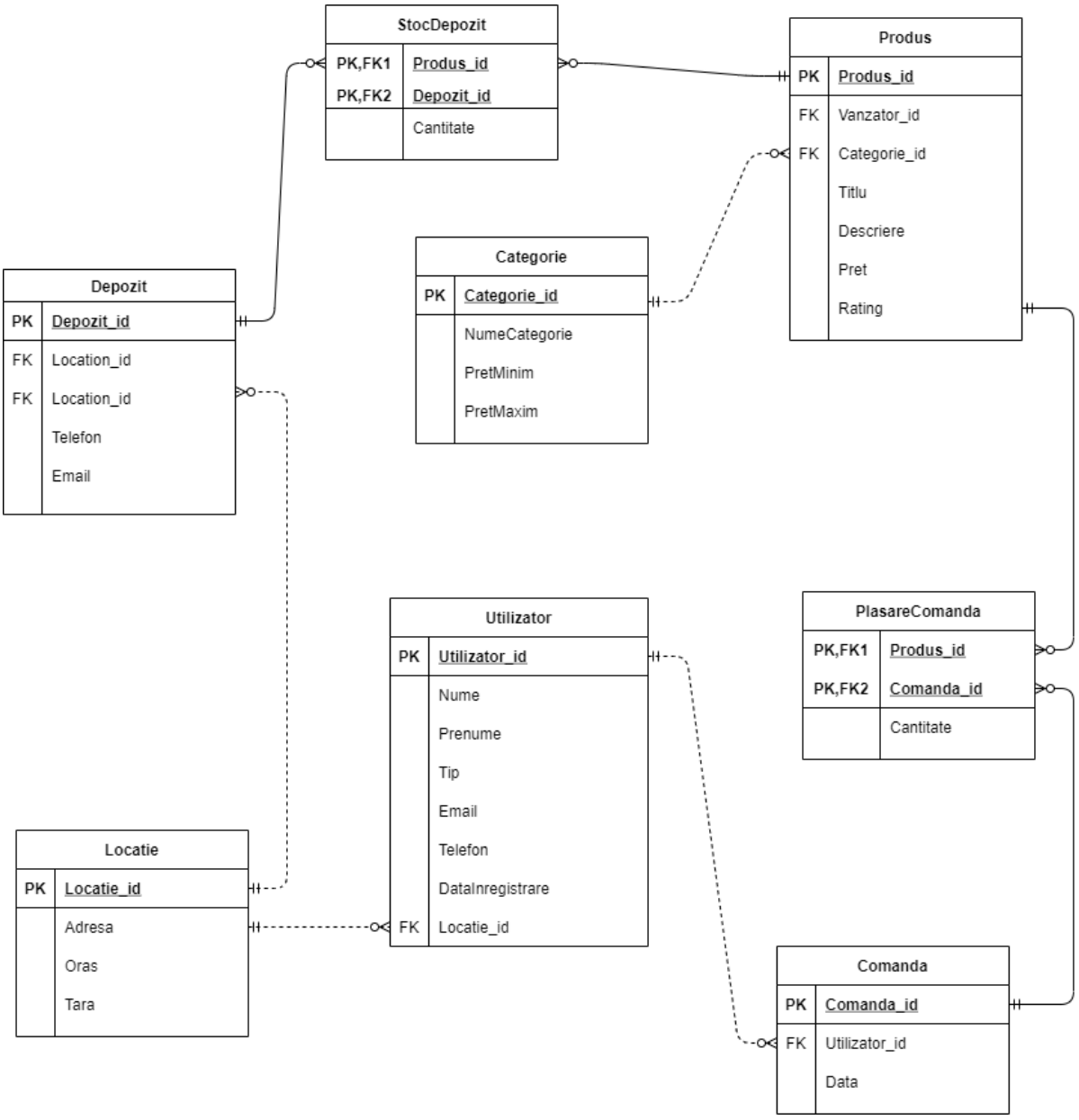


Diagrama Conceptuala



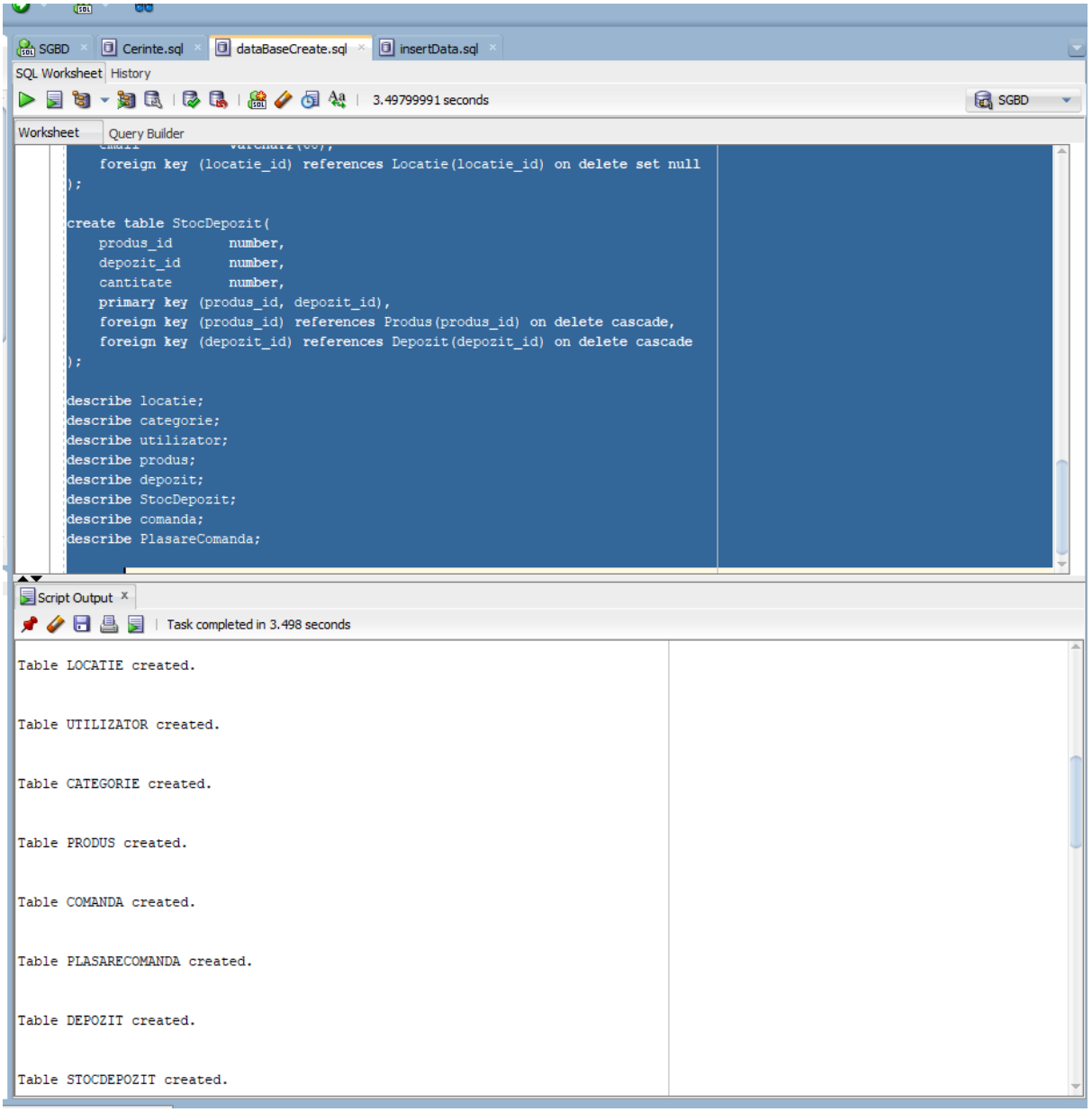
Crearea bazei de date

Codul pentru crearea bazei de date este generic. Trebuuie tinut cont de faptul exista tabele care depind de alte chei straine. Codul trebuie rulat in ordinea din acest document.

```
create table Locatie(
      locatie_id number primary key,
      adresa     varchar2(100),
      oras       varchar2(100),
      tara     varchar2(100)
);

create table Utilizator(
      utilizator_id           number primary key,
      nume                    varchar2(20) not null,
      prenume                       varchar2(20),
      tip                     varchar2(20) not null,
      email                   varchar2(60),
      telefon                       varchar2(10),
      DataInregistrare  date,
      locatie_id              number not null,
      foreign key (locatie_id) references Locatie(locatie_id) on delete
set null
);

create table Categorie(
      categorie_id      number primary key,
      numeCategorie     varchar2(50),
      PretMinim         number,
      pretMaxim         number
);

create table Produs(
      produs_id         number primary key,
      vanzator_id       number not null,
```

```
      categorie_id      number not null,
      titlu             varchar2(200),
      descriere         varchar2(3000),
      pret              number(10, 2),
      rating                  number(2, 1),
      foreign key (vanzator_id) references Utilizator(utilizator_id) on
delete cascade,
      foreign key (categorie_id) references Categorie(categorie_id) on
delete cascade
);

create table Comanda(
      comanda_id        number primary key,
      utilizator_id     number not null,
      data              date,
      foreign key (utilizator_id) references Utilizator(utilizator_id)
on delete cascade
);

create table PlasareComanda(
      produs_id         number,
      comanda_id        number,
      cantitate         number,
      primary key (produs_id, comanda_id),
      foreign key (produs_id) references Produs(produs_id) on delete
cascade,
      foreign key (comanda_id) references Comanda(comanda_id) on delete
cascade
);

create table Depozit(
      depozit_id        number primary key,
      locatie_id        number not null,
      telefon                 varchar2(10),
      email             varchar2(60),
      foreign key (locatie_id) references Locatie(locatie_id) on delete
set null
);

create table StocDepozit(
      produs_id         number,
      depozit_id        number,
      cantitate         number,
      primary key (produs_id, depozit_id),
      foreign key (produs_id) references Produs(produs_id) on delete
cascade,
```

```
    foreign key (depozit_id) references Depozit(depozit_id) on delete
cascade
);
```

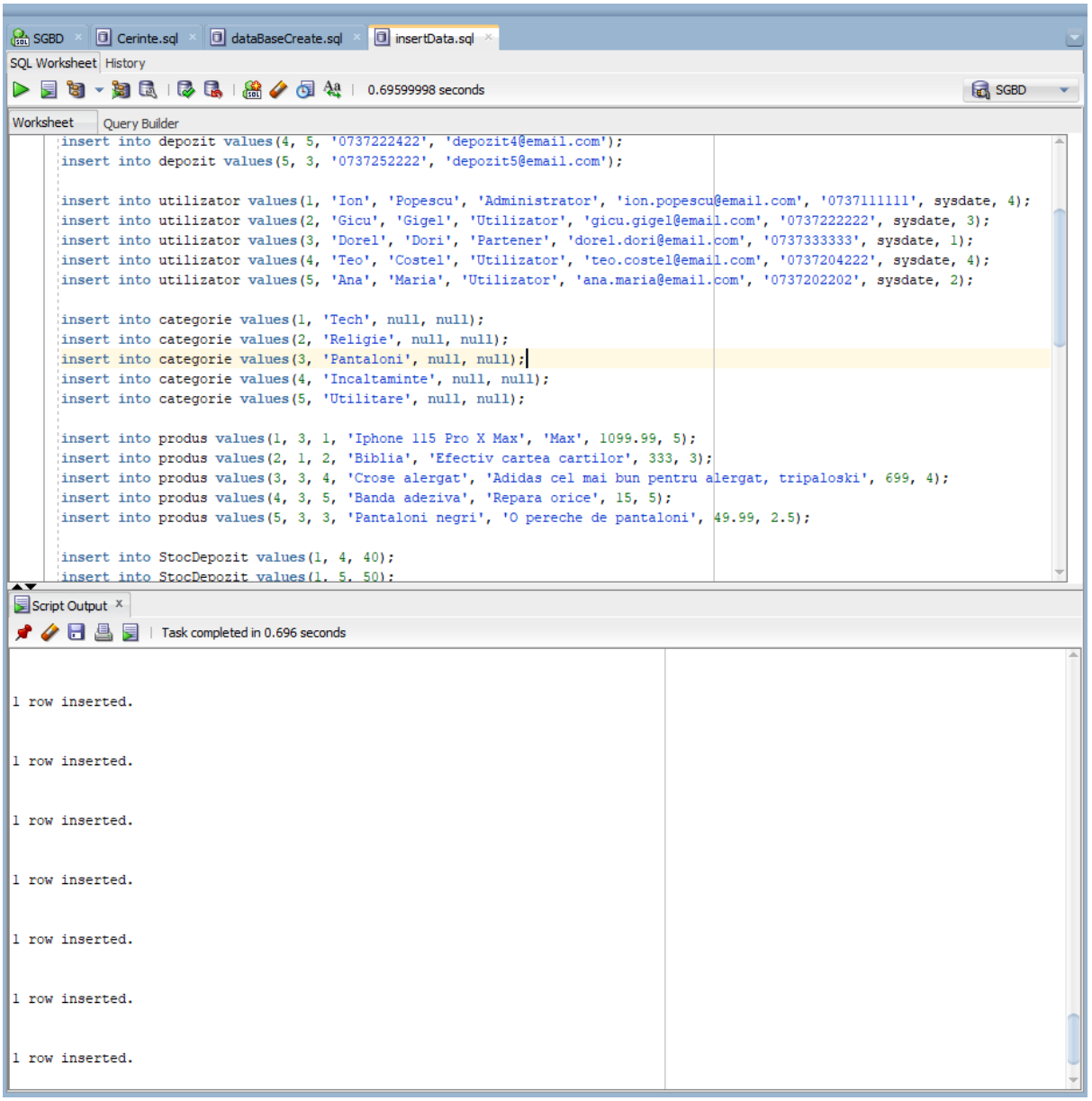


Popularea bazei de date

Am ales sa creez un set simplu de date, respectand conditile impuse.

```
insert into locatie values(1, 'Splaiul Independentei 204', 'Bucuresti',
'Romania');
insert into locatie values(2, 'Bulevardul Iuliu Maniua 104',
'Bucuresti', 'Romania');
insert into locatie values(3, 'Strada Zavoiului 39', 'Chisinau',
'Republica Moldova');
insert into locatie values(4, 'Bergen Strasse 3', 'Berlin', 'Germania');
insert into locatie values(5, 'Bulevardul Alba Iulia 69', 'Cluj-Napoca',
'Romania');

insert into depozit values(1, 1, '0747111111', 'depozit1@email.com');
insert into depozit values(2, 2, '0737222222', 'depozit2@email.com');
insert into depozit values(3, 4, '0737222322', 'depozit3@email.com');
insert into depozit values(4, 5, '0737222422', 'depozit4@email.com');
insert into depozit values(5, 3, '0737252222', 'depozit5@email.com');

insert into utilizator values(1, 'Ion', 'Popescu', 'Administrator',
'ion.popescu@email.com', '0737111111', sysdate, 4);
insert into utilizator values(2, 'Gicu', 'Gigel', 'Utilizator',
'gicu.gigel@email.com', '0737222222', sysdate, 3);
insert into utilizator values(3, 'Dorel', 'Dori', 'Partener',
'dorel.dori@email.com', '0737333333', sysdate, 1);
insert into utilizator values(4, 'Teo', 'Costel', 'Utilizator',
'teo.costel@email.com', '0737204222', sysdate, 4);
insert into utilizator values(5, 'Ana', 'Maria', 'Utilizator',
'ana.maria@email.com', '0737202202', sysdate, 2);

insert into categorie values(1, 'Tech', null, null);
insert into categorie values(2, 'Religie', null, null);
insert into categorie values(3, 'Pantaloni', null, null);
insert into categorie values(4, 'Incaltaminte', null, null);
insert into categorie values(5, 'Utilitare', null, null);

insert into produs values(1, 3, 1, 'Iphone 115 Pro X Max', 'Max',
1099.99, 5);
insert into produs values(2, 1, 2, 'Biblia', 'Efectiv cartea cartilor',
333, 3);
insert into produs values(3, 3, 4, 'Crose alergat', 'Adidas cel mai bun
pentru alergat, tripaloski', 699, 4);
insert into produs values(4, 3, 5, 'Banda adeziva', 'Repara orice', 15,
5);
insert into produs values(5, 3, 3, 'Pantaloni negri', 'O pereche de
pantaloni', 49.99, 2.5);

insert into StocDepozit values(1, 4, 40);
insert into StocDepozit values(1, 5, 50);
```

```
insert into StocDepozit values(1, 2, 100);
insert into StocDepozit values(1, 1, 220);
insert into StocDepozit values(2, 3, 100);
insert into StocDepozit values(2, 2, 10);
insert into StocDepozit values(2, 4, 25);
insert into StocDepozit values(3, 5, 69);
insert into StocDepozit values(3, 2, 420);
insert into StocDepozit values(3, 1, 12);
insert into StocDepozit values(4, 1, 55);
insert into StocDepozit values(4, 2, 69420);
insert into StocDepozit values(4, 3, 34);
insert into StocDepozit values(4, 4, 21);
insert into StocDepozit values(4, 5, 9);

insert into comanda values(1, 1, sysdate);
insert into comanda values(2, 5, sysdate);
insert into comanda values(3, 2, sysdate);
insert into comanda values(4, 3, sysdate);
insert into comanda values(5, 4, sysdate);

insert into PlasareComanda values(1, 1, 2);
insert into PlasareComanda values(1, 2, 4);
insert into PlasareComanda values(2, 4, 5);
insert into PlasareComanda values(2, 3, 8);
insert into PlasareComanda values(2, 5, 2);
insert into PlasareComanda values(3, 1, 1);
insert into PlasareComanda values(3, 2, 3);
insert into PlasareComanda values(3, 5, 1);
insert into PlasareComanda values(4, 4, 2);
insert into PlasareComanda values(4, 3, 7);
insert into PlasareComanda values(5, 3, 3);
insert into PlasareComanda values(5, 1, 2);

commit;
```



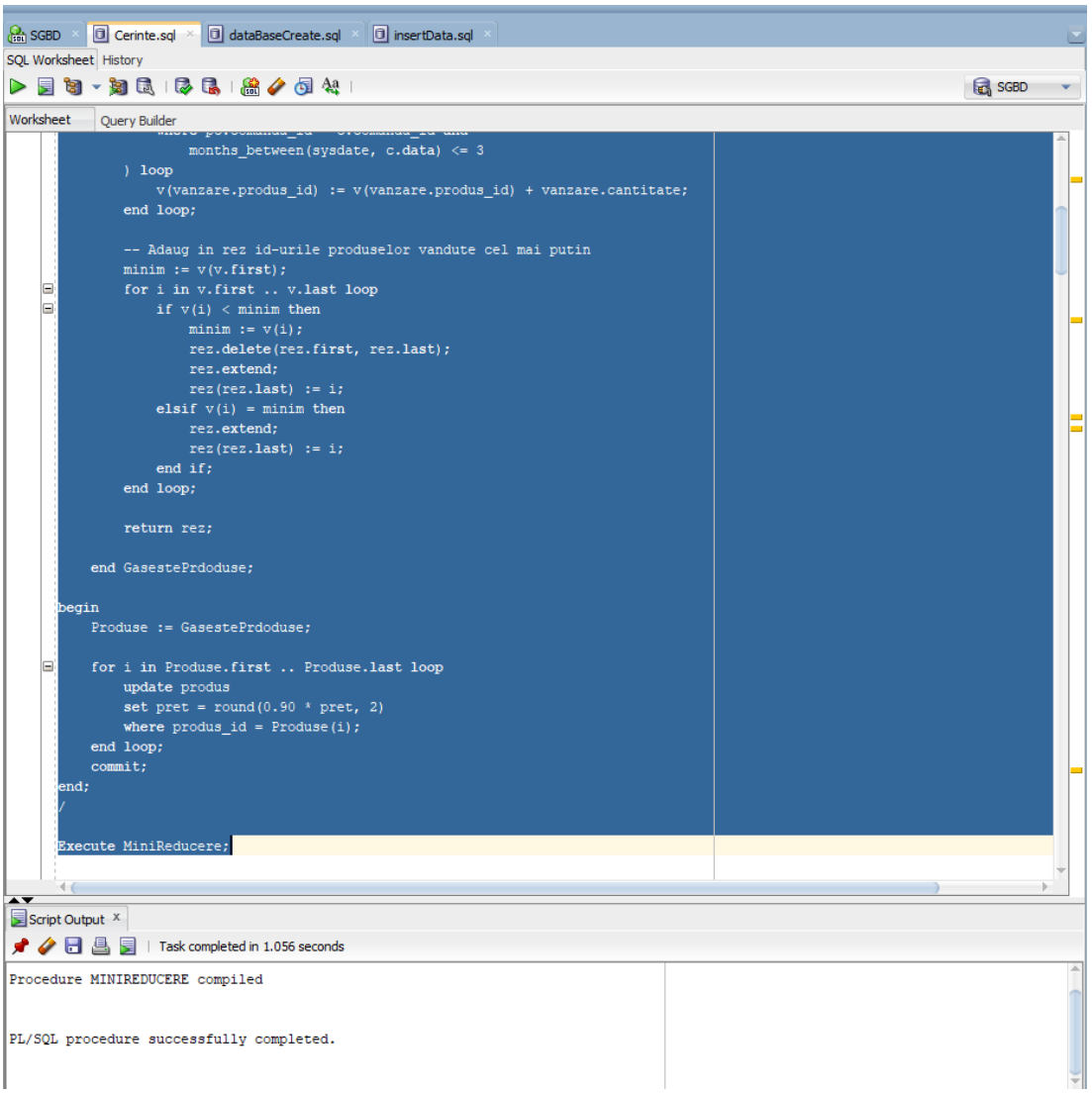
Cerinta 6

Pentru aceasta cerinta am ales sa fac o procedura numita *MiniReducere* care aplica o reducere de 10% celor mai slab vandute produse din ultimele 3 luni. Pentru a selecta produsele care indeplinesc conditiile din cerinta folosesc o functie *GasesteProduse* care cauta id-urile produselor care s-au vandut cel mai putin.

```
set serveroutput on;

create or replace procedure MiniReducere
```

```
is

    type tablou_imbr is table of number;
    Produse tablou_imbr;
    type tablou      is table of number index by binary_integer;

    function GasestePrdoduse
    return tablou_imbr
    is
        v       tablou;
        rez    tablou_imbr := tablou_imbr();
        minim      number;
    begin
        -- Selectez id-urile produselor si le folosesc ca index
        for prod in (
            select * from produs
        ) loop
            v(prod.produs_id) := 0;
        end loop;

        -- Gasesc cantitatea vanduta din fiecare produs vandut in
ultimele 3 luni
        for vanzare in (
            select pc.* from PlasareComanda pc, comanda c
            where pc.comanda_id = c.comanda_id and
                months_between(sysdate, c.data) <= 3
        ) loop
            v(vanzare.produs_id) := v(vanzare.produs_id) +
vanzare.cantitate;
        end loop;

        -- Adaug in rez id-urile produselor vandute cel mai putin
        minim := v(v.first);
        for i in v.first .. v.last loop
            if v(i) < minim then
                minim := v(i);
                rez.delete(rez.first, rez.last);
                rez.extend;
                rez(rez.last) := i;
            elsif v(i) = minim then
                rez.extend;
                rez(rez.last) := i;
            end if;
        end loop;

        return rez;
```

```
    end GasestePrdoduse;

begin
    Produse := GasestePrdoduse;

    for i in Produse.first .. Produse.last loop
        update produs
        set pret = round(0.90 * pret, 2)
        where produs_id = Produse(i);
    end loop;
    commit;
end;
/

Execute MiniReducere;
```



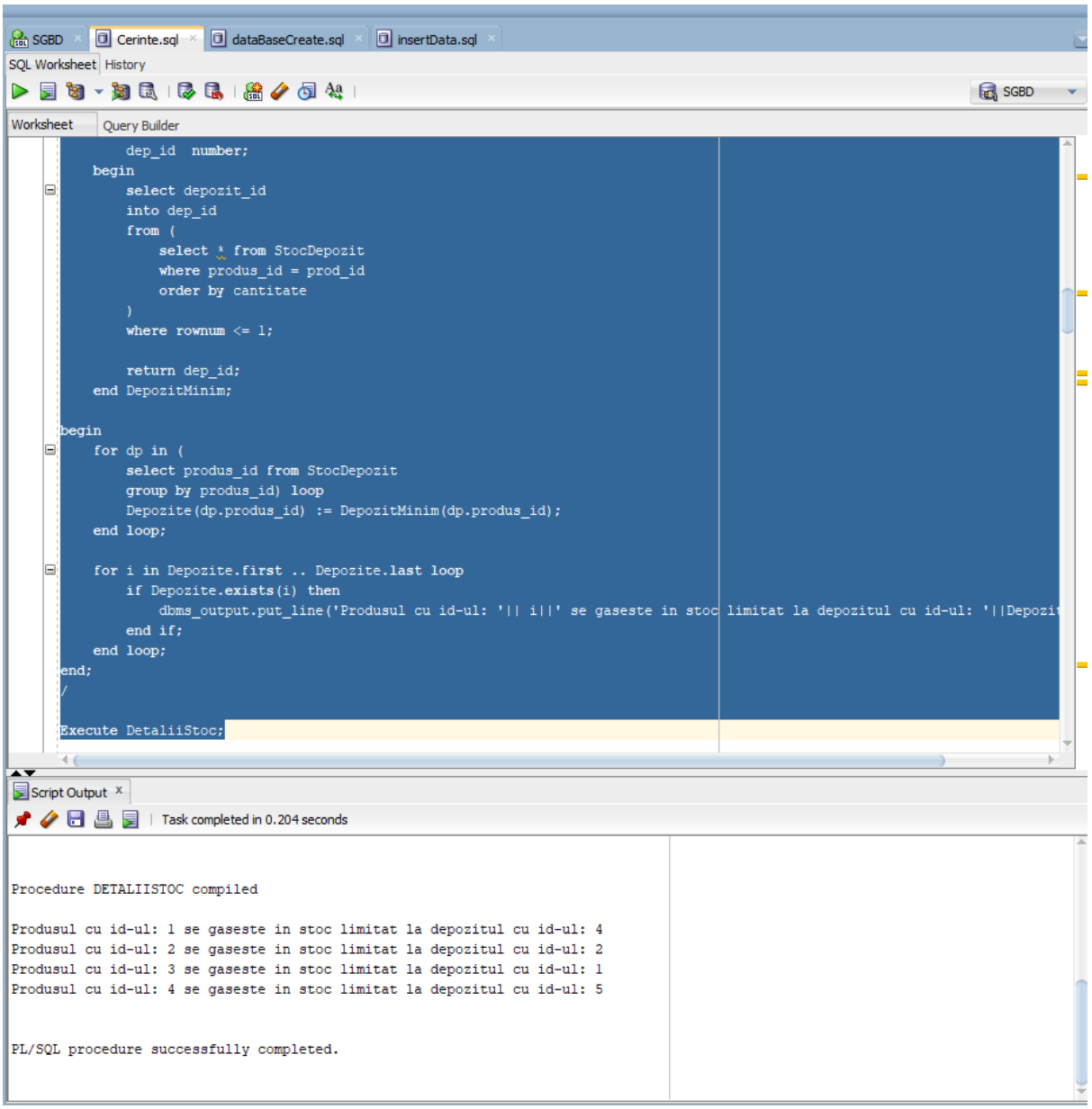
Cerinta 7

Pentru a putea face aprovizionari, este necesar sa aflam, pentru fiecare produs, care este depozitul unde produsul respectiv se alfa in cantitati limitate. Pentru a rezolva problema, am definit o procedura *DetaliiStoc* si o functie *DepozitMinim* care determina pentru fiecare produs depozitul corespunzator.

Am utiliat un ciclu cursor.

```
set serveroutput on;

create or replace procedure DetaliiStoc
is
    type tablou is table of number index by binary_integer;
    Depozite    tablou;
    prod_id      number;

    function DepozitMinim(
        prod_id     number
    )
    return number
    is
        dep_id  number;
    begin
        select depozit_id
        into dep_id
        from (
            select * from StocDepozit
            where produs_id = prod_id
            order by cantitate
        )
        where rownum <= 1;

        return dep_id;
    end DepozitMinim;

begin
    for dp in (
        select produs_id from StocDepozit
        group by produs_id) loop
        Depozite(dp.produs_id) := DepozitMinim(dp.produs_id);
    end loop;
```

```
    for i in Depozite.first .. Depozite.last loop
        if Depozite.exists(i) then
            dbms_output.put_line('Produsul cu id-ul: '|| i||' se gaseste
in stoc limitat la depozitul cu id-ul: '||Depozite(i));
        end if;
    end loop;
end;
/

Execute DetaliiStoc;
```



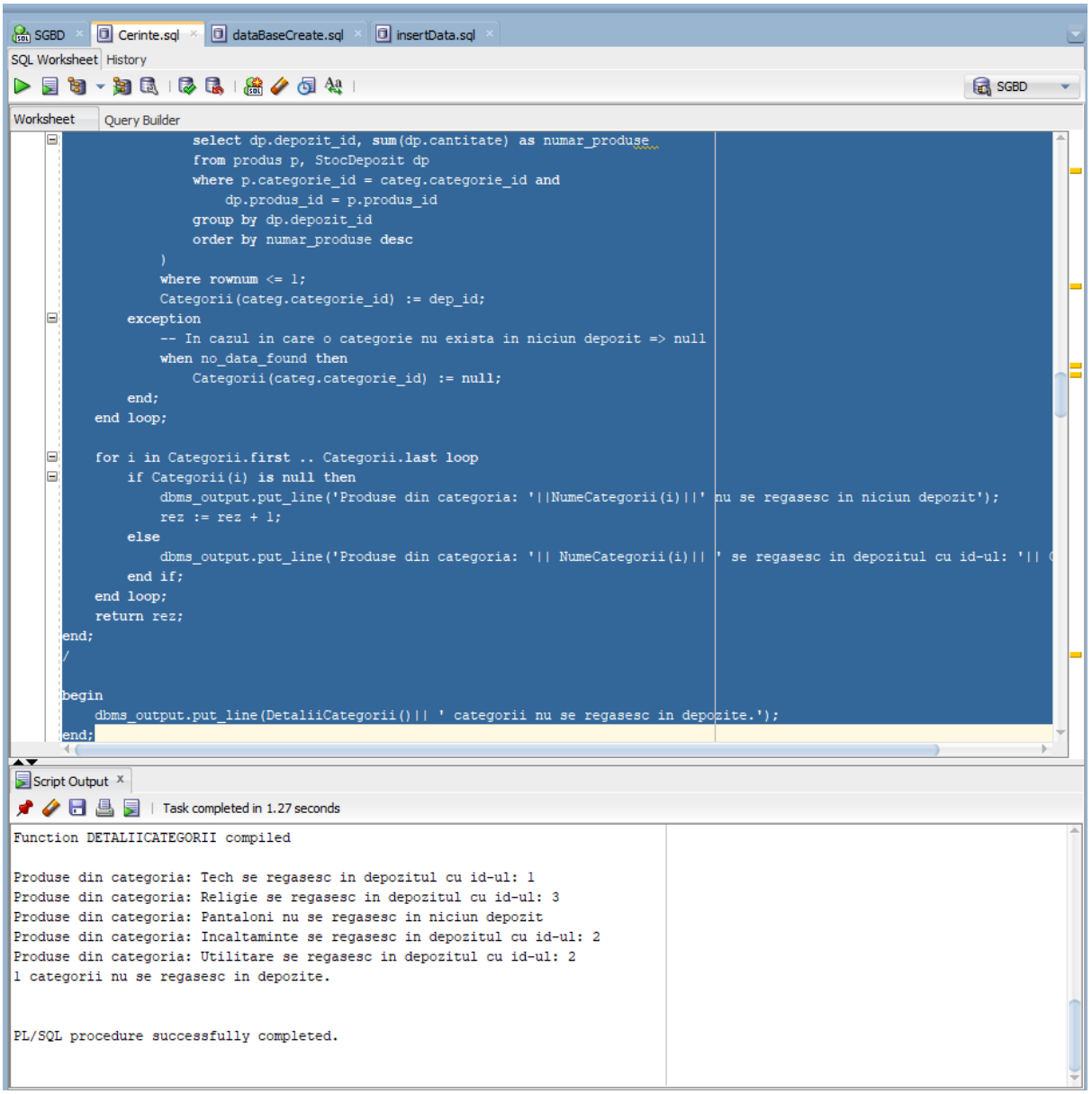
Cerinta 8

Pentru fiecare categorie de produs, dorim sa aflam care depozit are cele mai multe produse din categoria respectiva si sa afisam cate categorii nu se regasesc in niciun depozit, daca este cazul. A fost creata o functie *DetaliiCategorii* care afiseaza pentru fiecare categorie depozitul corespunzator si returneaza numarul de categorii care nu se gasesc in niciun depozit.

Se vor utiliza 3 tabele: *StocDepozit*, *Produs* si *Categorie*.

```
set serveroutput on;

-- Functia returneaza numarul categoriilor care nu se regasesc in niciun
depozit
create or replace function DetaliiCategorii
return number
is
    type tablou      is table of number index by binary_integer;
    Categorii        tablou;

    type tablou_string is table of categorie.NumeCategorie%type index by
binary_integer;
    NumeCategorii    tablou_string;

    rez              number := 0;
    dep_id           number;

begin
    for categ in (
        select * from categorie
    ) loop
        NumeCategorii(categ.categorie_id) := categ.NumeCategorie;
        begin
            select depozit_id into dep_id
            from (
                select dp.depozit_id, sum(dp.cantitate) as numar_produse
                from produs p, StocDepozit dp
                where p.categorie_id = categ.categorie_id and
                    dp.produs_id = p.produs_id
                group by dp.depozit_id
                order by numar_produse desc
            )
            where rownum <= 1;
```

```
            Categorii(categ.categorie_id) := dep_id;
        exception
            -- In cazul in care o categorie nu exista in niciun depozit
=> null
            when no_data_found then
                Categorii(categ.categorie_id) := null;
        end;
    end loop;

    for i in Categorii.first .. Categorii.last loop
        if Categorii(i) is null then
            dbms_output.put_line('Produse din categoria:
'||NumeCategorii(i)||' nu se regasesc in niciun depozit');
            rez := rez + 1;
        else
            dbms_output.put_line('Produse din categoria: '||
NumeCategorii(i)|| ' se regasesc in depozitul cu id-ul: '||
Categorii(i));
        end if;
    end loop;
    return rez;
end;
/

begin
    dbms_output.put_line(DetaliiCategorii()|| ' categorii nu se regasesc
in depozite.');
end;
```



Cerinta 9

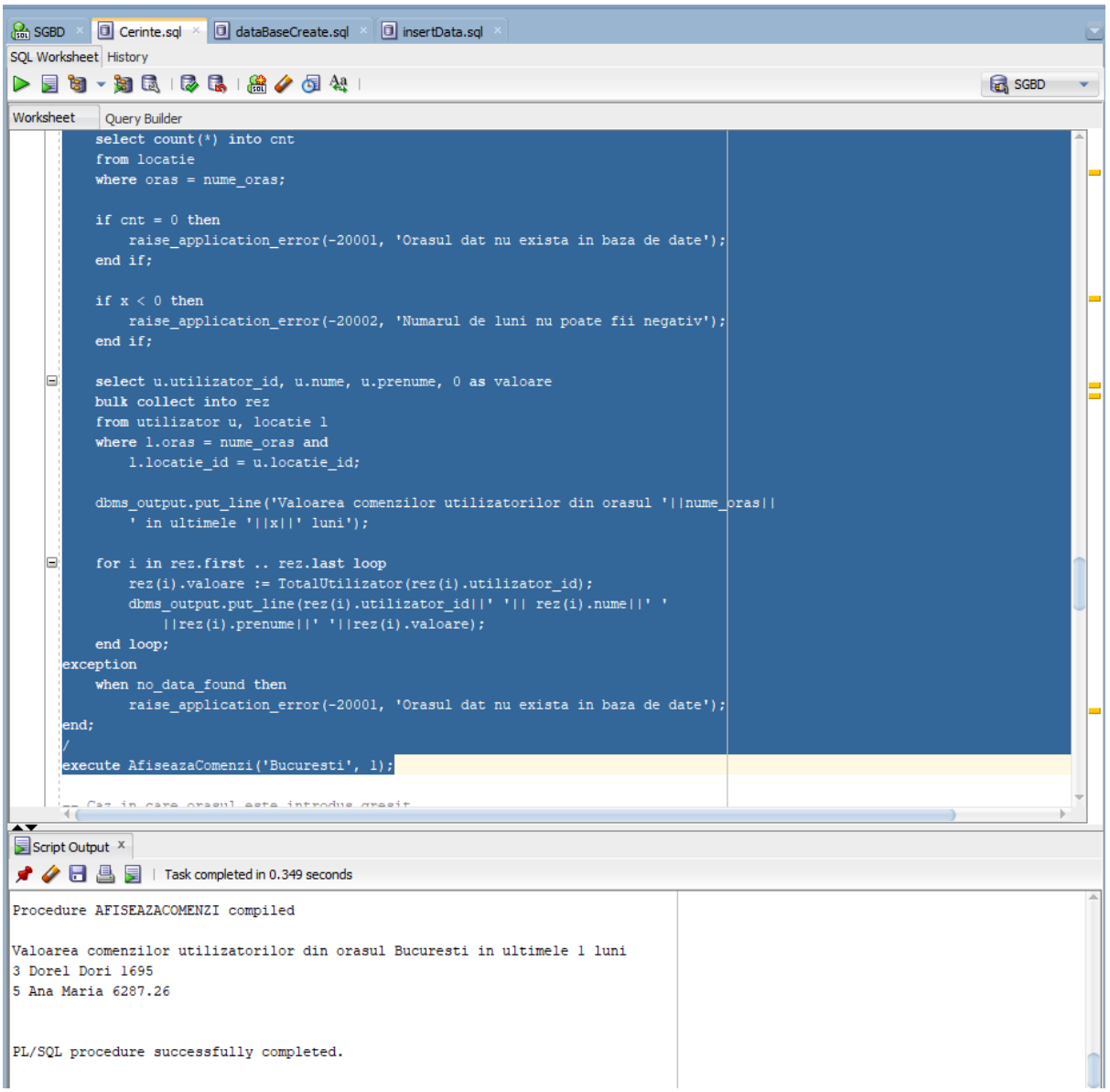
Dorim sa aflam cat a cumparat fiecare utilizator dintr-un oras x in ultimele y luni ale anului. Pentru a rezolva problema am creeat procedura *AfiseazaComenzi*. Aceasta utilizeaza 2 functii *GasestePret* si *TotalUtilizator* care determina pretul unui produs dupa id, respectiv calculeaza totalul unui utilizator.

Sunt utilizate urmatoarele tabele: *Comanda*, *Produs*, *Utilizator*, *PlasareComanda* si *Comanda*.

```
set serveroutput on;

create or replace procedure AfiseazaComenzi(
    nume_oras    locatie.oras%type,
    x            integer
)
is
    type tip_rez is record (utilizator_id
utilizator.utilizator_id%type,
                            nume            utilizator.nume%type,
                            prenume         utilizator.prenume%type,
                            valoare         number);
    type tablou is table of tip_rez;
    rez tablou;
    cnt     integer;

    function GasestePret(
        prod_id     produs.produs_id%type
    ) return produs.pret%type
    is
        prod_pret   number;
    begin
        select pret into prod_pret
        from produs
        where produs_id = prod_id;

        return prod_pret;
    exception
        when no_data_found then
            raise_application_error(-20003, 'Produsul cu id-ul dat nu
exista in baza de date');
        -- nu putem avea exceptia too many rows doarece produs_id este
cheie primara
    end GasestePret;

    function TotalUtilizator(
        u_id    utilizator.utilizator_id%type
    )
    return number
    is
        suma        number:=0;
        dataReg     date;
    begin
        dataReg := SYSDATE;
        for my_comanda in (
            select pc.* from comanda c, PlasareComanda pc
```

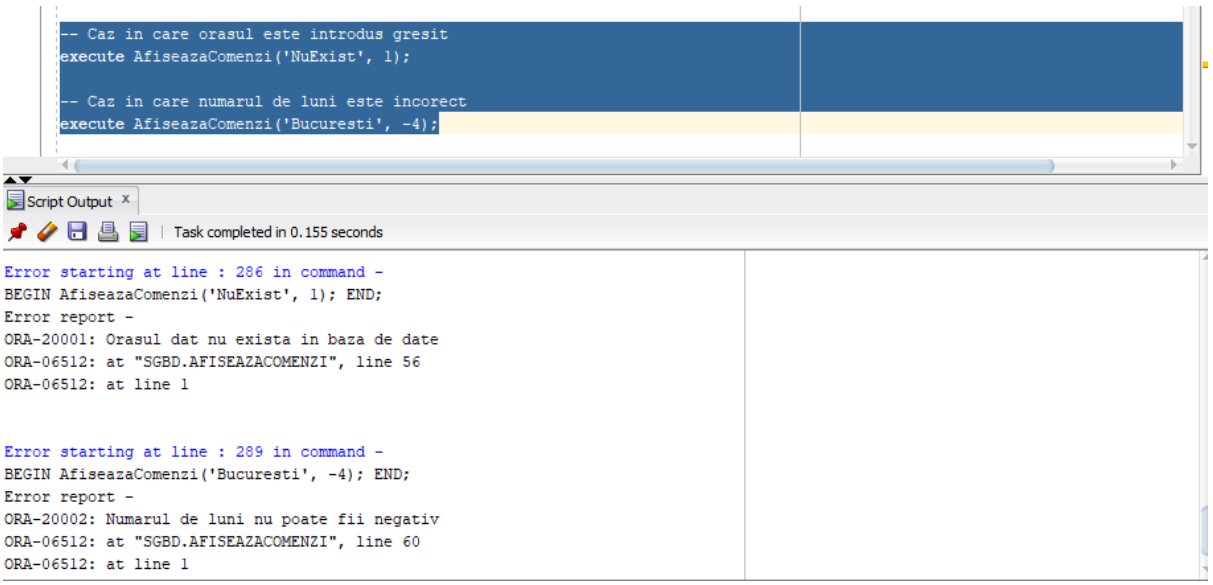
```
            where utilizator_id = u_id and
                months_between(data, dataReg) <= x and
                c.comanda_id = pc.comanda_id
        ) loop
            suma := suma + my_comanda.cantitate *
GasestePret(my_comanda.produs_id);
        end loop;
        return suma;
    end TotalUtilizator;
begin
    select count(*) into cnt
    from locatie
    where oras = nume_oras;

    if cnt = 0 then
        raise_application_error(-20001, 'Orasul dat nu exista in baza de
date');
    end if;

    if x < 0 then
        raise_application_error(-20002, 'Numarul de luni nu poate fii
negativ');
    end if;

    select u.utilizator_id, u.nume, u.prenume, 0 as valoare
    bulk collect into rez
    from utilizator u, locatie l
    where l.oras = nume_oras and
        l.locatie_id = u.locatie_id;

    dbms_output.put_line('Valoarea comenzilor utilizatorilor din orasul
'||nume_oras||
        ' in ultimele '||x||' luni');

    for i in rez.first .. rez.last loop
        rez(i).valoare := TotalUtilizator(rez(i).utilizator_id);
        dbms_output.put_line(rez(i).utilizator_id||' '|| rez(i).nume||'
'
            ||rez(i).prenume||' '||rez(i).valoare);
    end loop;
exception
    when no_data_found then
        raise_application_error(-20001, 'Orasul dat nu exista in baza de
date');
end;
/
```

```
execute AfiseazaComenzi('Bucuresti', 1);
```



Posibile exceptii:

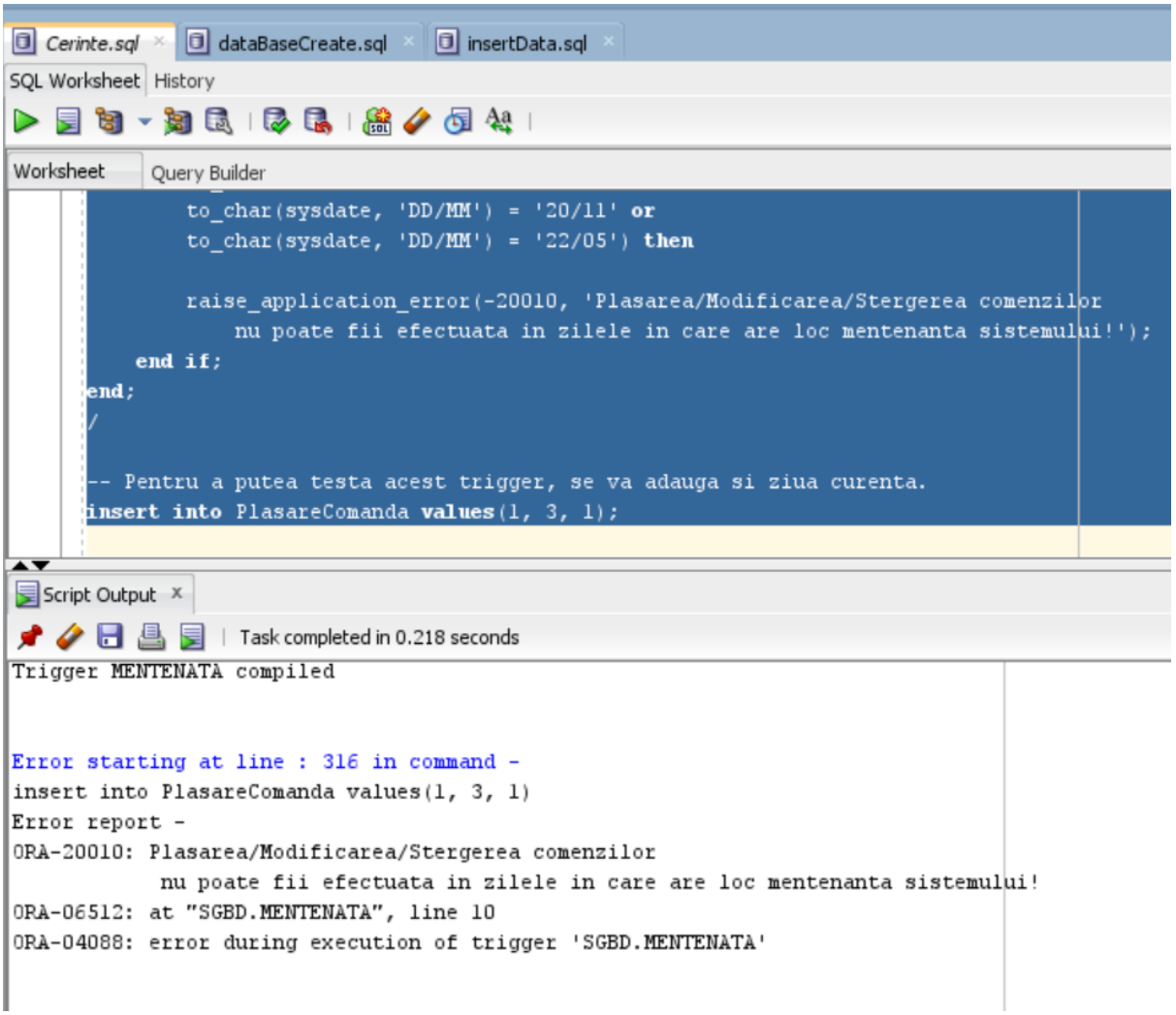
```
-- Caz in care orasul este introdus gresit
execute AfiseazaComenzi('NuExist', 1);

-- Caz in care numarul de luni este incorect
execute AfiseazaComenzi('Bucuresti', -4);
```



Cerinta 10

La interval de cate 2 luni are loc mentenanta sistemului. Din acest motiv vrem sa restrictionam posibilitatea plasarii comenzilor in acele zile. Din acest motiv am defint un trigger LDD numit *Mentenanta*.

```
set serveroutput on;

create or replace trigger Mentenata
before insert or delete or update on PlasareComanda
begin
    if (to_char(sysdate, 'DD/MM') = '20/01' or
        to_char(sysdate, 'DD/MM') = '20/03' or
        to_char(sysdate, 'DD/MM') = '20/05' or
        to_char(sysdate, 'DD/MM') = '20/07' or
        to_char(sysdate, 'DD/MM') = '20/09' or
        to_char(sysdate, 'DD/MM') = '20/11' or
        to_char(sysdate, 'DD/MM') = '28/05') then

        raise_application_error(-20010, 'Plasarea/Modificarea/Stergerea
comenzilor
            nu poate fii efectuata in zilele in care are loc mentenanta
sistemului!');
    end if;
end;
/
```

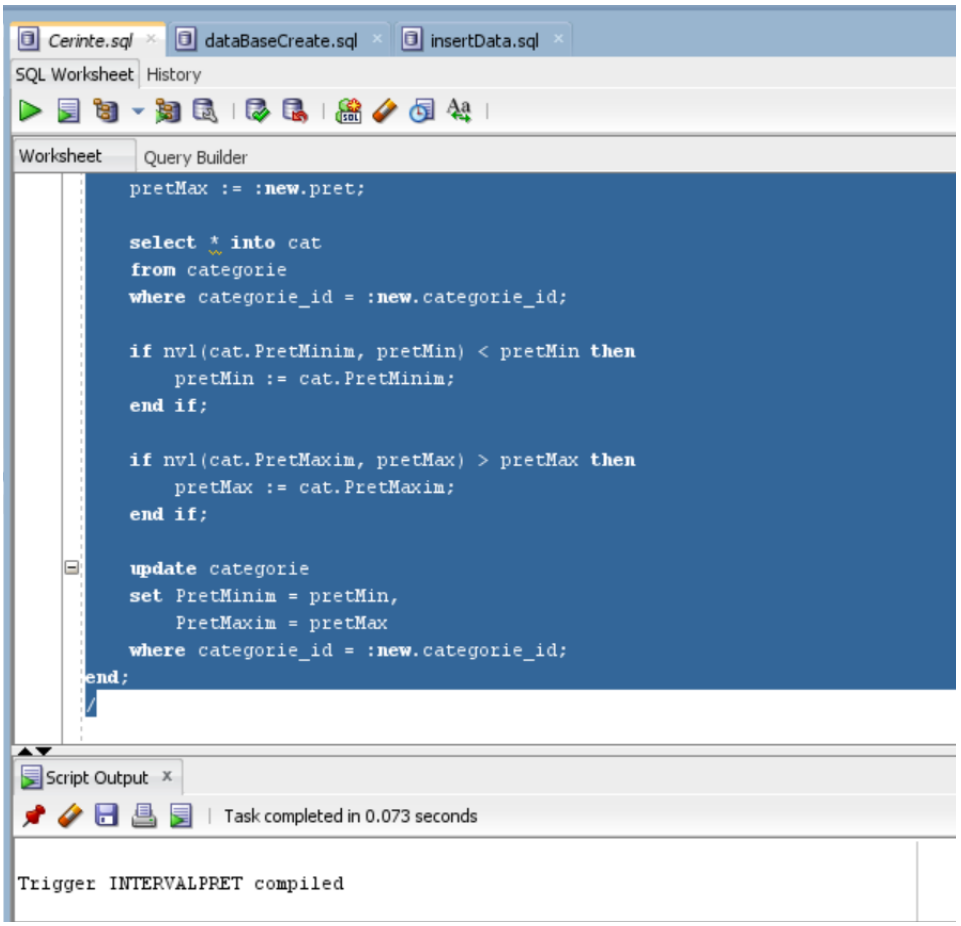


Cerinta 11

Pentru a putea filtra produsele dintr-o categorie, retinem pentru fiecare categorie cel mai mic si cel mai mare pret. Acest lucru devine complicat in momentul in care dorim sa adaugam produse noi intr-o categorie. Din acest motiv am definit un trigger LDD la nivel de line denumit *IntervalPret*.

```
create or replace trigger IntervalPret
after insert or update on produs for each row
declare
    pretMin      number;
    pretMax      number;
    cat       categorie%rowtype;
begin
```

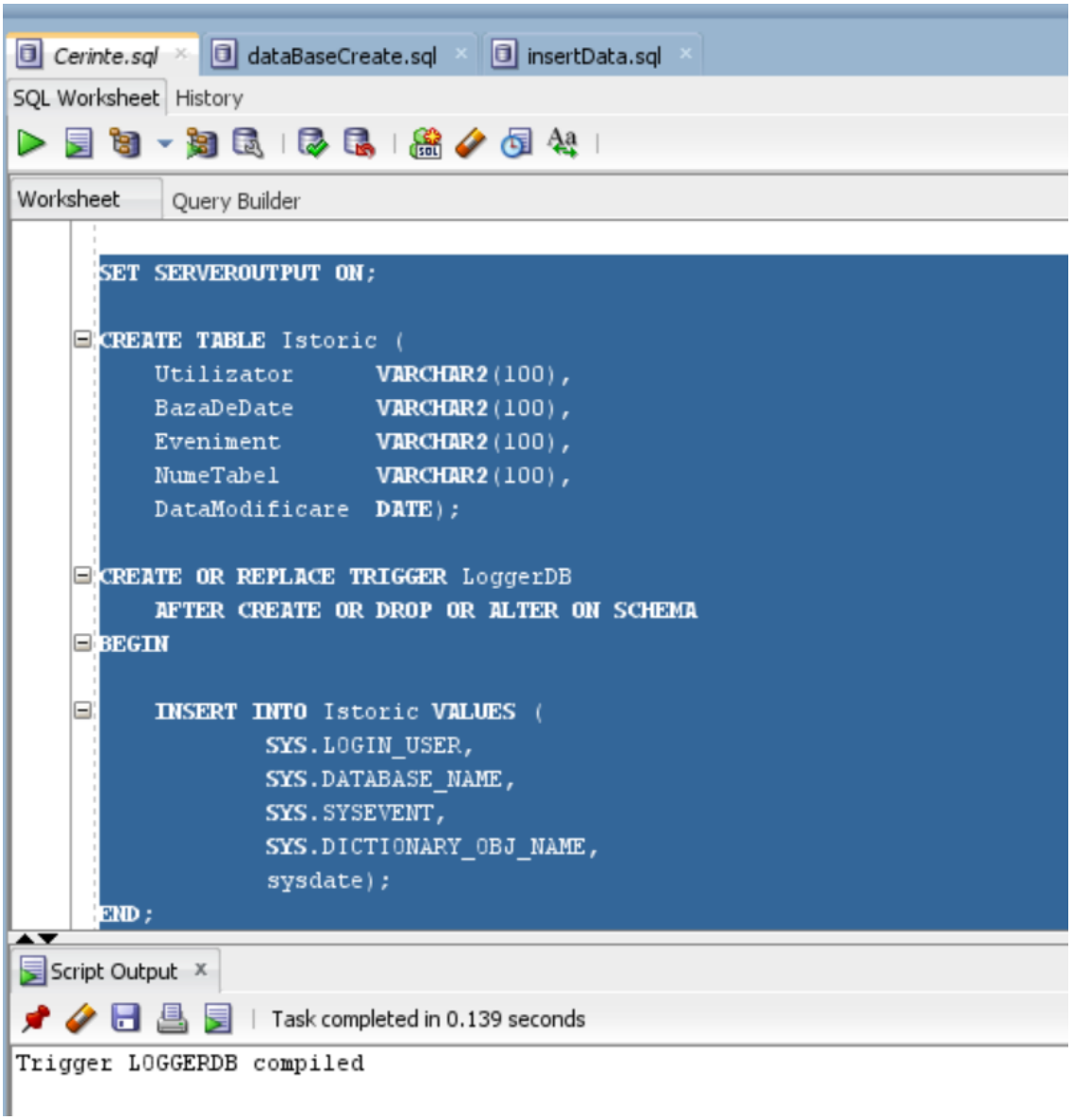
```
    pretMin := :new.pret;
    pretMax := :new.pret;

    select * into cat
    from categorie
    where categorie_id = :new.categorie_id;

    if nvl(cat.PretMinim, pretMin) < pretMin then
        pretMin := cat.PretMinim;
    end if;

    if nvl(cat.PretMaxim, pretMax) > pretMax then
        pretMax := cat.PretMaxim;
    end if;

    update categorie
    set PretMinim = pretMin,
        PretMaxim = pretMax
    where categorie_id = :new.categorie_id;
end;
/
```



Cerinta 12

Baza de date este o componenta foarte importanta a oricarei aplicatii. Din acest motiv dorim sa avem un istoric al modificarilor facute asupra shemei. Pentru a realiza acest lucru am creat un nou tabel *Istoric* si un trigger LMD *LoggerDB* care introduce date despre fiecare modificare in istoric.

```
SET SERVEROUTPUT ON;

CREATE TABLE Istoric (
    Utilizator      VARCHAR2(100),
    BazaDeDate      VARCHAR2(100),
    Eveniment       VARCHAR2(100),
    NumeTabel       VARCHAR2(100),
    DataModificare  DATE);

CREATE OR REPLACE TRIGGER LoggerDB
    AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN

    INSERT INTO Istoric VALUES (
            SYS.LOGIN_USER,
            SYS.DATABASE_NAME,
            SYS.SYSEVENT,
            SYS.DICTIONARY_OBJ_NAME,
            sysdate);
END;
/
```


Cerinte.sql
dataBaseCreate.sql
insertData.sql
SQL Worksheet
History
Worksheet
Query Builder
SET SERVEROUTPUT ON;
CREATE TABLE Istoric (
Utilizator VARCHAR2(100),
BazaDeDate VARCHAR2(100),
Eveniment VARCHAR2(100),
NumeTabel VARCHAR2(100),
DataModificare DATE);
CREATE OR REPLACE TRIGGER LoggerDB
AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
INSERT INTO Istoric VALUES (
SYS.LOGIN_USER,
SYS.DATABASE_NAME,
SYS.SYSEVENT,
SYS.DICTIONARY_OBJ_NAME,
sysdate);
END;
Script Output
Task completed in 0.139 seconds
Trigger LOGGERDB compiled