

**ACADEMIA DE STUDII ECONOMICE BUCUREȘTI
FACULTATEA DE CIBERNETICĂ STATISTICĂ ȘI
INFORMATICĂ ECONOMICĂ**



**PROIECT
SGBD ORACLE**

**GESTIONAREA COMENZILOR ONLINE ȘI
LIVRĂRILOR LA DOMICILIU ÎNTR-UN RESTAURANT**

Titular disciplină :
Conferențiar univ. dr. Iuliana Botha

Realizat de:

Nume prenume	Tabuncic Valeria
Grupa	1065

București, 2024- 2025

OBIECTIVUL BAZEI DE DATE

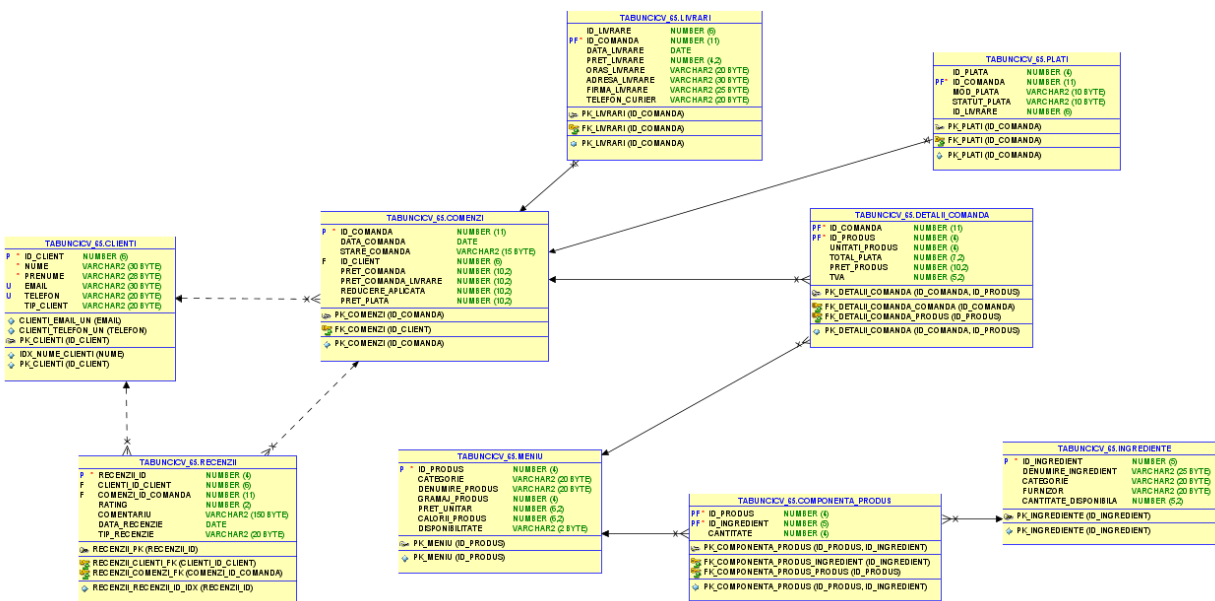
Perioada curentă se remarcă printr-o digitalizare continuă a economiei, motiv pentru care printre principalele sale manifestări se numără creșterea comerțului online și schimbarea comportamentului consumatorilor. Pornind de la premisa că livrarea exclusiv online reprezintă un model de afaceri în plină dezvoltare, având avantaje precum costuri de operare reduse, flexibilitate, adaptabilitate și eliminarea necesității unui spațiu fizic, am ales să aplic această abordare în cazul unui restaurant.

Astfel, în cadrul acestui proiect îmi propun realizarea unei baze de date relaționale care să faciliteze procesul de gestionare a comenzilor pentru un restaurant ce funcționează exclusiv în regim online. Scopul principal al acestei baze de date este automatizarea și respectiv eficientizarea proceselor cheie ce asigură funcționarea restaurantului, de la gestionarea meniului până la livrarea comenzilor și analiza feedback-ului oferit de clienții unității respective.

Baza de date vizează stocarea informațiilor necesare pentru parcurgerea următoarelor etape:

- ✓ Consultarea Meniului – Clienții au acces la o listă actualizată de feluri de mâncare, unde pot verifica disponibilitatea, componența (ingredientele) și prețurile unitare;
- ✓ Realizarea Comenzii – Clientul poate selecta produsele dorite din meniu, acestea fiind înregistrate împreună cu datele sale de contact în sistem;
- ✓ Achitarea Comenzii – Odată ce plata este procesată este declanșat procesul de pregătire și livrare a comenzii;
- ✓ Livrarea la Domiciliu – Comanda este livrată la domiciliul clientului, baza de date monitorizând statusul comenzii și respectiv eficiența procesului;
- ✓ Feedback -ul – Clientul are posibilitatea să ofere o recenzie pentru produsele și respectiv serviciile restaurantului.

SCHEMA CONCEPTUALĂ

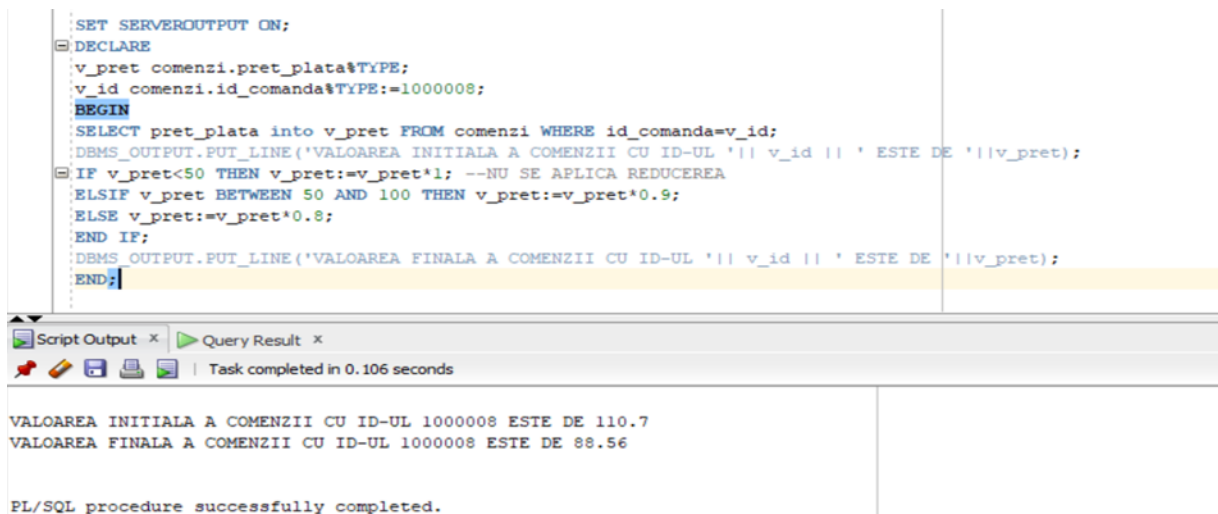


STRUCTURI DE CONTROL

1. În funcție de prețul de plată asociat comenzii cu ID-ul 1000008, se va afișa pe ecran valoarea modificată a acesteia, după aplicarea reducerilor. Reducerea se va aplica conform următoarelor reguli:

- Comenzile cu un preț sub 50 lei nu beneficiază de nicio reducere;
- Comenzile cu un preț cuprins între 50 și 100 lei beneficiază de o reducere de 10%;
- Comenzile cu un preț mai mare de 100 lei beneficiază de o reducere de 20%.

```
SET SERVEROUTPUT ON;
DECLARE
v_pret comenzi.pret_plata%TYPE;
v_id comenzi.id_comanda%TYPE:=1000008;
BEGIN
SELECT pret_plata into v_pret FROM comenzi WHERE id_comanda=v_id;
DBMS_OUTPUT.PUT_LINE('VALOAREA INITIALA A COMENZII CU ID-UL '|| v_id || ' ESTE DE '||v_pret);
IF v_pret<50 THEN v_pret:=v_pret*1; --NU SE APLICA REDUCEREA
ELSIF v_pret BETWEEN 50 AND 100 THEN v_pret:=v_pret*0.9;
ELSE v_pret:=v_pret*0.8;
END IF;
DBMS_OUTPUT.PUT_LINE('VALOAREA FINALA A COMENZII CU ID-UL '|| v_id || ' ESTE DE '||v_pret);
END;
/
```



```
SET SERVEROUTPUT ON;
DECLARE
v_pret comenzi.pret_plata%TYPE;
v_id comenzi.id_comanda%TYPE:=1000008;
BEGIN
SELECT pret_plata into v_pret FROM comenzi WHERE id_comanda=v_id;
DBMS_OUTPUT.PUT_LINE('VALOAREA INITIALA A COMENZII CU ID-UL '|| v_id || ' ESTE DE '||v_pret);
IF v_pret<50 THEN v_pret:=v_pret*1; --NU SE APLICA REDUCEREA
ELSIF v_pret BETWEEN 50 AND 100 THEN v_pret:=v_pret*0.9;
ELSE v_pret:=v_pret*0.8;
END IF;
DBMS_OUTPUT.PUT_LINE('VALOAREA FINALA A COMENZII CU ID-UL '|| v_id || ' ESTE DE '||v_pret);
END;
```

Script Output x Query Result x

Task completed in 0.106 seconds

VALOAREA INITIALA A COMENZII CU ID-UL 1000008 ESTE DE 110.7
VALOAREA FINALA A COMENZII CU ID-UL 1000008 ESTE DE 88.56

PL/SQL procedure successfully completed.

2. Să se realizeze o clasificare a produselor din meniul restaurantului în funcție de conținutul lor caloric. Având în vedere că numărul de calorii este specificat pentru 100 g de produs, se va calcula mai întâi numărul total de calorii per porție, utilizând gramajul specific fiecărui produs. Pe baza acestui rezultat, fiecare produs va fi încadrat într-o categorie calorică corespunzătoare. (TESTUL SE VA REALIZA PENTRU PRODUSUL CU ID-UL 101 CITIT DE LA TASTATURĂ.)

```

SET SERVEROUTPUT ON;
DECLARE
v_id meniu.id_produs%TYPE:=&i;
v_denumire meniu.denumire_produs%TYPE;
v_nrCalorii meniu.calorii_produs%TYPE;
v_gramaj meniu.gramaj_produs%TYPE;
v_caloriiPortie NUMBER;
v_categorie VARCHAR2(35);
BEGIN
SELECT denumire_produs,calorii_produs, gramaj_produs, (calorii_produs*gramaj_produs)/100
INTO v_denumire,v_nrCalorii,v_gramaj,v_caloriiPortie
FROM meniu
WHERE id_produs=v_id;
DBMS_OUTPUT.PUT_LINE ('Produsul '||v_denumire||' cu id-ul '||v_id||' are gramajul de '||v_gramaj|| ' si
un total de '||v_caloriiPortie||' calorii per portie. ');

CASE WHEN v_caloriiPortie<200 THEN v_categorie:='NR. CALORII SCAZUT';
      WHEN v_caloriiPortie BETWEEN 200 AND 700 THEN v_categorie:='NR. CALORII MODERAT';
      WHEN v_caloriiPortie BETWEEN 700 AND 1200 THEN v_categorie:='NR. CALORII RIDICAT';
      ELSE v_categorie:='NR. CALORII IN EXCES';
END CASE;
DBMS_OUTPUT.PUT_LINE( 'Produsul face parte din categoria produselor cu '||v_categorie);
END;
/

```

The screenshot displays the Oracle SQL Developer environment. The top pane shows a PL/SQL script that has been executed. The script declares variables for product details and calculates the calories per portion. It then uses a CASE statement to categorize the product based on its calories per portion. The bottom pane shows the output of the script, which includes the product name, its ID, weight, and the calculated calories per portion, followed by the category assignment.

```

SET SERVEROUTPUT ON;
DECLARE
v_id meniu.id_produs%TYPE:=&i;
v_denumire meniu.denumire_produs%TYPE;
v_nrCalorii meniu.calorii_produs%TYPE;
v_gramaj meniu.gramaj_produs%TYPE;
v_caloriiPortie NUMBER;
v_categorie VARCHAR2(35);
BEGIN
SELECT denumire_produs,calorii_produs, gramaj_produs, (calorii_produs*gramaj_produs)/100
INTO v_denumire,v_nrCalorii,v_gramaj,v_caloriiPortie
FROM meniu
WHERE id_produs=v_id;
DBMS_OUTPUT.PUT_LINE ('Produsul '||v_denumire||' cu id-ul '||v_id||' are gramajul de '||v_gramaj|| ' si un total de '||v_caloriiPortie||' calorii per portie. ');

CASE WHEN v_caloriiPortie<200 THEN v_categorie:='NR. CALORII SCAZUT';
      WHEN v_caloriiPortie BETWEEN 200 AND 700 THEN v_categorie:='NR. CALORII MODERAT';
      WHEN v_caloriiPortie BETWEEN 700 AND 1200 THEN v_categorie:='NR. CALORII RIDICAT';
      ELSE v_categorie:='NR. CALORII IN EXCES';
END CASE;
DBMS_OUTPUT.PUT_LINE( 'Produsul face parte din categoria produselor cu '||v_categorie);
END;
/

```

Script Output x Query Result x

Task completed in 4.361 seconds

Produsul Ciorbă de burtă cu id-ul 101 are gramajul de 400 si un total de 600 calorii per portie.
 Produsul face parte din categoria produselor cu NR. CALORII MODERAT

PL/SQL procedure successfully completed.

3. Într-un bloc PL/SQL să se parcurgă toți clienții care au înregistrat cel puțin 2 comenzi.

```

SET SERVEROUTPUT ON;
DECLARE
v_id comenzi.id_client%TYPE;

```

```

v_min_id comenzi.id_client%TYPE;
v_max_id v_min_id%TYPE;
v_nume clienti.nume%TYPE;
v_nrComenzi NUMBER;
BEGIN
SELECT min(cl.id_client),max(cl.id_client)
INTO v_min_id, v_max_id
FROM clienti cl, comenzi co
WHERE cl.id_client=co.id_client;
FOR i IN v_min_id..v_max_id LOOP
SELECT COUNT(co.id_comanda)
INTO v_nrComenzi
FROM comenzi co
WHERE co.id_client=i;
IF v_nrComenzi>=2 THEN SELECT cl.nume, cl.id_client INTO v_nume, v_id FROM clienti cl WHERE
cl.id_client=i;
DBMS_OUTPUT.PUT_LINE('Clientul '||v_nume||' cu id-ul '||v_id ||' a incheiat '||v_nrComenzi||'
comenzi. ');
END IF;
END LOOP;
END;
/

```

The screenshot shows a SQL Developer window with a worksheet containing a PL/SQL script. The script is designed to find clients who have placed at least 2 orders. It uses a loop to iterate through the range of client IDs, counting the number of orders for each. If a client has 2 or more orders, their name and ID are stored in variables, and a message is printed using DBMS_OUTPUT.PUT_LINE.

The output window at the bottom shows the results of the script execution, listing five clients and their respective order counts:

```

Clientul Tabuncic cu id-ul 1 a incheiat 2 comenzi.
Clientul Ceban cu id-ul 2 a incheiat 2 comenzi.
Clientul Georgescu cu id-ul 5 a incheiat 2 comenzi.
Clientul Babin cu id-ul 6 a incheiat 2 comenzi.
Clientul Raru cu id-ul 7 a incheiat 2 comenzi.

```

4. Să se afișeze toate comenzile încheiate până în anul 2025 și care au id-urile cuprinse între 1000001 și 1000020 .

```

SET SERVEROUTPUT ON;
DECLARE
v_id comenzi.id_comanda%TYPE:=1000001;
v_data comenzi.data_comanda%TYPE;
v NUMBER;

```

```

BEGIN
WHILE v_id<=1000020 LOOP
SELECT COUNT(id_comanda) INTO v FROM comenzi
WHERE id_comanda=v_id AND data_comanda<TO_DATE('2024-12-31','YYYY-MM-DD');
IF v>0 THEN
SELECT data_comanda INTO v_data FROM comenzi
WHERE id_comanda=v_id AND data_comanda<TO_DATE('2024-12-31','YYYY-MM-DD');
DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul '||v_id||' a fost inregistrata in data de '||v_data||'.');
ELSE
DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul '||v_id||' nu exista sau nu a fost inregistrata pana in anul
2025.');
```

```

END IF;
v_id:=v_id+1;
END LOOP;
END;
/

Comanda cu id-ul 1000001 a fost inregistrata in data de 25-DEC-24.
Comanda cu id-ul 1000002 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000003 a fost inregistrata in data de 28-DEC-24.
Comanda cu id-ul 1000004 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000005 a fost inregistrata in data de 30-DEC-24.
Comanda cu id-ul 1000006 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000007 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000008 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000009 a fost inregistrata in data de 30-DEC-24.
Comanda cu id-ul 1000010 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000011 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000012 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000013 a fost inregistrata in data de 25-DEC-24.
Comanda cu id-ul 1000014 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000015 a fost inregistrata in data de 30-DEC-24.
Comanda cu id-ul 1000016 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000017 a fost inregistrata in data de 29-DEC-24.
Comanda cu id-ul 1000018 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000019 nu exista sau nu a fost inregistrata pana in anul 2025.
Comanda cu id-ul 1000020 nu exista sau nu a fost inregistrata pana in anul 2025.

PL/SQL procedure successfully completed.
```

**5.Să se afișeze toate livrările care au drept oraș de livrare municipiul București.
(Blocuri imbricate => Ulterior exercițiul se va realiza utilizând cursori pentru eficiență.)**

```

SET SERVEROUTPUT ON;
DECLARE
v_min livrari.id_livrare%type;
v_max livrari.id_livrare%type;
v_id livrari.id_livrare%type;
v_oras livrari.oras_livrare%type;
BEGIN
select min(id_livrare), max(id_livrare)
into v_min,v_max
from livrari;

for i in v_min..v_max LOOP
begin
select oras_livrare, id_livrare
into v_oras,v_id
```

```

from livrari
where id_livrare=i;
if v_oras='București' then
dbms_output.put_line('Livrarea cu ID '||v_id|| ' are orasul de livrare '||v_oras);
end if;
exception
when no_data_found then null;
end;
end loop;
end;
/

```

```

Livrarea cu ID 912 are orasul de livrare București
Livrarea cu ID 916 are orasul de livrare București
Livrarea cu ID 920 are orasul de livrare București
Livrarea cu ID 924 are orasul de livrare București
Livrarea cu ID 928 are orasul de livrare București

PL/SQL procedure successfully completed.

```

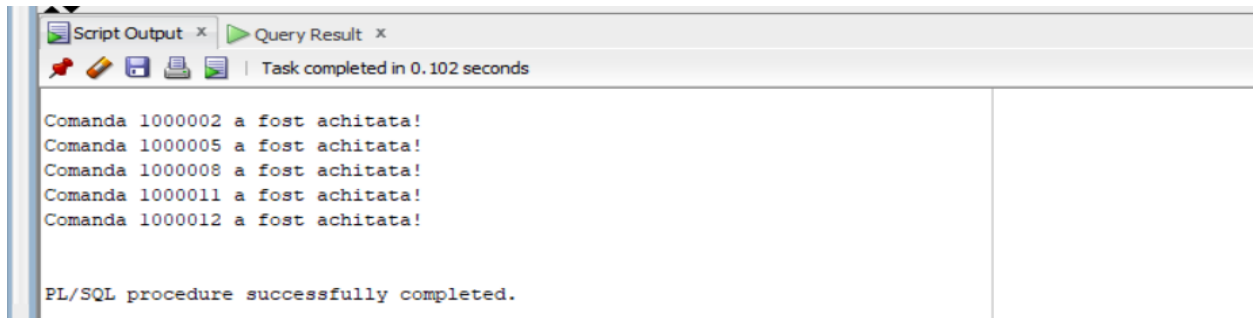
6. Să se implementeze un bloc PL/SQL care parcurge tabela Plati. Să se identifice și afișeze datele aferente primelor 5 comenzi cu statutul Achitat.

```

SET SERVEROUTPUT ON;
DECLARE
v_id_co plati.id_comanda%type;
v_id_plata plati.id_plata%type:=2000;
v_statut plati.statut_plata%type;
v number;
i number:=0;
BEGIN
LOOP
SELECT COUNT(id_plata) INTO v FROM plati
where id_plata=v_id_plata and statut_plata='Achitat';
IF v>0 THEN
select id_comanda, id_plata,statut_plata
into v_id_co,v_id_plata,v_statut
from plati
where id_plata=v_id_plata and statut_plata='Achitat';
dbms_output.put_line('Comanda ' ||v_id_co|| ' a fost achitata! ');
i:=i+1;
END IF;
v_id_plata:=v_id_plata+1;
EXIT WHEN i=5;

```

```
END LOOP;
END;
/
```



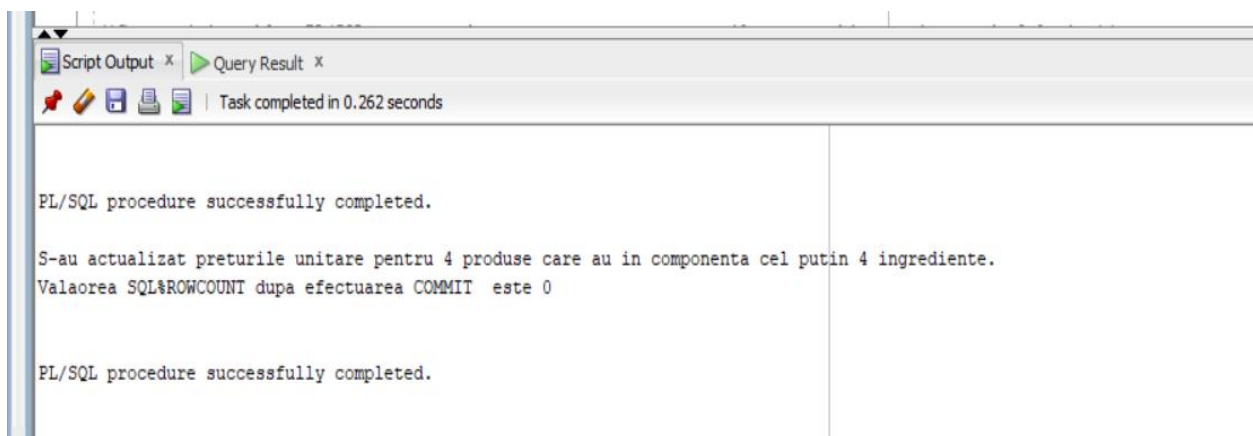
CURSURI

I.CURSURI IMPLICIȚI

1.Construiți un bloc PL/SQL care are drept scop actualizarea prețului unitar cu 10% pentru produsele care au în componență cel puțin 4 ingrediente.

```
SET SERVEROUTPUT ON;
DECLARE
BEGIN
UPDATE meniu
SET pret_unitar=pret_unitar*1.1
WHERE id_produc IN (SELECT id_produc
                     FROM componenta_produc
                     GROUP BY id_produc
                     HAVING COUNT(ID_INGREDIENT)>=4);
IF SQL%FOUND THEN
DBMS_OUTPUT.PUT_LINE('S-au actualizat preturile unitare pentru ||sql%rowcount|| produse care au
in componenta cel putin 4 ingrediente. ');
ELSE
DBMS_OUTPUT.PUT_LINE('Nu au fost gasite produse care sa respecte criteriile de actualizare. ');
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE('Valoarea SQL%ROWCOUNT dupa efectuarea COMMIT este
'||sql%rowcount);
END;
/
```

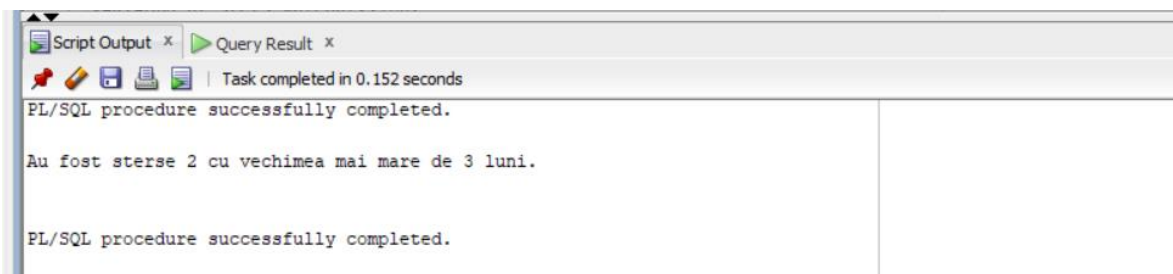
-- După o instrucțiune COMMIT (analog după ROLLBACK) , SQL%ROWCOUNT are valoarea 0.



2. Construiți un bloc PL/SQL care are drept scop ștergerea comenzilor cu vechimea mai mare de 3 luni.

```
SET SERVEROUTPUT ON;
DECLARE
BEGIN
DELETE FROM comenzi
WHERE DATA_COMANDA < ADD_MONTHS(SYSDATE,-3);
IF sql%found THEN
DBMS_OUTPUT.PUT_LINE('Au fost sterse '||sql%rowcount||' cu vechimea mai mare de 3 luni. ');
else
dbms_output.put_line('Nu au fost identificate comenzi cu vechimea mai mare de un an. ');
end if;
ROLLBACK;
END;
```

--add_months(data,nr_luni) – adaugă sau scade un anumit număr de luni



3. Să se șteargă din tabela Comenzi, comanda al cărei ID este introdus de utilizator prin intermediul variabilei de substituție g_id_comanda . Mesajul este afișat folosind variabila globală nr_sters.(--model seminar4 completari)

```
ACCEPT g_id_comanda PROMPT 'Introduceti id-ul comenzii:'
VARIABLE nr_sters varchar2(100)
DECLARE
BEGIN
DELETE FROM comenzi WHERE id_comanda = &g_id_comanda;
:nr_sters := TO_CHAR(SQL%ROWCOUNT) || ' comanda a fost stearsa. ';
END;
/
PRINT nr_sters
ROLLBACK;
```



II.CURSURI EXPLIȚIȚI

4. Construiți un bloc PL/SQL prin care să se afișeze informații despre comenzile care au drept oraș de livrare Cluj .

```
SET SERVEROUTPUT ON;
DECLARE
CURSOR c IS SELECT id_comanda, data_comanda, oras_livrare, adresa_livrare, pret_plata
              FROM livrari join comenzi using(id_comanda)
              WHERE oras_livrare='Cluj';
v c%rowtype;
BEGIN
OPEN c;
LOOP
FETCH c INTO v;
EXIT WHEN c%notfound;
DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul '||v.id_comanda||' a fost plasata la data de
'||v.data_comanda||'. Comanda este livrata in '||v.oras_livrare||' ,
la adresa '||v.adresa_livrare||' si are pretul final de '||v.pret_plata||' . ');
END LOOP;
CLOSE c;
END;
```

```
Comanda cu id-ul 1000002 a fost plasata la data de 03-JAN-25. Comanda este livrata in Cluj ,
la adresa Str. Libertății, Nr. 20 si are pretul final de 130.09 .
Comanda cu id-ul 1000006 a fost plasata la data de 02-JAN-25. Comanda este livrata in Cluj ,
la adresa Str. Eroilor, Nr. 22 si are pretul final de 69.73 .
Comanda cu id-ul 1000010 a fost plasata la data de 07-JAN-25. Comanda este livrata in Cluj ,
la adresa Str. Ferdinand, Nr. 19 si are pretul final de 43.03 .
Comanda cu id-ul 1000014 a fost plasata la data de 06-JAN-25. Comanda este livrata in Cluj ,
la adresa Str. Dorobanților, Nr. 3 si are pretul final de 105.16 .

PL/SQL procedure successfully completed.
```

5. Construiți un bloc PL/SQL prin care să se afișeze primii 5 clienți care au cele mai scumpe comenzi. (EXEMPLIFICAREA TUTUROR METODELOR)

--M1 - Cea mai riguroasă

```
-----
SET SERVEROUTPUT ON;
DECLARE
Cursor c IS SELECT nume,id_comanda,data_comanda, pret_plata
FROM comenzi JOIN clienti using(id_client)
ORDER BY pret_plata desc
FETCH FIRST 5 ROWS ONLY;
v_nume clienti.nume%TYPE;
v_id_com comenzi.id_comanda%TYPE;
```

```

v_data comenzi.data_comanda%TYPE;
v_pret comenzi.pret_plata%TYPE;
BEGIN
OPEN c;
LOOP
FETCH c INTO v_nume, v_id_com,v_data,v_pret;
EXIT WHEN c%notfound;
DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul '||v_id_com||' apartine clientului cu numele '||v_nume||'
si a fost plasata la data de '||v_data||
' avand pretul de '||v_pret||' lei.');
```

--M2 - Utilizam ROWTYPE

```

SET SERVEROUTPUT ON;
DECLARE
Cursor c IS SELECT nume,id_comanda,data_comanda, pret_plata
FROM comenzi JOIN clienti using(id_client)
ORDER BY pret_plata desc
FETCH FIRST 5 ROWS ONLY;
v c%rowtype;
BEGIN
OPEN c;
LOOP
FETCH c INTO v;
EXIT WHEN c%notfound;
DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul '||v.id_comanda||' apartine clientului cu numele
' ||v.nume||' si a fost plasata la data de '||v.data_comanda||
' avand pretul de '||v.pret_plata||' lei.');
```

--M3 - Bucla FOR

```

SET SERVEROUTPUT ON;
DECLARE
Cursor c IS SELECT nume,id_comanda,data_comanda, pret_plata
FROM comenzi JOIN clienti using(id_client)
ORDER BY pret_plata desc
FETCH FIRST 5 ROWS ONLY;
BEGIN
FOR v IN c LOOP
DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul '||v.id_comanda||' apartine clientului cu numele
' ||v.nume||' si a fost plasata la data de '||v.data_comanda||
' avand pretul de '||v.pret_plata||' lei.');
```

--M4 - cea mai simpla

```
SET SERVEROUTPUT ON;
BEGIN
FOR v IN ( SELECT nume,id_comanda,data_comanda, pret_plata
FROM comenzi JOIN clienti using(id_client)
ORDER BY pret_plata desc
FETCH FIRST 5 ROWS ONLY)
LOOP
DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul '||v.id_comanda||' apartine clientului cu numele
'||v.nume||' si a fost plasata la data de '||v.data_comanda||
' avand pretul de '||v.pret_plata||' lei. ');
END LOOP;
END;
```

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

```
Comanda cu id-ul 1000004 apartine clientului cu numele Ceban si a fost plasata la data de 05-JAN-25 avand pretul de 163.8 lei.
Comanda cu id-ul 1000002 apartine clientului cu numele Tabuncic si a fost plasata la data de 03-JAN-25 avand pretul de 130.09 lei.
Comanda cu id-ul 1000007 apartine clientului cu numele Georgescu si a fost plasata la data de 06-JAN-25 avand pretul de 120.15 lei.
Comanda cu id-ul 1000017 apartine clientului cu numele Enache si a fost plasata la data de 29-DEC-24 avand pretul de 116.1 lei.
Comanda cu id-ul 1000008 apartine clientului cu numele Georgescu si a fost plasata la data de 07-JAN-25 avand pretul de 110.7 lei.
```

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

6. Construiești un bloc PL/SQL care are drept scop afișarea valorii totale a comenzilor plasate de fiecare client.

```
SET SERVEROUTPUT ON;
DECLARE
CURSOR c IS SELECT cl.id_client, cl.nume, cl.prenume, nvl(sum(co.pret_plata),0) as total,
count(co.id_comanda) as nr_com
FROM clienti cl, comenzi co
WHERE cl.id_client =co.id_client(+) -- INCLUDEM INCLUSIV CLIENTII CARE NU AU COMENZI
GROUP BY cl.id_client,cl.nume,cl.prenume;
BEGIN
FOR v IN c LOOP
IF v.total>0 THEN
DBMS_OUTPUT.PUT_LINE('Clientul '||v.nume||' '||v.prenume||' a plasat '||v.nr_com||' comenzi in
valoare de '||v.total||' lei');
ELSE
DBMS_OUTPUT.PUT_LINE('Clientul '||v.nume||' '||v.prenume||' nu a plasat nici o comanda. ');
END IF;
END LOOP;
END;
```

```

Clientul Tabuncic Valeria a plasat 2 comenzi in valoare de 184.24 lei
Clientul Ceban Vitalina a plasat 2 comenzi in valoare de 247.87 lei
Clientul Vasilescu Ion a plasat 1 comenzi in valoare de 49.4 lei
Clientul Tanase Andrada a plasat 1 comenzi in valoare de 69.73 lei
Clientul Georgescu Alexandru a plasat 2 comenzi in valoare de 230.85 lei
Clientul Babin Loredana a plasat 2 comenzi in valoare de 148.33 lei
Clientul Raru Gabriel a plasat 2 comenzi in valoare de 167.47 lei
Clientul Craciun Simona a plasat 1 comenzi in valoare de 44.17 lei
Clientul Marinescu Elena a plasat 1 comenzi in valoare de 105.16 lei
Clientul Simon Cristina a plasat 1 comenzi in valoare de 82.65 lei
Clientul Caliman Alexandru a plasat 1 comenzi in valoare de 100.8 lei
Clientul Enache Constanta a plasat 1 comenzi in valoare de 116.1 lei
Clientul Raru Eugen nu a plasat nici o comanda.
Clientul Tiberiu Simona a plasat 1 comenzi in valoare de 90.9 lei
Clientul Lupascu Victor a plasat 1 comenzi in valoare de 95.85 lei

PL/SQL procedure successfully completed.

```

7. Construiți un bloc PL/SQL prin care să se afișeze pentru fiecare comanda (id_comanda) informații despre produsele incluse (id_produș, unitati_produș). Să se afișeze la nivelul fiecărei comenzi și nr total de unitati incluse.

```

SET SERVEROUTPUT ON;
DECLARE
CURSOR c is select id_comanda FROM detalii_comanda;
v_total_unitati NUMBER;
BEGIN
FOR v IN c LOOP
DBMS_OUTPUT.PUT_LINE('Comanda '||v.id_comanda);
v_total_unitati:=0;
FOR prod IN (SELECT id_produș,unitati_produș from detalii_comanda
              where id_comanda=v.id_comanda)
LOOP
dbms_output.put_line(prod.id_produș||' , '||prod.unitati_produș||' unitati');
v_total_unitati:=v_total_unitati+prod.unitati_produș;
END LOOP;
DBMS_OUTPUT.PUT_LINE('Total unitati comandate: '||v_total_unitati);
dbms_output.put_line(' ');
END LOOP;
END;
/

```

```

Total unitati comandate: 5

Comanda 1000017
101 , 1 unitati
108 , 4 unitati
Total unitati comandate: 5

Comanda 1000019
110 , 2 unitati
113 , 3 unitati
Total unitati comandate: 5

Comanda 1000019
110 , 2 unitati
113 , 3 unitati
Total unitati comandate: 5

Comanda 1000020
109 , 2 unitati
112 , 1 unitati
Total unitati comandate: 3

Comanda 1000020
109 , 2 unitati
112 , 1 unitati
Total unitati comandate: 3

PL/SQL procedure successfully completed.

```

8. Construiți un bloc PL/SQL care are drept scop stabilirea fezabilității unei comenzi în funcție de disponibilitatea produselor. Disponibilitatea fiecărui produs poate fi consultată în tabela Meniu , iar componenta comenzilor în Detalii_Comanda. ANTERIOR ÎN PROIECT AM VERIFICAT DISPONIBILITATEA FIECĂRUI PRODUS DIN MENU ÎN FUNCȚIE DE CANTITĂȚILE DE INGREDIENTE NECESARE ȘI DISPONIBILITATEA ACESTORA ÎN STOC.

```

SET SERVEROUTPUT ON;
DECLARE
CURSOR c IS SELECT distinct id_comanda FROM detalii_comanda;

```



```

v_disponibilitate meniu.disponibilitate%type;
v_com_fezabila BOOLEAN := TRUE;
BEGIN
FOR v IN c LOOP
v_com_fezabila := TRUE;
FOR prod IN (SELECT distinct id_produs FROM detalii_comanda WHERE id_comanda = v.id_comanda)
LOOP
SELECT disponibilitate INTO v_disponibilitate
FROM meniu
WHERE id_produs = prod.id_produs;
IF v_disponibilitate = 'LT' THEN v_com_fezabila := FALSE;
DBMS_OUTPUT.PUT_LINE('Produsul ' || prod.id_produs || ' nu este disponibil , astfel comanda cu id-ul
' || v.id_comanda || ' nu poate fi realizată. ');
EXIT;
END IF;
END LOOP;
END LOOP;
IF v_com_fezabila THEN
DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul ' || v.id_comanda || ' poate fi realizată!');
END IF;
END LOOP;
END;

```

```

PL/SQL procedure successfully completed.

Comanda cu id-ul 1000001 poate fi realizată!
Produsul 112 nu este disponibil , astfel comanda cu id-ul 1000002 nu poate fi realizată.
Produsul 109 nu este disponibil , astfel comanda cu id-ul 1000003 nu poate fi realizată.
Comanda cu id-ul 1000004 poate fi realizată!
Comanda cu id-ul 1000005 poate fi realizată!
Comanda cu id-ul 1000006 poate fi realizată!
Comanda cu id-ul 1000007 poate fi realizată!
Comanda cu id-ul 1000008 poate fi realizată!
Comanda cu id-ul 1000009 poate fi realizată!
Comanda cu id-ul 1000010 poate fi realizată!
Comanda cu id-ul 1000011 poate fi realizată!
Comanda cu id-ul 1000012 poate fi realizată!
Comanda cu id-ul 1000013 poate fi realizată!
Comanda cu id-ul 1000014 poate fi realizată!
Produsul 109 nu este disponibil , astfel comanda cu id-ul 1000015 nu poate fi realizată.
Produsul 112 nu este disponibil , astfel comanda cu id-ul 1000016 nu poate fi realizată.
Comanda cu id-ul 1000017 poate fi realizată!
Comanda cu id-ul 1000019 poate fi realizată!
Produsul 109 nu este disponibil , astfel comanda cu id-ul 1000020 nu poate fi realizată.

```

EXCEPȚII

1.Creați un bloc PL/SL prin care creșteți cu 20% prețul produselor al căror cod este citit de la tastatură. În cazul în care produsul cu acest id nu există se invocă o excepție.

```

SET SERVEROUTPUT ON;
DECLARE
v_id meniu.id_produs%type:=&a;
v_pr_init meniu.pret_unitar%type;
v_pr_final meniu.pret_unitar%type;

```

```

BEGIN
select pret_unitar into v_pr_init from meniu where id_produs=v_id;
dbms_output.put_line('Pret initial: '||v_pr_init||' lei');
update meniu
set pret_unitar=pret_unitar*1.2
where id_produs=v_id;
select pret_unitar into v_pr_final from meniu where id_produs=v_id;
dbms_output.put_line('Pretul a fost actualizat cu succes la '||v_pr_final||'lei');
EXCEPTION
when no_data_found then dbms_output.put_line('Produsul cu id-ul '||v_id||' nu a fost identificat !');
END;

```

```

END;
Pret initial: 27.23 lei
Pretul a fost actualizat cu succes la 32.68lei

PL/SQL procedure successfully completed.

```

```

END;
Produsul cu id-ul 106 nu a fost identificat !

PL/SQL procedure successfully completed.

```

2. Să se actualizeze cu 30% prețul produselor incluse într-o anumită categorie citită de la tastatură (ex: Aperitive). Se va ține cont de faptul că într-o categorie sunt incluse mai multe produse. Dacă categoria nu există, se va genera o excepție explicită.

```

SET SERVEROUTPUT ON;
DECLARE
v_categorie meniu.categorie%type;
v_id meniu.id_produs%type;
v_pr_init meniu.pret_unitar%type;
v_pr_final meniu.pret_unitar%type;

categorie_inexistenta Exception;

Cursor c is select id_produs,pret_unitar
              from meniu
              where categorie=v_categorie;
BEGIN
v_categorie := '&categorie';
dbms_output.put_line(v_categorie);

```



```

--verificam initial daca categoria exista
open c;
fetch c into v_id,v_pr_init;
if c%notfound then raise categorie_inexistenta;
end if;
close c;

--facem actualizarile necesare
for v in c loop
v_pr_init:=v.pret_unitar;
update meniu
set pret_unitar=pret_unitar*1.3
where id_produș=v.id_produș;
select pret_unitar into v_pr_final from meniu where id_produș=v.id_produș;
dbms_output.put_line('Pretul pentru produsul cu id-ul '||v.id_produș||' a fost actualizat de la '||v_pr_init||'
lei la '||v_pr_final||' lei. ');
end loop;
EXCEPTION
when categorie_inexistenta then dbms_output.put_line('Categoria solicitata nu exista. ');
when others then dbms_output.put_line('Eroare: '||sqlerrm);
END;

```

<pre> END; Aperitive Pretul pentru produsul cu id-ul 107 a fost actualizat de la 304.7 lei la 396.11 lei. Pretul pentru produsul cu id-ul 113 a fost actualizat de la 37.64 lei la 48.93 lei. Pretul pentru produsul cu id-ul 203 a fost actualizat de la 12.44 lei la 16.17 lei. PL/SQL procedure successfully completed. </pre>	
<pre> END; Ciocolata Categoria solicitata nu exista. PL/SQL procedure successfully completed. </pre>	

3. Într-un bloc PL/SQL, citiți de la tastatură ID-ul unui client. Să se afișeze datele clientului (nume, prenume) și numărul de comenzi plasate de acesta. Pentru clienții care au plasat comenzi, să se afișeze și valoarea totală a acestora.

- c1. Dacă clientul nu există, să se afișeze o excepție implicită corespunzătoare.**
- c2. Dacă clientul există și a plasat cel puțin o comandă, se afișează datele aferente.**
- c3. Dacă clientul există, dar nu a plasat nicio comandă, se va genera o excepție explicită.**
- c4. Tratați orice altă excepție cu o rutină de tratare corespunzătoare.**

```

SET SERVEROUTPUT ON;
DECLARE
v_id clienti.id_client%type:=&a;
v_numa clienti.numa%type;
v_prenume clienti.prenume%type;
v_nr_comenzi number:=0;
v_total number:=0;
fara_comenzi exception;

BEGIN
select nume, prenume into v_numa, v_prenume
from clienti
where id_client=v_id; --se declanseaza automat no_data_found daca clientul nu exista

select count(id_comanda), nvl(sum(pret_plata),0) into v_nr_comenzi, v_total
from comenzi
where id_client=v_id;
if v_nr_comenzi>0 then
dbms_output.put_line('Clientul '||v_numa||' '||v_prenume||' a plasat '||v_nr_comenzi||' comenzi in valoare de '||v_total||' lei.');
```

```

else
raise fara_comenzi;
end if;
EXCEPTION
when no_data_found then dbms_output.put_line('Clientul cu id-ul '||v_id||' nu exista.');
```

```

when fara_comenzi then dbms_output.put_line('Clientul cu id-ul '||v_id||' nu a plasat comenzi.');
```

```

when others then dbms_output.put_line(sqlerrm);
END;
```

```

END;
Clientul cu id-ul 16 nu exista.

PL/SQL procedure successfully completed.
```

```

END;
Clientul Tabuncic Valeria a plasat 2 comenzi in valoare de 184.24 lei.

PL/SQL procedure successfully completed.
```

```
END;  
Clientul cu id-ul 15 nu a plasat comenzi.  
  
PL/SQL procedure successfully completed.
```

4. Să se creeze un bloc PL/SQL care are drept scop identificarea lunilor în care s-a efectuat o singură comandă.

c1.Dacă nu au fost realizate comenzi, se va afișa excepția aferentă.

c2.Dacă au fost realizate între 2 și 20 comenzi, se va genera o excepție explicită cu mesajul „Numărul maxim de comenzi a fost atins”.

c3.Dacă s-au realizat mai mult de 20 de comenzi, se va genera excepția TOO_MANY_ROWS.

```
SET SERVEROUTPUT ON;  
DECLARE  
v_luna NUMBER := &a;  
v_nr NUMBER;  
v_id_com comenzi.id_comanda%type;  
v_data_com comenzi.data_comanda%type;  
maxim_comenzi EXCEPTION;  
  
BEGIN  
select count(id_comanda) into v_nr from comenzi  
where extract(month from data_comanda) = v_luna;  
  
if v_nr = 0 then raise no_data_found;  
elsif v_nr between 2 and 20 then raise maxim_comenzi;  
elsif v_nr > 20 then raise too_many_rows;  
else select id_comanda, data_comanda  
      into v_id_com, v_data_com  
      from comenzi  
      where extract(month from data_comanda) = v_luna and rownum = 1;  
  
dbms_output.put_line('In luna ' || v_luna || ' a existat o singura comanda.');
```

```
dbms_output.put_line('Comanda cu id-ul: ' || v_id_com || ' a fost comandata in data de ' ||  
TO_CHAR(v_data_com, 'DD-MM-YYYY'));  
end if;  
  
EXCEPTION  
when no_data_found then dbms_output.put_line('In luna ' || v_luna || ' nu s-au realizat comenzi.');
```

```
when maxim_comenzi then dbms_output.put_line('Numărul maxim de comenzi a fost atins in luna ' ||  
v_luna);  
when too_many_rows then dbms_output.put_line('In luna ' || v_luna || ' s-au realizat peste 20 comenzi.');
```

```
when others then dbms_output.put_line('Eroare: ' || sqlerrm);  
END;
```

```
END;  
In luna 3 a existat o singura comanda.  
Comanda cu id-ul: 1080 a fost comandata in data de 05-03-2025  
  
PL/SQL procedure successfully completed.
```

```
END;  
Numărul maxim de comenzi a fost atins in luna 1  
  
PL/SQL procedure successfully completed.
```

```
END;  
In luna 4 nu s-au realizat comenzi.  
  
PL/SQL procedure successfully completed.
```

5. Se va citi de la tastatură ID-ul unui produs și se va afișa denumirea acestuia.

Ulterior, se vor afișa toate produsele (denumirea) în compoziția cărora intră ingredientul respectiv, precum și comenzile în care apar aceste produse.

c1) Dacă ingredientul nu există, se va genera o excepție implicită.

c2) Dacă ingredientul există, dar nu intră în compoziția niciunui produs, se va genera o excepție explicită.

c3) Dacă ingredientul există, intră în compoziția unor produse, dar acestea nu au fost comandate, se va genera o excepție explicită.

c4) Tratați orice altă excepție cu o rutină de tratare corespunzătoare.

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
v_id_ing ingrediente.id_ingredient%type := &a;
```

```
v_den ingrediente.denumire_ingredient%type;
```

```
v_produs meniu.denumire_produs%type;
```

```
v_comanda detalii_comanda.id_comanda%type;
```

```
v_produs_comanda_exista BOOLEAN := FALSE;
```

```
fara_produse EXCEPTION;
```

```
fara_comenzi EXCEPTION;
```

```
--cursor pentru produsele care conțin ingredientul
```

```
cursor c1 is select distinct m.denumire_produs, m.id_produs
```

```

from meniu m, componenta_produs cp
where m.id_produs = cp.id_produs and cp.id_ingredient = v_id_ing;
--cursor pentru comenzile ce conțin produse ce au ingredientul respectiv
cursor c2 is select distinct dc.id_comanda
from detalii_comanda dc, componenta_produs cp
where dc.id_produs = cp.id_produs and cp.id_ingredient = v_id_ing;

BEGIN
select denumire_ingredient into v_den from ingrediente where id_ingredient = v_id_ing;
dbms_output.put_line('Ingredientul cu ID-ul ' || v_id_ing || ' (' || v_den || ') este asociat următoarelor
produse și comenzi:');
open c1;
fetch c1 into v_produs, v_comanda;
if c1%notfound then raise fara_produse;
end if;
loop
dbms_output.put_line('Produs: ' || v_produs);

open c2;
loop
fetch c2 into v_comanda;
exit when c2%notfound;
v_produs_comanda_exista := TRUE;
dbms_output.put_line(' - Comanda ID: ' || v_comanda);
end loop;
close c2;
if not v_produs_comanda_exista then raise fara_comenzi;
end if;
v_produs_comanda_exista := FALSE;
fetch c1 into v_produs, v_comanda;
exit when c1%notfound;
end loop;
close c1;

EXCEPTION
when no_data_found then dbms_output.put_line('Ingredientul cu id-ul ' || v_id_ing || ' nu exista.');
```

when fara_produse then dbms_output.put_line('Ingredientul cu id-ul ' || v_id_ing || ' nu apare in componenta niciunui produs.');

when fara_comenzi then dbms_output.put_line('Produsul in componenta caruia intra ingredientul cu id-ul ' || v_id_ing || ' nu a fost comandat.');

when others then dbms_output.put_line('Eroare: ' || sqlerrm);

END;

```

END;
Ingredientul cu id-ul 2987 nu exista.

PL/SQL procedure successfully completed.
```

<div> <div>Script Output x</div> <div>Query Result x</div> <div>Task completed in 5.282 seconds</div> </div> <div> <p>END;</p> <p>Ingredientul cu ID-ul 1003 (Carne tocată porc) este asociat următoarelor produse și comenzi:</p> <p>Produs: Platou țărănesc</p> <ul style="list-style-type: none"> - Comanda ID: 1000002 - Comanda ID: 1000003 - Comanda ID: 1000004 - Comanda ID: 1000008 - Comanda ID: 1000009 - Comanda ID: 1000012 - Comanda ID: 1000014 - Comanda ID: 1000015 - Comanda ID: 1000017 <p>Produs: Mici cu muștar</p> <ul style="list-style-type: none"> - Comanda ID: 1000002 - Comanda ID: 1000003 - Comanda ID: 1000004 - Comanda ID: 1000008 - Comanda ID: 1000009 - Comanda ID: 1000012 - Comanda ID: 1000014 - Comanda ID: 1000015 - Comanda ID: 1000017 <p>Produs: Friptură porc</p> <ul style="list-style-type: none"> - Comanda ID: 1000002 - Comanda ID: 1000003 - Comanda ID: 1000004 - Comanda ID: 1000008 - Comanda ID: 1000009 - Comanda ID: 1000012 - Comanda ID: 1000014 - Comanda ID: 1000015 </div>	
<p>Ingredientul cu ID-ul 205 (Ciuperci) este asociat următoarelor produse și comenzi:</p> <p>Produs: Ciuperci</p> <p>Produsul in componenta caruia intra ingredientul cu id-ul 205 nu a fost comandat.</p> <p>PL/SQL procedure successfully completed.</p>	
<p>Ingredientul cu ID-ul 205 (Ciuperci) este asociat următoarelor produse și comenzi:</p> <p>Produs: Ciuperci</p> <p>Produsul in componenta caruia intra ingredientul cu id-ul 205 nu a fost comandat.</p> <p>PL/SQL procedure successfully completed.</p>	

SUBPROGRAME: PROCEDURI & FUNCȚII

1. Sa se creeze o functie care verifica daca clientul cu un anumit id exista deja.

```
create or replace function client_exista(p_id in clienti.id_client%type) return number is
  v_nr NUMBER;
BEGIN
  select count(*) into v_nr
  from clienti
  where id_client=p_id;
  return v_nr;
END ;
/
```

2. Să se creeze o procedură care adaugă în tabela clienți o nouă înregistrare, iar ulterior modifică tipul clientului respectiv. Informatiile ce trebuie adaugate sunt furnizate drept parametri procedurii. Se trateaza cazul în care exista deja o functie cu codul introdus. SE VA UTILIZA FUNCȚIA CREATĂ ANTERIOR.

```
create or replace procedure adauga_client(p_id in clienti.id_client%type,
                                         p_nume in clienti.numa%type,
                                         p_prenume in clienti.prenume%type,
                                         p_email in clienti.email%type,
                                         p_telefon in clienti.telefon%type,
                                         p_tip in clienti.tip_client%type,
                                         p_tip_nou in clienti.tip_client%type)
is
  client_deja_existent exception;
  v_tip clienti.tip_client%type;
begin
  if (client_exista(p_id)=0)
  then insert into clienti(id_client,numa,prenume,email,telefon,tip_client)
  values (p_id,p_nume,p_prenume,p_email,p_telefon,p_tip);
  select tip_client into v_tip from clienti where id_client=p_id;
  dbms_output.put_line('Tip initial client: '||v_tip);
  else
  raise client_deja_existent;
end if;
  update clienti
  set tip_client=p_tip_nou
  where id_client=p_id;
  select tip_client into v_tip from clienti where id_client=p_id;
  dbms_output.put_line('Tip final client: '||v_tip);

exception
  when client_deja_existent then dbms_output.put_line('Clientul cu acest id deja exista!');
  when others then dbms_output.put_line('Eroare: '||sqlerrm);
end;
/
set serveroutput on;
begin
```

```

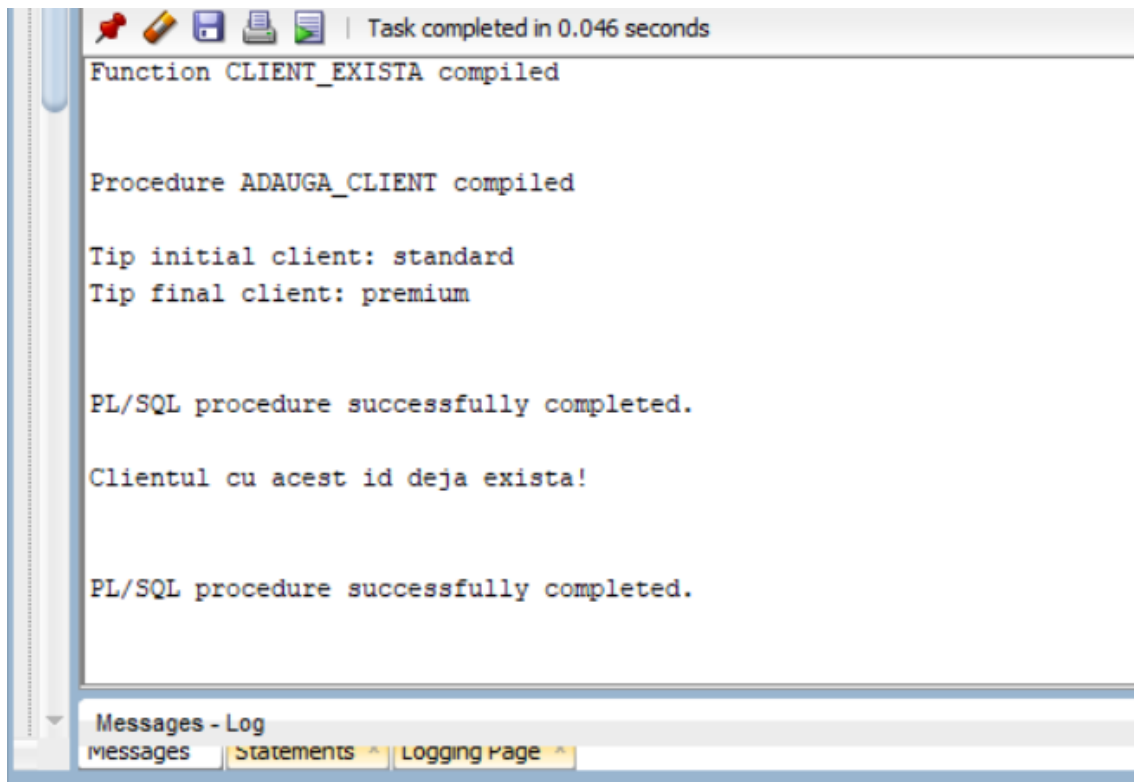
adauga_client(19921, 'Cojocaru','Elena','elena.coj@email.com','0727463876','standard','premium');
end;
/

```

```

--pentru a verifica dacă functia client_exista functioneaza
set serveroutput on;
begin
adauga_client(19921, 'Cojocaru','Elena','elena.coj@email.com','0727463876','standard','premium');
end;
/

```



3.Să se construiască o procedură care va actualiza starea comenzilor ca fiind "Finalizată" pentru toate comenzile plasate în anul 2024 și care au avut drept oras_livrare municipiul București.

```

SET SERVEROUTPUT ON;
create or replace procedure finalizeaza_comenzi_2024 as
cursor c is select id_comanda from comenzi join livrari using(id_comanda)
           where extract(year from data_comanda)=2024 and upper(oras_livrare)='BUCUREȘTI';

v_id comenzi.id_comanda%type;
i number:=0;

BEGIN

```



```

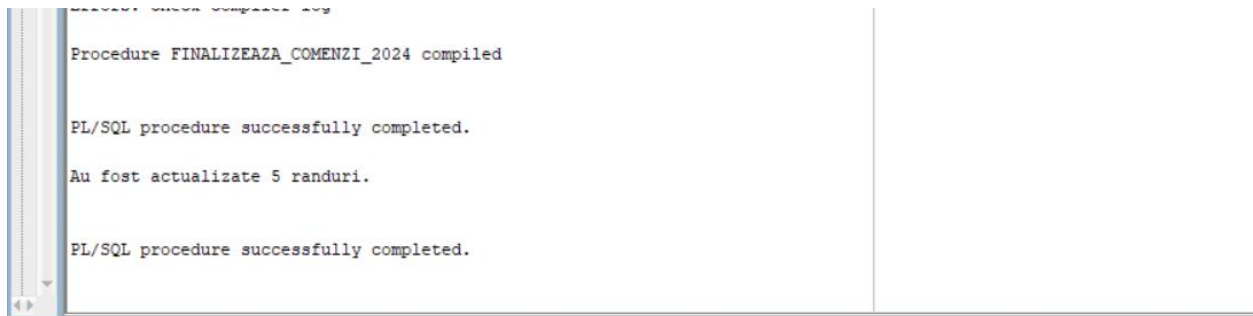
open c;
loop
fetch c into v_id;
exit when c%notfound;

update comenzi
set stare_comanda='Finalizată'
where id_comanda=v_id;
i:=i+1;
end loop;
close c;

if i=0 then raise no_data_found;
else
dbms_output.put_line('Au fost actualizate ' ||i||' randuri. ');
end if;

EXCEPTION
when no_data_found then dbms_output.put_line('Nu au fost plasate comenzi pentru București în 2024');
when others then dbms_output.put_line('Eroare: '||SQLERRM);
END;
/
execute finalizeaza_comenzi_2024;

```



4. Construiți procedura care să dubleze cantitatea disponibilă a ingredientelor incluse în categoria indicată ca parametru și o triplează pentru ingredientele aparținente altor categorii. Să se trateze cazul în care categoria indicată nu există și respectiv cel în care categoria indicată nu conține ingrediente. (exemple de categorii : Carne, Cereale, Legume)

```

CREATE OR REPLACE PROCEDURE dubleaza_tripleaza_cantitatea(p_categorie in
ingrediente.categorie%type)
AS
cursor c is select id_ingredient, categorie from ingrediente;
v_id_ingredient ingrediente.id_ingredient%type;
v_categorie ingrediente.categorie%type;
v_categ_exista BOOLEAN:=FALSE;
categorie_goala EXCEPTION;
categorie_inexistenta EXCEPTION;
BEGIN
open c;

```

```

loop
fetch c into v_id_ingredient,v_categorie;
exit when c%notfound;

if v_categorie is not null and LOWER(v_categorie)=LOWER(p_categorie) then
update ingrediente
set cantitate_disponibila=cantitate_disponibila*2
where id_ingredient=v_id_ingredient;
v_categ_exista:=TRUE;
elsif v_categorie is not null and LOWER(v_categorie)<>LOWER(p_categorie) then
update ingrediente
set cantitate_disponibila=cantitate_disponibila*3
where id_ingredient=v_id_ingredient;

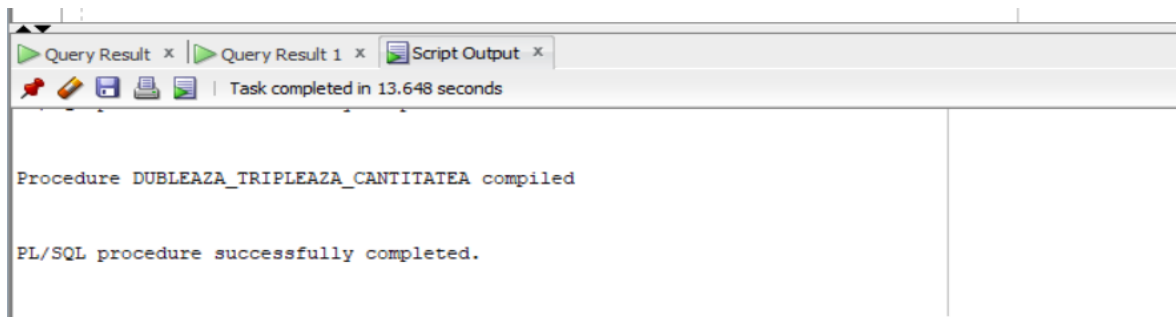
elsif v_categorie is null
then raise categorie_goala;
end if;
end loop;
close c;

if not v_categ_exista then raise categorie_inexistenta;
end if;
EXCEPTION
when categorie_goala then dbms_output.put_line('In categoria respectiva nu exista ingrediente');
when categorie_inexistenta then dbms_output.put_line('Numele categoriei a fost indicat gresit');
when others then dbms_output.put_line('Eroare: '||SQLERRM);
END;
/
execute dubleaza_tripleaza_cantitatea(&a);

```

INIȚIAL :

ID_INGREDIENT	DENUMIRE_INGREDIENT	CATEGORIE	FURNIZOR	CANTITATE_DISPONIBILA
1	1001 Burtă de vită	Carne	Furnizor Carnuri SRL	200
2	1002 Piept de pui	Carne	Avicola Pui SRL	200
3	1003 Carne tocată porc	Carne	Furnizor Carnuri SRL	200
4	1004 Smântână	Lactate	Fabrica Lactate SRL	200
5	1005 Brânză telemea	Lactate	Fabrica Lactate SRL	200
6	1006 Ouă	Lactate	Avicola Pui SRL	200
7	1007 Morcov	Legume	AgroLegume SRL	200
8	1008 Țelină	Legume	AgroLegume SRL	200
9	1009 Varză murată	Legume	Conserva SRL	200
10	1010 Orez	Cereale	Cereale SRL	200
11	1011 Porumb	Cereale	Cereale SRL	200
12	1012 Făină	Cereale	Cereale SRL	200
13	1013 Dulceață	Conserve	Conserva SRL	200
14	1014 Roșii	Legume	AgroLegume SRL	200
15	1015 Cartofi	Legume	AgroLegume SRL	200
16	1021 Ardei	Legume	AgroLegume SRL	200



DUPĂ APLICAREA PROCEDURII:

ID_INGREDIENT	DENUMIRE_INGREDIENT	CATEGORIE	FURNIZOR	CANTITATE_DISPONIBILA
1	1001 Burtă de vită	Carne	Furnizor Carnuri SRL	400
2	1002 Piept de pui	Carne	Avicola Pui SRL	400
3	1003 Carne tocată porc	Carne	Furnizor Carnuri SRL	400
4	1004 Smântână	Lactate	Fabrica Lactate SRL	600
5	1005 Brânză telemea	Lactate	Fabrica Lactate SRL	600
6	1006 Ouă	Lactate	Avicola Pui SRL	600
7	1007 Morcov	Legume	AgroLegume SRL	600
8	1008 Țelină	Legume	AgroLegume SRL	600
9	1009 Varză murată	Legume	Conserva SRL	600
10	1010 Orez	Cereale	Cereale SRL	600
11	1011 Porumb	Cereale	Cereale SRL	600
12	1012 Făină	Cereale	Cereale SRL	600
13	1013 Dulceață	Conserve	Conserva SRL	600
14	1014 Roșii	Legume	AgroLegume SRL	600
15	1015 Cartofi	Legume	AgroLegume SRL	600
16	1021 Ardei	Legume	AgroLegume SRL	600
17	1022 Ceapă	Legume	AgroLegume SRL	600

5. Să se construiască o procedură care va afișa informațiile despre cea mai recentă comandă plasată de un client dat(id_client este indicat ca parametru).

```

CREATE OR REPLACE PROCEDURE afiseaza_comanda(p_id_client IN clienti.id_client%type) IS
TYPE rec IS RECORD (id_comanda comenzi.id_comanda%type,
                     data_comanda comenzi.data_comanda%type,
                     stare_comanda comenzi.stare_comanda%type,
                     reducere_aplicata comenzi.reducere_aplicata%type,
                     pret_plata comenzi.pret_plata%type);

```

```

v_comanda rec;

```

```

BEGIN

```

```

select id_comanda,data_comanda,stare_comanda,reducere_aplicata,pret_plata
into v_comanda

```

```

from (select id_comanda,data_comanda,stare_comanda,reducere_aplicata,pret_plata
      from comenzi
      where id_client=p_id_client
      order by data_comanda DESC)
where rownum=1;

```

```

dbms_output.put_line('Cea mai recenta comanda a clientului cu id-ul '||p_id_client ||' are urmatorul id: '||

```

```
v_comanda.id_comanda||' .Comanda a fost plasata in data de '||TO_CHAR(v_comanda.data_comanda,'dd-mm-yyyy')|| ' si are statutul '||v_comanda.stare_comanda||' .S-a aplicat o reducere de '||v_comanda.reducere_aplicata||' lei si s-au achitat in final '||v_comanda.pret_plata||' lei.');
```

```
EXCEPTION
```

```
when no_data_found then dbms_output.put_line('Clientul cu acest id nu a plasat comenzi.');
```

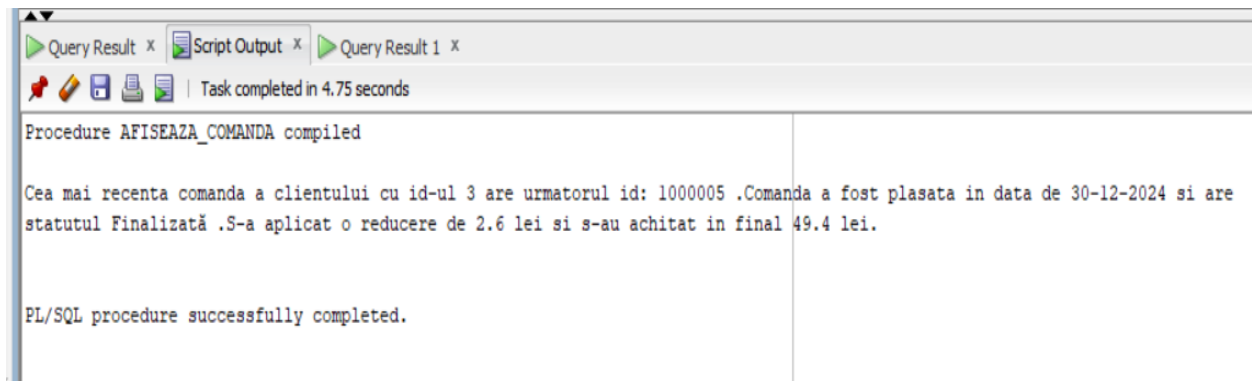
```
when others then dbms_output.put_line('Eroare: '||SQLERRM);
```

```
END;
```

```
/
```

```
SET SERVEROUTPUT ON;
```

```
execute afiseaza_comanda(&a);
```



6.Să se construiască funcția verifica_total care va calcula suma totală a comenzilor efectuate de un anumit client. Ulterior va returna TRUE dacă totalul depășește un anumit prag prestabilit și FALSE în caz contrar. Dacă clientul nu există sau nu are comenzi, funcția va returna NULL.

```
CREATE OR REPLACE FUNCTION verifica_total(p_id in clienti.id_client%type, p_prag in number)
```

```
RETURN BOOLEAN
```

```
is
```

```
v_total comenzi.pret_plata%type;
```

```
BEGIN
```

```
select sum(pret_plata) into v_total
```

```
from comenzi
```

```
where id_client=p_id;
```

```
if v_total>p_prag then
```

```
return true;
```

```
else
```

```
return false;
```

```
end if;
```

```
EXCEPTION
```

```
when no_data_found then return null;
```

```
end;
```

```
/
```

```
--apelul functiei dintr-un bloc anonim
```

```
set serveroutput on;
```

```

declare
v_prag NUMBER:=200;
begin
--apel1
if verifica_total(1,v_prag) is null then dbms_output.put_line('Client cu acest id nu exista sau nu a plasat comenzi!');
elsif verifica_total(1,v_prag) then dbms_output.put_line('Valoarea comenzilor acestui client a depasit pragul');
else dbms_output.put_line('Valoarea comenzilor este sub prag!');
end if;
--apel2
if verifica_total(5,v_prag) is null then dbms_output.put_line('Client cu acest id nu exista sau nu a plasat comenzi!');
elsif verifica_total(5,v_prag) then dbms_output.put_line('Valoarea comenzilor acestui client a depasit pragul');
else dbms_output.put_line('Valoarea comenzilor este sub prag!');
end if;
end;

```

```

Function VERIFICA_TOTAL compiled

Valoarea comenzilor este sub prag!
Valoarea comenzilor acestui client a depasit pragul

PL/SQL procedure successfully completed.

```

7. Să se construiască funcția calculeaza_durata_livrării care parcurge toate comenzile unui client și întoarce durata medie în zile dintre data plasării comenzii și data livrării. Dacă clientul nu există sau nu are comenzi livrate, funcția returnează NULL

```

create or replace function calculeaza_durata_livrării(p_id in clienti.id_client%type)
return number is
cursor c is select data_comanda,data_livrare
from comenzi join livrari using(id_comanda)
where id_client=p_id;

v_zile NUMBER:=0;
v_nr_comenzi NUMBER:=0;
BEGIN
for v in c loop
    if v.data_livrare is not null and v.data_comanda is not null then
        v_zile:=v_zile+(v.data_livrare-v.data_comanda);
        v_nr_comenzi:=v_nr_comenzi+1;
    end if;
end loop;

if v_nr_comenzi=0 then return null;

```

```

else return v_zile/v_nr_comenzi;
end if;
EXCEPTION
when others then return null;
end;
/
SET SERVEROUTPUT ON;
DECLARE
v_medie NUMBER;
BEGIN
v_medie:=calculeaza_durata_livrarii(1);
if v_medie is null then
dbms_output.put_line('Clientul nu a inregistrat comenzi. ');
else
dbms_output.put_line('Durata medie a livrarilor a fost de '||v_medie||' zile. ');
end if;

END;
/

```

```
Function CALCULEAZA_DURATA_LIVRARII compiled
```

```
Durata medie a livrarilor a fost de 1 zile.
```

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

8.Să se construiască o funcție care returnează numărul de apariții ale unui ingredient în componența produselor comandate.

```

create or replace function calculeaza_nr_aparitii_ingredient(p_id ingrediente.id_ingredient%type)
return number is
cursor c is select distinct dc.id_comanda
            from detalii_comanda dc, componenta_produc cp
            where dc.id_produc=cp.id_produc(+)
            and cp.id_ingredient=p_id;

i number:=0;
ingredient_neutilizat exception;
produc_necomandat exception;
BEGIN
select count(*) into i
from componenta_produc

```

```
where id_ingredient=p_id;
```

```
if i=0 then raise ingredient_neutilizat;  
end if;
```

```
for v in c loop  
i:=i+1;  
end loop;  
if i=0 then raise produs_necomandat;  
end if;  
return i;
```

```
EXCEPTION
```

```
when no_data_found then dbms_output.put_line('Ingredientul cu acest id nu exista!');  
when ingredient_neutilizat then dbms_output.put_line('Ingredienul nu intra in componenta nici unui  
produs.');
```

```
return null;  
when produs_necomandat then dbms_output.put_line('Ingredientul se regaseste in anumite produse, dar  
acestea nu au fost comandate.');
```

```
return null;  
when others then dbms_output.put_line('Eroare:'||sqlerrm);  
end;  
/
```

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
v_numar_comenzi number;
```

```
BEGIN
```

```
v_numar_comenzi := calculeaza_nr_aparitii_ingredient(1007);
```

```
if v_numar_comenzi is null then
```

```
dbms_output.put_line('Eroare la calcularea numărului de comenzi.');
```

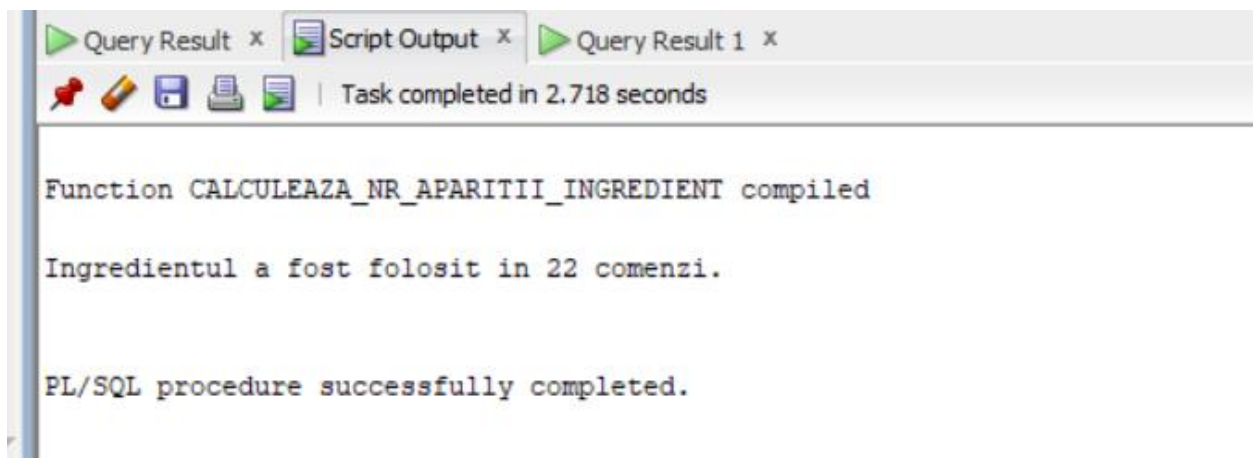
```
else
```

```
dbms_output.put_line('Ingredientul a fost folosit in ' || v_numar_comenzi || ' comenzi.');
```

```
end if;
```

```
END;
```

```
/
```



PACHETE DE SUBPROGRAME

1. Construiți un pachet care să conțină:

- o funcție care calculează prețul mediu al produselor existente în meniu.
- o funcție care returnează numărul produselor cu prețul sub medie (utilizând funcția precedentă).
- o procedură care afișează toate produsele dintr-o categorie dată drept parametru și care au prețul peste cel mediu.
- o procedură care șterge toate produsele cu preț sub o valoare dată. ROLLBACK ulterior.

Să se apeleze subprogramele din cadrul pachetului.

```
-----
create or replace package pac1 is
    function calculeaza_pret_mediu
    return number;
    function nr_produce_sub_medie
    return number;
    procedure afiseaza_produce_peste_medie(p_categorie in meniu.categorie%type);
    procedure sterge_produce_sub_medie;
end;
/
create or replace package body pac1 is
--1
    function calculeaza_pret_mediu return number
    is
        v_pret_mediu number;
    begin
        select avg(pret_unitar) into v_pret_mediu from meniu;
        return v_pret_mediu;
    exception
        when others then return null;
    end;
--2
    function nr_produce_sub_medie return number
    is
        v_pret_m number;
        v_nr_prod number;
    begin
        v_pret_m:=calculeaza_pret_mediu;
        select count(*) into v_nr_prod
            from meniu
            where pret_unitar<v_pret_m;
        return v_nr_prod;
    exception
        when others then return null;
    end;
--3
    procedure afiseaza_produce_peste_medie(p_categorie in meniu.categorie%type)
    is
```



```

v_pret_m number;
x number;
categorie_inexistenta exception;
begin
v_pret_m:=calculeaza_pret_meniu;

select count(*) into x
from meniu
where upper(categorie)=upper(p_categorie);

if x=0 then raise categorie_inexistenta;
end if;

for rec in (select denumire_produus, pret_unitar, categorie
            from meniu
            where upper(categorie)=upper(p_categorie) and pret_unitar>v_pret_m)
loop
dbms_output.put_line('Produsul : '||rec.denumire_produus||' Pret unitar: '||rec.pret_unitar||' Categoria:
'||rec.categorie);
end loop;
exception
when categorie_inexistenta then dbms_output.put_line('Categoria '||p_categorie||' nu exista!');
end;
--4.
procedure sterge_produce_sub_medie
is
v_pret_m number;
begin
v_pret_m:=calculeaza_pret_meniu;

delete from meniu where pret_unitar<v_pret_m;
dbms_output.put_line(sql%rowcount||' produse au fost sterse. ');
rollback;
end;
end;
/
set serveroutput on;
declare
pret_meniu number;
sub_medie number;
begin
pret_meniu:=pac1.calculeaza_pret_meniu;
dbms_output.put_line('Pretul mediu al produselor: '||pret_meniu);

sub_medie:=pac1.nr_produce_sub_medie;
dbms_output.put_line('Numarul produselor sub medie: '||sub_medie);

pac1.afiseaza_produce_peste_medie('Aperitive');
pac1.sterge_produce_sub_medie;
end;
/

```

```
Package PAC1 compiled

Package Body PAC1 compiled

Pretul mediu al produselor: 39.731333333333333333333333333333333333
Numarul produselor sub medie: 13
Produsul : Platou țărănesc Pret unitar: 304.7 Categoria: Aperitive
13 produse au fost sterse.

PL/SQL procedure successfully completed.
```

[illegible]

```
Package PAC1 compiled

Package Body PAC1 compiled

Pretul mediu al produselor: 39.7313333333333333333333333333333333333
Numarul produselor sub medie: 13
Produsul : Platou țărănesc Pret unitar: 304.7 Categoria: Aperitive
13 produse au fost sterse.

PL/SQL procedure successfully completed.
```

[illegible]

```
Package PAC1 compiled

Package Body PAC1 compiled

Pretul mediu al produselor: 39.73133333333333333333333333333333333333
Numarul produselor sub medie: 13
Produsul : Platou țărănesc Pret unitar: 304.7 Categoria: Aperitive
13 produse au fost sterse.

PL/SQL procedure successfully completed.
```

```
Package PAC1 compiled

Package Body PAC1 compiled

Pretul mediu al produselor: 39.731333333333333333333333333333333333
Numarul produselor sub medie: 13
Produsul : Platou țărănesc Pret unitar: 304.7 Categoria: Aperitive
13 produse au fost sterse.

PL/SQL procedure successfully completed.
```

```
Package PAC1 compiled

Package Body PAC1 compiled

Pretul mediu al produselor: 39.731333333333333333333333333333333333
Numarul produselor sub medie: 13
Produsul : Platou țărănesc Pret unitar: 304.7 Categoria: Aperitive
13 produse au fost sterse.

PL/SQL procedure successfully completed.
```

2. Construiți un pachet care să conțină:

- funcție care returnează valoarea totală a comenzilor unui client. (Un client poate avea una sau mai multe comenzi).
- procedura care returnează cei mai activi n clienți după valoarea totală a comenzilor si respectiv cei mai puțin activi n clienti.
- procedură care actualizează tipul clientului (PREMIUM/ STANDARD / INCEPATOR) în funcție de valoarea comenzilor. (actualizăm tip_client pentru toți clienții)
- procedură care șterge clienții fără comenzi. În caz contrar se ridică o excepție. (în funcție de id)
- procedură care aplică o reducere particularizată în funcție de valoarea comenzilor .(în funcție de id)

Să se apeleze subprogramele din cadrul pachetului.

- ## 2. Construiți un pachet care să conțină:
- funcție care returnează valoarea totală a comenzilor unui client. (Un client poate avea una sau mai multe comenzi).
 - procedura care returnează cei mai activi n clienți după valoarea totală a comenzilor si respectiv cei mai puțin activi n clienti.
 - procedură care actualizează tipul clientului (PREMIUM/ STANDARD / INCEPATOR) în funcție de valoarea comenzilor. (actualizăm tip_client pentru toți clienții)
 - procedură care șterge clienții fără comenzi. În caz contrar se ridică o excepție. (în funcție de id)
 - procedură care aplică o reducere particularizată în funcție de valoarea comenzilor .(în funcție de id)
- Să se apeleze subprogramele din cadrul pachetului.

2. Construiți un pachet care să conțină:

- funcție care returnează valoarea totală a comenzilor unui client. (Un client poate avea una sau mai multe comenzi).
- procedura care returnează cei mai activi n clienți după valoarea totală a comenzilor si respectiv cei mai puțin activi n clienti.
- procedură care actualizează tipul clientului (PREMIUM/ STANDARD / INCEPATOR) în funcție de valoarea comenzilor. (actualizăm tip_client pentru toți clienții)
- procedură care șterge clienții fără comenzi. În caz contrar se ridică o excepție. (în funcție de id)
- procedură care aplică o reducere particularizată în funcție de valoarea comenzilor .(în funcție de id)

Să se apeleze subprogramele din cadrul pachetului.

2. Construiți un pachet care să conțină:

- funcție care returnează valoarea totală a comenzilor unui client. (Un client poate avea una sau mai multe comenzi).
- procedura care returnează cei mai activi n clienți după valoarea totală a comenzilor si respectiv cei mai puțin activi n clienti.
- procedură care actualizează tipul clientului (PREMIUM/ STANDARD / INCEPATOR) în funcție de valoarea comenzilor. (actualizăm tip_client pentru toți clienții)
- procedură care șterge clienții fără comenzi. În caz contrar se ridică o excepție. (în funcție de id)
- procedură care aplică o reducere particularizată în funcție de valoarea comenzilor .(în funcție de id)

Să se apeleze subprogramele din cadrul pachetului.

```

client_inexistent exception;
begin
select count(*) into x
from clienti
where id_client=p_id;

if x=0 then raise client_inexistent;
end if;

select nvl(sum(pret_produs*unitati_produs),0) into v_totala
      from detalii_comanda join comenzi using(id_comanda)
      where id_client=p_id;
return v_totala;
end;
--2.1
procedure cei_mai_activi(n in number)
as
cursor c is select id_client, nume, prenume, valoarea_totala(id_client) as val
            from clienti
            order by val desc
            fetch first n rows only;
type rec is record( id_client clienti.id_client%type,
                    nume clienti.nume%type,
                    prenume clienti.prenume%type,
                    val number);

v rec;

begin
open c;
loop
fetch c into v;
exit when c%notfound;
dbms_output.put_line(v.id_client||' '||v.nume||' '||v.prenume||' Valoarea comenzilor: '||v.val);
end loop;
close c;
end;

--2.2.
procedure cei_mai_putin_activi(n in number)
as
begin
for v in ( select id_client, nume, prenume, valoarea_totala(id_client) as val
          from clienti
          order by val
          fetch first n rows only )
loop
dbms_output.put_line(v.id_client||' '||v.nume||' '||v.prenume||' Valoarea comenzilor: '||v.val);
end loop;
end;

--3.

```

```

procedure actualizare_tip_client
is
v_id clienti.id_client%type;
v_total number;
v_tip clienti.tip_client%type;

cursor c is select id_client from clienti;
begin
open c;
loop
fetch c into v_id;
exit when c%notfound;
v_total:=valoarea_totala(v_id);

if v_total>500 then v_tip:='Premium';
elsif v_total>100 then v_tip:='Standard';
else v_tip:='Inceptor';
end if;

update clienti
set tip_client=v_tip
where id_client=v_id;
end loop;
close c;
commit;
end;

```

--4.

```

procedure stergere_client_fara_comenzi(p_id in clienti.id_client%type)
is
nr_com number;
client_are_com exception;
begin
select count(*) into nr_com from comenzi where id_client=p_id;
if nr_com>0 then raise client_are_com;
else
delete from clienti where id_client=p_id;
dbms_output.put_line('Clientul cu id-ul '||p_id||' a fost sters.');
end if;
exception
when client_are_com then dbms_output.put_line('Clientul cu acest id are comenzi');
end;

```

--5.

```

procedure reducere(p_id in number)
is
v_total number;
v_reducere number;

begin
v_total:=valoarea_totala(p_id);

```

```

    if v_total<100 then v_reducere:=0;
    elsif v_total<200 then v_reducere:=10;
    else v_reducere:=15;
    end if;
    dbms_output.put_line('Clientul beneficiaza de o reducere de '||v_reducere ||' %');
    end;

end;
/

```

```

set serveroutput on;
declare
v_total number;
begin
v_total:=pac2.valoarea_totala(10);
dbms_output.put_line('Total comenzi: '||v_total);
dbms_output.put_line('Top 3 clienti activi: ');
pac2.cei_mai_activi(3);
dbms_output.put_line('Top 3 clienti inactivi: ');
pac2.cei_mai_putin_activi(2);
pac2.actualizare_tip_client;
for v in (select * from clienti)
loop
dbms_output.put_line(v.id_client||' '||v.num||' '||v.preume||' '||v.tip_client);
end loop;
pac2.stergere_client_fara_comenzi(19921);
pac2.reducere(1);
end;
/

```

```
Task completed in 0.063 seconds
Package PAC2 compiled

Package Body PAC2 compiled

Total comenzi: 120
Top 3 clienti activi:
2 Ceban Vitalina Valoarea comenzilor: 764.5
5 Georgescu Alexandru Valoarea comenzilor: 729
1 Tabuncic Valeria Valoarea comenzilor: 396.5
Top 3 clienti inactivi:
13 Raru Eugen Valoarea comenzilor: 0
8 Craciun Simona Valoarea comenzilor: 27.5
1 Tabuncic Valeria Standard
2 Ceban Vitalina Premium
3 Vasilescu Ion Incepator
4 Tanase Andrada Standard
5 Georgescu Alexandru Premium
6 Babin Loredana Standard
7 Raru Gabriel Standard
8 Craciun Simona Incepator
9 Marinescu Elena Standard
10 Simon Cristina Standard
11 Caliman Alexandru Standard
12 Enache Constanta Standard
13 Raru Eugen Incepator
14 Tiberiu Simona Standard
15 Lupascu Victor Standard
Clientul cu id-ul 19921 a fost sters.
Clientul beneficiaza de o reducere de 15 %
```

3. Construiți un pachet de subprograme care să conțină:

- funcție care verifică dacă produsul cu un anumit id există în meniu;
- procedură care ne returnează numărul de ingrediente incluse în componenta produselor din meniu;
- procedură care ne afișează cantitatea disponibilă din fiecare ingredient necesar pentru un anumit produs din meniu;
- procedura care mărește cantitatea disponibilă a ingredientelor care au un anumit furnizor;

Să se apeleze subprogramele din cadrul pachetului;

```
-----
create or replace package pac3 is
function produs_exista(p_id in meniu.id_produs%type)
return boolean;
procedure numar_ingredientes;
procedure afisare_cantitati(p_id in meniu.id_produs%type);
procedure marire_cantitate(p_furnizor in ingrediente.furnizor%type, p_adaos in number);
```

```

end;
/
create or replace package body pac3 is
--1.
function produs_exista(p_id in meniu.id_produs%type)
return boolean
is
x number;
begin
select count(*) into x
from meniu
where id_produs=p_id;

if x>0 then return true;
else return false;
end if;
exception
when no_data_found then return false;
end;

--2.
procedure numar_ingrediente
is
cursor c is select id_produs,denumire_produs
            from meniu;

v c%rowtype;
v_nr_ing number;
begin
for v in c
loop
select count(*) into v_nr_ing
      from componenta_produs
      where id_produs=v.id_produs;
dbms_output.put_line('Produsul '||v.denumire_produs|| ' contine '||v_nr_ing||' ingrediente in componenta
sa.');
```

```

end loop;
end;

--3.
procedure afisare_cantitati(p_id in meniu.id_produs%type)
is
cursor c is select id_ingredient,denumire_ingredient, cantitate_disponibila,cantitate
            from ingrediente join componenta_produs using(id_ingredient)
            where id_produs=p_id;

v c%rowtype;
den_pr varchar2(40);
begin
select denumire_produs into den_pr from meniu where id_produs=p_id;
dbms_output.put_line('Ingredientele necesare pentru produsul '||den_pr|| ':' );
```

```

open c;
loop
fetch c into v;
exit when c%notfound;
dbms_output.put_line('ID Ingredient: ' || v.id_ingredient ||', Denumire: ' || v.denumire_ingredient ||
', Cantitate disponibila: ' || v.cantitate_disponibila ||', Cantitate necesara: ' || v.cantitate);
end loop;
close c;
end;
--4.
procedure marire_cantitate(p_furnizor in ingrediente.furnizor%type, p_adaos in number)
is
begin
update ingrediente
set cantitate_disponibila=cantitate_disponibila+p_adaos
where furnizor=p_furnizor;

commit;
dbms_output.put_line('Cantitatea a fost modificata pentru ' || sql%rowcount || ' ingrediente.');
```

```

end;
/
set serveroutput on;
begin
if pac3.produs_exista(107) then dbms_output.put_line('Produsul cu id-ul 107 exista!');
else dbms_output.put_line('Produsul cu id-ul 107 nu exista!');
end if;

pac3.numar_ingrediente;
dbms_output.put_line("");
pac3.afisare_cantitati(101);
pac3.marire_cantitate('Furnizor Carnuri SRL',134);
end;
```

```
Package Body PAC3 compiled
```

```

Produsul cu id-ul 107 exista!
Produsul Ciorbă de burtă contine 4 ingrediente in componenta sa.
Produsul Platou țărănesc contine 6 ingrediente in componenta sa.
Produsul Mici cu muștar contine 3 ingrediente in componenta sa.
Produsul Friptură porc contine 4 ingrediente in componenta sa.
Produsul Papanăși contine 4 ingrediente in componenta sa.
Produsul Plăcintă brânză contine 3 ingrediente in componenta sa.
Produsul Coaste porc contine 3 ingrediente in componenta sa.
Produsul Salată vinete contine 3 ingrediente in componenta sa.
Produsul Gogoși pufoase contine 2 ingrediente in componenta sa.
Produsul Ciorbă fasole contine 3 ingrediente in componenta sa.
Produsul Pâine integrală contine 1 ingrediente in componenta sa.
Produsul Orez basmati contine 1 ingrediente in componenta sa.
Produsul Sos pesto contine 1 ingrediente in componenta sa.
Produsul Fructe de pădure contine 1 ingrediente in componenta sa.
Produsul Ciuperci contine 1 ingrediente in componenta sa.
```


Produsul Ciuperci contine 1 ingrediente in componenta sa.

Ingredientele necesare pentru produsul Ciorbă de burtă :

ID Ingredient: 1001, Denumire: Burtă de vită, Cantitate disponibila: 802, Cantitate necesara: 200

ID Ingredient: 1004, Denumire: Smântână, Cantitate disponibila: 600, Cantitate necesara: 50

ID Ingredient: 1007, Denumire: Morcov, Cantitate disponibila: 600, Cantitate necesara: 50

ID Ingredient: 1008, Denumire: Țelină, Cantitate disponibila: 600, Cantitate necesara: 30

Cantitatea a fost modificata pentru 2 ingrediente.

DECLANȘATORI

1. Se creează un trigger pentru a nu permite depășirea limitei de minim 5 lei și maxim 50 lei pentru livrare.

create or replace trigger verifica_pret_livrare

before insert or update on livrari

for each row

begin

if :new.pret_livrare < 5 then

raise_application_error(-20002, 'Pretul livrării nu trebuie să fie sub 5 lei');

elsif :new.pret_livrare > 50 then

raise_application_error(-20003, 'Pretul livrării nu trebuie să fie peste 50 de lei');

end if;

end;

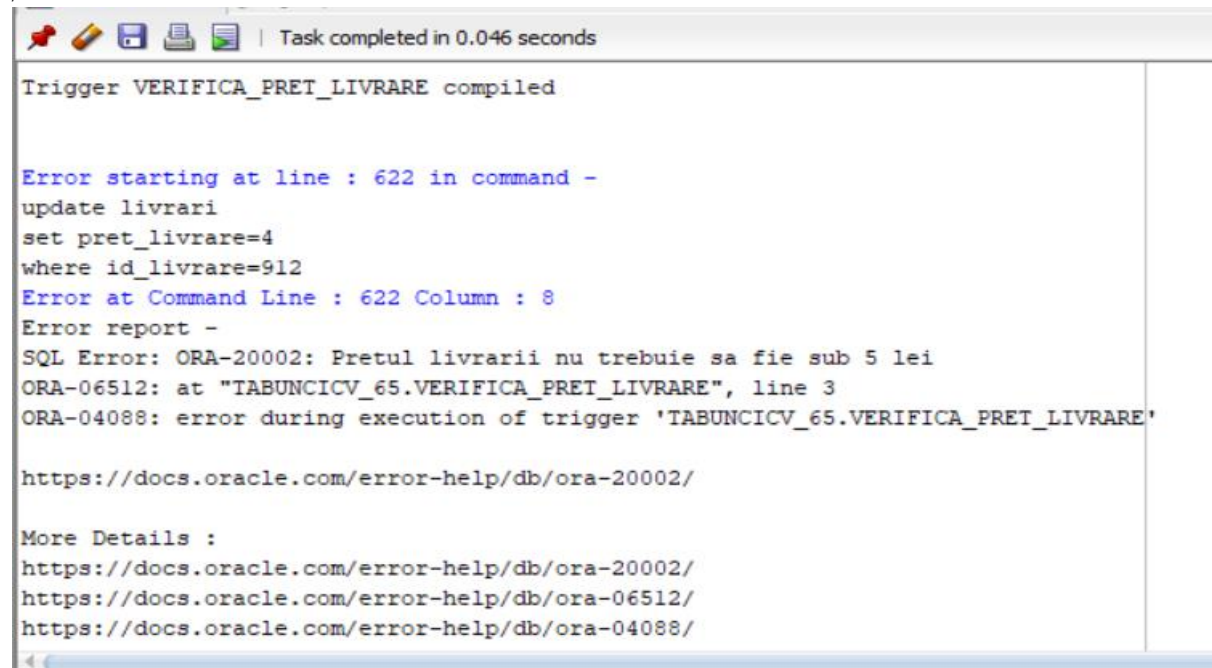
/

update livrari

set pret_livrare=4

where id_livrare=912;

/



The screenshot shows the SQL Developer interface. At the top, a status bar indicates 'Task completed in 0.046 seconds'. Below it, a message box states 'Trigger VERIFICA_PRET_LIVRARE compiled'. The main text area displays the following error message:

```
Error starting at line : 622 in command -
update livrari
set pret_livrare=4
where id_livrare=912
Error at Command Line : 622 Column : 8
Error report -
SQL Error: ORA-20002: Pretul livrării nu trebuie să fie sub 5 lei
ORA-06512: at "TABUNCICV_65.VERIFICA_PRET_LIVRARE", line 3
ORA-04088: error during execution of trigger 'TABUNCICV_65.VERIFICA_PRET_LIVRARE'
```

Below the error message, there are three links to Oracle documentation:

- <https://docs.oracle.com/error-help/db/ora-20002/>
- <https://docs.oracle.com/error-help/db/ora-06512/>
- <https://docs.oracle.com/error-help/db/ora-04088/>

2. Se creează un trigger pentru a nu permite depășirea unei cantități disponibile pentru ingredientele dintr-o anumită categorie.

```
create or replace trigger stop_marire_cantitate
before insert or update on ingrediente
for each row
declare
cantitate_maxima number;
begin

case :new.categorie
when 'Carne' then cantitate_maxima:=10;
when 'Lactate' then cantitate_maxima:=500;
when 'Legume' then cantitate_maxima:=700;
when 'Cereale' then cantitate_maxima:=150;
when 'Conserva' then cantitate_maxima:=400;
when 'Carbohidrați' then cantitate_maxima:=300;
else cantitate_maxima:=100;
end case;

if :new.cantitate_disponibila>cantitate_maxima then
raise_application_error(-20001, 'Cantitatea disponibila depaseste limita maxima pentru categoria
'||:new.categorie);
end if;
end;
/
insert into ingrediente(id_ingredient,denumire_ingredient,categorie,furnizor,cantitate_disponibila)
values(2896, 'Ton', 'Carne','Furnizor F', 12);
/
```

```
Trigger STOP_MARIRE_CANTITATE compiled

Error starting at line : 579 in command -
insert into ingrediente(id_ingredient,denumire_ingredient,categorie,furnizor,cantitate_disponibila)
values(2896, 'Ton', 'Carne','Furnizor F', 12)
Error at Command Line : 579 Column : 13
Error report -
SQL Error: ORA-20001: Cantitatea disponibila depaseste limita maxima pentru categoria Carne
ORA-06512: at "TABUNCICV_65.STOP_MARIRE_CANTITATE", line 16
ORA-04088: error during execution of trigger 'TABUNCICV_65.STOP_MARIRE_CANTITATE'

https://docs.oracle.com/error-help/db/ora-20001/

More Details :
https://docs.oracle.com/error-help/db/ora-20001/
https://docs.oracle.com/error-help/db/ora-06512/
https://docs.oracle.com/error-help/db/ora-04088/
```

3. Să se creeze un trigger care nu permite modificarea reducerii aplicate pentru comenzile cu o vechime mai mare de 2 luni.

```
create or replace trigger trg_stop_reducere
before update of reducere_aplicata on comenzi
for each row
declare
v_vechime number;
begin
v_vechime:=months_between(sysdate,:old.data_comanda);
if v_vechime>2 then
raise_application_error(-20004,'Modificarea reducerii nu este permisa deoarece au trecut peste 2 luni de la
plasarea comenzii.');
```

```
end if;
```

```
end;
```

```
/
```

```
update comenzi
set reducere_aplicata=80
where id_comanda=1000001;
```

```
/
```

```
Trigger TRG_STOP_REUCERE compiled
```

```
Error starting at line : 709 in command -
```

```
update comenzi
```

```
set reducere_aplicata=80
```

```
where id_comanda=1000001
```

```
Error at Command Line : 709 Column : 8
```

```
Error report -
```

```
SQL Error: ORA-20004: Modificarea reducerii nu este permisa deoarece au trecut peste 2 luni de la plasarea comenzii.
```

```
ORA-06512: at "TABUNCICV_65.TRG_STOP_REUCERE", line 6
```

```
ORA-04088: error during execution of trigger 'TABUNCICV_65.TRG_STOP_REUCERE'
```

```
https://docs.oracle.com/error-help/db/ora-20004/
```

```
More Details :
```

```
https://docs.oracle.com/error-help/db/ora-20004/
```

```
https://docs.oracle.com/error-help/db/ora-06512/
```

```
https://docs.oracle.com/error-help/db/ora-04088/
```

4. Să se creeze un trigger asupra tabelii Comenzi , prin care, la modificarea valorii id_comanda din tabela părinte , aceasta să se modifice automat și în înregistrările corespondente din tabela copil.

```
create or replace trigger trg_update_comaenzi
after update of id_comanda on comenzi
for each row
begin
update detalii_comanda
set id_comanda = :new.id_comanda
where id_comanda = :old.id_comanda;
```

```
UPDATE plati
```

```
SET id_comanda = :new.id_comanda
```

```
WHERE id_comanda = :old.id_comanda;
```

```
UPDATE livrari
```

```
SET id_comanda = :new.id_comanda
```

```
WHERE id_comanda = :old.id_comanda;
```

```
end;
```

```
/
```

```
update comenzi
```

```
set id_comanda=1234
```

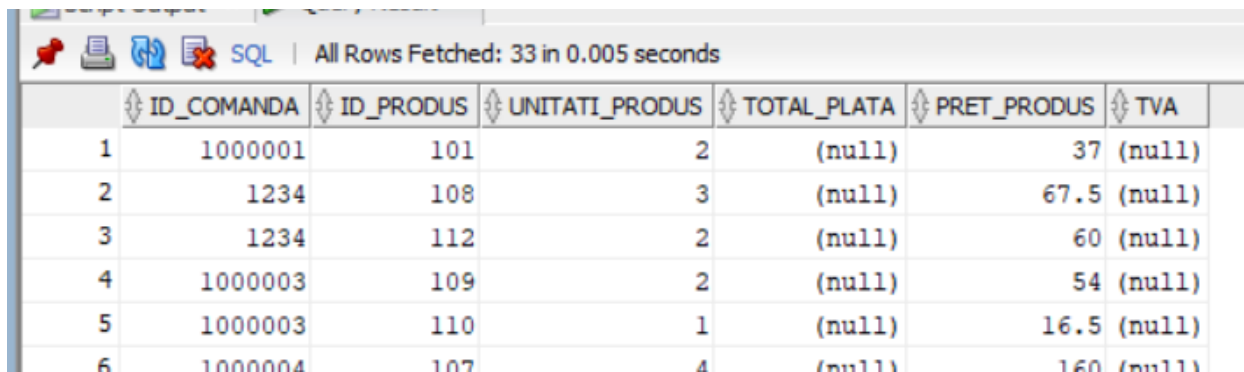
```
where id_comanda=1000002;
```

```
/
```

Trigger TRG_UPDATE_COMAENZI compiled

1 row updated.

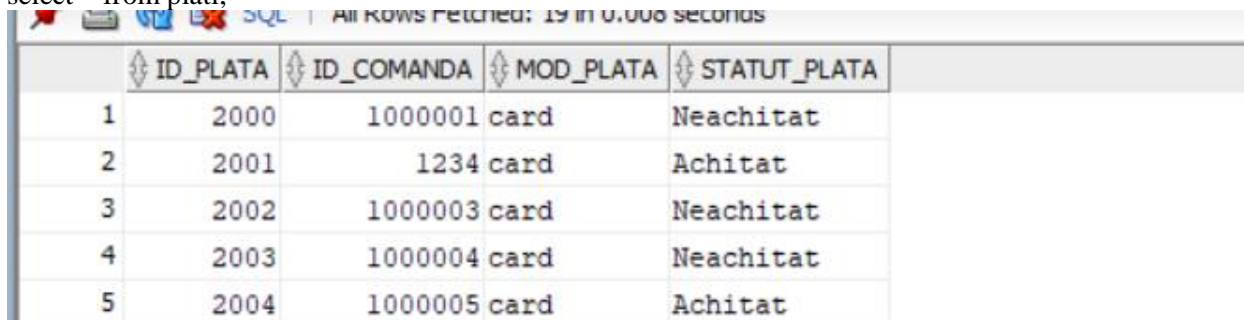
```
select * from detalii_comanda;
```



The screenshot shows a SQL query result in a database client. The status bar indicates "All Rows Fetched: 33 in 0.005 seconds". The table has 7 columns: ID_COMANDA, ID_PRODUS, UNITATI_PRODUS, TOTAL_PLATA, PRET_PRODUS, and TVA. There are 6 rows of data displayed.

	ID_COMANDA	ID_PRODUS	UNITATI_PRODUS	TOTAL_PLATA	PRET_PRODUS	TVA
1	1000001	101	2	(null)	37	(null)
2	1234	108	3	(null)	67.5	(null)
3	1234	112	2	(null)	60	(null)
4	1000003	109	2	(null)	54	(null)
5	1000003	110	1	(null)	16.5	(null)
6	1000004	107	4	(null)	160	(null)

```
select * from plati;
```



The screenshot shows a SQL query result in a database client. The status bar indicates "All Rows Fetched: 19 in 0.008 seconds". The table has 5 columns: ID_PLATA, ID_COMANDA, MOD_PLATA, and STATUT_PLATA. There are 5 rows of data displayed.

	ID_PLATA	ID_COMANDA	MOD_PLATA	STATUT_PLATA
1	2000	1000001	card	Neachitat
2	2001	1234	card	Achitat
3	2002	1000003	card	Neachitat
4	2003	1000004	card	Neachitat
5	2004	1000005	card	Achitat

5. Să se creeze un trigger care în momentul modificării statutului platii într-o înregistrare din tabela plati , va adăuga automat o nouă înregistrare în tabela istoric_plăți.

```
create table istoric_plati (id_istoric number primary key,
                           id_plata number,
                           id_comanda number,
                           statut_vechi varchar2(20),
                           statut_nou varchar2(20));

/

create sequence secventa_istoric_plati
start with 1
increment by 1
maxvalue 100
nocycle;

/

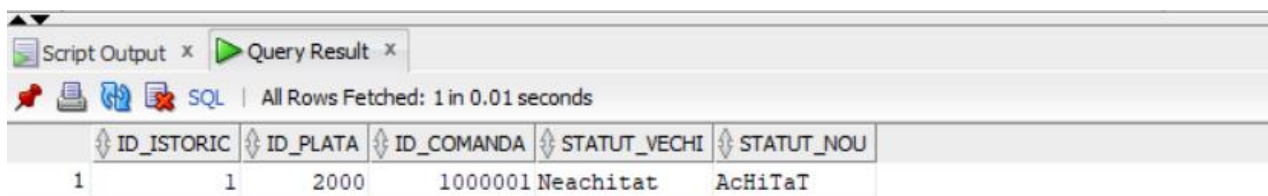
create or replace trigger trg_modif_statut
before update of statut_plata on plati
for each row
begin
insert into istoric_plati(id_istoric,id_plata,id_comanda,statut_vechi,statut_nou)
values(secventa_istoric_plati.nextval,:old.id_plata,:old.id_comanda,:old.statut_plata,:new.statut_plata);
end;

/

update plati
set statut_plata='AcHiTaT'
where id_plata=2000;

/

select * from istoric_plati;
```



The screenshot shows a SQL query result window with a single row of data. The window has tabs for 'Script Output' and 'Query Result'. The 'Query Result' tab is active, showing a table with 5 columns: ID_ISTORIC, ID_PLATA, ID_COMANDA, STATUT_VECHI, and STATUT_NOU. The first row contains the values 1, 1, 2000, 1000001, and Neachitat, respectively. The STATUT_NOU column is highlighted in blue.

ID_ISTORIC	ID_PLATA	ID_COMANDA	STATUT_VECHI	STATUT_NOU
1	1	2000	1000001	Neachitat

6. Să se creeze un trigger care asigură unicitatea identificatorului clientului folosind valorile generate de o secventa.

```
create sequence clienti_secventa
start with 16
increment by 1
maxvalue 100
nocycle;

create or replace trigger generare_id_client
before insert on clienti
```

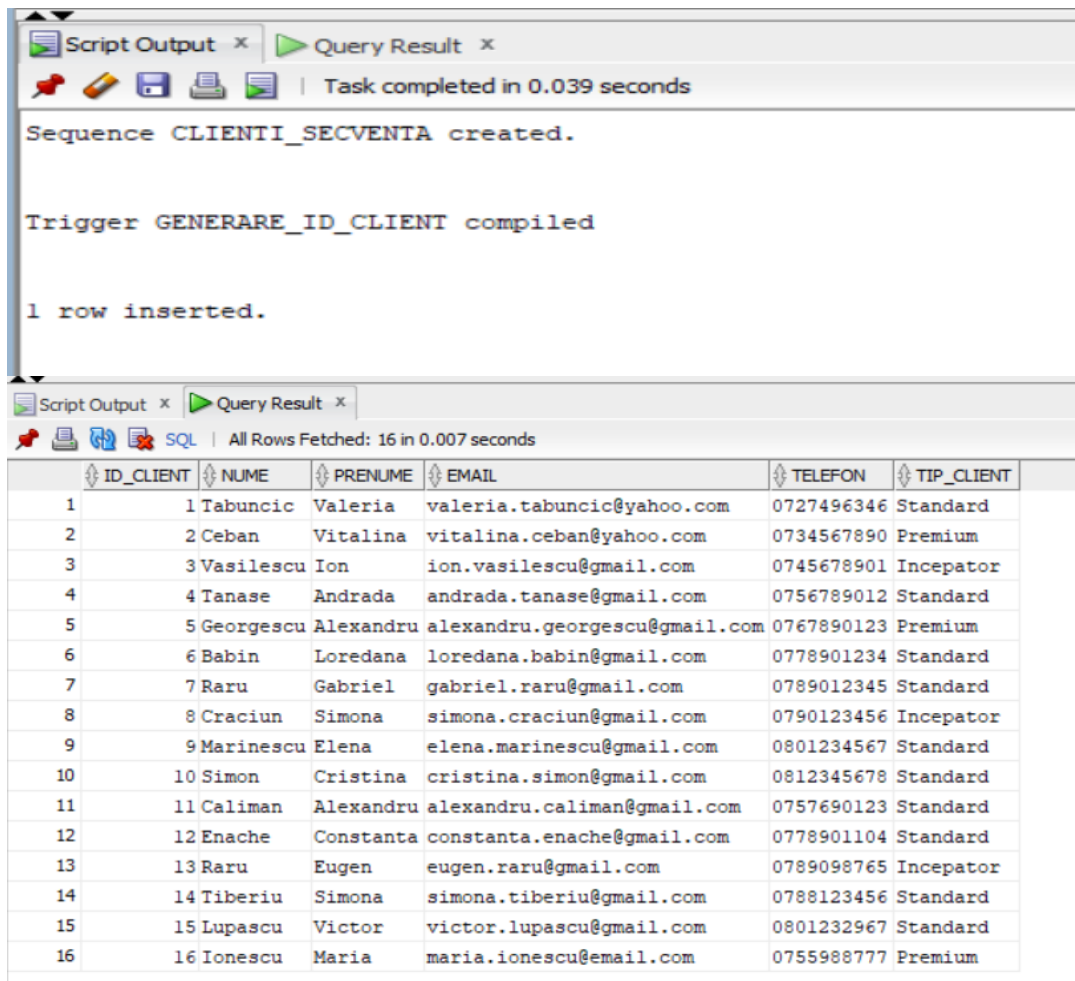
```

for each row
begin
select clienti_secventa.nextval into :new.id_client from dual;
end;
/

insert into clienti (nume, prenume, email, telefon, tip_client)
values ('Ionescu', 'Maria', 'maria.ionescu@email.com', '0755988777', 'Premium');
/

select * from clienti;
/

```



The screenshot shows a database management tool interface. The top panel, titled 'Script Output' and 'Query Result', displays the execution of SQL scripts. The first script successfully created the 'CLIENTI_SECVENTA' sequence, compiled the 'GENERARE_ID_CLIENT' trigger, and inserted one row. The second script fetched all 16 rows from the 'CLIENTI' table, which are displayed in the table below.

ID_CLIENT	NUME	PRENUME	EMAIL	TELEFON	TIP_CLIENT
1	1 Tabuncic	Valeria	valeria.tabuncic@yahoo.com	0727496346	Standard
2	2 Ceban	Vitalina	vitalina.ceban@yahoo.com	0734567890	Premium
3	3 Vasilescu	Ion	ion.vasilescu@gmail.com	0745678901	Incepator
4	4 Tanase	Andrada	andrada.tanase@gmail.com	0756789012	Standard
5	5 Georgescu	Alexandru	alexandru.georgescu@gmail.com	0767890123	Premium
6	6 Babin	Loredana	loredana.babin@gmail.com	0778901234	Standard
7	7 Raru	Gabriel	gabriel.raru@gmail.com	0789012345	Standard
8	8 Craciun	Simona	simona.craciun@gmail.com	0790123456	Incepator
9	9 Marinescu	Elena	elena.marinescu@gmail.com	0801234567	Standard
10	10 Simon	Cristina	cristina.simon@gmail.com	0812345678	Standard
11	11 Caliman	Alexandru	alexandru.caliman@gmail.com	0757690123	Standard
12	12 Enache	Constanta	constantina.enache@gmail.com	0778901104	Standard
13	13 Raru	Eugen	eugen.raru@gmail.com	0789098765	Incepator
14	14 Tiberiu	Simona	simona.tiberiu@gmail.com	0788123456	Standard
15	15 Lupascu	Victor	victor.lupascu@gmail.com	0801232967	Standard
16	16 Ionescu	Maria	maria.ionescu@email.com	0755988777	Premium