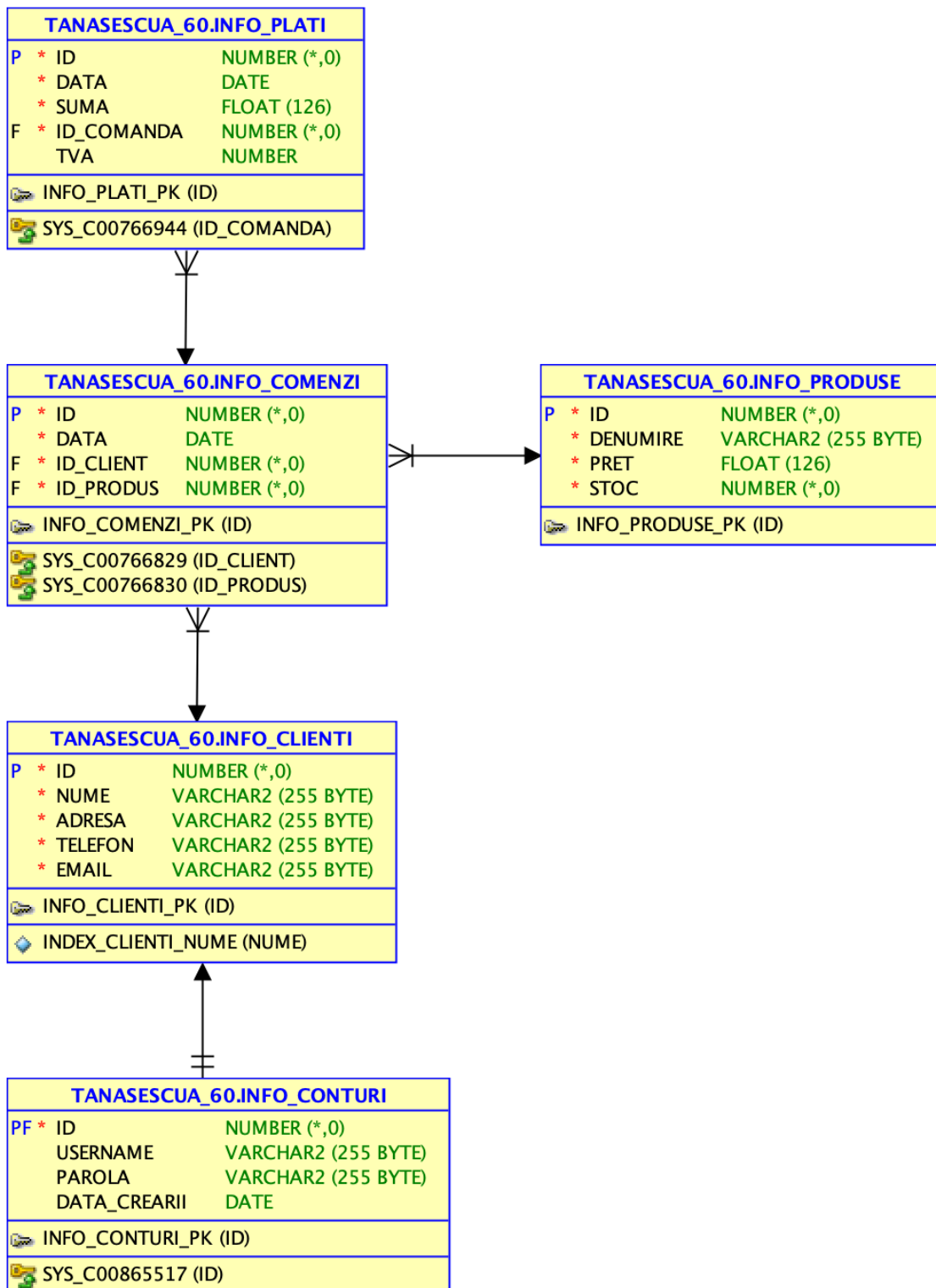


Proiect SGBD

Am ales sa realizez baza a unui magazin cu scopul de a stoca ierarhia datelor.

Relatiile dintre tabele sunt:

- fiecare comanda plasata are unul sau mai multe produse;
- fiecare client poate avea una sau mai multe comenzi;
- fiecare comanda conține informații despre plata;
- fiecare client are un singur cont, salvat in tabela info_conturi.



```

create table info_clienti (
    id int primary key,
    nume varchar2(255) not null,
    adresa varchar2(255) not null,
    telefon varchar2(255) not null,
    email varchar2(255) not null
);

create table info_produce (
    id int primary key,
    denumire varchar2(255) not null,
    pret float not null,
    stoc int not null
);

create table info_comenzi (
    id int GENERATED BY DEFAULT AS IDENTITY primary key,
    data date not null,
    id_client int not null,
    id_product int not null,
    foreign key(id_client) references info_clienti(id),
    foreign key(id_product) references info_produce(id)
);

create table info_plati (
    id int GENERATED BY DEFAULT AS IDENTITY primary key,
    data date not null,
    suma float not null,
    id_comanda int not null,
    foreign key(id_comanda) references info_comenzi(id),
    tva number default 19
);

create table info_conturi (
    id int primary key,
    username varchar2(255),
    parola varchar(255),
    data_crearii date,
    foreign key(id) REFERENCES info_clienti(id)
);

```

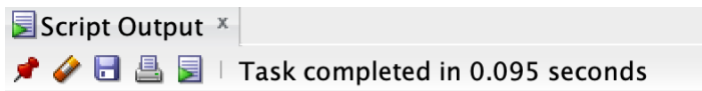
Structuri de control + 3 cursori implicite + 3 cursori explicite

Bloc 1 - utilizează o structură de control 'IF' pentru a verifica dacă suma totală a plăților depășește 1000. Dacă da, TVA-ul este redus la 16%, altfel rămâne la valoarea implicită de 19%.

```

DECLARE
    v_suma_plati float;
    v_tva number := 19;
BEGIN
    SELECT SUM(suma) INTO v_suma_plati
    FROM info_plati;
    IF v_suma_plati > 1000 THEN
        v_tva := 16;
    END IF;
    UPDATE info_plati SET tva = v_tva;
    DBMS_OUTPUT.PUT_LINE('Suma totala a platilor este: ' || v_suma_plati);
    DBMS_OUTPUT.PUT_LINE('Valoarea TVA-ului este: ' || v_tva);
END;

```



Suma totala a platilor este: 587,1538044
Valoarea TVA-ului este: 19

PL/SQL procedure successfully completed.

--Bloc 2 - utilizează un cursor și afișează denumirea produsului și valoarea totala din suma cu tva pe baza id-ului clientului introdus de la tastatura, cursor explicit

```
DECLARE
  v_denumire_produs info_produce.denumire%TYPE;
  v_valoare_totala NUMBER;
  v_id_client info_clienti.id%TYPE;
  CURSOR c_produce (p_id_client info_clienti.id%TYPE) IS
    SELECT denumire, SUM(pret + pret * tva / 100) AS valoare_totala
    FROM info_produce
    JOIN info_comenzi ON info_produce.id = info_comenzi.id_produs
    JOIN info_plati ON info_comenzi.id = info_plati.id_comanda
    WHERE info_comenzi.id_client = p_id_client
    GROUP BY denumire;
BEGIN
  dbms_output.put_line('Introduceti id-ul clientului:');
  v_id_client := &id_client;

  OPEN c_produce(v_id_client);
  LOOP
    FETCH c_produce INTO v_denumire_produs, v_valoare_totala;
    EXIT WHEN c_produce%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_denumire_produs || ': ' || v_valoare_totala);
  END LOOP;
  CLOSE c_produce;
END;
```

```
    CLOSE c_produce;
END;
Introduceti id-ul clientului:
Switch 64 ports 10Gb: 244,465984
```

PL/SQL procedure successfully completed.

--Bloc 3 - citește de la tastatura id-ul unei persoane și afișează numărul de comenzi al acesteia, cursor implicit

```
DECLARE
  v_id_client int;
  v_nr_comenzi int := 0;
BEGIN
  v_id_client := &id_client;
  FOR c IN (SELECT COUNT(*) AS nr_comenzi FROM info_comenzi WHERE id_client = v_id_client) LOOP
    v_nr_comenzi := c.nr_comenzi;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Numarul de comenzi ale clientului cu ID-ul ' || v_id_client || ' este ' || v_nr_comenzi);
END;
```

Script Output x
Task completed in 1.861 seconds

```

END LOOP;
DBMS_OUTPUT.PUT_LINE('Numarul de comenzi ale clientului cu
END;
Numarul de comenzi ale clientului cu ID-ul 2 este 1

```

PL/SQL procedure successfully completed.

-- Bloc 4 - citește de la tastatura id_ul persoanei și afișează prețul total pe care îl are de achitat cu suma plus tva la toate comenzile, cursor explicit

```

DECLARE
client_id info_clienti.id%TYPE;
total_plata FLOAT := 0;
CURSOR c_plati IS
SELECT ip.suma * ip.tva / 100 + ip.suma AS total
FROM info_comenzi ic
JOIN info_plati ip ON ic.id = ip.id_comanda
WHERE ic.id_client = client_id;
BEGIN
client_id := &id_client;

OPEN c_plati;
FETCH c_plati INTO total_plata;
IF c_plati%NOTFOUND THEN
DBMS_OUTPUT.PUT_LINE('Nu exista comenzi si plati pentru clientul cu id-ul ' || client_id || '.');
ELSE
DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul ' || client_id || ' are de achitat ' || total_plata || ' lei. ');
END IF;
CLOSE c_plati;
END;

```

Script Output x
Task completed in 1.414 seconds

```

DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul ' || clien
END IF;
END;
Clientul cu id-ul 1 are de achitat 390,3708249 lei.

PL/SQL procedure successfully completed.

```

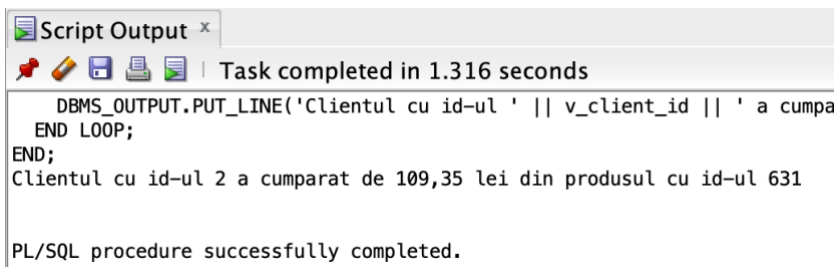
-- Bloc 5 - citește de la tastatura id-ul clientului și calculează prețul total pentru fiecare produs al clientului, cursor implicit

```

DECLARE
v_client_id info_clienti.id%TYPE;
v_product_id info_produce.id%TYPE;
v_product_price info_produce.pret%TYPE;
v_total_price info_plati.suma%TYPE;
BEGIN
v_client_id := &v_client_id;
FOR c IN (
SELECT id_produc, data
FROM info_comenzi
WHERE id_client = v_client_id
)
LOOP
v_product_id := c.id_produc;
SELECT pret INTO v_product_price FROM info_produce WHERE id = v_product_id;
SELECT SUM(suma) INTO v_total_price FROM info_plati WHERE id_comanda IN (
SELECT id FROM info_comenzi WHERE id_client = v_client_id AND id_produc = v_product_id

```

```
);
DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul ' || v_client_id || ' a cumparat de ' || v_total_price || ' lei din produsul
cu id-ul ' || v_product_id);
END LOOP;
END;
```



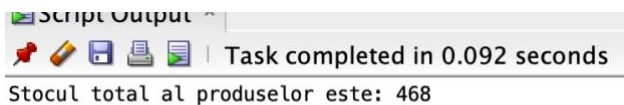
```
DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul ' || v_client_id || ' a cumpa
END LOOP;
END;
Clientul cu id-ul 2 a cumparat de 109,35 lei din produsul cu id-ul 631

PL/SQL procedure successfully completed.
```

-- Bloc 6 – calculeaza stocul total al produselor utilizand un curosor implicit

```
DECLARE
v_stoc_total INT := 0;
BEGIN
FOR C IN (SELECT stoc FROM info_produce)
LOOP
v_stoc_total := v_stoc_total + C.stoc;
END LOOP;

dbms_output.put_line('Stocul total al produselor este: ' || v_stoc_total);
END;
```

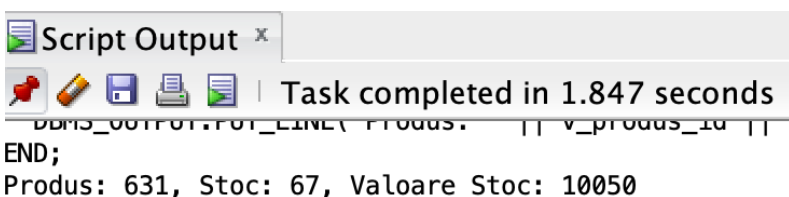


```
Stocul total al produselor este: 468
```

PL/SQL procedure successfully completed.

-- Bloc 7 – am folosit un cursor explicit pentru a afisa denumire, stocul si valoarea stocului pentru id ul produsului citit de la tastatura

```
DECLARE
v_total_stoc int := 0;
v_produus_id info_produce.id%TYPE;
v_valoare_stoc FLOAT := 0;
BEGIN
v_produus_id := &id_produus;
SELECT stoc INTO v_total_stoc FROM info_produce WHERE id = v_produus_id;
SELECT pret INTO v_valoare_stoc FROM info_produce WHERE id = v_produus_id;
v_valoare_stoc := v_total_stoc * v_valoare_stoc;
DBMS_OUTPUT.PUT_LINE('Produs: ' || v_produus_id || ', Stoc: ' || v_total_stoc || ', Valoare Stoc: ' || v_valoare_stoc);
END;
```



```
DBMS_OUTPUT.PUT_LINE('Produs: ' || v_produus_id || ', Stoc: ' || v_total_stoc || ', Valoare Stoc: ' || v_valoare_stoc);
END;
Produs: 631, Stoc: 67, Valoare Stoc: 10050
```

PL/SQL procedure successfully completed.

Tratarea exceptiilor: 2 exceptii implicite + 2 exceptii explicite

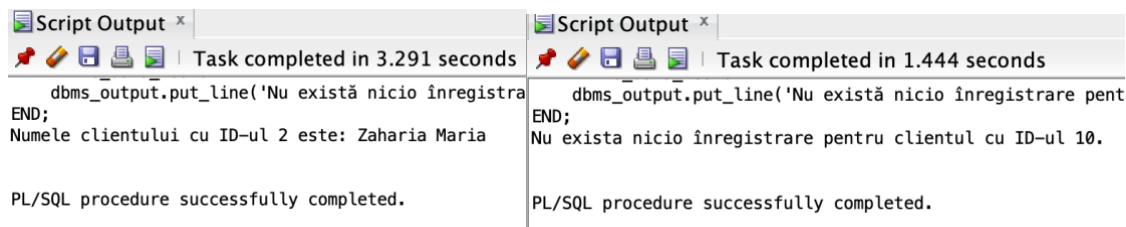
-- 1. Blocul contine o exceptie implicita (no_data_found) care este afisata atunci cand nu exista clientul cu id-ul citit de la tastatura

```
DECLARE
    v_client_id info_clienti.ID%TYPE;
    v_client_nume info_clienti.nume%TYPE;

BEGIN
    v_client_id := &client_id;

    SELECT nume INTO v_client_nume FROM info_clienti WHERE ID = v_client_id;
    dbms_output.put_line('Numele clientului cu ID-ul ' || v_client_id || ' este: ' || v_client_nume);

EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('Nu există nicio înregistrare pentru clientul cu ID-ul ' || v_client_id || '.');
END;
```



-- 2. Blocul contine o exceptie implicita (no_data_found) care este afisata cu mesajul “Nu există comenzi pentru clientul cu ID-ul” atunci cand nu exista comenzi pentru clientul cu id-ul citit de la tastatura

```
DECLARE
    v_client_id info_clienti.ID%TYPE;
    v_nr_comenzi INT := 0;

CURSOR c_comenzi (p_id_client info_clienti.ID%TYPE) IS
    SELECT ID, DATA
    FROM info_comenzi
    WHERE id_client = p_id_client;

    v_comanda_id info_comenzi.ID%TYPE;
    v_comanda_data info_comenzi.DATA%TYPE;

BEGIN
    v_client_id := &client_id;

    SELECT COUNT(*) INTO v_nr_comenzi
    FROM info_comenzi
    WHERE id_client = v_client_id;

    IF v_nr_comenzi > 0 THEN
        dbms_output.put_line('Clientul cu ID-ul ' || v_client_id || ' a plasat ' || v_nr_comenzi || ' comenzi.');

OPEN c_comenzi(v_client_id);



LOOP
    FETCH c_comenzi INTO v_comanda_id, v_comanda_data;
    EXIT WHEN c_comenzi%notfound;
    dbms_output.put_line('Comanda cu ID-ul ' || v_comanda_id || ', Data: ' || v_comanda_data);
END LOOP;



CLOSE c_comenzi;

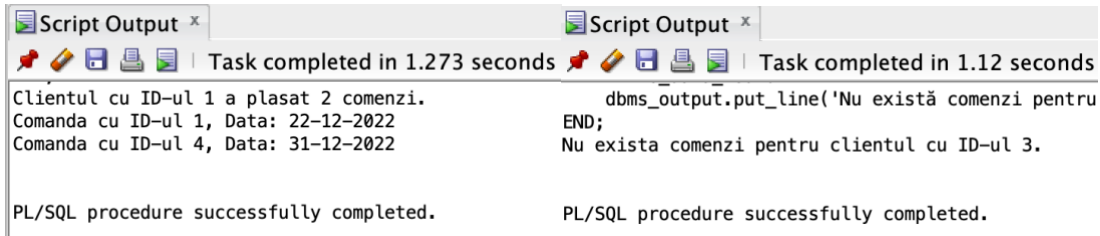

```

```

ELSE
    RAISE no_data_found;
END IF;

EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('Nu există comenzi pentru clientul cu ID-ul ' || v_client_id || ');
END;

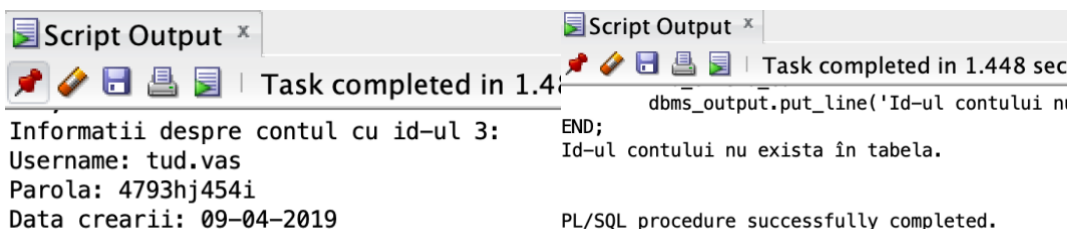
```



-- 3. Blocul contine o exceptie explicita (nu_exista_cont) care verifica pe baza id ului al contului citit de la tastatura daca contul exista sau nu, in cazul in care exista se afiseaza informatiile despre cont, in caz contrar este afisata exceptia

```

DECLARE
    v_count NUMBER;
    nu_exista_cont EXCEPTION;
    p_id_cont info_conturi.ID%TYPE;
    v_username info_conturi.username%TYPE;
    v_parola info_conturi.parola%TYPE;
    v_data_crearii info_conturi.data_crearii%TYPE;
    CURSOR c IS
        SELECT COUNT(*) FROM info_conturi WHERE ID = p_id_cont;
BEGIN
    p_id_cont := &id_cont;
    OPEN c;
    FETCH c INTO v_count;
    CLOSE c;
    IF v_count = 0 THEN
        RAISE nu_exista_cont;
    ELSE
        SELECT username, parola, data_crearii
        INTO v_username, v_parola, v_data_crearii
        FROM info_conturi
        WHERE ID = p_id_cont;
        dbms_output.put_line('Informatii despre contul cu id-ul ' || p_id_cont || ':');
        dbms_output.put_line('Username: ' || v_username);
        dbms_output.put_line('Parola: ' || v_parola);
        dbms_output.put_line('Data creării: ' || v_data_crearii);
    END IF;
EXCEPTION
    WHEN nu_exista_cont THEN
        dbms_output.put_line('Id-ul contului nu există în tabela.');
```



-- 4. Blocul contine o exceptie explicita (stoc_epuizat) care verifica pe baza id-ului citit de la tastatura al produsului daca este pe stoc, in cazului in care este disponibil afiseaza si numarul de bucati, in caz contrar afiseaza mesajul exceptiei

SET SERVEROUTPUT ON

DECLARE

```
v_count INT;
v_id_produs info_produce.ID%TYPE;
v_stoc info_produce.stoc%TYPE;
stoc_epuizat EXCEPTION;
CURSOR C IS
    SELECT stoc FROM info_produce WHERE ID = v_id_produs;
```

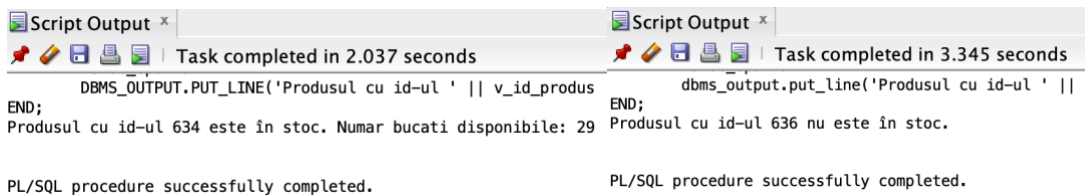
BEGIN

```
v_id_produs := &id;
OPEN C;
FETCH C INTO v_count;
CLOSE C;
IF v_count <= 0 THEN
    RAISE stoc_epuizat;
ELSE
    SELECT stoc INTO v_stoc FROM info_produce WHERE ID = v_id_produs;
    dbms_output.put_line('Produsul cu id-ul ' || v_id_produs || ' este în stoc. Numar bucati disponibile: ' || v_stoc);
END IF;
```

EXCEPTION

```
WHEN stoc_epuizat THEN
    dbms_output.put_line('Produsul cu id-ul ' || v_id_produs || ' nu este în stoc.');
```

END;



3 Functii + 3 Proceduri + 2 Pachete:

-- 1. Functie care calculeaza valoarea totala a comenzii pe baza id-ului de comanda citit de la tastatura.

CREATE OR REPLACE FUNCTION calculeaza_total_comanda(p_id_comanda IN info_comenzi.ID%TYPE)

RETURN NUMBER

IS

```
v_total NUMBER := 0;
```

BEGIN

```
SELECT SUM(ip.pret) INTO v_total
FROM info_produce ip
JOIN info_comenzi ic ON ip.ID = ic.id_produs
WHERE ic.ID = p_id_comanda;
```

```
RETURN v_total;
```

EXCEPTION

```
WHEN no_data_found THEN
```

```
RETURN 0;
```

END;

/

DECLARE

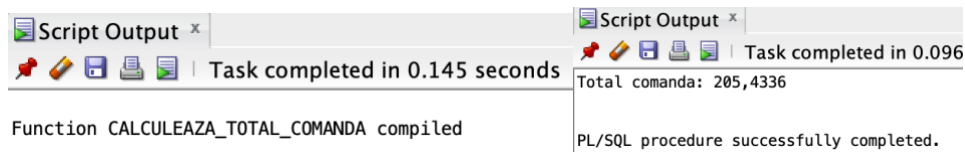
```
v_total NUMBER;
```

BEGIN


```

v_total := calculeaza_total_comanda(2);
dbms_output.put_line('Total comanda: ' || v_total);
END;

```



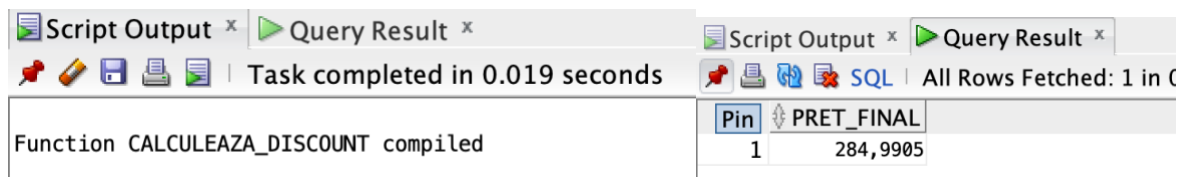
-- 2. Functia calculeaza si aplica un discount pentru produse, in functie de anumite criterii.

set serveroutput on

```

CREATE OR REPLACE FUNCTION calculeaza_discount(p_id_produc IN INT) RETURN FLOAT IS
    discount FLOAT;
    pret_final FLOAT;
    v_pret info_produce.pret%type;
BEGIN
    SELECT pret INTO v_pret
    FROM info_produce
    WHERE id = p_id_produc;
    IF v_pret <= 150 THEN
        discount := 0.1;
    ELSE
        discount := 0.05;
    END IF;
    pret_final:=v_pret - (v_pret * discount);
    RETURN pret_final;
END;
/
SELECT calculeaza_discount(630) AS Pret_final FROM dual;

```



-- 3. Functia calculeaza valoarea totala a comenzilor inregistrate in tabela info_comenzi, utilizand un cursor.

SET SERVEROUTPUT ON

```

CREATE OR REPLACE FUNCTION calcul_valoare_totala_comenzi RETURN NUMBER IS
    v_valoare_totala NUMBER := 0;
    CURSOR C IS
        SELECT id_produc, SUM(pret) AS valoare_totala
        FROM info_comenzi
        JOIN info_produce ON info_comenzi.id_produc = info_produce.ID
        GROUP BY id_produc;
BEGIN
    FOR rec_info_comenzi IN C LOOP
        v_valoare_totala := v_valoare_totala + rec_info_comenzi.valoare_totala;
    END LOOP;

    RETURN v_valoare_totala;
END;
/

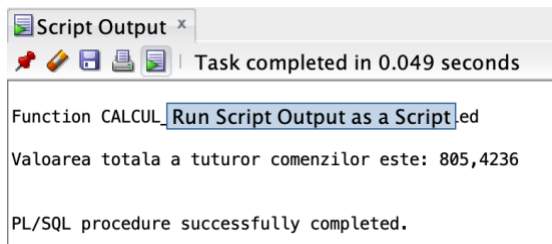
DECLARE
    v_total_comenzi NUMBER;

```

```

BEGIN
    v_total_comenzi := calcul_valoare_totala_comenzi;
    dbms_output.put_line('Valoarea totală a tuturor comenzilor este: ' || v_total_comenzi);
END;

```



-- 4. Procedura procesare_clienti verifica daca datele clientilor sunt complete, daca nu se aplica exceptii cu afisarea unui mesaj, s-a utilizat un cursor pentru parcurgerea datelor din tabela info_clienti

```

CREATE OR REPLACE PROCEDURE procesare_clienti IS
    v_id_client info_clienti.ID%TYPE;
    v_nume_client info_clienti.nume%TYPE;
    v_adresa_client info_clienti.adresa%TYPE;
    v_telefon_client info_clienti.telefon%TYPE;
    v_email_client info_clienti.email%TYPE;
    client_invalid EXCEPTION;
    CURSOR C IS
        SELECT ID, nume, adresa, telefon, email
        FROM info_clienti;
BEGIN
    OPEN C;
    LOOP
        FETCH C INTO v_id_client, v_nume_client, v_adresa_client, v_telefon_client, v_email_client;
        EXIT WHEN C%notfound;

        BEGIN
            -- Verificarea datelor clientului
            IF TRIM(v_nume_client) IS NULL OR TRIM(v_adresa_client) IS NULL OR TRIM(v_telefon_client) IS
            NULL OR TRIM(v_email_client) IS NULL THEN
                RAISE client_invalid;
            END IF;

            -- Procesarea datelor clientului
            -- ...

            dbms_output.put_line('Client procesat: ' || v_nume_client);
        EXCEPTION
            WHEN client_invalid THEN
                dbms_output.put_line('Datele clientului cu id-ul ' || v_id_client || ' sunt incomplete. Client ignorat.');
```

```

END;

```

```

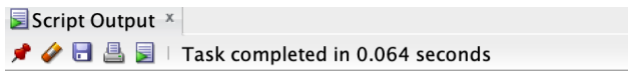
/

```

```

EXECUTE procesare_clienti;

```



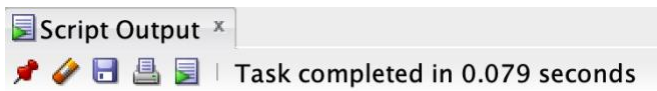
Procedure PROCESARE_CLIENTI compiled

Client procesat: Grigore Andrei
Datele clientului cu id-ul 6 sunt incomplete. Client ignorat.
Client procesat: Popescu Ion
Client procesat: Zaharia Maria
Client procesat: Vasile Tudor
Client procesat: Maftei Valeria

PL/SQL procedure successfully completed.

-- 5. Procedura verificare_stoc_produs verifica daca produsul este pe stoc, atunci afiseaza numarul bucatilor, daca este stoc epuizat se afiseaza mesajul exceptiei, iar daca nu exista produsul se afiseaza mesajul exceptiei no_data_found

```
CREATE OR REPLACE PROCEDURE verificare_stoc_produs (  
    p_id_produs IN INT  
) IS  
    v_stoc INT;  
    stoc_insuficient EXCEPTION;  
BEGIN  
    SELECT stoc INTO v_stoc  
    FROM info_produce  
    WHERE ID = p_id_produs;  
    IF v_stoc > 0 THEN  
        dbms_output.put_line('Stocul produsului este disponibil: ' || v_stoc);  
    ELSE  
        RAISE stoc_insuficient;  
  
    END IF;  
EXCEPTION  
    WHEN stoc_insuficient THEN  
        dbms_output.put_line('Stoc indisponibil');  
    WHEN no_data_found THEN  
        dbms_output.put_line('Produsul specificat nu există');  
END;  
/  
EXECUTE verificare_stoc_produs(631);  
  
EXECUTE verificare_stoc_produs(639);  
  
EXECUTE verificare_stoc_produs(636);
```



Procedure VERIFICARE_STOC_PRODUS compiled

Stocul produsului este disponibil: 61

PL/SQL procedure successfully completed.

Produsul specificat nu exista

PL/SQL procedure successfully completed.

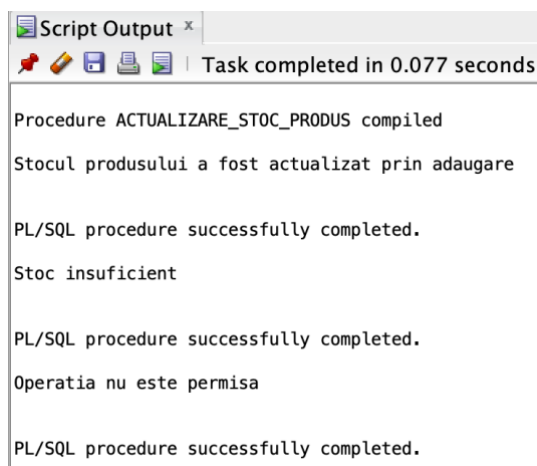
Stoc indisponibil

PL/SQL procedure successfully completed.

-- 6. Procedura actualizare_stoc_produș modifica stocul unui produs prin actualizarea numărului de bucăți, prin adăugare, scădere, dacă este posibilă, existent restricții la scădere, precum și o excepție în cazul introducerii unei operații invalide

```
CREATE OR REPLACE PROCEDURE actualizare_stoc_produș (  
  p_id_produș INT,  
  p_cantitate INT,  
  p_operatie VARCHAR2  
) IS  
  v_stoc INT;  
  stoc_insuficient EXCEPTION;  
  operatie_invalida EXCEPTION;  
BEGIN  
  SELECT stoc INTO v_stoc  
  FROM info_produșe  
  WHERE ID = p_id_produș;  
  IF p_operatie = 'ADUGARE' THEN  
    UPDATE info_produșe  
    SET stoc = stoc + p_cantitate  
    WHERE ID = p_id_produș;  
    dbms_output.put_line('Stocul produsului a fost actualizat prin adăugare');  
  ELSIF p_operatie = 'SCADERE' THEN  
    IF v_stoc >= p_cantitate THEN  
      UPDATE info_produșe  
      SET stoc = stoc - p_cantitate  
      WHERE ID = p_id_produș;  
      dbms_output.put_line('Stocul produsului a fost actualizat prin scădere');  
    ELSE  
      RAISE stoc_insuficient;  
    END IF;  
  ELSE  
    RAISE operatie_invalida;  
  END IF;  
  COMMIT;  
EXCEPTION  
  when stoc_insuficient then  
    dbms_output.put_line('Stoc insuficient');  
  when operatie_invalida THEN  
    dbms_output.put_line('Operatia nu este permisa');  
  WHEN no_data_found THEN  
    dbms_output.put_line('Produsul specificat nu există');  
END;
```

```
EXECUTE actualizare_stoc_produș(633, 11, 'ADUGARE');  
EXECUTE actualizare_stoc_produș(636, 5, 'SCADERE');  
EXECUTE actualizare_stoc_produș(634, 10, 'INMULTIRE');
```



```
Script Output x  
Task completed in 0.077 seconds  
  
Procedure ACTUALIZARE_STOC_PRODUS compiled  
Stocul produsului a fost actualizat prin adăugare  
  
PL/SQL procedure successfully completed.  
Stoc insuficient  
  
PL/SQL procedure successfully completed.  
Operatia nu este permisa  
  
PL/SQL procedure successfully completed.
```

-- 7. Pachetul info_client contine o functie care returneaza numarul total de client si o procedura care foloseste functia pentru a selecta toti clientii si a modifica adresa unui client

```
CREATE OR REPLACE PACKAGE info_client AS
    FUNCTION numar_total_clienti RETURN INT;
    PROCEDURE actualizare_adresa_client(p_id_client INT, p_adresa_noua VARCHAR2);
END info_client;
/

CREATE OR REPLACE PACKAGE BODY info_client AS
    FUNCTION numar_total_clienti RETURN INT IS
        v_numar_clienti INT;
    BEGIN
        SELECT COUNT(*) INTO v_numar_clienti
        FROM info_clienti;

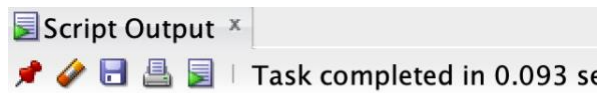
        RETURN v_numar_clienti;
    END numar_total_clienti;

    PROCEDURE actualizare_adresa_client(p_id_client INT, p_adresa_noua VARCHAR2) IS
    BEGIN
        UPDATE info_clienti
        SET adresa = p_adresa_noua
        WHERE ID = p_id_client;

        COMMIT;
    END actualizare_adresa_client;
END info_client;
/
```

```
DECLARE
    v_numar_clienti INT;
BEGIN
    SELECT info_client.numar_total_clienti INTO v_numar_clienti FROM dual;
    dbms_output.put_line('Numarul total de clienti: ' || v_numar_clienti);
END;
/

BEGIN
    info_client.actualizare_adresa_client(1, 'Prahova, Ploiesti, Str. Castanului, nr.5');
    dbms_output.put_line('Adresa clientului a fost actualizata.');
```



Package INFO_CLIENT compiled

Package Body INFO_CLIENT compiled

Numarul total de clienti: 6

PL/SQL procedure successfully completed.

Adresa clientului a fost actualizata.

PL/SQL procedure successfully completed.

-- 8. Pachetul comenzi_plasate este compus dintr-o functie care reține numele clientului și o procedura care plasează comenzi, procedura având excepții

```
CREATE OR REPLACE PACKAGE comenzi_plasate AS
  FUNCTION nume_client(p_id_client INT) RETURN VARCHAR2;
  PROCEDURE plasare_comanda(p_id_client INT, p_id_produș INT, p_cantitate INT);
  stoc_insuficient EXCEPTION;
END comenzi_plasate;
/
```

```
CREATE OR REPLACE PACKAGE BODY comenzi_plasate AS
  FUNCTION nume_client(p_id_client INT) RETURN VARCHAR2 IS
    v_nume VARCHAR2(255);
  BEGIN
    SELECT nume INTO v_nume
    FROM info_clienti
    WHERE ID = p_id_client;
    RETURN v_nume;
  END nume_client;

  PROCEDURE plasare_comanda(p_id_client INT, p_id_produș INT, p_cantitate INT) IS
    v_stoc INT;
    v_count INT := 0;
  BEGIN
    SELECT stoc INTO v_stoc
    FROM info_produșe
    WHERE ID = p_id_produș;

    IF v_stoc >= p_cantitate THEN
      v_stoc := v_stoc - p_cantitate;
      UPDATE info_produșe
      SET stoc = v_stoc
      WHERE ID = p_id_produș;
      SELECT COUNT(*)
      INTO v_count
      FROM info_comenzi
      WHERE id_client = p_id_client AND id_produș = p_id_produș;

      IF v_count > 0 THEN
        UPDATE info_comenzi
        SET DATA = sysdate
        WHERE id_client = p_id_client AND id_produș = p_id_produș;
      ELSE
        INSERT INTO info_comenzi (DATA, id_client, id_produș)
        VALUES (sysdate, p_id_client, p_id_produș);
      END IF;

      COMMIT;
      dbms_output.put_line('Comanda a fost plasata cu succes.');
```

```
    ELSE
      RAISE stoc_insuficient;
    END IF;
  EXCEPTION
    WHEN stoc_insuficient THEN
      dbms_output.put_line('Stoc insuficient pentru produsul cu ID-ul ' || p_id_produș);
    WHEN no_data_found THEN
      dbms_output.put_line('Nu exista un client sau un produs cu ID-urile specificate.');
```

```
  END plasare_comanda;
END comenzi_plasate;
/
```

Package COMENZI_PLASATE compiled

Package Body COMENZI_PLASATE compiled

DECLARE

v_num_client VARCHAR2(100);

BEGIN

v_num_client := comenzi_plasate.num_client(1);

dbms_output.put_line('Numele clientului cu ID-ul 1: ' || v_num_client);

END;

/

BEGIN

comenzi_plasate.plasare_comanda(4, 635, 5);

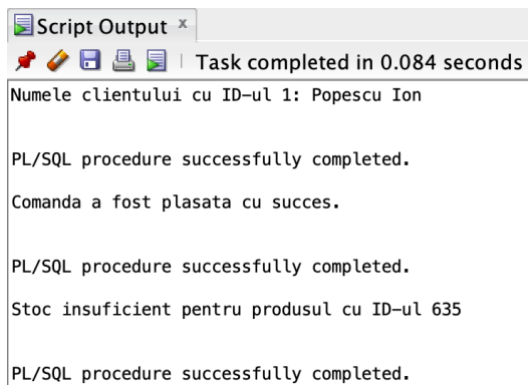
END;

/

BEGIN

comenzi_plasate.plasare_comanda(1, 635, 25);

END;



Script Output x

Task completed in 0.084 seconds

Numele clientului cu ID-ul 1: Popescu Ion

PL/SQL procedure successfully completed.

Comanda a fost plasata cu succes.

PL/SQL procedure successfully completed.

Stoc insuficient pentru produsul cu ID-ul 635

PL/SQL procedure successfully completed.

3 Declansatori:

-- 1. Declansatorul update_pret actualizeaza pretul prduselor din acel interval dupa fiecare insert in tabela info_produce.

CREATE OR REPLACE TRIGGER update_pret

BEFORE insert ON info_produce

FOR EACH ROW

DECLARE

v_pret_vechi info_produce.pret%TYPE;

v_pret_nou info_produce.pret%TYPE;

v_procent NUMBER(6, 2) := 10;

BEGIN

IF :OLD.pret >= 100 AND :OLD.pret <= 200 THEN

v_pret_vechi := :OLD.pret;

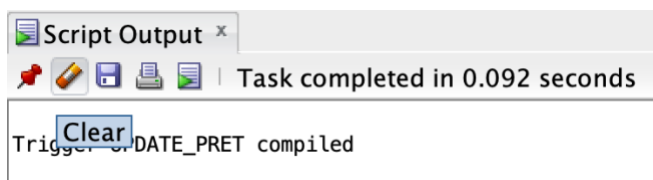
v_pret_nou := v_pret_vechi + (v_pret_vechi * v_procent / 100);

:NEW.pret := v_pret_nou;

END IF;

END;

/



Script Output x

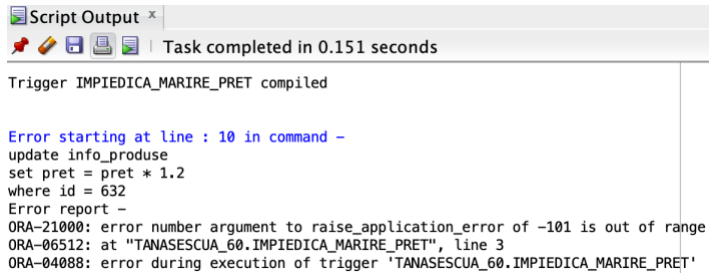
Task completed in 0.092 seconds

Clear

Trigger DATE_PRET compiled

-- 2. Declansatorul impiedica_marire_pret nu permite modificarea preturilor mai mari de 200

```
CREATE OR REPLACE TRIGGER impiedica_marire_pret
BEFORE UPDATE ON info_produce
FOR EACH ROW
BEGIN
    IF :NEW.pret > 200 THEN
        RAISE_APPLICATION_ERROR(-101, 'Nu se permite marirea prețului pentru produsele mai scumpe de 200!');
    END IF;
END;
```



Script Output x

Task completed in 0.151 seconds

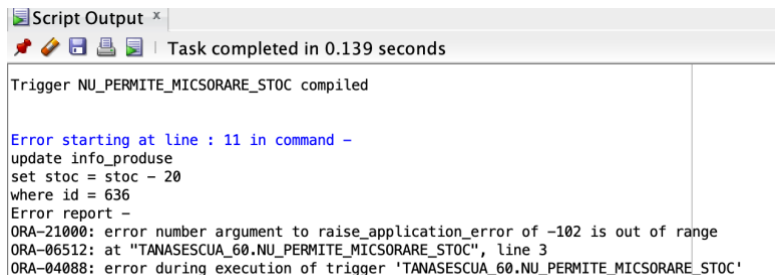
Trigger IMPIEDICA_MARIRE_PRET compiled

Error starting at line : 10 in command -

```
update info_produce
set pret = pret * 1.2
where id = 632
Error report -
ORA-21000: error number argument to raise_application_error of -101 is out of range
ORA-06512: at "TANASESCUA_60.IMPIEDICA_MARIRE_PRET", line 3
ORA-04088: error during execution of trigger 'TANASESCUA_60.IMPIEDICA_MARIRE_PRET'
```

-- 3. Declansatorul nu_permite_micsorare_stoc impiedica micsorarea stocului produselor a carui valoare este sub 10

```
CREATE OR REPLACE TRIGGER nu_permite_micsorare_stoc
BEFORE UPDATE ON info_produce
FOR EACH ROW
BEGIN
    IF :NEW.stoc < 10 AND :NEW.stoc < :OLD.stoc THEN
        RAISE_APPLICATION_ERROR(-102, 'Nu se permite micsorarea stocului pentru produsele cu cantitate sub 10!');
    END IF;
END;
```



Script Output x

Task completed in 0.139 seconds

Trigger NU_PERMITE_MICSORARE_STOC compiled

Error starting at line : 11 in command -

```
update info_produce
set stoc = stoc - 20
where id = 636
Error report -
ORA-21000: error number argument to raise_application_error of -102 is out of range
ORA-06512: at "TANASESCUA_60.NU_PERMITE_MICSORARE_STOC", line 3
ORA-04088: error during execution of trigger 'TANASESCUA_60.NU_PERMITE_MICSORARE_STOC'
```