# Proiect SGBD

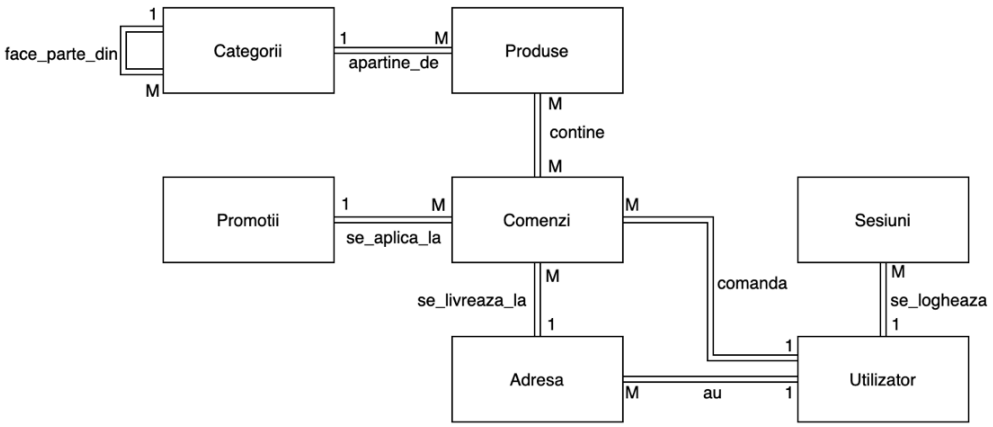
Realizat de *Buhai Darius*
*Grupa 234*

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

În acest proiect voi crea baza de date a unui magazin online. Magazinul online va conține: produse, categorii, promoții (coduri promoționale), comenzi, utilizatori și sesiuni de autentificare. Voi folosi o tabelă asociativă pentru a stabili produsele comandate.

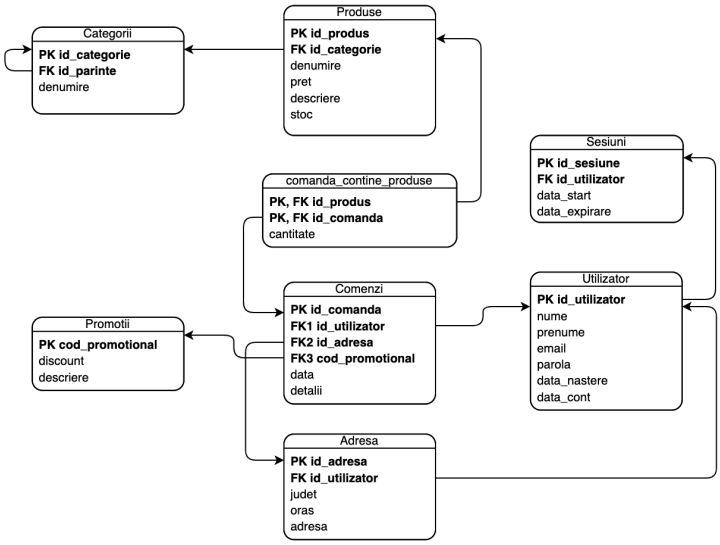
## 2. Realizați diagrama entitate-relație (ERD).

Diagrama entitate-relație aferentă bazei de date, este descrisă mai jos:




## 3. Realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.

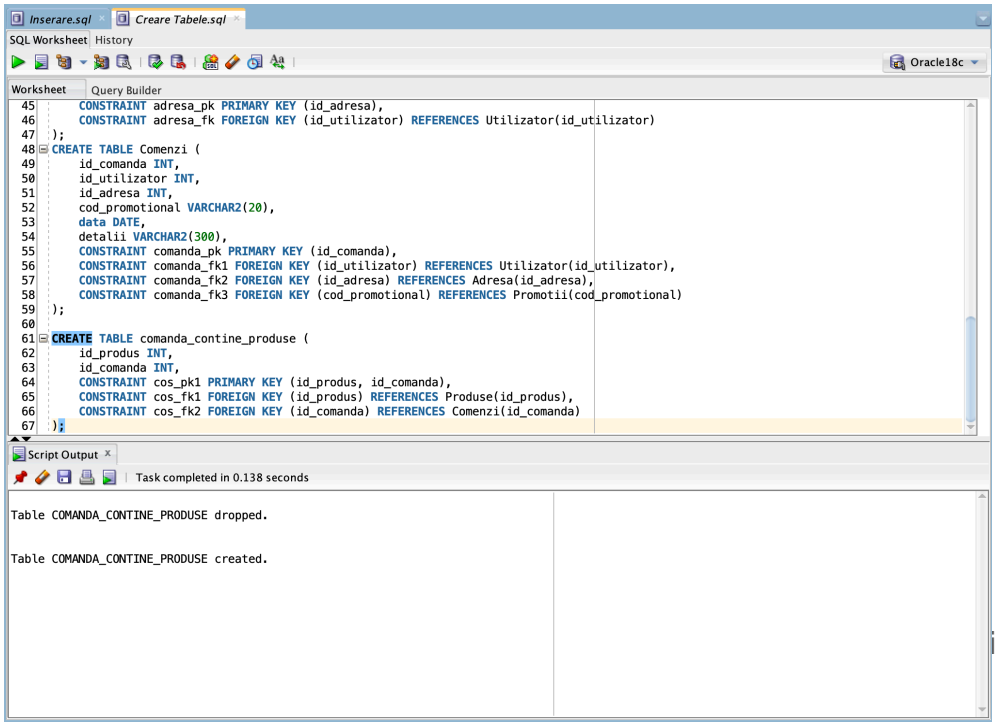
Diagrama conceptuală a modelului propus este descrisă mai jos:



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

```
CREATE TABLE Categorii (
    id_categorie INT NOT NULL,
    id_parinte INT,
    denumire VARCHAR2(20),
    CONSTRAINT categorie_pk PRIMARY KEY (id_categorie),
    CONSTRAINT categorie_fk FOREIGN KEY (id_parinte) REFERENCES Categorii(id_categorie)
);
CREATE TABLE Produse (
    id_produs INT NOT NULL,
    id_categorie INT NOT NULL,
    denumire VARCHAR2(20),
    pret INTEGER,
    descriere VARCHAR2(300),
    stoc INT,
    CONSTRAINT produs_pk PRIMARY KEY (id_produs),
    CONSTRAINT produs_fk FOREIGN KEY (id_categorie) REFERENCES Categorii(id_categorie)
);
CREATE TABLE Promotii (
    cod_promotional VARCHAR2(20) NOT NULL,
    discount INT,
    descriere VARCHAR2(300),
    CONSTRAINT promotie_pk PRIMARY KEY (cod_promotional)
);
CREATE TABLE Utilizator (
    id_utilizator INT NOT NULL,
    nume VARCHAR2(20) NOT NULL,
    prenume VARCHAR2(20) NOT NULL,
    email VARCHAR2(30),
    parola VARCHAR2(100),
    data_nastere DATE,
    data_cont DATE,
    CONSTRAINT utilizator_pk PRIMARY KEY (id_utilizator)
);
CREATE TABLE Sesiuni (
    id_sesiune INT NOT NULL,
    id_utilizator INT NOT NULL,
    data_start DATE NOT NULL,
    data_expirare DATE NOT NULL,
    CONSTRAINT sesiune_pk PRIMARY KEY (id_sesiune),
    CONSTRAINT sesiune_fk FOREIGN KEY (id_utilizator) REFERENCES Utilizator(id_utilizator)
);
```

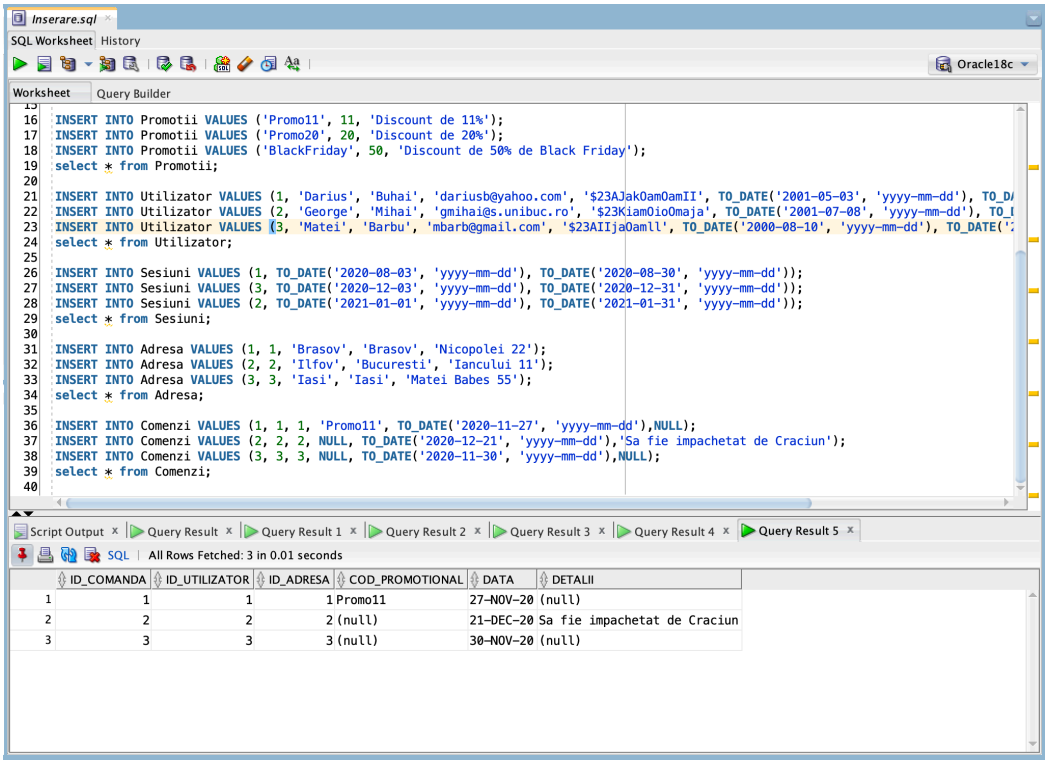
```
CREATE TABLE Adresa (
    id_adresa INT NOT NULL,
    id_utilizator INT NOT NULL,
    judet VARCHAR2(50),
    oras VARCHAR2(50),
    adresa VARCHAR2(50),
    CONSTRAINT adresa_pk PRIMARY KEY (id_adresa),
    CONSTRAINT adresa_fk FOREIGN KEY (id_utilizator) REFERENCES Utilizator(id_utilizator)
);
CREATE TABLE Comenzi (
    id_comanda INT NOT NULL,
    id_utilizator INT NOT NULL,
    id_adresa INT NOT NULL,
    cod_promotional VARCHAR2(20),
    data DATE,
    detalii VARCHAR2(300),
    CONSTRAINT comanda_pk PRIMARY KEY (id_comanda),
    CONSTRAINT comanda_fk1 FOREIGN KEY (id_utilizator) REFERENCES Utilizator(id_utilizator),
    CONSTRAINT comanda_fk2 FOREIGN KEY (id_adresa) REFERENCES Adresa(id_adresa),
    CONSTRAINT comanda_fk3 FOREIGN KEY (cod_promotional) REFERENCES Promotii(cod_promotional)
);
CREATE TABLE comanda_contine_produse (
    id_produs INT NOT NULL,
    id_comanda INT NOT NULL,
    cantitate INT,
    CONSTRAINT cos_pk1 PRIMARY KEY (id_produs, id_comanda),
    CONSTRAINT cos_fk1 FOREIGN KEY (id_produs) REFERENCES Produse(id_produs),
    CONSTRAINT cos_fk2 FOREIGN KEY (id_comanda) REFERENCES Comenzi(id_comanda)
);
```



## 5. Adăugați informații coerente în tabelele create (minim 3-5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

```
INSERT INTO Categorii VALUES (1, NULL, 'Tricouri');
INSERT INTO Categorii VALUES (2, 1, 'Polo');
INSERT INTO Categorii VALUES (3, 1, 'Clasic');
INSERT INTO Categorii VALUES (4, NULL, 'Pantaloni');
INSERT INTO Categorii VALUES (5, 4, 'Costum');
INSERT INTO Categorii VALUES (6, 4, 'Blugi');

INSERT INTO Produse VALUES (1, 2, 'Tricou Guess Alb', 211, 'Tricou de calitate superioara GUESS', 50);
INSERT INTO Produse VALUES (2, 2, 'Tricou Guess Maro', 211, 'Tricou de calitate superioara GUESS', 30);
INSERT INTO Produse VALUES (3, 3, 'Tricou Zara', 211, 'Tricou din materiale reciclabile',22);
INSERT INTO Produse VALUES (4, 6, 'Pantaloni Zara', 211, 'Pantaloni din materiale reciclabile', 15);
INSERT INTO Produse VALUES (5, 6, 'Pantaloni Reserved', 211, 'Pantaloni din materiale reciclabile', 189);

INSERT INTO Promotii VALUES ('Promo11', 11, 'Discount de 11%');
INSERT INTO Promotii VALUES ('Promo20', 20, 'Discount de 20%');
INSERT INTO Promotii VALUES ('BlackFriday', 50, 'Discount de 50% de Black Friday');

INSERT INTO Utilizator VALUES (1, 'Darius', 'Buhai', 'dariusb@yahoo.com', '$23AJakOamOamII',
TO_DATE('2001-05-03', 'yyyy-mm-dd'), TO_DATE('2020-08-03', 'yyyy-mm-dd'));
INSERT INTO Utilizator VALUES (2, 'George', 'Mihai', 'gmihai@s.unibuc.ro', '$23KiamOioOmaja',
TO_DATE('2001-07-08', 'yyyy-mm-dd'), TO_DATE('2020-08-05', 'yyyy-mm-dd'));
INSERT INTO Utilizator VALUES (3, 'Matei', 'Barbu', 'mbarb@gmail.com', '$23AIljaOamII', TO_DATE('2000-08-10',
'yyyy-mm-dd'), TO_DATE('2020-08-07', 'yyyy-mm-dd'));

INSERT INTO Sesiuni VALUES (1, 1, TO_DATE('2020-08-03', 'yyyy-mm-dd'), TO_DATE('2020-08-30', 'yyyy-mm-
dd'));
INSERT INTO Sesiuni VALUES (2, 3, TO_DATE('2020-12-03', 'yyyy-mm-dd'), TO_DATE('2020-12-31', 'yyyy-mm-
dd'));
INSERT INTO Sesiuni VALUES (3, 2, TO_DATE('2021-01-01', 'yyyy-mm-dd'), TO_DATE('2021-01-31', 'yyyy-mm-
dd'));

INSERT INTO Adresa VALUES (1, 1, 'Brasov', 'Brasov', 'Nicopolei 22');
INSERT INTO Adresa VALUES (2, 2, 'Ilfov', 'Bucuresti', 'Iancului 11');
INSERT INTO Adresa VALUES (3, 3, 'Iasi', 'Iasi', 'Matei Babes 55');

INSERT INTO Comenzi VALUES (1, 1, 1, 'Promo11', TO_DATE('2020-11-27', 'yyyy-mm-dd'),NULL);
INSERT INTO Comenzi VALUES (2, 2, 2, NULL, TO_DATE('2020-12-21', 'yyyy-mm-dd'),'Sa fie impachetat de
Craciun');
INSERT INTO Comenzi VALUES (3, 3, 3, 'Promo11', TO_DATE('2020-11-30', 'yyyy-mm-dd'),NULL);

INSERT INTO comanda_contine_produse VALUES (1, 1, 1);
INSERT INTO comanda_contine_produse VALUES (2, 3, 3);
```

```
INSERT INTO comanda_contine_produse VALUES (5, 2, 1);
INSERT INTO comanda_contine_produse VALUES (3, 2, 4);
```

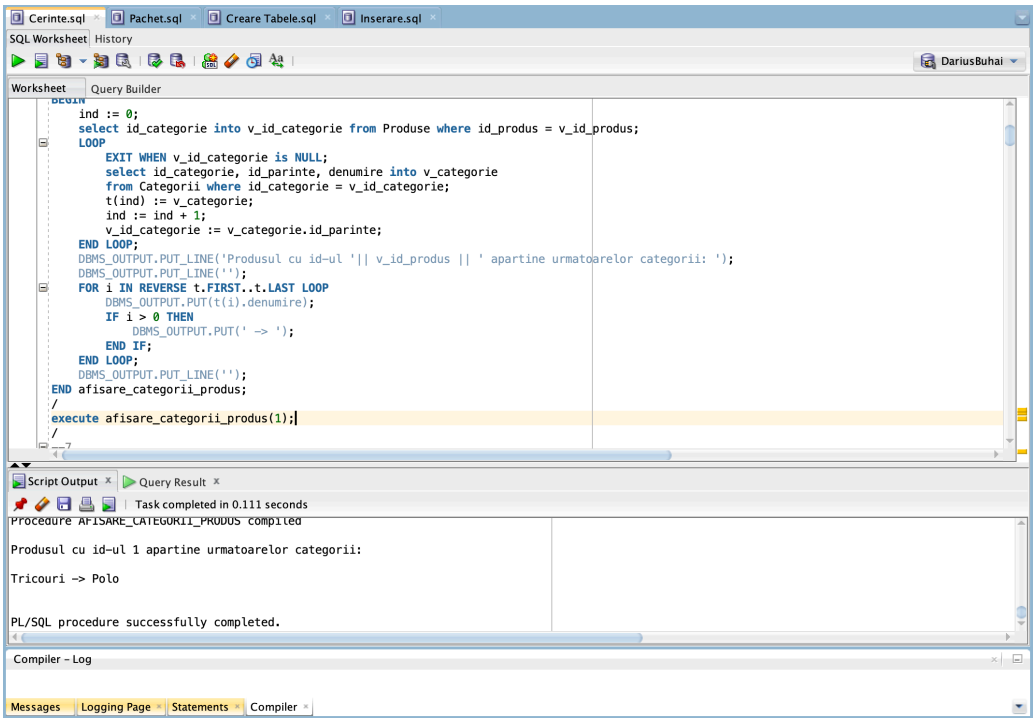



## 6. Definiți un subprogram stocat care să utilizeze un tip de colecție studiat. Apelați subprogramul.

Pentru un produs dat (id_produs), salvați si afișați categoriile din care face parte:

```
CREATE OR REPLACE PROCEDURE afisare_categorii_produs
   (v_id_produs Produse.id_produs%TYPE)
AS
   TYPE tablou_indexat IS TABLE OF Categorii%ROWTYPE INDEX BY PLS_INTEGER;
   t tablou_indexat;
   v_categorie Categorii%ROWTYPE;
   v_id_categorie Categorii.id_categorie%TYPE;
   ind NUMBER;
BEGIN
   ind := 0;
   select id_categorie into v_id_categorie from Produse where id_produs = v_id_produs;
   LOOP
      EXIT WHEN v_id_categorie is NULL;
      select id_categorie, id_parinte, denumire into v_categorie
      from Categorii where id_categorie = v_id_categorie;
```

```
    t(ind) := v_categorie;
    ind := ind + 1;
    v_id_categorie := v_categorie.id_parinte;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Produsul cu id-ul '|| v_id_produs || ' apartine urmatoarelor categorii: ');
  DBMS_OUTPUT.PUT_LINE('');
  FOR i IN REVERSE t.FIRST..t.LAST LOOP
    DBMS_OUTPUT.PUT(t(i).denumire);
    IF i > 0 THEN
      DBMS_OUTPUT.PUT(' -> ');
    END IF;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('');
END afisare_categorii_produs;
/
execute afisare_categorii_produs(1);
```

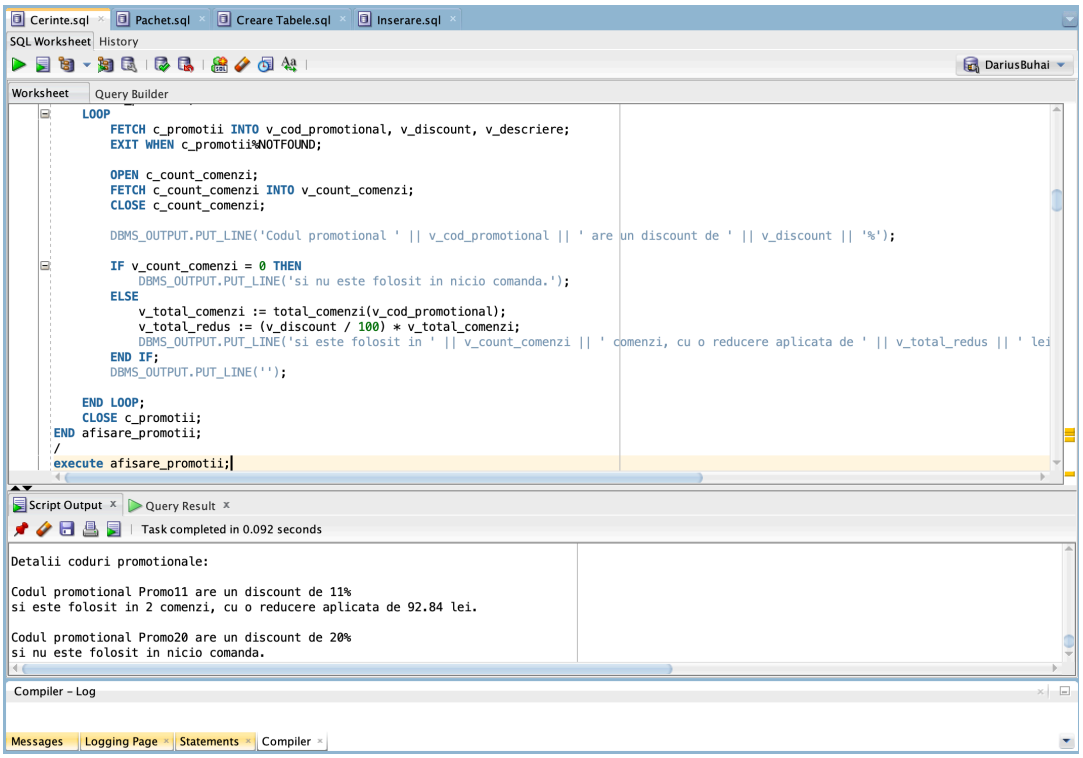



## 7. Definiți un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

Afișați pentru fiecare promoție in parte:
codul promoțional, discount-ul, numărul de comenzi pe care este aplicat și suma totală redusă.

```
CREATE OR REPLACE FUNCTION total_comenzi
   (v_cod_promotional Comenzi.cod_promotional%TYPE)
RETURN NUMBER IS
   v_total NUMBER;
BEGIN
   select sum(p.pret * cp.cantitate) into v_total
   from Comenzi c
   join comanda_contine_produse cp on (c.id_comanda = cp.id_comanda)
   join Produse p on (p.id_produs = cp.id_produs)
   where c.cod_promotional = v_cod_promotional;
   RETURN v_total;
END total_comenzi;
/
CREATE OR REPLACE PROCEDURE afisare_promotii AS
   v_count_comenzi NUMBER;
   v_total_comenzi NUMBER;
   v_total_redus NUMBER;

   TYPE t_detalii_promotie IS RECORD(
      cod_promotional Promotii.cod_promotional%TYPE,
      discount Promotii.discount%TYPE,
      descriere Promotii.descriere%TYPE
   );
   detalii_promotie t_detalii_promotie;

   CURSOR c_promotii RETURN t_detalii_promotie IS
   select cod_promotional, discount, descriere
   from Promotii;

   CURSOR c_count_comenzi IS
   select count(id_comanda) from Comenzi
   where cod_promotional = detalii_promotie.cod_promotional;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Detalii coduri promotionale: ');
   DBMS_OUTPUT.PUT_LINE('');
   OPEN c_promotii;
   LOOP
      FETCH c_promotii INTO detalii_promotie;
      EXIT WHEN c_promotii%NOTFOUND;

      OPEN c_count_comenzi;
         FETCH c_count_comenzi INTO v_count_comenzi;
      CLOSE c_count_comenzi;

      DBMS_OUTPUT.PUT_LINE('Codul promotional ' || detalii_promotie.cod_promotional || ' are un discount de ' ||
detalii_promotie.discount || '%');
```

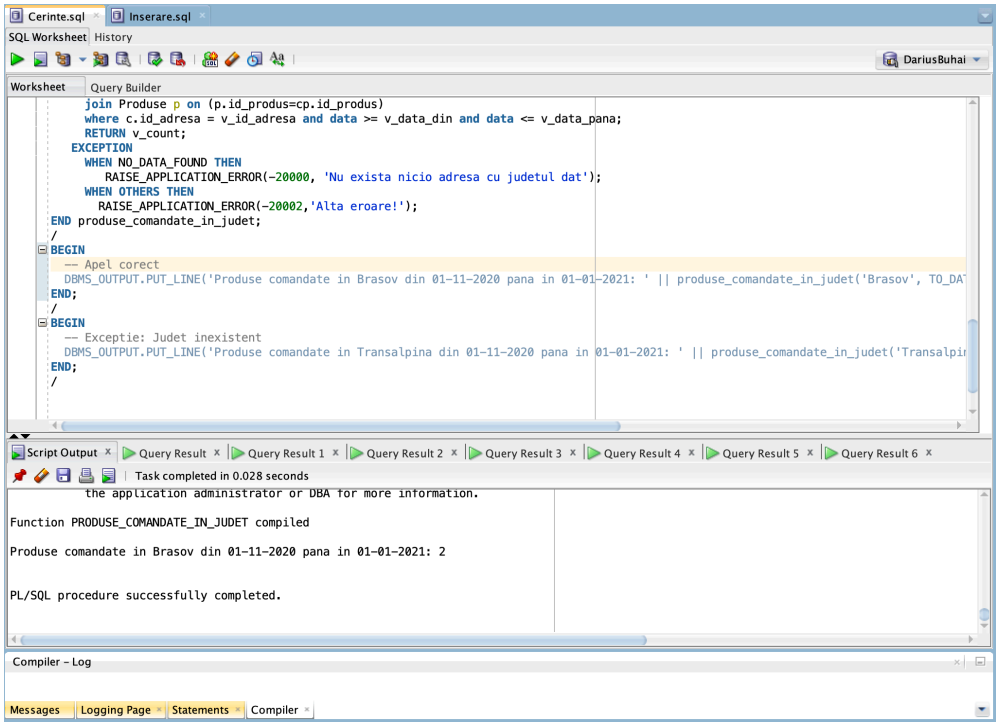
```
      IF v_count_comenzi = 0 THEN
         DBMS_OUTPUT.PUT_LINE('si nu este folosit in nicio comanda.');
      ELSE
         v_total_comenzi := total_comenzi(detalii_promotie.cod_promotional);
         v_total_redus := (detalii_promotie.discount / 100) * v_total_comenzi;
         DBMS_OUTPUT.PUT_LINE('si este folosit in ' || v_count_comenzi || ' comenzi, cu o reducere aplicata de ' ||
v_total_redus || ' lei.');
      END IF;
      DBMS_OUTPUT.PUT_LINE('');

   END LOOP;
   CLOSE c_promotii;
END afisare_promotii;
/
execute afisare_promotii;
```



## 8. Definiți un subprogram stocat de tip funcție care să utilizeze 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Creați o funcție care să returneze numărul de produse comandate dintr-un județ dat într-un anumit interval de timp:

```
CREATE OR REPLACE FUNCTION produse_comandate_in_judet
  (v_judet Adresa.judet%TYPE, v_data_din DATE, v_data_pana DATE)
RETURN NUMBER IS
    v_id_adresa Adresa.id_adresa%TYPE;
    v_count NUMBER;
  BEGIN
    select id_adresa into v_id_adresa from Adresa
    where judet = v_judet;
    select count(p.id_produs) into v_count
    from Comenzi c
    join comanda_contine_produse cp on (c.id_comanda=cp.id_comanda)
    join Produse p on (p.id_produs=cp.id_produs)
    where c.id_adresa = v_id_adresa and data >= v_data_din and data <= v_data_pana;
    RETURN v_count;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RAISE_APPLICATION_ERROR(-20000, 'Nu exista nicio adresa cu judetul dat');
    WHEN OTHERS THEN
     RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END produse_comandate_in_judet;
/
BEGIN
  -- Apel corect
  DBMS_OUTPUT.PUT_LINE('Produse comandate in Brasov din 01-11-2020 pana in 01-01-2021: ' ||
produse_comandate_in_judet('Brasov', TO_DATE('01-11-2020', 'dd-mm-yyyy'), TO_DATE('01-01-2021', 'dd-mm-
yyyy')));
END;
/
BEGIN
  -- Exceptie: Judet inexistent
  DBMS_OUTPUT.PUT_LINE('Produse comandate in Transalpina din 01-11-2020 pana in 01-01-2021: ' ||
produse_comandate_in_judet('Transalpina', TO_DATE('01-11-2020', 'dd-mm-yyyy'), TO_DATE('01-01-2020', 'dd-
mm-yyyy')));
END;
```



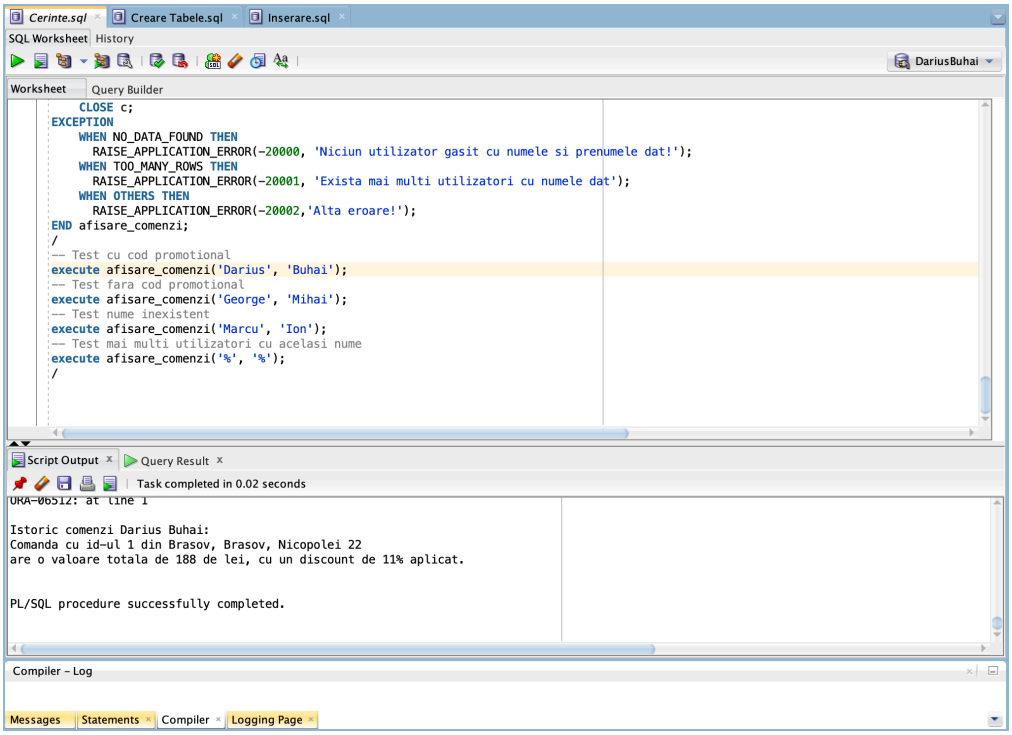

## 9. Definiți un subprogram stocat de tip procedură care să utilizeze 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Afișați toate comenzile cu adresa și totalul lor (aplicând codurile promoționale) făcute de un utilizator dat (nume și prenume):

```
CREATE OR REPLACE PROCEDURE afisare_comenzi
   (v_nume Utilizator.nume%TYPE, v_prenume Utilizator.prenume%TYPE)
IS
   v_id_utilizator Utilizator.id_utilizator%TYPE;

   v_total_comanda INT;

   v_discount Promotii.discount%TYPE;

   TYPE t_detalii_comanda IS RECORD(
      id_comanda INT,
      judet Adresa.judet%TYPE,
      oras Adresa.oras%TYPE,
      text_adresa Adresa.adresa%TYPE,
      cod_promotional Promotii.cod_promotional%TYPE
   );
   TYPE t_detalii_produs IS RECORD(
      pret Produse.pret%TYPE,
      cantitate comanda_contine_produse.cantitate%TYPE
   );
```

```
    detalii_comanda t_detalii_comanda;
    detalii_produs t_detalii_produs;

    CURSOR c_comanda RETURN t_detalii_comanda IS
    select c.id_comanda, a.judet, a.oras, a.adresa, c.cod_promotional
    from Comenzi c join Adresa a on (c.id_adresa=a.id_adresa)
    where c.id_utilizator = v_id_utilizator;

    CURSOR c_produs RETURN t_detalii_produs IS
    select p.pret, cp.cantitate from Produse p
    join comanda_contine_produse cp using(id_produs)
    where cp.id_comanda = detalii_comanda.id_comanda;

BEGIN
    select id_utilizator into v_id_utilizator from Utilizator
    where lower(nume) like lower(v_nume) and lower(prenume) like lower(v_prenume);
    DBMS_OUTPUT.PUT_LINE('Istoric comenzi ' || v_nume || ' ' || v_prenume || ': ');
    DBMS_OUTPUT.PUT_LINE('');
    OPEN c_comanda;
        LOOP
            FETCH c_comanda INTO detalii_comanda;
            EXIT WHEN c_comanda%NOTFOUND;

            v_discount := 0;
            v_total_comanda := 0;

            OPEN c_produs;
                LOOP
                    FETCH c_produs INTO detalii_produs;
                    EXIT WHEN c_produs%NOTFOUND;
                    v_total_comanda := v_total_comanda + detalii_produs.pret * detalii_produs.cantitate;
                END LOOP;
            CLOSE c_produs;

            DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul ' || detalii_comanda.id_comanda || ' din ' ||
detalii_comanda.judet || ', ' || detalii_comanda.oras || ', ' || detalii_comanda.text_adresa);

            IF detalii_comanda.cod_promotional IS NOT NULL THEN
                select discount into v_discount from Promotii
                where cod_promotional = detalii_comanda.cod_promotional;
                v_total_comanda := v_total_comanda - v_discount / 100 * v_total_comanda;
                DBMS_OUTPUT.PUT_LINE('are o valoare totala de ' || v_total_comanda || ' de lei, cu un discount de ' ||
v_discount || '% aplicat.');
            ELSE
                DBMS_OUTPUT.PUT_LINE('are o valoare totala de ' || v_total_comanda || ' de lei, si nu are nicio promotie
aplicata.');
            END IF;
```

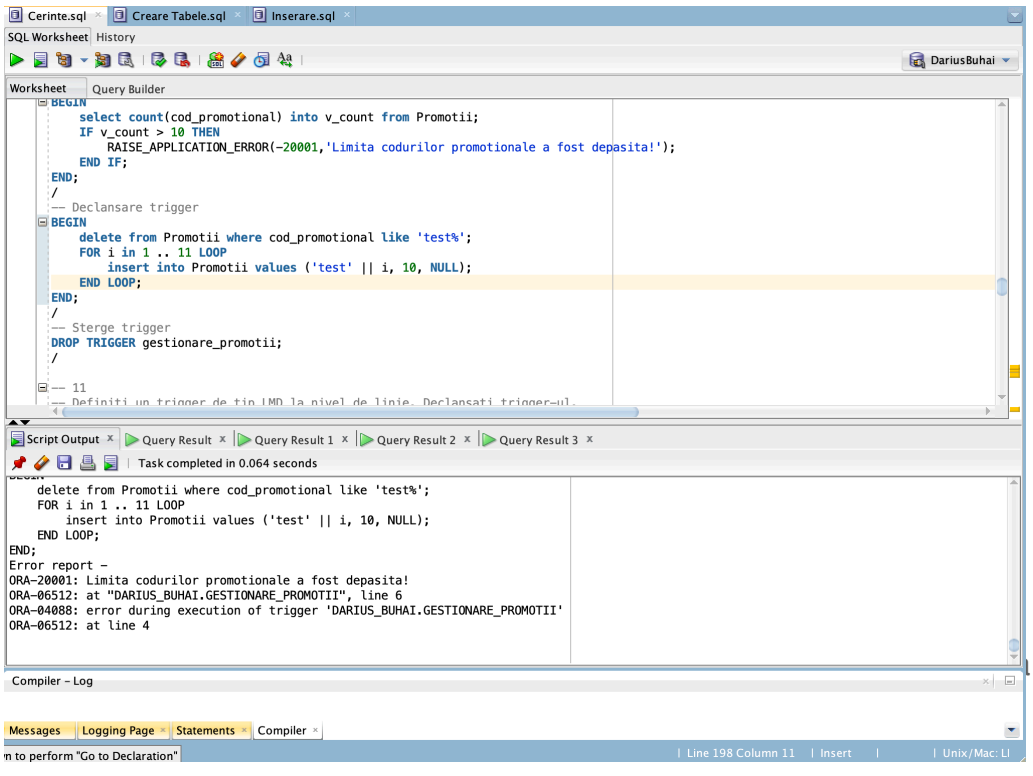
```
    END LOOP;
  CLOSE c_comanda;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20000, 'Niciun utilizator gasit cu numele si prenumele dat!');
  WHEN TOO_MANY_ROWS THEN
    RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi utilizatori cu numele dat');
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END afisare_comenzi;
/
-- Test cu cod promotional
execute afisare_comenzi('Darius', 'Buhai');
-- Test fara cod promotional
execute afisare_comenzi('George', 'Mihai');
-- Test nume inexistent
execute afisare_comenzi('Marcu', 'Ion');
-- Test mai multi utilizatori cu acelasi nume
execute afisare_comenzi('%', '%');
```



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

Creați un trigger de tip LMD la nivel de comandă care să nu permită inserarea a mai multor coduri promoționale decât 10 (pentru a descuraja reducerile de preț excesive).

```
CREATE OR REPLACE TRIGGER gestionare_promotii
  BEFORE INSERT ON Promotii
DECLARE
   v_count INT;
BEGIN
   select count(cod_promotional) into v_count from Promotii;
   IF v_count > 10 THEN
      RAISE_APPLICATION_ERROR(-20001,'Limita codurilor promotionale a fost depasita!');
   END IF;
END;
/
-- Declansare trigger
BEGIN
   FOR i in 1 .. 11 LOOP
      insert into Promotii values ('test' || i, 10, NULL);
   END LOOP;
   delete from Promotii where cod_promotional like 'test%';
END;
/
-- Sterge trigger
DROP TRIGGER gestionare_promotii;
```

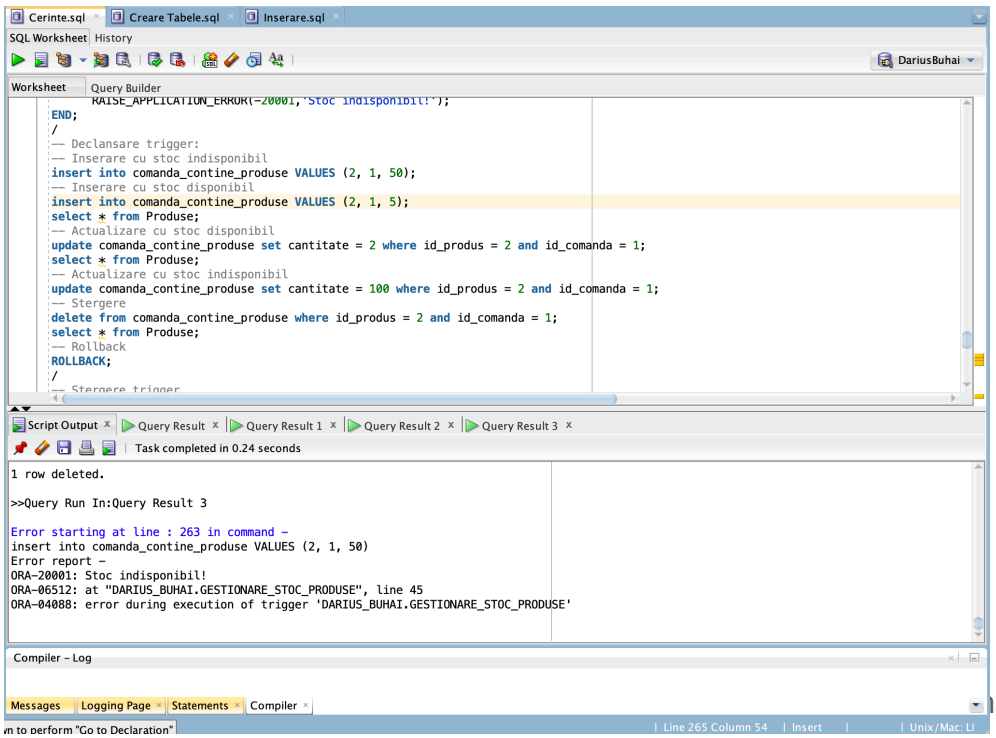


## 11. Definiți un *trigger* de tip LMD la nivel de linie. Declanșați *trigger*-ul.

Creați un trigger de tip LMD la nivel de linie care să verifice și să actualizeze stocul produselor comandate, ținând cont de toate cazurile posibile (insert, update, delete):

```
CREATE OR REPLACE TRIGGER gestionare_stoc_produse
   BEFORE INSERT OR UPDATE OR DELETE ON comanda_contine_produse
   FOR EACH ROW
DECLARE
 v_stoc Produse.stoc%TYPE;
 v_stoc_vechi Produse.stoc%TYPE;
 exceptie_stoc EXCEPTION;
BEGIN
   IF INSERTING THEN
      select stoc into v_stoc from Produse where id_produs = :NEW.id_produs;

      IF :NEW.cantitate > v_stoc THEN
         RAISE exceptie_stoc;
      END IF;

      v_stoc := v_stoc - :NEW.cantitate;
      update Produse set stoc = v_stoc where id_produs = :NEW.id_produs;
   ELSIF UPDATING THEN
      select stoc into v_stoc from Produse where id_produs = :NEW.id_produs;
      IF :OLD.id_produs != :NEW.id_produs THEN

         select stoc into v_stoc_vechi from Produse where id_produs = :OLD.id_produs;
         v_stoc_vechi := v_stoc_vechi + :OLD.cantitate;
         update Produse set stoc = v_stoc_vechi where id_produs = :OLD.id_produs;
         IF :NEW.cantitate > v_stoc THEN
            RAISE exceptie_stoc;
         END IF;
         v_stoc := v_stoc - :NEW.cantitate;
      ELSE
         IF :NEW.cantitate-:OLD.cantitate > v_stoc THEN
            RAISE exceptie_stoc;
         END IF;
         v_stoc := v_stoc - (:NEW.cantitate - :OLD.cantitate);
      END IF;
      update Produse set stoc = v_stoc where id_produs = :NEW.id_produs;
   ELSE
      select stoc into v_stoc_vechi from Produse where id_produs = :OLD.id_produs;
      v_stoc_vechi := v_stoc_vechi + :OLD.cantitate;
      update Produse set stoc = v_stoc_vechi where id_produs = :OLD.id_produs;
```

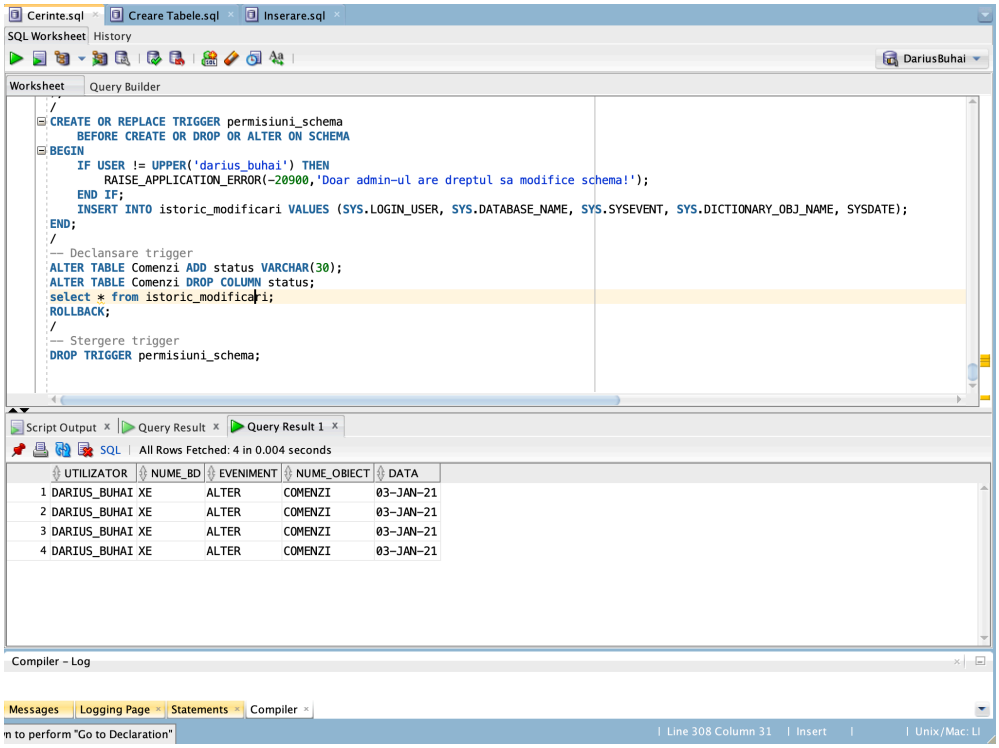
```
   END IF;
EXCEPTION
   WHEN exceptie_stoc THEN
     RAISE_APPLICATION_ERROR(-20001,'Stoc indisponibil!');
END;
/
-- Declansare trigger:
-- Inserare cu stoc indisponibil
insert into comanda_contine_produse VALUES (2, 1, 50);
-- Inserare cu stoc disponibil
insert into comanda_contine_produse VALUES (2, 1, 5);
select * from Produse;
-- Actualizare cu stoc disponibil
update comanda_contine_produse set cantitate = 2 where id_produs = 2 and id_comanda = 1;
select * from Produse;
-- Actualizare cu stoc indisponibil
update comanda_contine_produse set cantitate = 100 where id_produs = 2 and id_comanda = 1;
-- Stergere
delete from comanda_contine_produse where id_produs = 2 and id_comanda = 1;
select * from Produse;
-- Rollback
ROLLBACK;
/
-- Stergere trigger
DROP TRIGGER gestionare_stoc_produse;
```



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

Definiți un trigger de tip LDD care să permită modificarea schemei doar de către utilizatorul darius_buhai. Salvați toate modificările făcute asupra schemei în tabela istoric_modificari:

```
CREATE TABLE istoric_modificari (
    utilizator VARCHAR(30),
    nume_bd VARCHAR(50),
    eveniment VARCHAR(20),
    nume_obiect VARCHAR(30),
    data DATE
);
/
CREATE OR REPLACE TRIGGER permisiuni_schema
    BEFORE CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
    IF USER != UPPER('darius_buhai') THEN
        RAISE_APPLICATION_ERROR(-20900,'Doar admin-ul are dreptul sa modifice schema!');
    END IF;
    INSERT INTO istoric_modificari VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT,
SYS.DICTIONARY_OBJ_NAME, SYSDATE);
END;
/
-- Declansare trigger
ALTER TABLE Comenzi ADD status VARCHAR(30);
ALTER TABLE Comenzi DROP COLUMN status;
select * from istoric_modificari;
ROLLBACK;
/
-- Stergere trigger
DROP TRIGGER permisiuni_schema;
```




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

```
CREATE OR REPLACE PACKAGE proiect_bhd AS
   PROCEDURE afisare_categorii_produs(v_id_produs Produse.id_produs%TYPE);
   PROCEDURE afisare_promotii;
   FUNCTION produse_comandate_in_judet (v_judet Adresa.judet%TYPE, v_data_din DATE, v_data_pana DATE)
RETURN NUMBER;
   PROCEDURE afisare_comenzi (v_nume Utilizator.nume%TYPE, v_prenume Utilizator.prenume%TYPE);
END proiect_bhd;
/
CREATE OR REPLACE PACKAGE BODY proiect_bhd AS
   --6
   -- Definiți un subprogram stocat care să utilizeze un tip de colecție studiat. Apelați subprogramul.

   -- Pentru un produs dat (id_produs), salvati si afisati categoriile din care face parte

   PROCEDURE afisare_categorii_produs
      (v_id_produs Produse.id_produs%TYPE)
   AS
      TYPE tablou_indexat IS TABLE OF Categorii%ROWTYPE INDEX BY PLS_INTEGER;
      t tablou_indexat;
      v_categorie Categorii%ROWTYPE;
      v_id_categorie Categorii.id_categorie%TYPE;
      ind NUMBER;
```

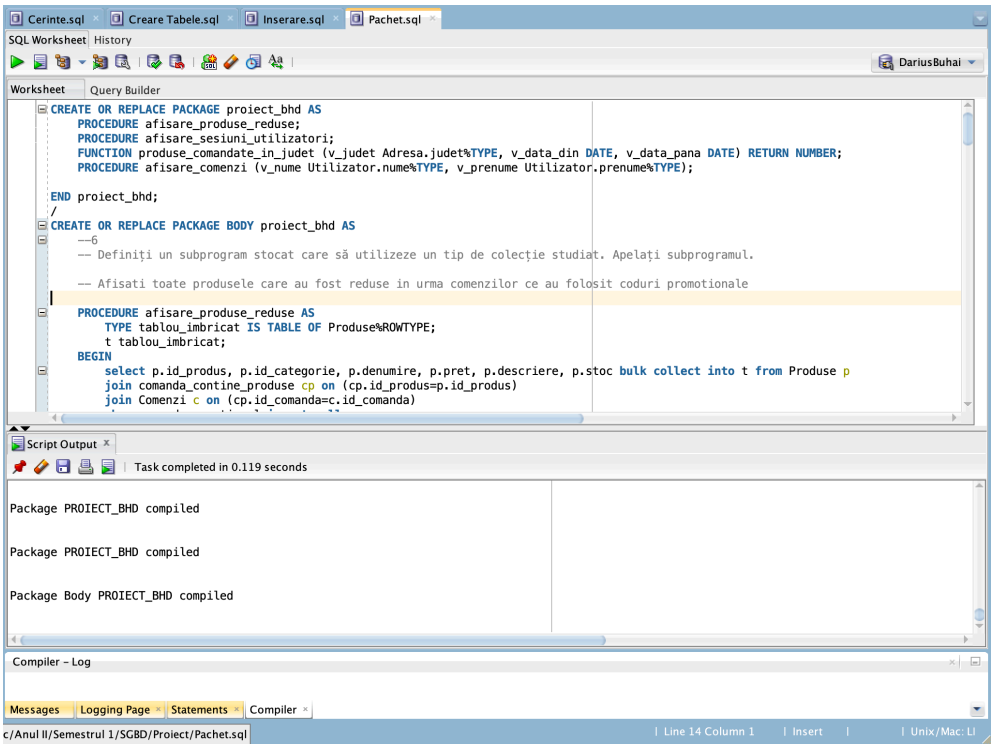
```
BEGIN
    ind := 0;
    select id_categorie into v_id_categorie from Produse where id_produs = v_id_produs;
    LOOP
        EXIT WHEN v_id_categorie is NULL;
        select id_categorie, id_parinte, denumire into v_categorie
        from Categorii where id_categorie = v_id_categorie;
        t(ind) := v_categorie;
        ind := ind + 1;
        v_id_categorie := v_categorie.id_parinte;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Produsul cu id-ul '|| v_id_produs || ' apartine urmatoarelor categorii: ');
    DBMS_OUTPUT.PUT_LINE('');
    FOR i IN REVERSE t.FIRST..t.LAST LOOP
        DBMS_OUTPUT.PUT(t(i).denumire);
        IF i > 0 THEN
            DBMS_OUTPUT.PUT(' -> ');
        END IF;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('');
END afisare_categorii_produs;

--7
-- Definiți un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

-- Afisati pentru fiecare Promotie in parte:
-- codul promotional, discount-ul, numarul de comenzi pe care este aplicat si suma totala redusa.

PROCEDURE afisare_promotii AS
    v_count_comenzi NUMBER;
    v_total_comenzi NUMBER;
    v_total_redus NUMBER;

    TYPE t_detalii_promotie IS RECORD(
        cod_promotional Promotii.cod_promotional%TYPE,
        discount Promotii.discount%TYPE,
        descriere Promotii.descriere%TYPE
    );
    detalii_promotie t_detalii_promotie;

    CURSOR c_promotii RETURN t_detalii_promotie IS
    select cod_promotional, discount, descriere
    from Promotii;

    CURSOR c_count_comenzi IS
    select count(id_comanda) from Comenzi
```

```
    where cod_promotional = detalii_promotie.cod_promotional;
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Detalii coduri promotionale: ');
    DBMS_OUTPUT.PUT_LINE('');
    OPEN c_promotii;
    LOOP
      FETCH c_promotii INTO detalii_promotie;
      EXIT WHEN c_promotii%NOTFOUND;

      OPEN c_count_comenzi;
        FETCH c_count_comenzi INTO v_count_comenzi;
      CLOSE c_count_comenzi;

      DBMS_OUTPUT.PUT_LINE('Codul promotional ' || detalii_promotie.cod_promotional || ' are un discount de ' ||
detalii_promotie.discount || '%');

      IF v_count_comenzi = 0 THEN
        DBMS_OUTPUT.PUT_LINE('si nu este folosit in nicio comanda.');
      ELSE
        v_total_comenzi := total_comenzi(detalii_promotie.cod_promotional);
        v_total_redus := (detalii_promotie.discount / 100) * v_total_comenzi;
        DBMS_OUTPUT.PUT_LINE('si este folosit in ' || v_count_comenzi || ' comenzi, cu o reducere aplicata de ' ||
v_total_redus || ' lei.');
      END IF;
      DBMS_OUTPUT.PUT_LINE('');

    END LOOP;
    CLOSE c_promotii;
  END afisare_promotii;

  -- 8
  -- Definiți un subprogram stocat de tip funcție care să utilizeze 3 dintre tabelele definite. Tratați toate
  -- excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

  -- Creati o functie care sa returneze numarul de produse comandate dintr-un judet dat intr-un anumit interval de
timp

  FUNCTION produse_comandate_in_judet
    (v_judet Adresa.judet%TYPE, v_data_din DATE, v_data_pana DATE)
  RETURN NUMBER IS
     v_id_adresa Adresa.id_adresa%TYPE;
     v_count NUMBER;
    BEGIN
     select id_adresa into v_id_adresa from Adresa
     where judet = v_judet;
     select count(p.id_produs) into v_count
     from Comenzi c
```

```
      join comanda_contine_produse cp on (c.id_comanda=cp.id_comanda)
      join Produse p on (p.id_produs=cp.id_produs)
      where c.id_adresa = v_id_adresa and data >= v_data_din and data <= v_data_pana;
      RETURN v_count;
    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista nicio adresa cu judetul dat');
      WHEN OTHERS THEN
       RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
  END produse_comandate_in_judet;

  -- 9
  -- Definiți un subprogram stocat de tip procedură care să utilizeze 5 dintre tabelele definite. Tratați toate
  -- excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

  -- Afisati toate comenzile cu adresa si totalul lor (aplicand codurile promotionale) facute de un utilizator dat (nume
si prenume)

  PROCEDURE afisare_comenzi
    (v_nume Utilizator.nume%TYPE, v_prenume Utilizator.prenume%TYPE)
  IS
    v_id_utilizator Utilizator.id_utilizator%TYPE;

    v_total_comanda INT;

    v_discount Promotii.discount%TYPE;

    TYPE t_detalii_comanda IS RECORD(
      id_comanda INT,
      judet Adresa.judet%TYPE,
      oras Adresa.oras%TYPE,
      text_adresa Adresa.adresa%TYPE,
      cod_promotional Promotii.cod_promotional%TYPE
    );
    TYPE t_detalii_produs IS RECORD(
      pret Produse.pret%TYPE,
      cantitate comanda_contine_produse.cantitate%TYPE
    );
    detalii_comanda t_detalii_comanda;
    detalii_produs t_detalii_produs;

    CURSOR c_comanda RETURN t_detalii_comanda IS
    select c.id_comanda, a.judet, a.oras, a.adresa, c.cod_promotional
    from Comenzi c join Adresa a on (c.id_adresa=a.id_adresa)
    where c.id_utilizator = v_id_utilizator;
```

```
    CURSOR c_produs RETURN t_detalii_produs IS
    select p.pret, cp.cantitate from Produse p
    join comanda_contine_produse cp using(id_produs)
    where cp.id_comanda = detalii_comanda.id_comanda;

  BEGIN
    select id_utilizator into v_id_utilizator from Utilizator
    where lower(nume) like lower(v_nume) and lower(prenume) like lower(v_prenume);
    DBMS_OUTPUT.PUT_LINE('Istoric comenzi ' || v_nume || ' ' || v_prenume || ': ');
    DBMS_OUTPUT.PUT_LINE('');
    OPEN c_comanda;
      LOOP
        FETCH c_comanda INTO detalii_comanda;
        EXIT WHEN c_comanda%NOTFOUND;

        v_discount := 0;
        v_total_comanda := 0;

        OPEN c_produs;
          LOOP
            FETCH c_produs INTO detalii_produs;
            EXIT WHEN c_produs%NOTFOUND;
            v_total_comanda := v_total_comanda + detalii_produs.pret * detalii_produs.cantitate;
          END LOOP;
        CLOSE c_produs;

        DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul ' || detalii_comanda.id_comanda || ' din ' ||
detalii_comanda.judet || ', ' || detalii_comanda.oras || ', ' || detalii_comanda.text_adresa);

        IF detalii_comanda.cod_promotional IS NOT NULL THEN
          select discount into v_discount from Promotii
          where cod_promotional = detalii_comanda.cod_promotional;
          v_total_comanda := v_total_comanda - v_discount / 100 * v_total_comanda;
          DBMS_OUTPUT.PUT_LINE('are o valoare totala de ' || v_total_comanda || ' de lei, cu un discount de ' ||
v_discount || '% aplicat.');
        ELSE
          DBMS_OUTPUT.PUT_LINE('are o valoare totala de ' || v_total_comanda || ' de lei, si nu are nicio
promotie aplicata.');
        END IF;

      END LOOP;
    CLOSE c_comanda;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
     RAISE_APPLICATION_ERROR(-20000, 'Niciun utilizator gasit cu numele si prenumele dat!');
    WHEN TOO_MANY_ROWS THEN
     RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi utilizatori cu numele dat');
```

```
      WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
   END afisare_comenzi;

END proiect_bhd;
/
-- Testare pachet
-- 6
execute proiect_bhd.afisare_categorii_produs(1);
-- 7
execute proiect_bhd.afisare_promotii();
-- 8
/
BEGIN
   -- Apel corect
   DBMS_OUTPUT.PUT_LINE('Produse comandate in Brasov din 01-11-2020 pana in 01-01-2021: ' ||
proiect_bhd.produse_comandate_in_judet('Brasov', TO_DATE('01-11-2020', 'dd-mm-yyyy'), TO_DATE('01-01-2021',
'dd-mm-yyyy')));
END;
/
BEGIN
   -- Exceptie: Judet inexistent
   DBMS_OUTPUT.PUT_LINE('Produse comandate in Transalpina din 01-11-2020 pana in 01-01-2021: ' ||
proiect_bhd.produse_comandate_in_judet('Transalpina', TO_DATE('01-11-2020', 'dd-mm-yyyy'),
TO_DATE('01-01-2020', 'dd-mm-yyyy')));
END;
/
-- 9
-- Test cu cod promotional
execute proiect_bhd.afisare_comenzi('Darius', 'Buhai');
-- Test fara cod promotional
execute proiect_bhd.afisare_comenzi('George', 'Mihai');
-- Test nume inexistent
execute proiect_bhd.afisare_comenzi('Marcu', 'Ion');
-- Test mai multi utilizatori cu acelasi nume
execute proiect_bhd.afisare_comenzi('%', '%');
```




## 14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare pentru acțiuni integrate.

```
CREATE OR REPLACE PACKAGE proiect_bhd_2 AS

  TYPE tablou_indexat IS TABLE OF Categorii%ROWTYPE INDEX BY PLS_INTEGER;

  TYPE t_detalii_promotie IS RECORD(
     cod_promotional Promotii.cod_promotional%TYPE,
     discount Promotii.discount%TYPE,
     descriere Promotii.descriere%TYPE
  );

  CURSOR c_promotii RETURN t_detalii_promotie;

  FUNCTION total_comenzi (v_cod_promotional Comenzi.cod_promotional%TYPE) RETURN NUMBER;

  TYPE t_detalii_comanda IS RECORD(
     id_comanda INT,
     judet Adresa.judet%TYPE,
     oras Adresa.oras%TYPE,
     text_adresa Adresa.adresa%TYPE,
     cod_promotional Promotii.cod_promotional%TYPE
  );
  TYPE t_detalii_produs IS RECORD(
     pret Produse.pret%TYPE,
     cantitate comanda_contine_produse.cantitate%TYPE
```

```
   );

END proiect_bhd_2;
/
CREATE OR REPLACE PACKAGE BODY proiect_bhd_2 AS

   CURSOR c_promotii RETURN t_detalii_promotie IS
   select cod_promotional, discount, descriere
   from Promotii;

   FUNCTION total_comenzi
      (v_cod_promotional Comenzi.cod_promotional%TYPE)
   RETURN NUMBER IS
      v_total NUMBER;
   BEGIN
      select sum(p.pret * cp.cantitate) into v_total
      from Comenzi c
      join comanda_contine_produse cp on (c.id_comanda = cp.id_comanda)
      join Produse p on (p.id_produs = cp.id_produs)
      where c.cod_promotional = v_cod_promotional;
      RETURN v_total;
   END total_comenzi;

END proiect_bhd_2;
```