**PROIECT**

**SISTEME DE GESTIUNE A BAZELOR DE DATE**

Restaurant

Moisii Lucian 241

**CUPRINS**

1. Prezentarea bazei de date
2. Diagrama entitate-relație (ERD)
3. Diagrama conceptuală
4. Implementarea tabelelor în Oracle
5. Inserarea datelor în tabele
6. Subprogram stocat care utilizează două tipuri de colecții
7. Subprogram stocat care utilizează doua tipuri de cursoare
8. Subprogram stocat de tip funcț ie care să utilizeze într-o singură comandă SQL 3 tabele definite
9. Subprogram stocat de tip procedură care să utilizeze într-o singură comandă de tip SQL 5 dintre tabelele definite
10. Definiți un trigger de tip LMD la nivel de comandă
11. Definiți un trigger de tip LMD la nivel de linie
12. Definiți un trigger de tip LDD
13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului
14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

**1. Prezentarea bazei de date**

Acest proiect cuprinde proiectarea unei baze de date și rezolvarea unor probleme pe aceasta.

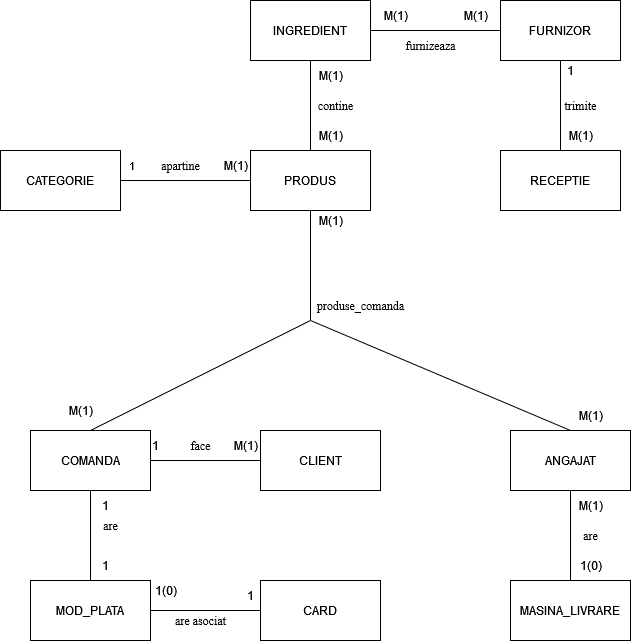
Tema aleasă pentru această bază de date are în centrul său gestiunea unui restaurant, în cadrul căreia sunt reținute informații despre personalul, comenzile, clienții, meniul, furnizorii și livrarea produselor ce aparțin acestei afaceri.

Restaurantul are o locație fizică unde se pot efectua comenzi ce vor urma a fi servite în locație, la pachet sau livrate. De asemenea, restaurantul dispune și de o platformă online prin cadrul căreia clienții pot da comandă.

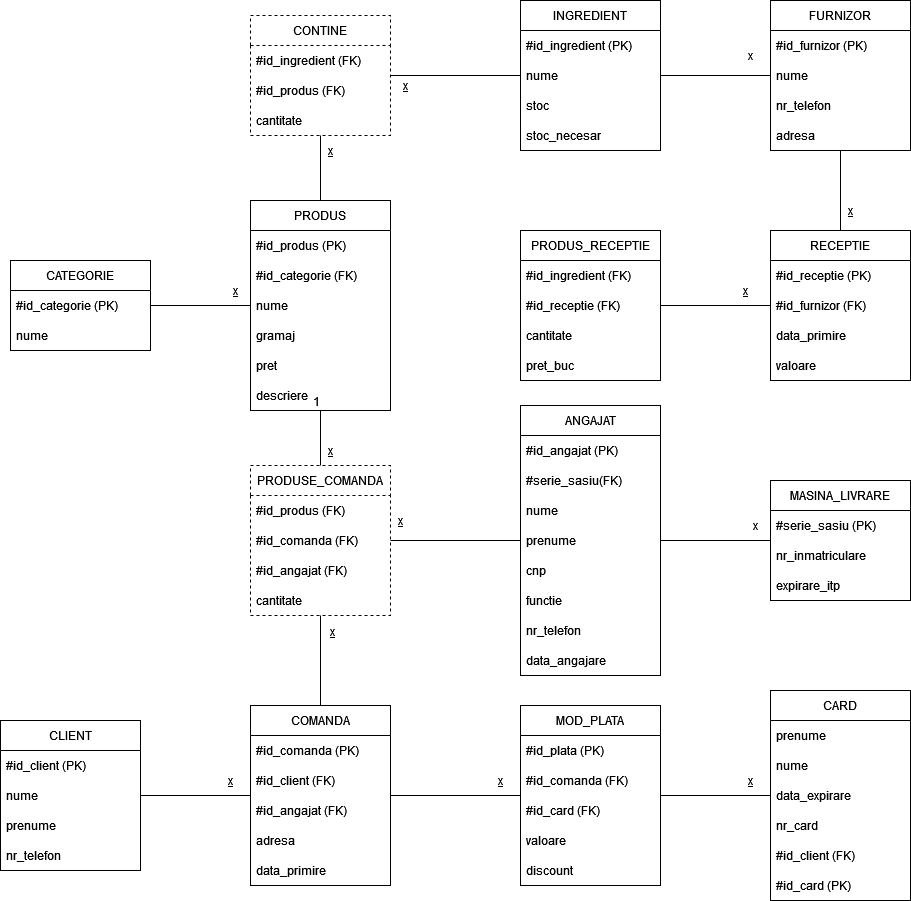
Personalul restaurantului este format din șef (sau șefi daca sunt mai multe persoane care dețin împreuna restaurantul), manageri, bucătari-șefi, bucătari, ajutor de bucătar, pizzeri, ajutor de pizzeri, barmani, ajutor de barman, ospătari și livratori.

Furnizorii restaurantului sunt diferite firme locale sau lanțuri mari de magazinee prin care acesta se aprovizioneaza.

**2. Diagrama entitate-relație (ERD)**

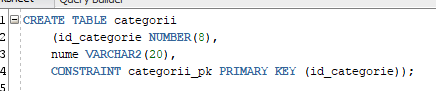
****

**3. Diagrama conceptuală**

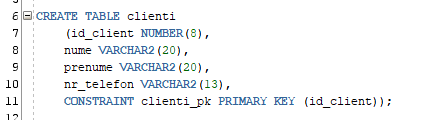
****

**4. Implementarea tabelelor in SQL**

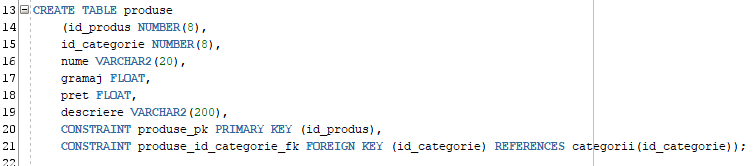
1. Crearea tabelului CATEGORII:



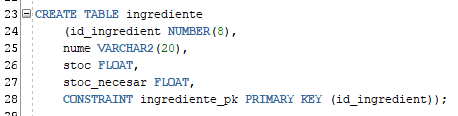
1. Crearea tabelului CLIENTI:



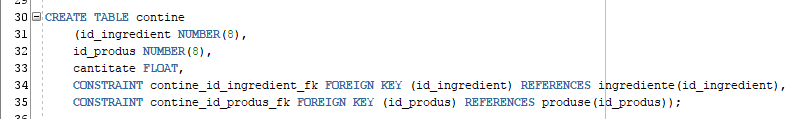
1. Crearea tabelului PRODUSE:



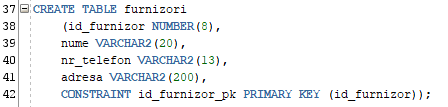
1. Crearea tabelului INGREDIENTE:



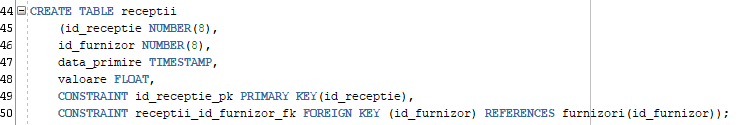
1. Crearea tabelului CONTINE:



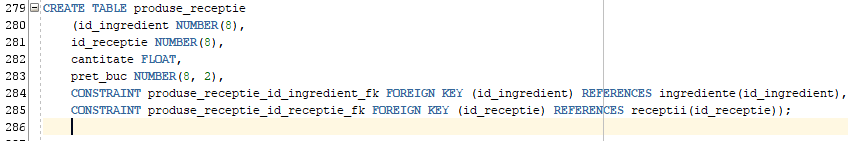
1. Crearea tabelului FURNIZORI:



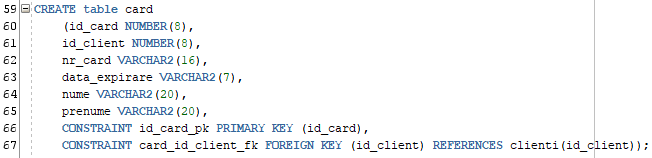
1. Crearea tabelului RECEPTII:



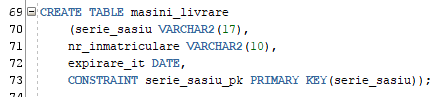
1. Crearea tabelului PRODUSE\_RECEPTIE:



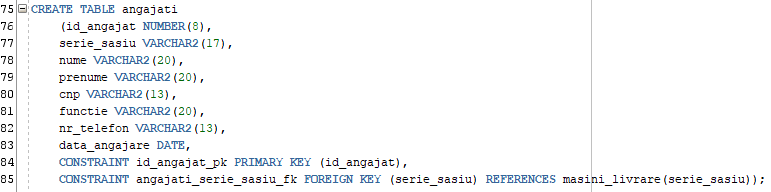
1. Crearea tabelului CARD:



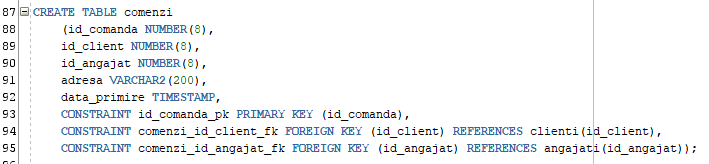
1. Crearea tabelului MASINI\_LIVRARE:



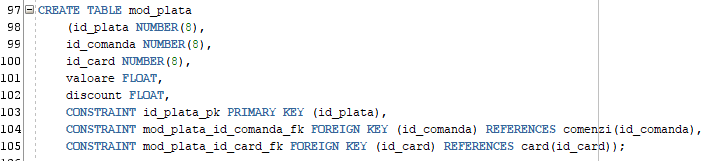
1. Crearea tabelului ANGAJATI:



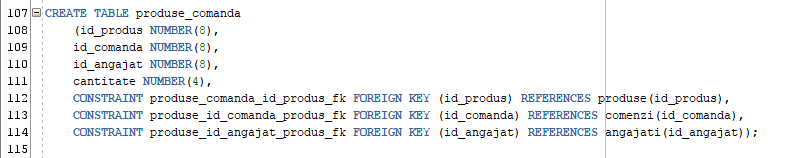
1. Crearea tabelului COMENZI:



1. Crearea tabelului MOD\_PLATA:

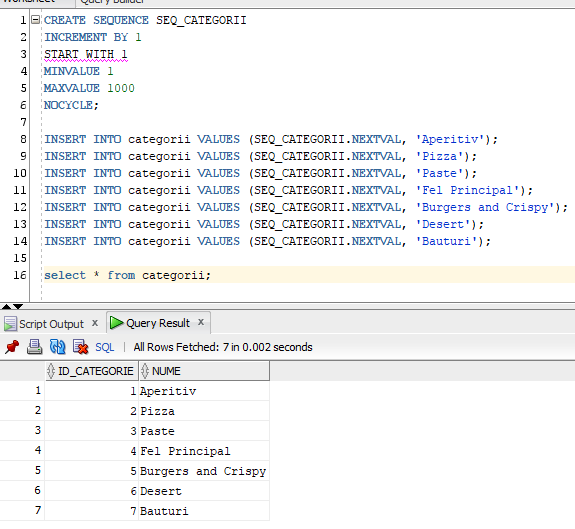


1. Crearea tabelului PRODUSE\_COMANDA:

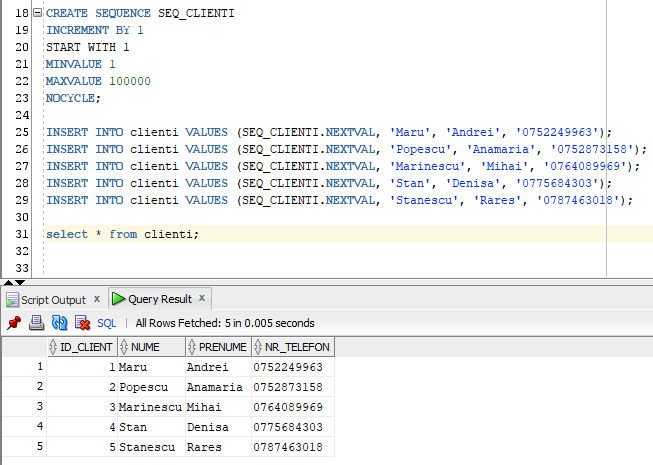


**5. Inserarea datelor în tabele**

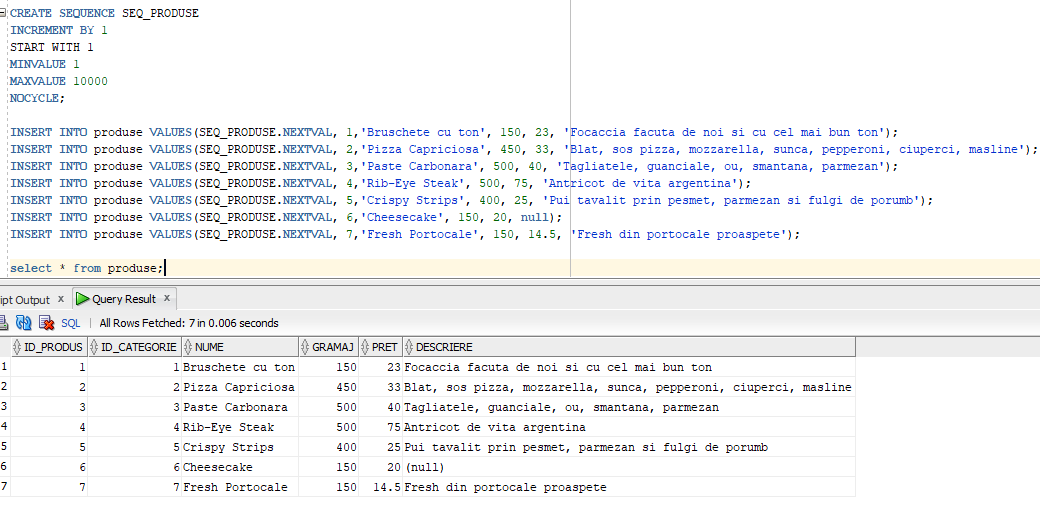
1. Inserarea datelor în tabelul CATEGORII:

****

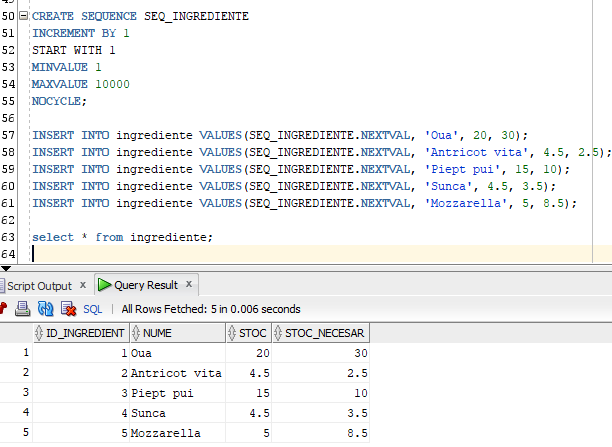
2. Inserarea datelor în tabelul CLIENTI:

****

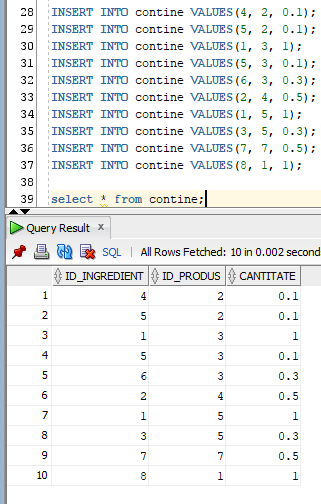
3. Inserarea datelor în tabelul PRODUSE:

****

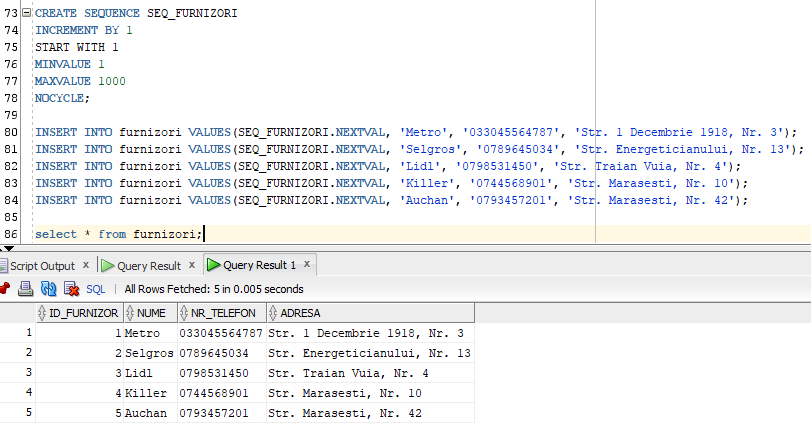
4. Inserarea datelor în tabelul INGREDIENTE:

****

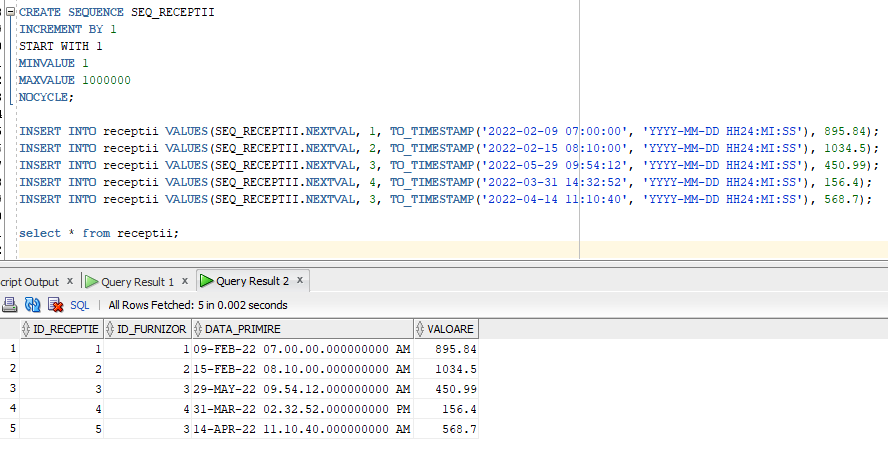
5. Inserarea datelor în tabelul CONTINE:

****

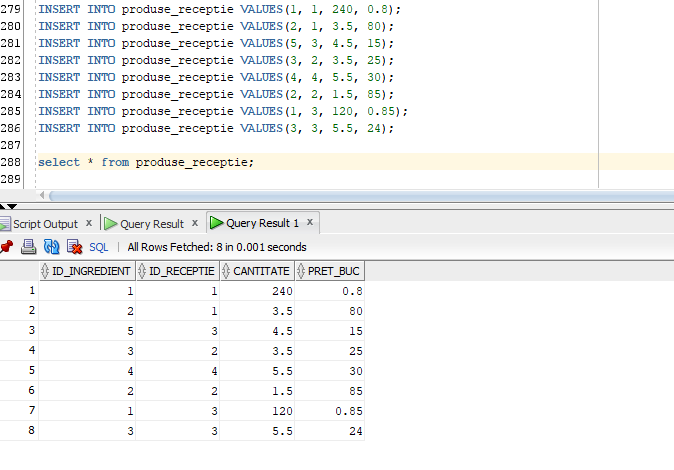
6. Inserarea datelor în tabelul FURNIZORI:

****

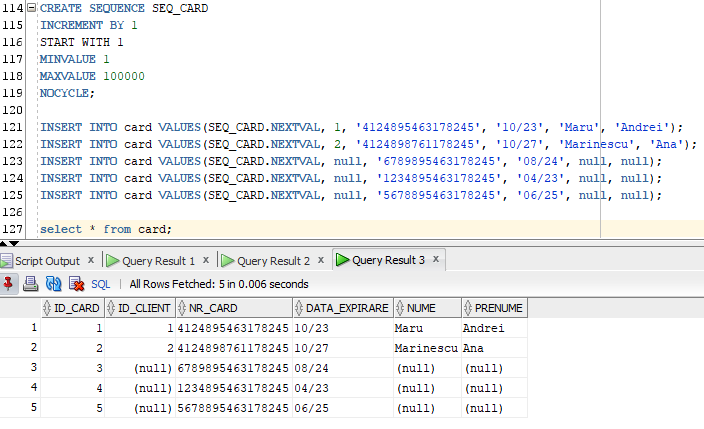
7. Inserarea datelor în tabelul RECEPTII:

****

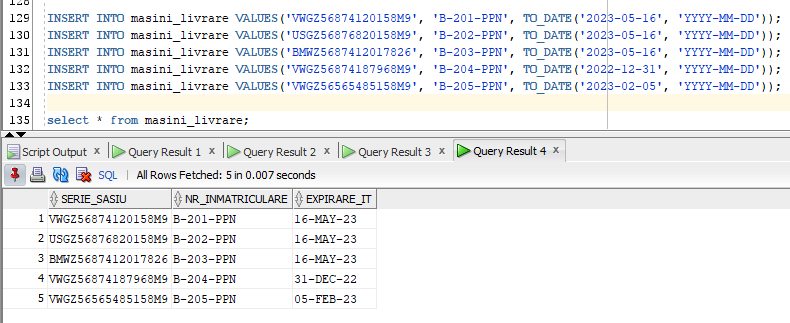
8. Inserarea datelor în tabelul PRODUSE\_RECEPTIE:

****

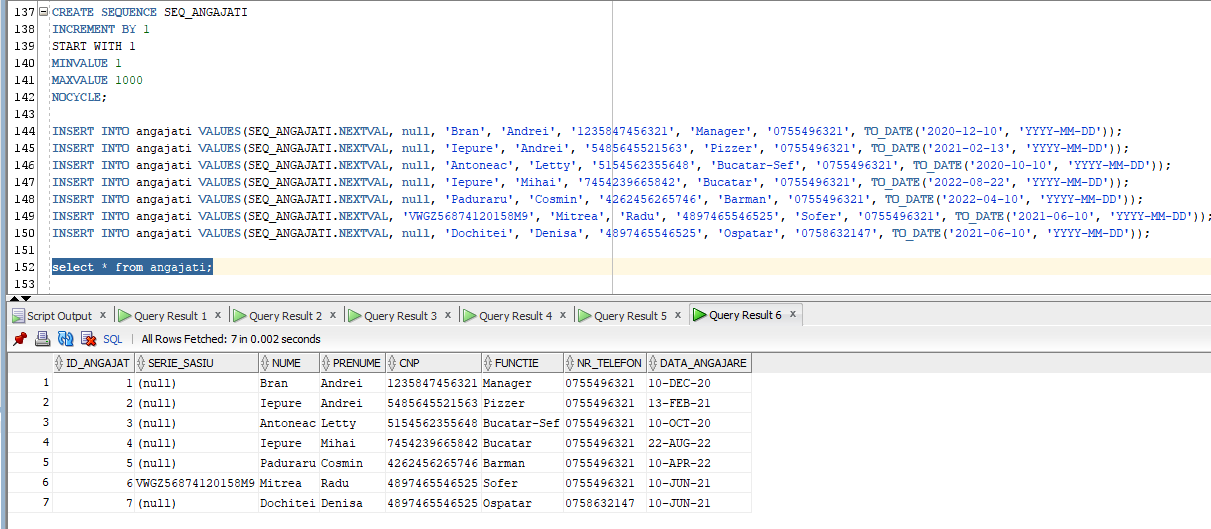
9. Inserarea datelor în tabelul CARD:

****

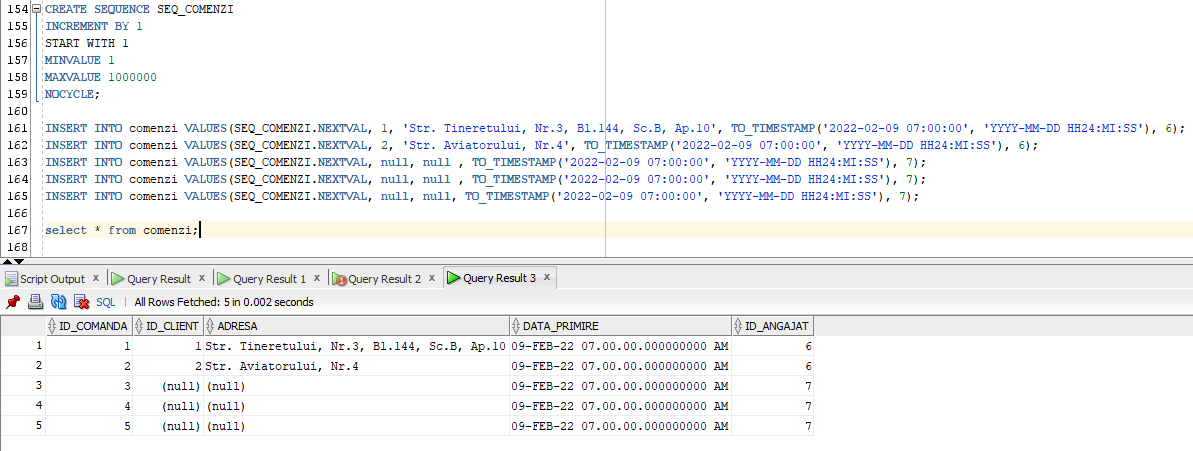
10. Inserarea datelor în tabelul MASINI\_LIVRARE:

****

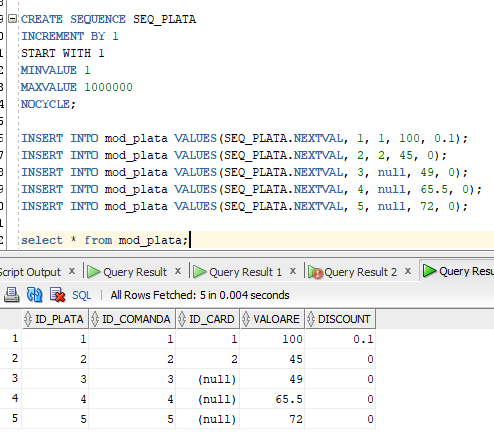
11. Inserarea datelor în tabelul ANGAJATI:

****

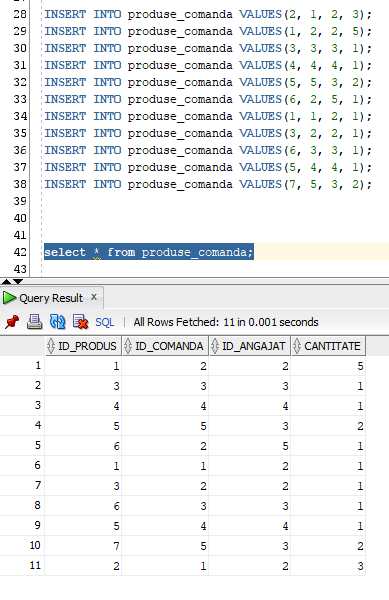
12. Inserarea datelor în tabelul COMENZI:

****

13. Inserarea datelor în tabelul MOD\_PLATA:

****

14. Inserarea datelor în tabelul PRODUSE\_COMANDA:

****

**6. Subprogram stocat care utilizează două tipuri de colecții**

*Scrieti o procedura ce afișează pentru fiecare categorie produsele din acea categorie, iar pentru fiecare produs ingredientele sale.*

CREATE OR REPLACE PROCEDURE ex\_6

IS

TYPE produs IS RECORD (id\_produs produse.id\_produs%TYPE, nume produse.nume%TYPE);

TYPE vec\_categorii IS TABLE OF categorii%ROWTYPE INDEX BY PLS\_INTEGER;

TYPE vec\_produse IS TABLE OF produs;

TYPE vec\_contine IS VARRAY(20) OF (ingrediente.id\_ingredient%TYPE);

v\_produse vec\_produse := vec\_produse();

v\_categorii vec\_categorii;

v\_contine vec\_contine := vec\_contine();

j INTEGER;

v\_nume ingrediente.nume%TYPE;

a NUMBER(4);

BEGIN

SELECT \* BULK COLLECT INTO v\_categorii FROM categorii;

FOR i IN v\_categorii.FIRST..v\_categorii.LAST LOOP

SELECT id\_produs, nume BULK COLLECT INTO v\_produse FROM produse WHERE id\_categorie = v\_categorii(i).id\_categorie;

DBMS\_OUTPUT.PUT\_LINE('Categoria ' || v\_categorii(i).nume);

j := v\_produse.FIRST;

WHILE j <= v\_produse.LAST LOOP

DBMS\_OUTPUT.PUT(v\_produse(j).nume || ' : ');

SELECT COUNT(\*) INTO a FROM contine WHERE id\_produs = v\_produse(j).id\_produs;

IF a > 0 THEN

a := a - 1;

SELECT id\_ingredient BULK COLLECT INTO v\_contine FROM contine WHERE v\_produse(j).id\_produs = id\_produs;

FOR k in v\_contine.FIRST..v\_contine.LAST LOOP

SELECT nume INTO v\_nume FROM ingrediente WHERE id\_ingredient = v\_contine(k);

DBMS\_OUTPUT.PUT(v\_nume);

IF a > 0 THEN

DBMS\_OUTPUT.PUT(', ');

a := a - 1;

END IF;

END LOOP;

ELSE

DBMS\_OUTPUT.PUT('Nu exista inca ingrediente introduse pentru acest produs');

END IF;

j := v\_produse.NEXT(j);

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

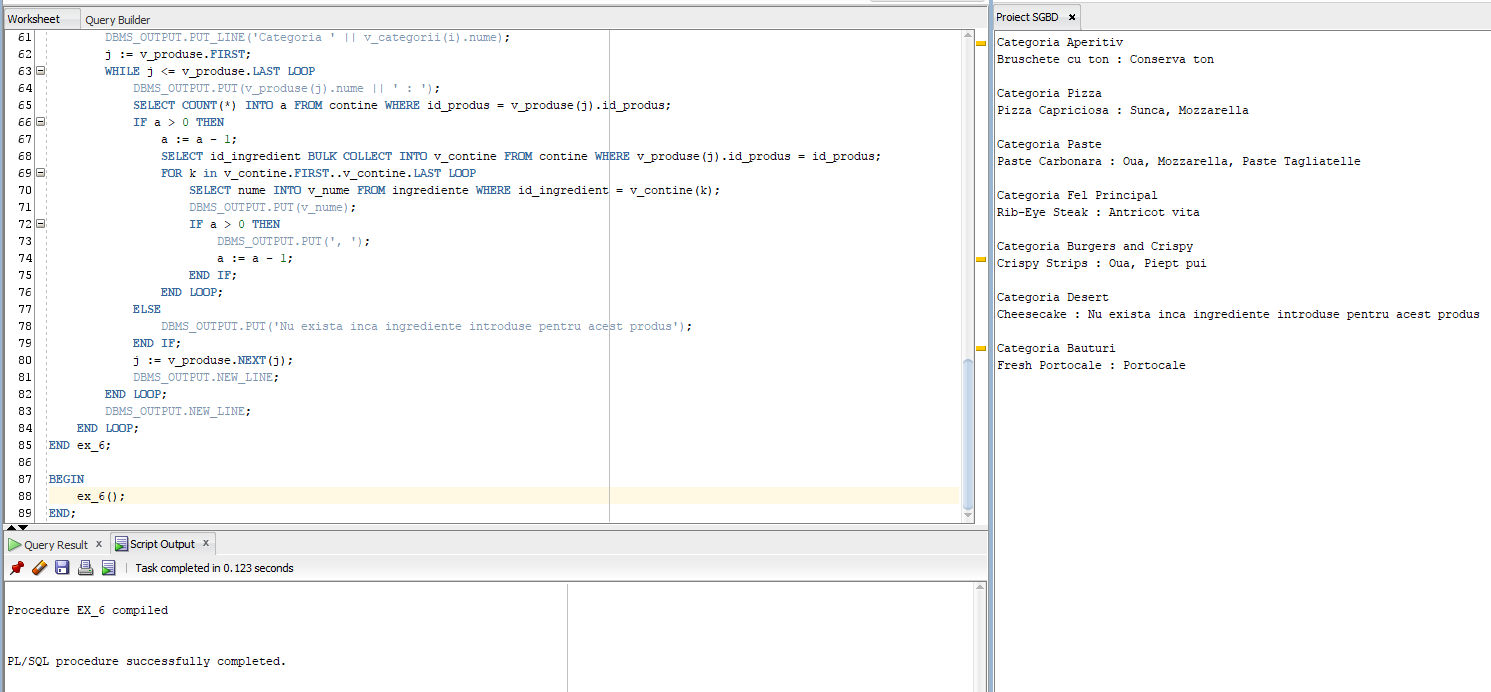
END LOOP;

END ex\_6;

BEGIN

ex\_6();

END;



**7. Subprogram stocat care utilizează doua tipuri de cursoare**

*Să se scrie o procedură ce primește ca parametri numele și prenumele unui client și afișeaza data fiecărei comenzi precum și produsele comandate.*

CREATE OR REPLACE PROCEDURE ex\_7

(p\_nume clienti.nume%TYPE,

p\_prenume clienti.prenume%TYPE)

IS

g\_id\_comanda comenzi.id\_comanda%TYPE;

g\_data\_primire comenzi.data\_primire%TYPE;

g\_nume\_produs produse.nume%TYPE;

g\_index NUMBER(4) := 1;

g\_comma BOOLEAN;

CURSOR c\_comenzi\_client

IS

SELECT id\_comanda, data\_primire FROM comenzi WHERE id\_client = (SELECT id\_client

FROM clienti

WHERE nume = p\_nume AND prenume = p\_prenume);

CURSOR c\_produse\_comanda (v\_id\_comanda comenzi.id\_comanda%TYPE)

IS

SELECT id\_produs FROM produse\_comanda WHERE id\_comanda = v\_id\_comanda;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Clientul cu numele ' || p\_nume || ' ' || p\_prenume || ' a dat comenzile:');

--cursor clasic

OPEN c\_comenzi\_client;

LOOP

FETCH c\_comenzi\_client INTO g\_id\_comanda, g\_data\_primire;

EXIT WHEN c\_comenzi\_client%NOTFOUND;

DBMS\_OUTPUT.PUT(g\_index || '. Comanda de la data de ' || TO\_CHAR(g\_data\_primire, 'YYYY-MM-DD HH24:MI:SS') || ' contine: ');

g\_comma := true;

--ciclu cursor cu parametru

FOR i IN c\_produse\_comanda(g\_id\_comanda) LOOP

SELECT nume INTO g\_nume\_produs FROM produse WHERE id\_produs = i.id\_produs;

IF g\_comma THEN

DBMS\_OUTPUT.PUT(g\_nume\_produs);

g\_comma := false;

ELSE

DBMS\_OUTPUT.PUT(', ' || g\_nume\_produs);

END IF;

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

g\_index := g\_index + 1;

END LOOP;

END ex\_7;

DECLARE

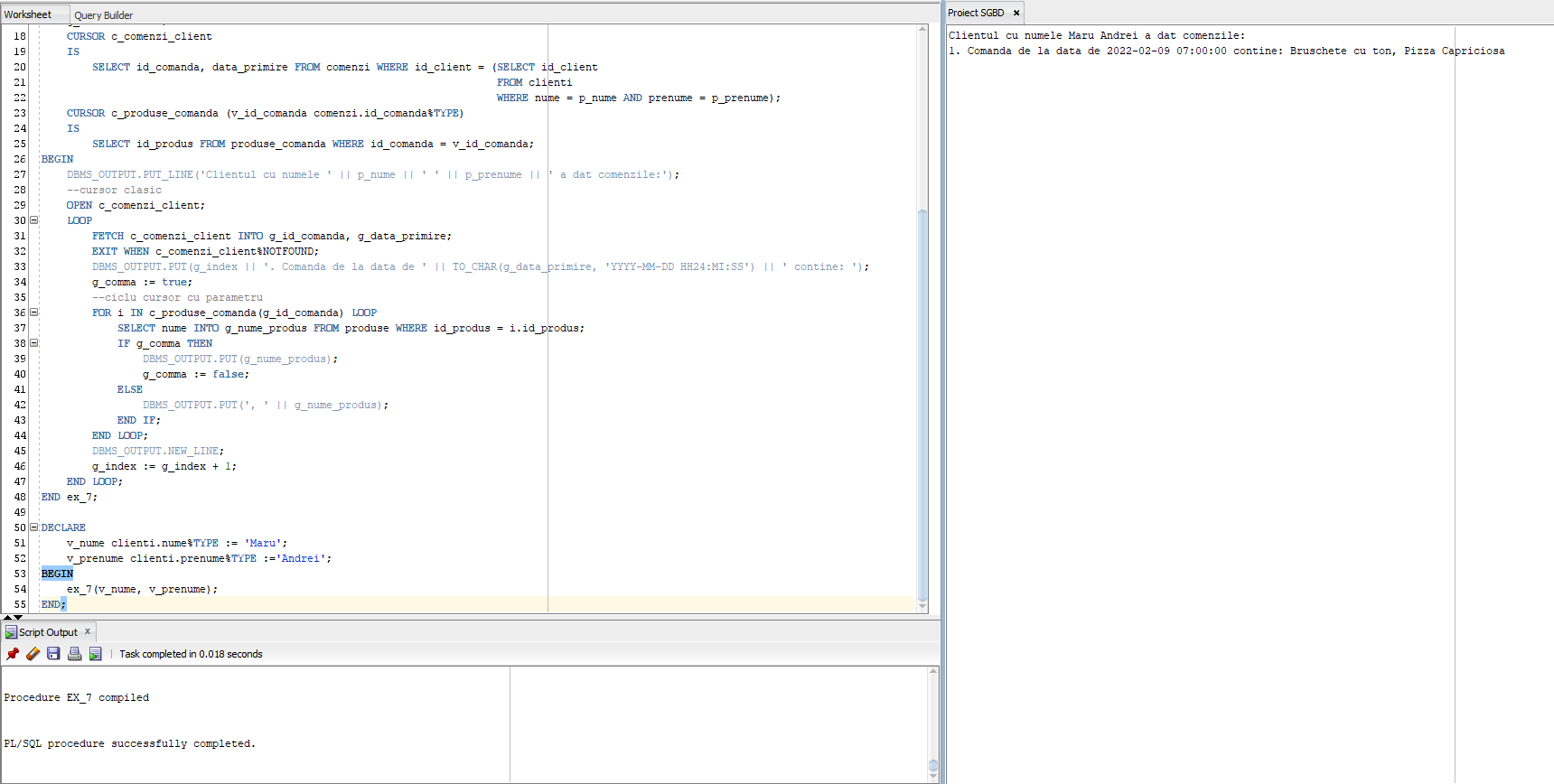
v\_nume clienti.nume%TYPE := 'Maru';

v\_prenume clienti.prenume%TYPE :='Andrei';

BEGIN

ex\_7(v\_nume, v\_prenume);

END;

****

**8. Subprogram stocat de tip funcție care să utilizeze într-o singură comandă SQL 3 tabele definite**

*Să se scrie o funcție care primește ca parametri numele, prenumele și data unei comenzi si returneaza valoarea comenzii. Tratați excepțiile necesare.*

CREATE OR REPLACE FUNCTION ex\_8 (f\_nume clienti.nume%TYPE,

f\_prenume clienti.prenume%TYPE,

f\_data comenzi.data\_primire%TYPE)

RETURN NUMBER

IS

TYPE price\_table\_datatype IS TABLE OF NUMBER(8, 2) INDEX BY PLS\_INTEGER;

price\_table price\_table\_datatype;

pret\_total NUMBER(8,2) := 0;

nr NUMBER(4);

v\_id\_client clienti.id\_client%TYPE;

zero\_clients\_exc EXCEPTION;

multiple\_clients\_exc EXCEPTION;

zero\_orders\_exc EXCEPTION;

BEGIN

SELECT COUNT(\*) INTO nr FROM clienti WHERE nume = f\_nume AND prenume = f\_prenume;

IF nr = 0 THEN

RAISE zero\_clients\_exc;

ELSIF nr > 1 THEN

RAISE multiple\_clients\_exc;

END IF;

SELECT id\_client INTO v\_id\_client FROM clienti WHERE UPPER(nume) = UPPER(f\_nume) AND UPPER(prenume) = UPPER(f\_prenume);

SELECT COUNT(\*) INTO nr FROM comenzi WHERE id\_client = v\_id\_client AND data\_primire = f\_data;

IF nr = 0 THEN

RAISE zero\_orders\_exc;

END IF;

SELECT pc.cantitate \* p.pret BULK COLLECT INTO price\_table FROM produse\_comanda pc

JOIN comenzi c

ON c.id\_comanda = pc.id\_comanda

JOIN produse p

ON p.id\_produs = pc.id\_produs

WHERE c.id\_client = v\_id\_client AND c.data\_primire = f\_data;

FOR i IN price\_table.FIRST..price\_table.LAST LOOP

pret\_total := pret\_total + price\_table(i);

END LOOP;

RETURN NVL(pret\_total, 0);

EXCEPTION

WHEN zero\_clients\_exc THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Nu exista nici un client cu numele dat');

WHEN multiple\_clients\_exc THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Exista mai multi clienti cu numele dat');

WHEN zero\_orders\_exc THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Nu exista nici o comanda data de acest client la data respectiva');

WHEN value\_error THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Nu exista nici un produs in comanda data');

END ex\_8;

DECLARE

v\_nume clienti.nume%TYPE := 'Maru';

v\_prenume clienti.prenume%TYPE := 'Andrei';

v\_data comenzi.data\_primire%TYPE := '09-FEB-22 07.00.00.000000000 AM';

-- exemplu pentru eroarea -20001

-- v\_nume clienti.nume%TYPE := 'Maru';

-- v\_prenume clienti.prenume%TYPE := 'Marius';

-- v\_data comenzi.data\_primire%TYPE := '09-FEB-22 07.00.00.000000000 AM';

---- exemplu pentru eroarea -20002

-- v\_nume clienti.nume%TYPE := 'Stanescu';

-- v\_prenume clienti.prenume%TYPE := 'Rares';

-- v\_data comenzi.data\_primire%TYPE := '09-FEB-22 07.00.00.000000000 AM';

---- exemplu pentru eroarea -20003

-- v\_nume clienti.nume%TYPE := 'Maru';

-- v\_prenume clienti.prenume%TYPE := 'Andrei';

-- v\_data comenzi.data\_primire%TYPE := '09-FEB-22 08.20.00.000000000 AM';

---- exemplu pentru eroarea -20004

-- v\_nume clienti.nume%TYPE := 'Stan';

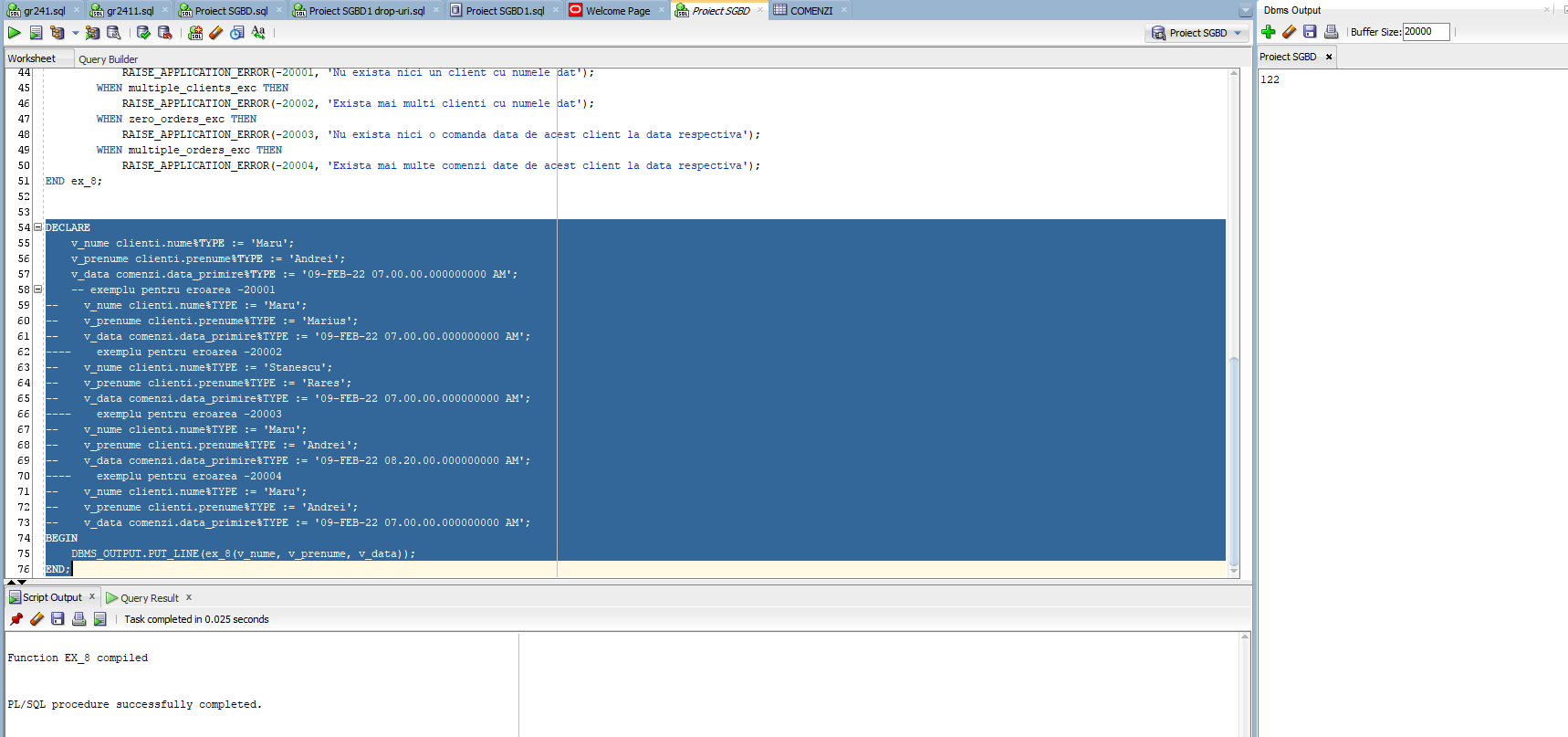
-- v\_prenume clienti.prenume%TYPE := 'Denisa';

-- v\_data comenzi.data\_primire%TYPE := '09-FEB-22 07.00.00.000000000 AM';

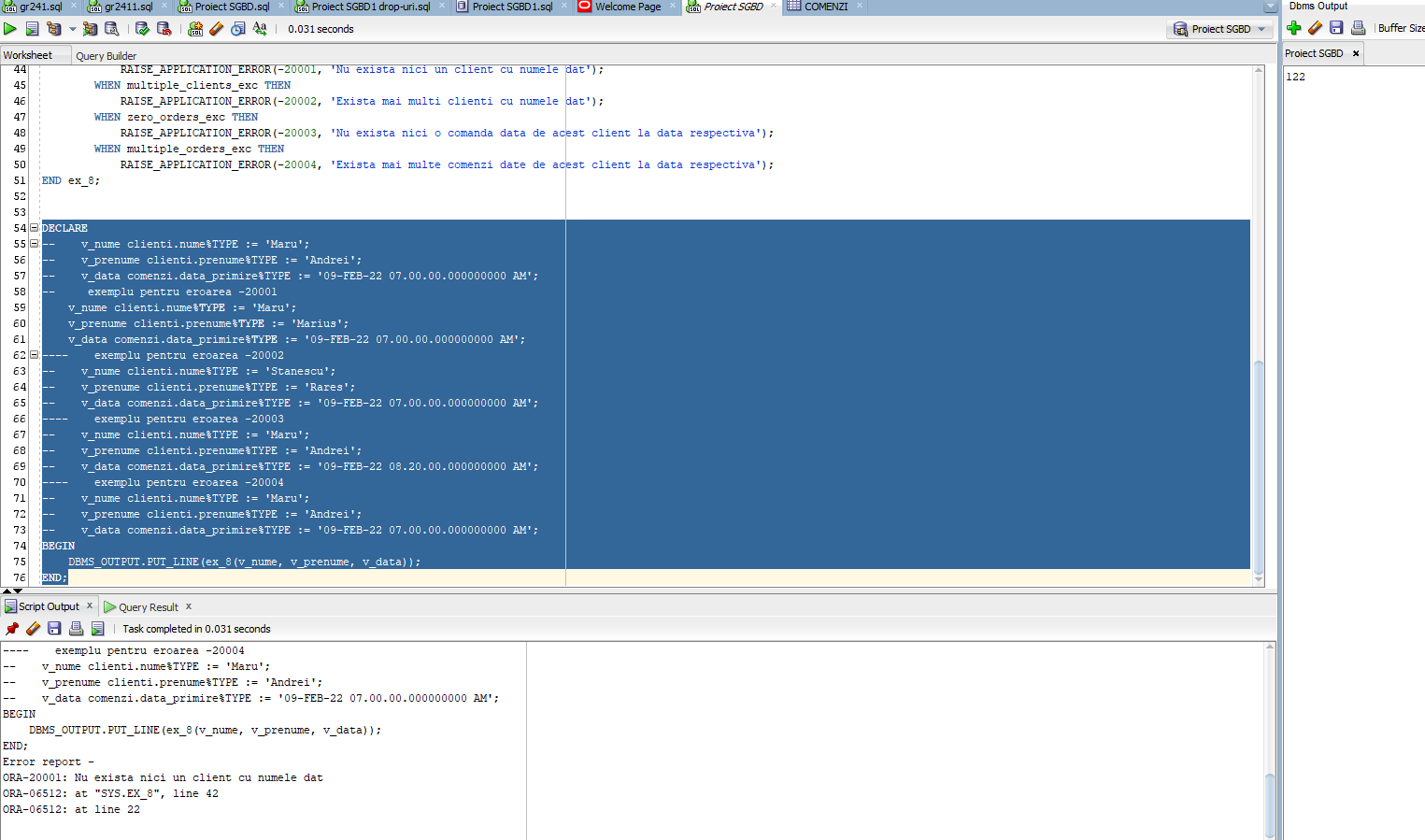
BEGIN

DBMS\_OUTPUT.PUT\_LINE('Valoarea comenzii este de ' || ex\_8(v\_nume, v\_prenume, v\_data) || 'RON');

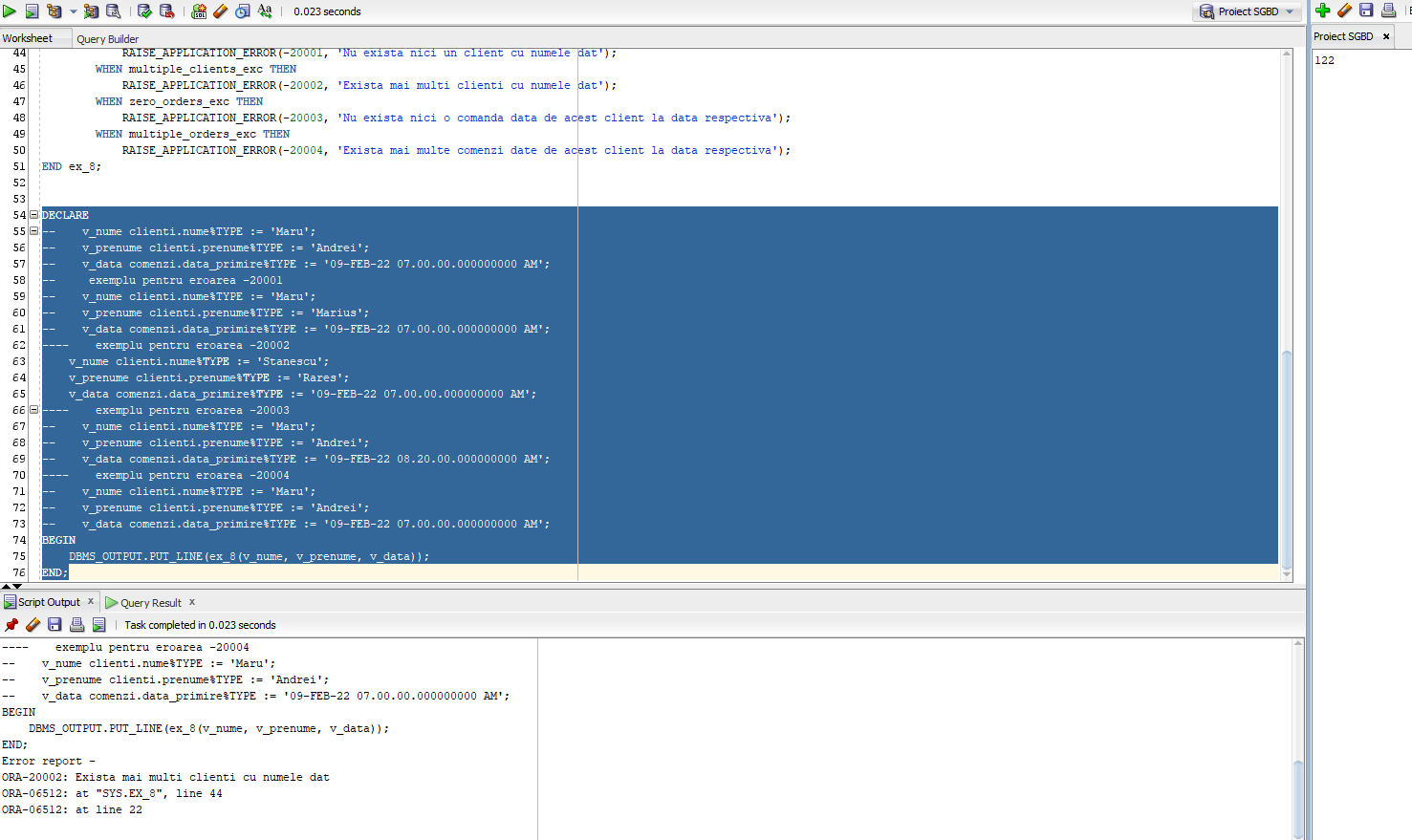
END;;



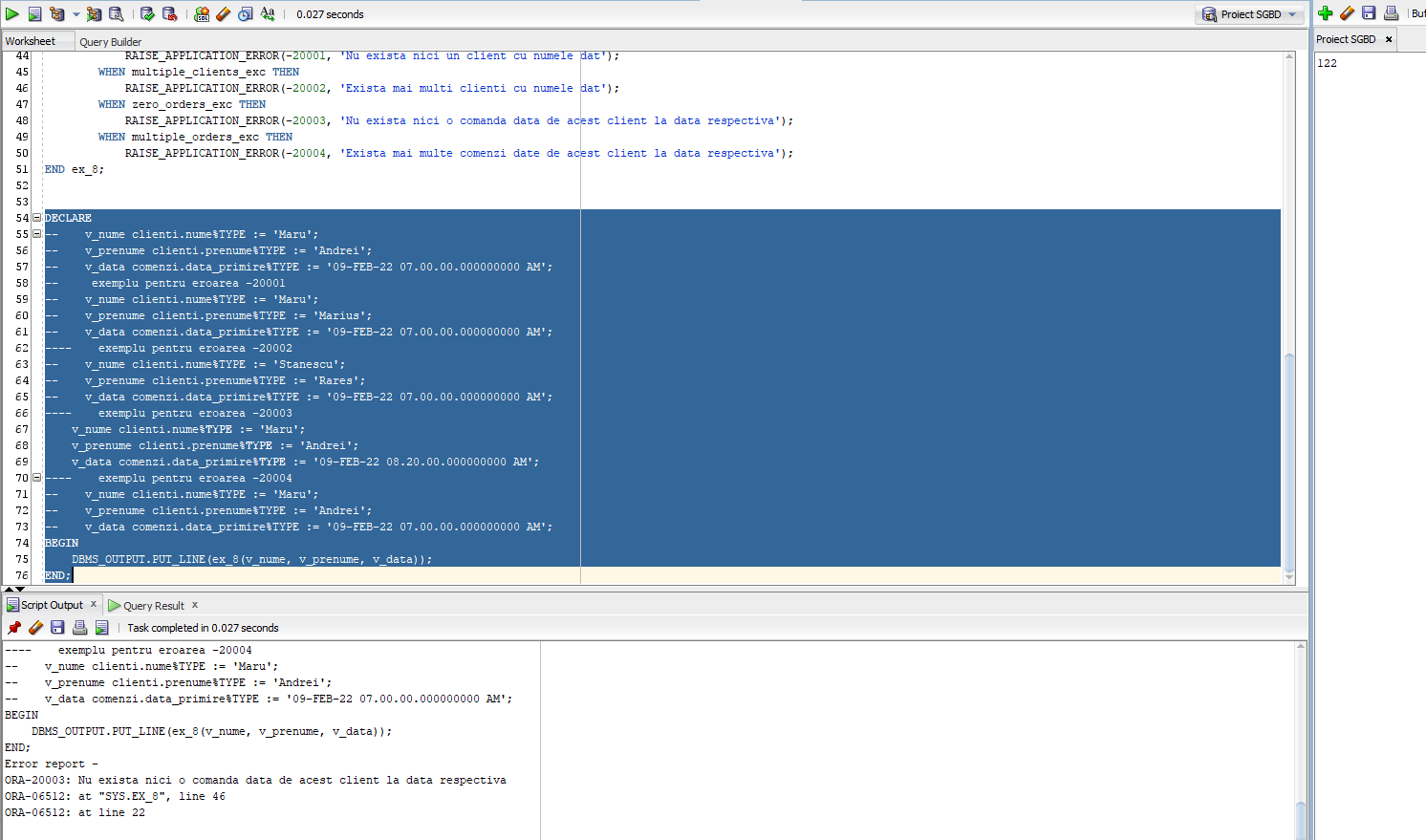
Exemplu pentru cazul în care nu există nici un client cu numele dat:



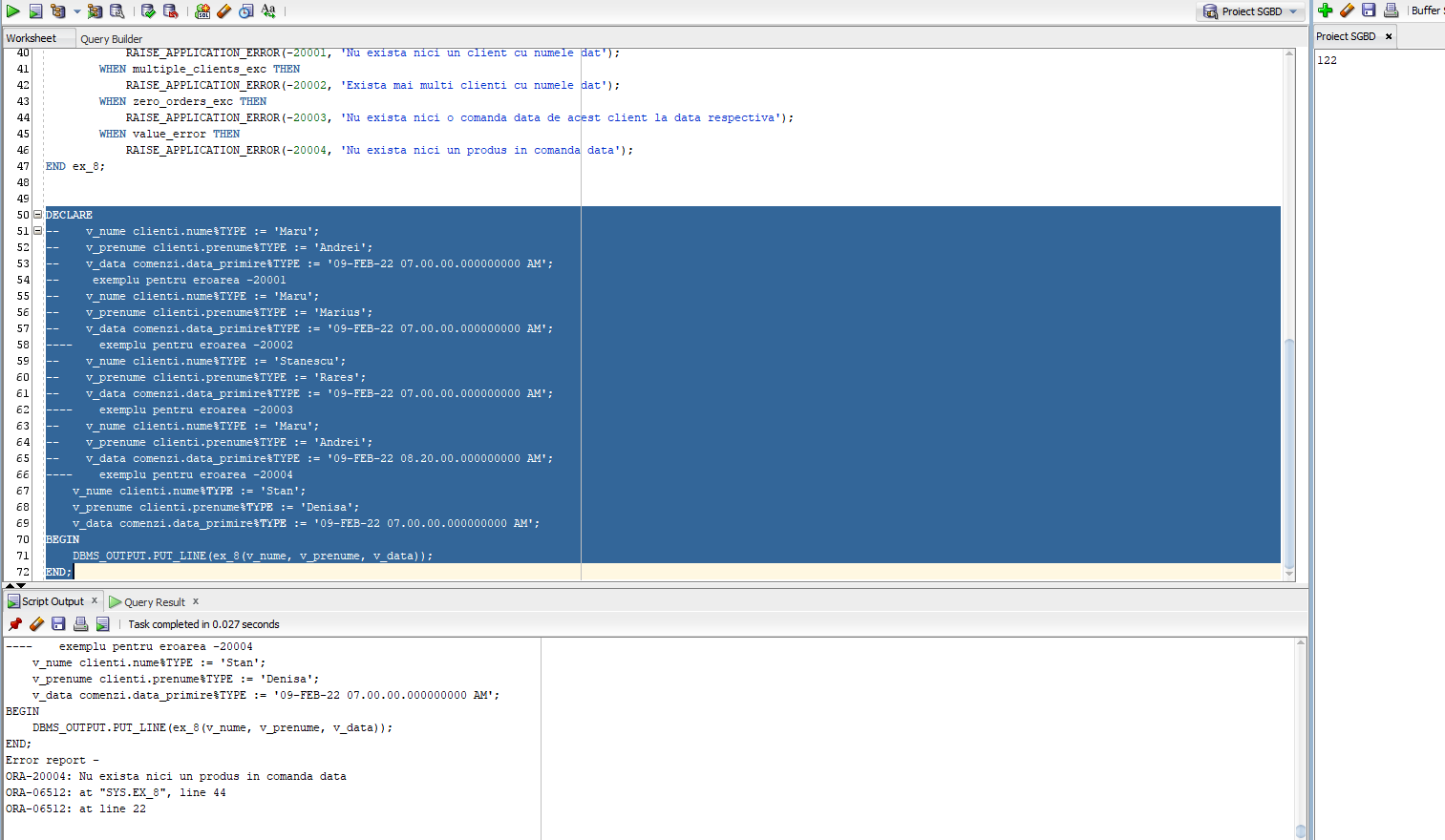
Exemplu pentru cazul în care există mai mulți clienți cu același nume:



Exemplu pentru cazul în care clientul nu a dat nici o comandă la ora specificată:



Exemplu pentru cazul în care nu există nici un produs în comanda respectivă:



**9. Subprogram stocat de tip procedură care să utilizeze într-o singură comandă de tip SQL 5 dintre tabelele definite**

*Definiți o procedură care afișează clientii, angajații și mașinile cu care au fost livrate comenzile ce conțin produsul cu cele mai multe ingrediente. Tratați excepțiile necesare.*

CREATE OR REPLACE PROCEDURE ex\_9

IS

v\_id\_produs produse.id\_produs%TYPE;

v\_nr\_ingrediente NUMBER(3);

v\_check BOOLEAN := false;

v\_nume\_produs produse.nume%TYPE;

CURSOR v\_cursor(c\_id\_produs produse.id\_produs%TYPE)

IS

SELECT clienti.nume clienti\_nume, clienti.prenume clienti\_prenume, angajati.nume angajati\_nume, angajati.prenume angajati\_prenume, nr\_inmatriculare, data\_primire

FROM produse\_comanda

JOIN comenzi ON comenzi.id\_comanda = produse\_comanda.id\_comanda

JOIN clienti ON clienti.id\_client = comenzi.id\_client

JOIN angajati ON angajati.id\_angajat = produse\_comanda.id\_angajat

JOIN masini\_livrare ON masini\_livrare.serie\_sasiu = angajati.serie\_sasiu

WHERE produse\_comanda.id\_produs = c\_id\_produs;

BEGIN

SELECT MAX(COUNT(\*)) INTO v\_nr\_ingrediente FROM contine GROUP BY id\_produs;

SELECT id\_produs INTO v\_id\_produs FROM (SELECT id\_produs, COUNT(\*) as nr FROM contine GROUP BY id\_produs) c WHERE nr = v\_nr\_ingrediente;

SELECT nume INTO v\_nume\_produs FROM produse WHERE id\_produs = v\_id\_produs;

FOR i IN v\_cursor(v\_id\_produs) LOOP

v\_check := true;

DBMS\_OUTPUT.PUT\_LINE('Clientul ' || i.clienti\_nume || ' ' || i.clienti\_prenume || ' a comandat produsul ' || v\_nume\_produs || ' pe ' || TO\_CHAR(i.data\_primire, 'YYYY-MM-DD HH24:MI:SS') || ' si a fost livrata de ' || i.angajati\_nume || ' ' || i.angajati\_prenume || ' cu masina ' || i.nr\_inmatriculare);

END LOOP;

IF v\_check = false THEN

DBMS\_OUTPUT.PUT\_LINE('Nu a comandat inca nimeni produsul ' || v\_nume\_produs);

END IF;

EXCEPTION

WHEN too\_many\_rows THEN

RAISE\_APPLICATION\_ERROR(-20060, 'Exista mai multe produse ce au numar maxim de ingrediente');

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20061, 'Nu exista nici un produs');

WHEN others THEN

RAISE\_APPLICATION\_ERROR(-20062, 'A aparut alta eroare');

END ex\_9;

INSERT INTO produse\_comanda VALUES(3, 1, 6, 1); -- am adaugat produsul Paste Carbonara la comanda 1.

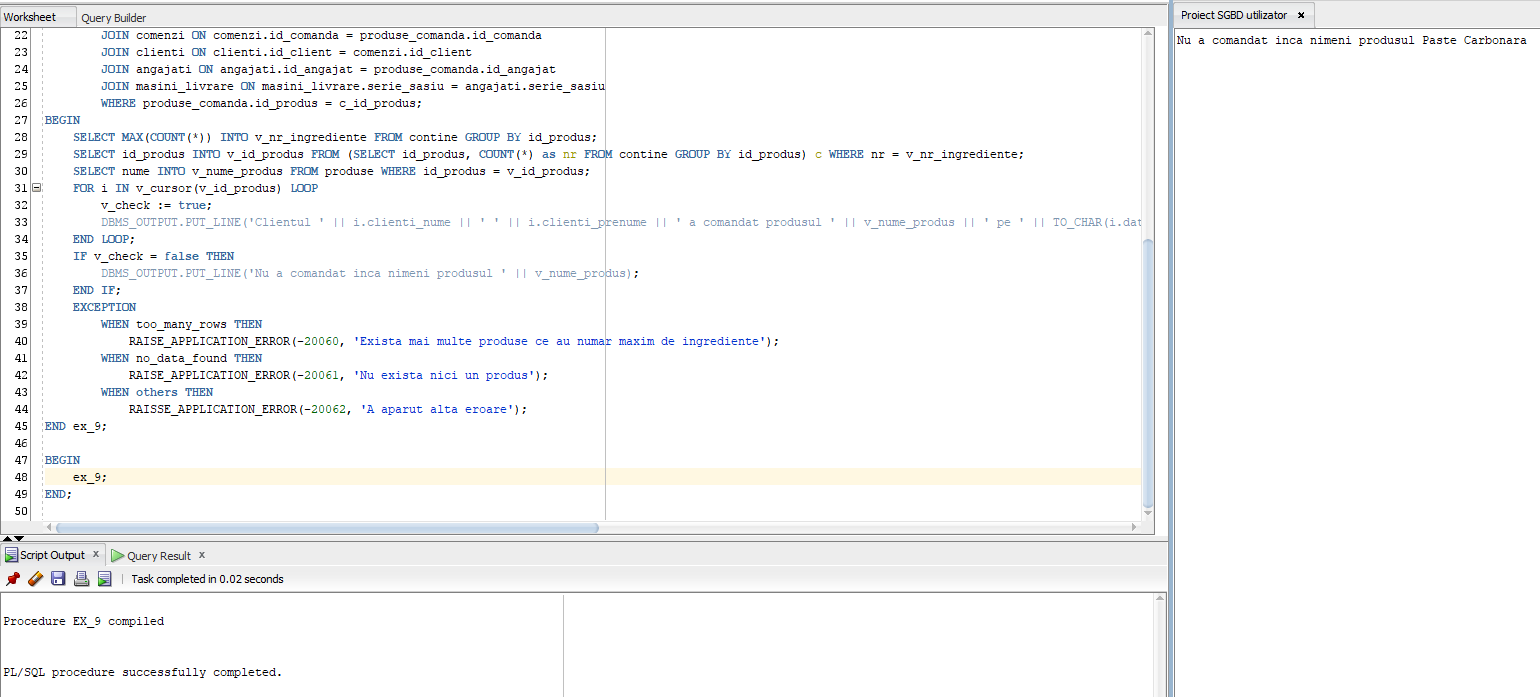
INSERT INTO contine VALUES(5, 5, 0.07); -- am adaugat ingredientul mozzarella la produsul Crispy Strips pentru a avea si acesta 3 ingrediente

BEGIN

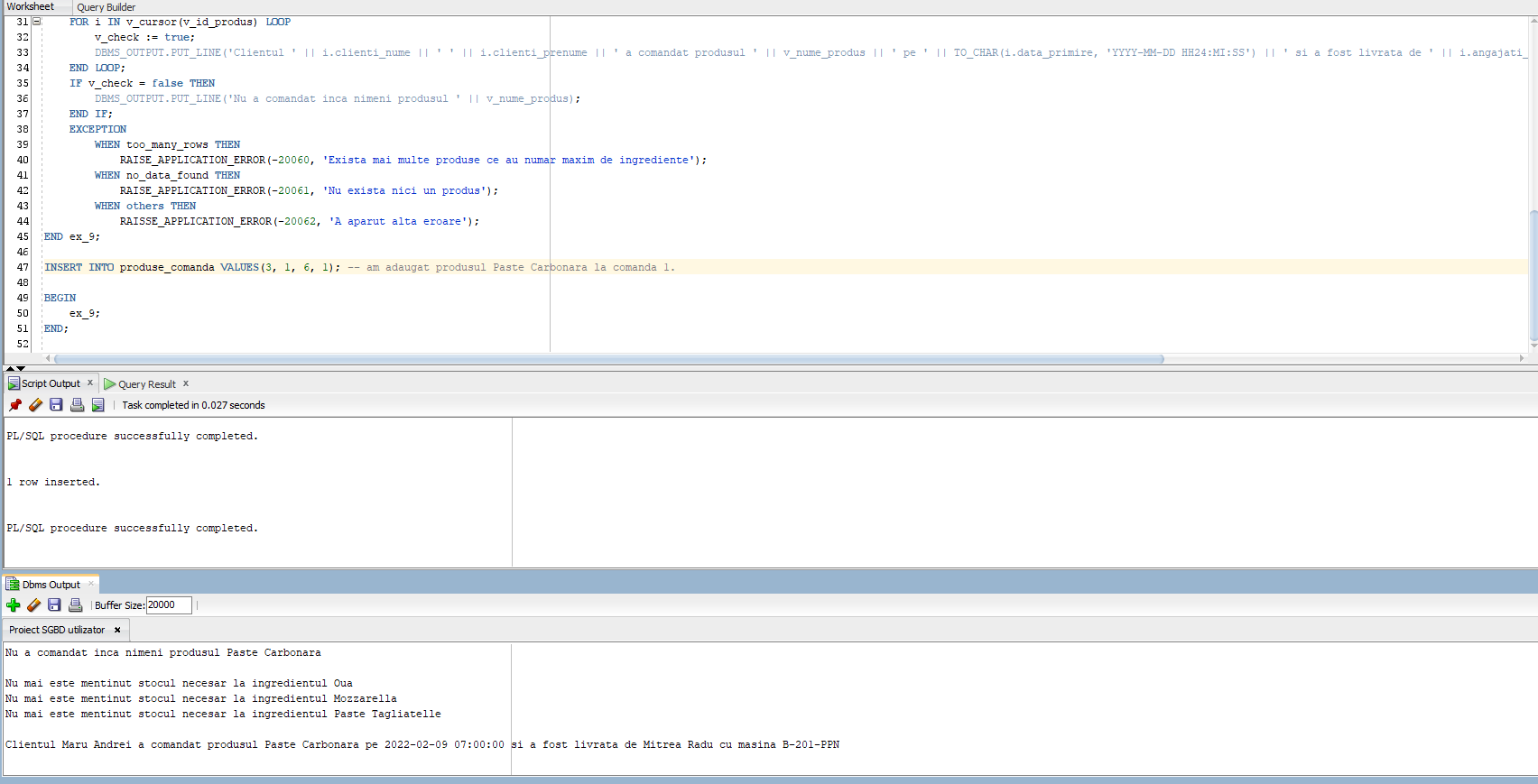
ex\_9;

END;

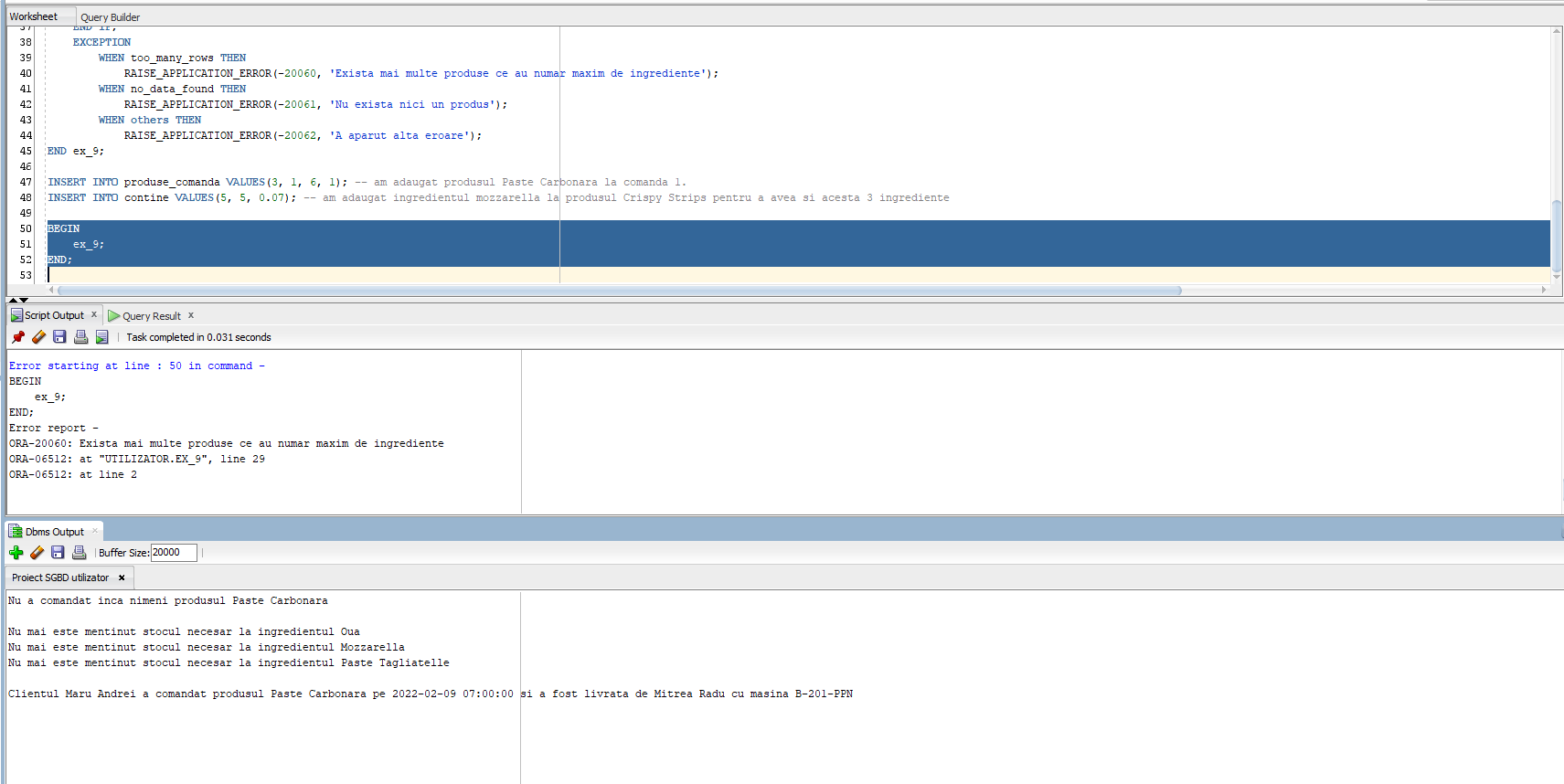
Exemplu de situație în care produsul cu cele mai multe ingrediente nu a fost comandat:

****

Exemplu de situație când produsul cu cele mai multe ingrediente a fost comandat:

**

Exemplu de situație când sunt mai multe produse cu număr maxim de ingrediente:



**10. Definiți un trigger de tip LMD la nivel de comandă**

*Definiți un trigger care nu permite adăugarea, ștergerea sau modificarea comenzilor în afara programului de funcționare a restaurantului. Programul de funcționare: L-V: 08-23, S-D: 09-23.*

CREATE OR REPLACE TRIGGER ex\_10

BEFORE INSERT OR UPDATE OR DELETE ON comenzi

BEGIN

IF ((TO\_CHAR(SYSDATE, 'DAY') = 'SATURDAY' OR TO\_CHAR(SYSDATE, 'DAY') = 'SUNDAY')

AND

(TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24')) >= 23 OR TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24')) < 9))

OR

((TO\_CHAR(SYSDATE, 'DAY') != 'SATURDAY' AND TO\_CHAR(SYSDATE, 'DAY') != 'SUNDAY')

AND

(TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24')) >= 23 OR TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24')) < 8))

THEN

RAISE\_APPLICATION\_ERROR(-20020, 'Restaurantul este inchis');

END IF;

END ex\_10;

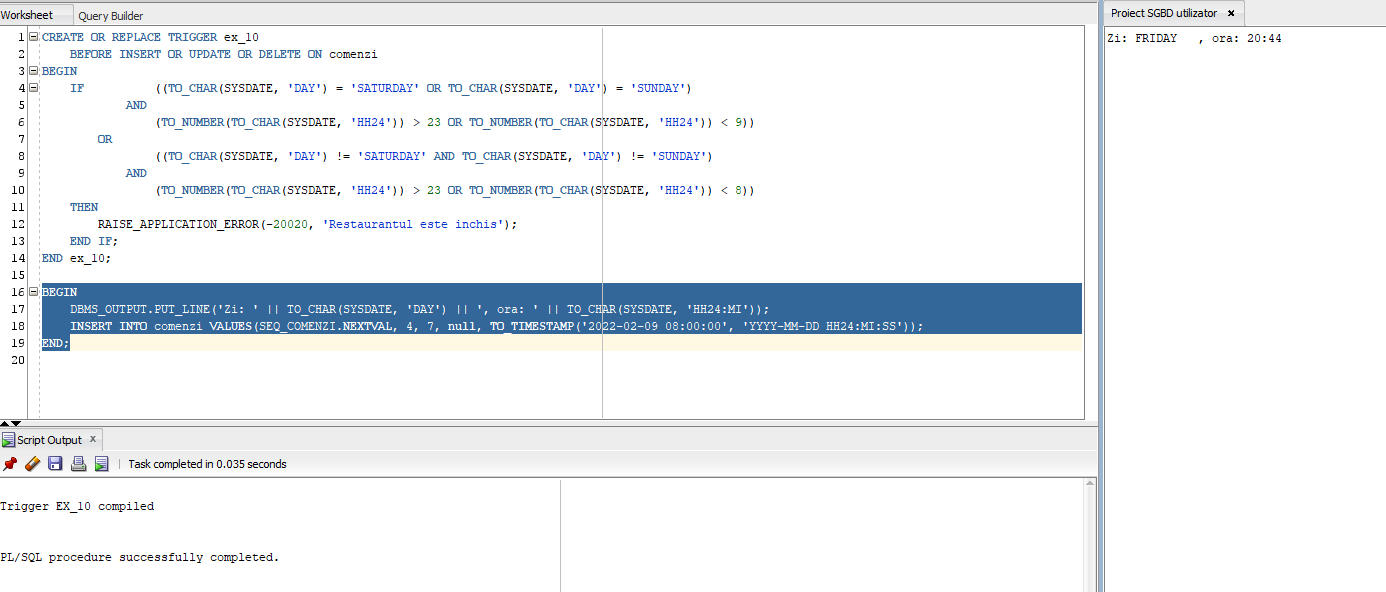
BEGIN

DBMS\_OUTPUT.PUT\_LINE('Zi: ' || TO\_CHAR(SYSDATE, 'DAY') || ', ora: ' || TO\_CHAR(SYSDATE, 'HH24:MI'));

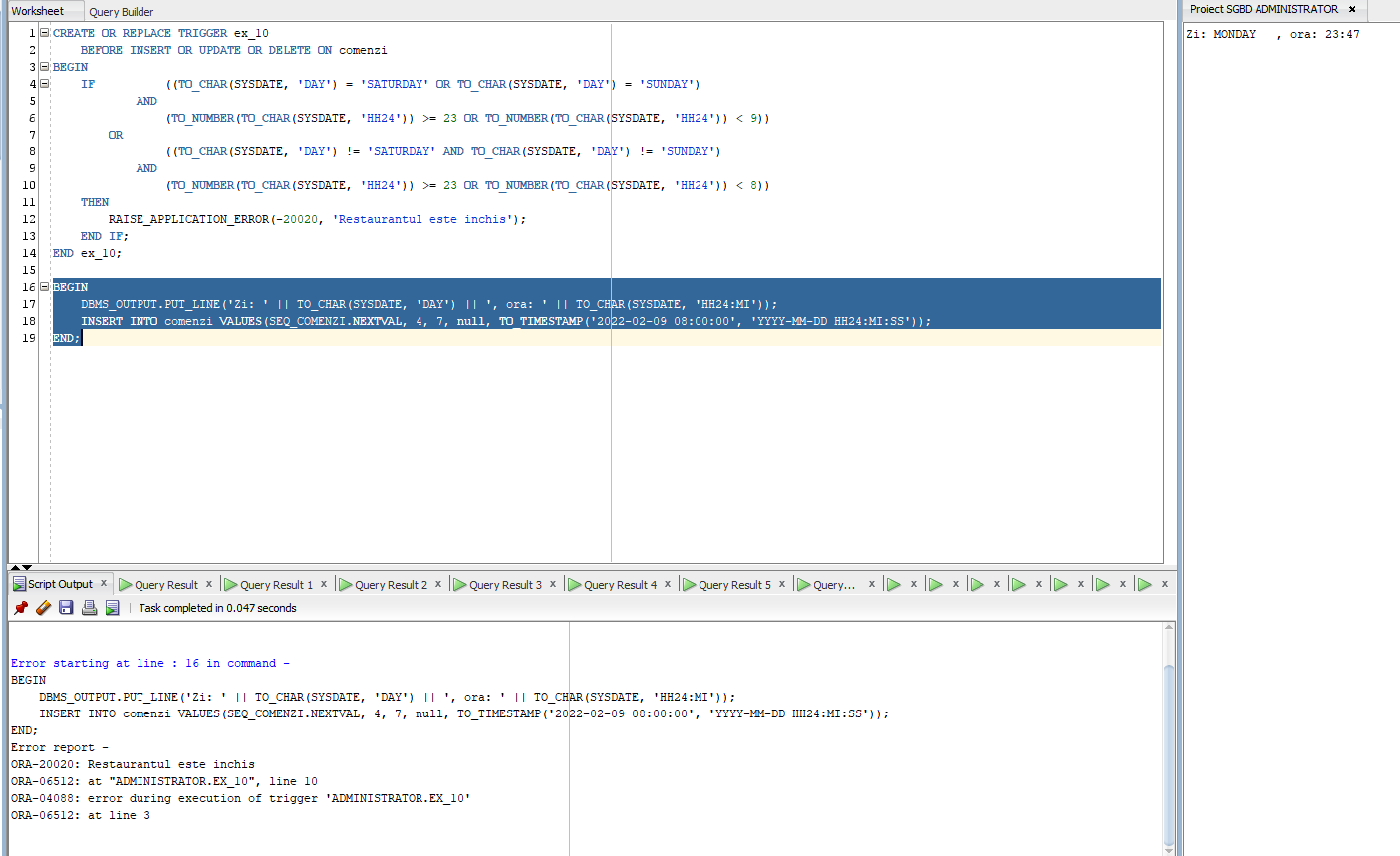
INSERT INTO comenzi VALUES(SEQ\_COMENZI.NEXTVAL, 4, 7, null, TO\_TIMESTAMP('2022-02-09 08:00:00', 'YYYY-MM-DD HH24:MI:SS'));

END;

Exemplu de situație când inserăm în timpul programului de funcționare:



Exemplu de situație când inserăm în afara programului de funcționare:



**11. Definiți un trigger de tip LMD la nivel de linie**

*Definiți un trigger care verifică la introducerea unui nou produs la o comandă dacă este suficient stoc de ingrediente pentru a crea produsul. Dacă nu sunt destule ingrediente se va genera o eroare corespunzătoare, iar daca stocul dupa actualizare va fi mai mic decât stocul necesar al unui ingredient se va afișa un mesaj corespunzator.*

CREATE OR REPLACE TRIGGER ex\_11

BEFORE INSERT OR UPDATE

ON produse\_comanda

FOR EACH ROW

DECLARE

v\_nume\_produs produse.nume%TYPE;

CURSOR ingredients (id\_prod produse.id\_produs%TYPE)

IS

SELECT \* FROM contine c

JOIN ingrediente i ON c.id\_ingredient = i.id\_ingredient

WHERE id\_produs = id\_prod;

BEGIN

SELECT nume INTO v\_nume\_produs FROM produse WHERE id\_produs = :NEW.id\_produs;

FOR i IN ingredients(:NEW.id\_produs) LOOP

IF :NEW.cantitate \* i.cantitate > i.stoc THEN

RAISE\_APPLICATION\_ERROR(-20030, 'Nu exista suficient stoc de ' || i.nume || ' pentru a produce produsul ' || v\_nume\_produs);

ELSIF i.stoc - :NEW.cantitate \* i.cantitate < i.stoc\_necesar THEN

DBMS\_OUTPUT.PUT\_LINE('Nu mai este mentinut stocul necesar la ingredientul ' || i.nume);

END IF;

END LOOP;

END ex\_11;

BEGIN

-- inserare la care suntem avertizati despre ingredientul mozzarella ca nu are stoc necesar, insa sunca are stoc necesar

INSERT INTO produse\_comanda VALUES (2, 8, 2, 1);

-- inserare la care suntem avertizati ca nu mai este mentinut stocul necesar nici la sunca nici la mozzarella

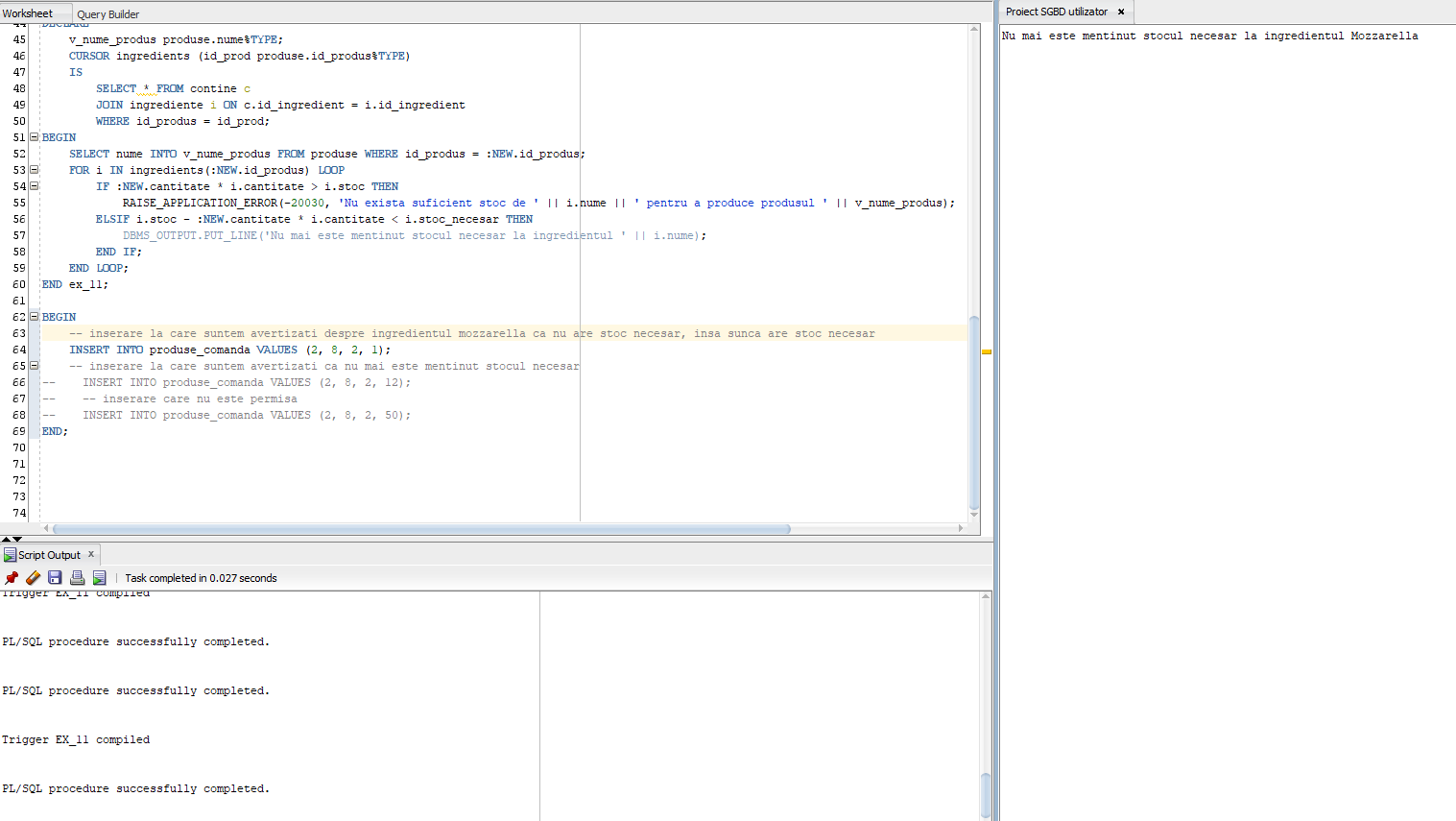
-- INSERT INTO produse\_comanda VALUES (2, 8, 2, 12);

-- -- inserare care nu este permisa

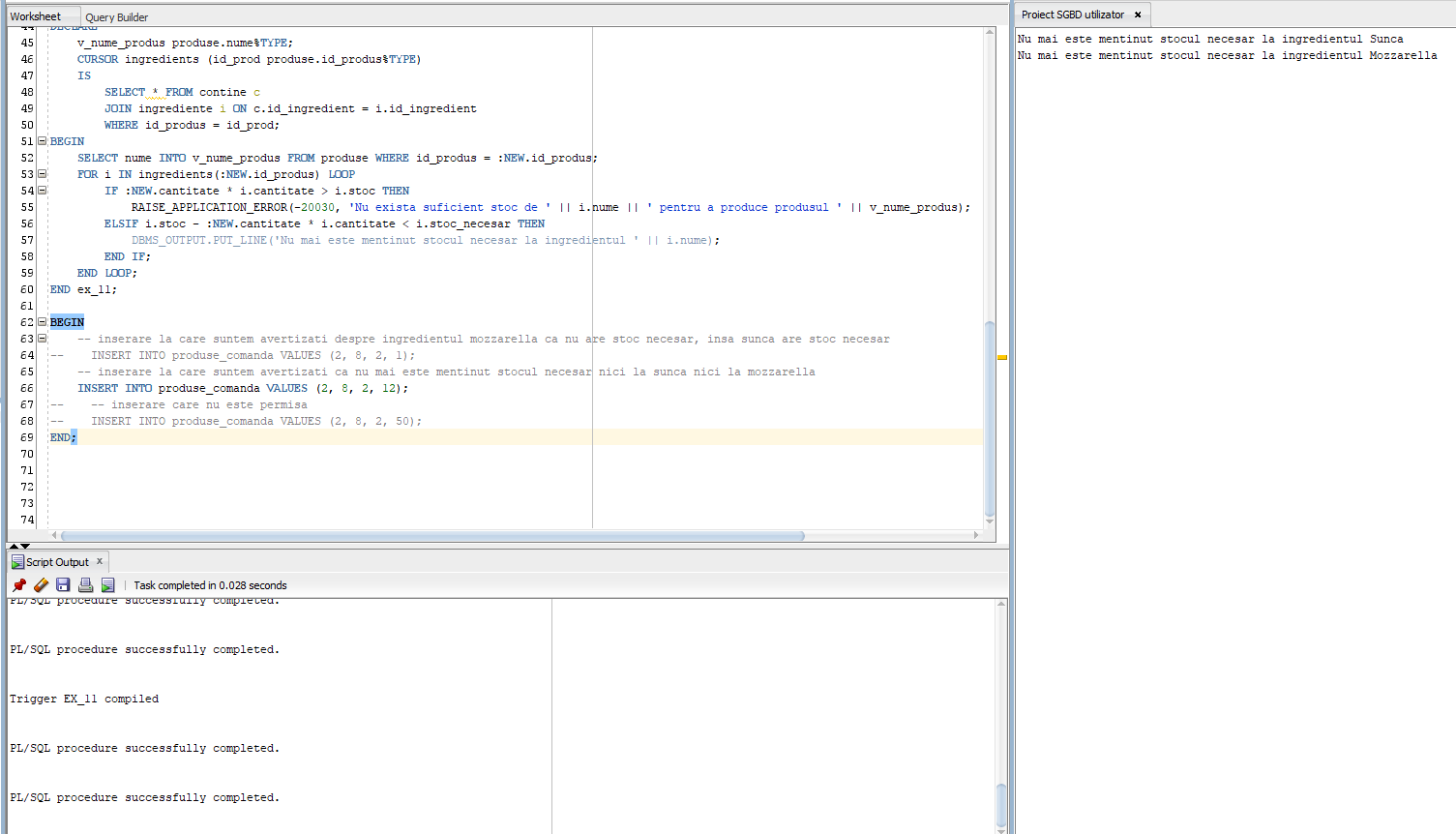
-- INSERT INTO produse\_comanda VALUES (2, 8, 2, 50);

END;

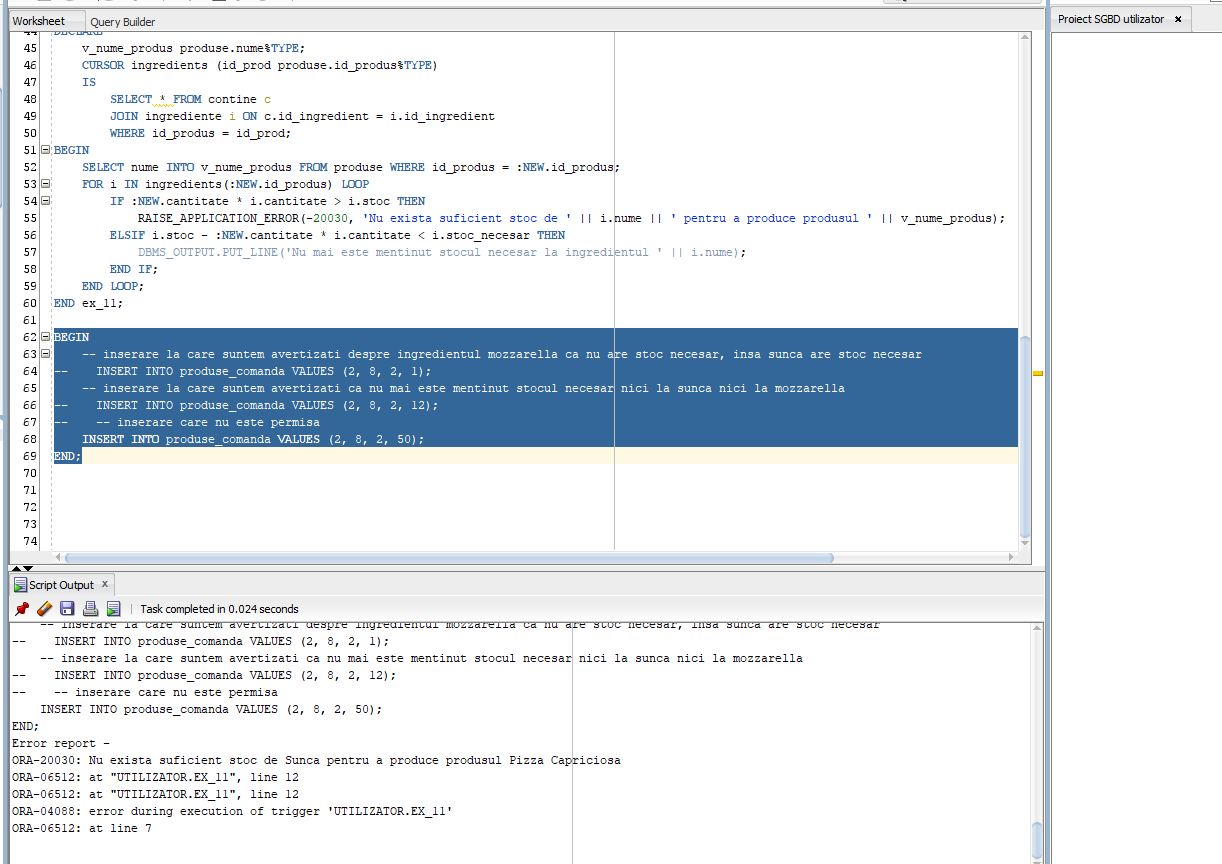
Exemplu la care suntem avertizați asupra ingredientului mozzarella:



Exemplu la care suntem avertizați asupra a doua ingrediente:



Exemplu la care ni se întoarce o eroare:



**12. Definiți un trigger de tip LDD**

*Definiți un trigger care nu permite modificarea schemei bazei de date decât de către utilizatorul ADMINISTRATOR în afara programului de funcționare a restaurantului.*

CREATE OR REPLACE TRIGGER ex\_12

BEFORE CREATE OR DROP OR ALTER ON SCHEMA

BEGIN

IF UPPER(USER) != 'ADMINISTRATOR' THEN

RAISE\_APPLICATION\_ERROR(-20040, 'Doar administratorul poate modifica schema bazei de date');

END IF;

IF TO\_CHAR(SYSDATE, 'DAY') = 'SUNDAY' OR TO\_CHAR(SYSDATE, 'DAY') = 'SATURDAY' THEN

IF TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24')) < 23 AND TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24')) >= 9 THEN

RAISE\_APPLICATION\_ERROR(-20041, 'Doar in afara programului de functionare a restaurantului puteti efectua modificari asupra schemei bazei de date');

END IF;

ELSE

IF TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24')) < 23 AND TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24')) >= 8 THEN

RAISE\_APPLICATION\_ERROR(-20041, 'Doar in afara programului de functionare a restaurantului puteti efectua modificari asupra schemei bazei de date');

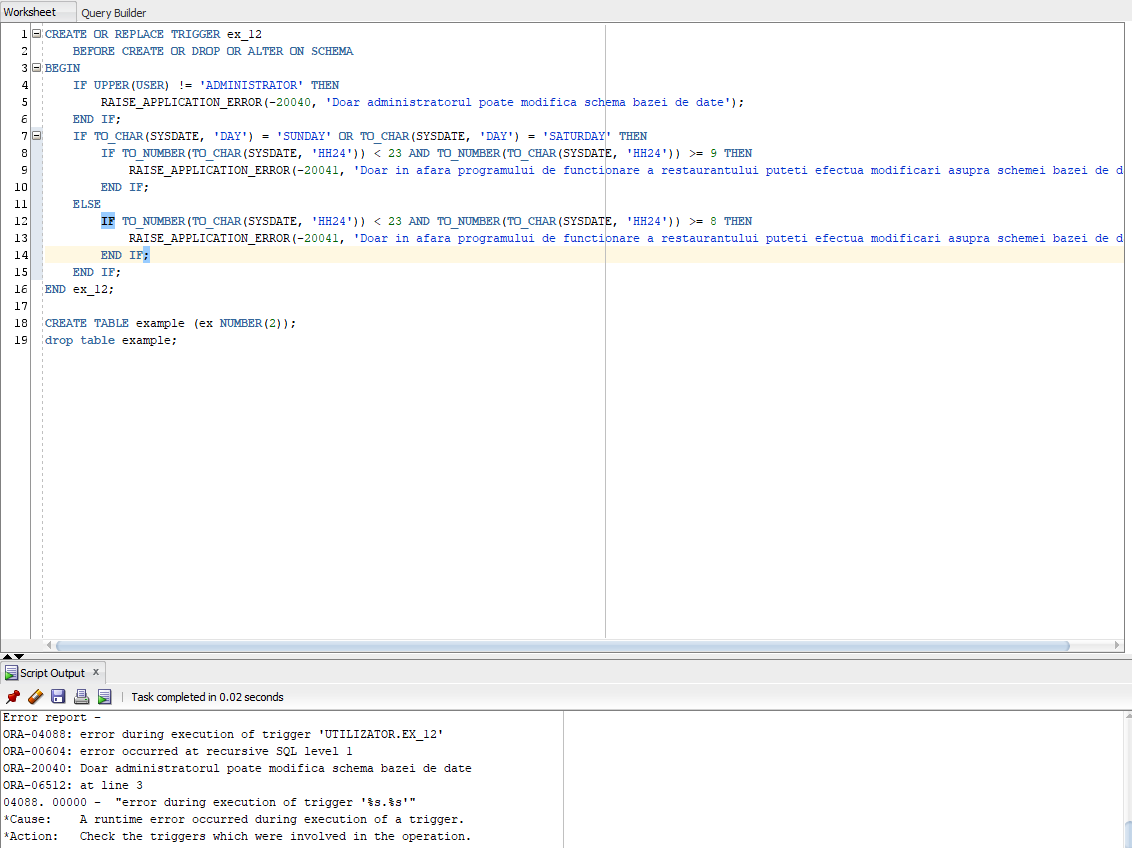
END IF;

END IF;

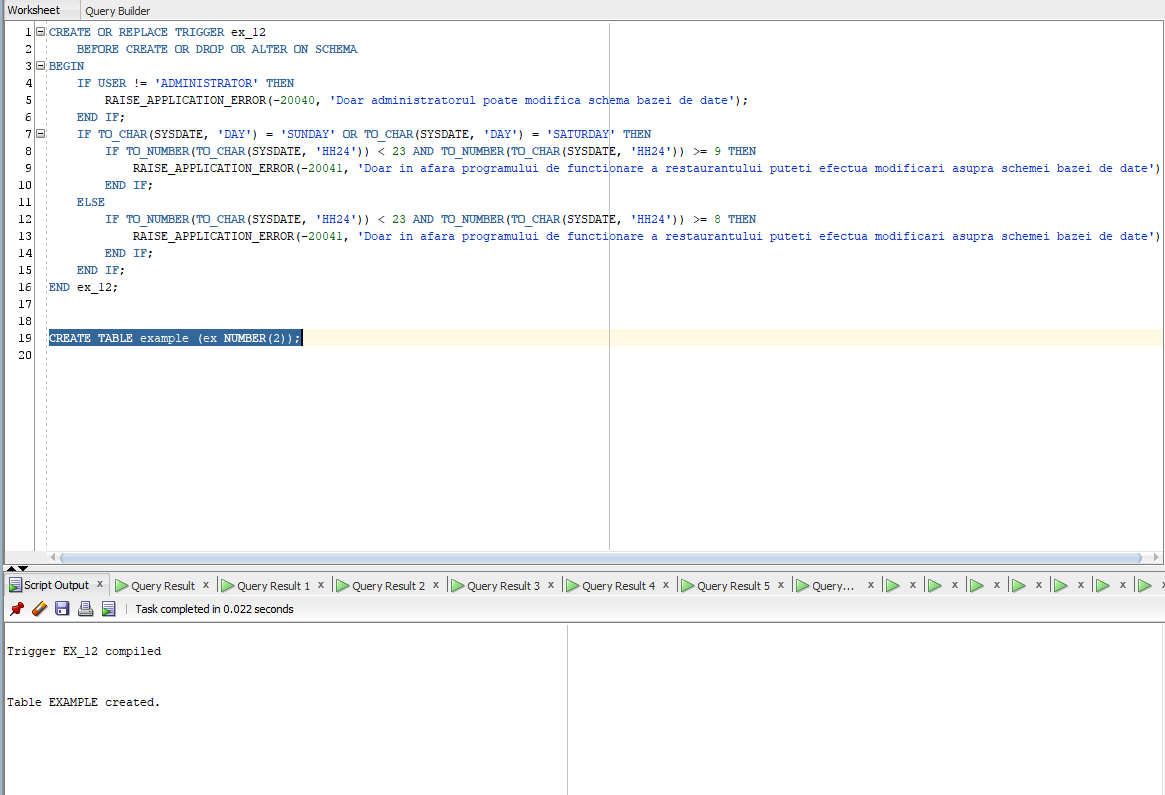
END ex\_12;

CREATE TABLE example (ex NUMBER(2));

Exemplu de situație în care ni se întoarce o eroare, deoarece nu suntem ADMIN:



Exemplu de situație în care putem modifica schema bazei de date:



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

CREATE OR REPLACE PACKAGE pachet\_ex\_13 AS

PROCEDURE ex\_6;

PROCEDURE ex\_7

(p\_nume clienti.nume%TYPE,

p\_prenume clienti.prenume%TYPE);

FUNCTION ex\_8 (f\_nume clienti.nume%TYPE,

f\_prenume clienti.prenume%TYPE,

f\_data comenzi.data\_primire%TYPE)

RETURN NUMBER;

PROCEDURE ex\_9;

END pachet\_ex\_13;

CREATE OR REPLACE PACKAGE BODY pachet\_ex\_13 AS

PROCEDURE ex\_6

IS

TYPE produs IS RECORD (id\_produs produse.id\_produs%TYPE, nume produse.nume%TYPE);

TYPE vec\_categorii IS TABLE OF categorii%ROWTYPE INDEX BY PLS\_INTEGER;

TYPE vec\_produse IS TABLE OF produs;

TYPE vec\_contine IS VARRAY(20) OF (ingrediente.id\_ingredient%TYPE);

v\_produse vec\_produse := vec\_produse();

v\_categorii vec\_categorii;

v\_contine vec\_contine := vec\_contine();

j INTEGER;

v\_nume ingrediente.nume%TYPE;

a NUMBER(4);

BEGIN

SELECT \* BULK COLLECT INTO v\_categorii FROM categorii;

FOR i IN v\_categorii.FIRST..v\_categorii.LAST LOOP

SELECT id\_produs, nume BULK COLLECT INTO v\_produse FROM produse WHERE id\_categorie = v\_categorii(i).id\_categorie;

DBMS\_OUTPUT.PUT\_LINE('Categoria ' || v\_categorii(i).nume);

j := v\_produse.FIRST;

WHILE j <= v\_produse.LAST LOOP

DBMS\_OUTPUT.PUT(v\_produse(j).nume || ' : ');

SELECT COUNT(\*) INTO a FROM contine WHERE id\_produs = v\_produse(j).id\_produs;

IF a > 0 THEN

a := a - 1;

SELECT id\_ingredient BULK COLLECT INTO v\_contine FROM contine WHERE v\_produse(j).id\_produs = id\_produs;

FOR k in v\_contine.FIRST..v\_contine.LAST LOOP

SELECT nume INTO v\_nume FROM ingrediente WHERE id\_ingredient = v\_contine(k);

DBMS\_OUTPUT.PUT(v\_nume);

IF a > 0 THEN

DBMS\_OUTPUT.PUT(', ');

a := a - 1;

END IF;

END LOOP;

ELSE

DBMS\_OUTPUT.PUT('Nu exista inca ingrediente introduse pentru acest produs');

END IF;

j := v\_produse.NEXT(j);

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END ex\_6;

PROCEDURE ex\_7

(p\_nume clienti.nume%TYPE,

p\_prenume clienti.prenume%TYPE)

IS

g\_id\_comanda comenzi.id\_comanda%TYPE;

g\_data\_primire comenzi.data\_primire%TYPE;

g\_nume\_produs produse.nume%TYPE;

g\_index NUMBER(4) := 1;

g\_comma BOOLEAN;

CURSOR c\_comenzi\_client

IS

SELECT id\_comanda, data\_primire FROM comenzi WHERE id\_client = (SELECT id\_client

FROM clienti

WHERE nume = p\_nume AND prenume = p\_prenume);

CURSOR c\_produse\_comanda (v\_id\_comanda comenzi.id\_comanda%TYPE)

IS

SELECT id\_produs FROM produse\_comanda WHERE id\_comanda = v\_id\_comanda;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Clientul cu numele ' || p\_nume || ' ' || p\_prenume || ' a dat comenzile:');

--cursor clasic

OPEN c\_comenzi\_client;

LOOP

FETCH c\_comenzi\_client INTO g\_id\_comanda, g\_data\_primire;

EXIT WHEN c\_comenzi\_client%NOTFOUND;

DBMS\_OUTPUT.PUT(g\_index || '. Comanda de la data de ' || TO\_CHAR(g\_data\_primire, 'YYYY-MM-DD HH24:MI:SS') || ' contine: ');

g\_comma := true;

--ciclu cursor cu parametru

FOR i IN c\_produse\_comanda(g\_id\_comanda) LOOP

SELECT nume INTO g\_nume\_produs FROM produse WHERE id\_produs = i.id\_produs;

IF g\_comma THEN

DBMS\_OUTPUT.PUT(g\_nume\_produs);

g\_comma := false;

ELSE

DBMS\_OUTPUT.PUT(', ' || g\_nume\_produs);

END IF;

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

g\_index := g\_index + 1;

END LOOP;

END ex\_7;

FUNCTION ex\_8 (f\_nume clienti.nume%TYPE,

f\_prenume clienti.prenume%TYPE,

f\_data comenzi.data\_primire%TYPE)

RETURN NUMBER

IS

TYPE price\_table\_datatype IS TABLE OF NUMBER(8, 2) INDEX BY PLS\_INTEGER;

price\_table price\_table\_datatype;

pret\_total NUMBER(8,2) := 0;

nr NUMBER(4);

v\_id\_client clienti.id\_client%TYPE;

zero\_clients\_exc EXCEPTION;

multiple\_clients\_exc EXCEPTION;

zero\_orders\_exc EXCEPTION;

BEGIN

SELECT COUNT(\*) INTO nr FROM clienti WHERE nume = f\_nume AND prenume = f\_prenume;

IF nr = 0 THEN

RAISE zero\_clients\_exc;

ELSIF nr > 1 THEN

RAISE multiple\_clients\_exc;

END IF;

SELECT id\_client INTO v\_id\_client FROM clienti WHERE UPPER(nume) = UPPER(f\_nume) AND UPPER(prenume) = UPPER(f\_prenume);

SELECT COUNT(\*) INTO nr FROM comenzi WHERE id\_client = v\_id\_client AND data\_primire = f\_data;

IF nr = 0 THEN

RAISE zero\_orders\_exc;

END IF;

SELECT pc.cantitate \* p.pret BULK COLLECT INTO price\_table FROM produse\_comanda pc

JOIN comenzi c

ON c.id\_comanda = pc.id\_comanda

JOIN produse p

ON p.id\_produs = pc.id\_produs

WHERE c.id\_client = v\_id\_client AND c.data\_primire = f\_data;

FOR i IN price\_table.FIRST..price\_table.LAST LOOP

pret\_total := pret\_total + price\_table(i);

END LOOP;

RETURN NVL(pret\_total, 0);

EXCEPTION

WHEN zero\_clients\_exc THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Nu exista nici un client cu numele dat');

WHEN multiple\_clients\_exc THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Exista mai multi clienti cu numele dat');

WHEN zero\_orders\_exc THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Nu exista nici o comanda data de acest client la data respectiva');

WHEN value\_error THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Nu exista nici un produs in comanda data');

END ex\_8;

PROCEDURE ex\_9

IS

v\_id\_produs produse.id\_produs%TYPE;

v\_nr\_ingrediente NUMBER(3);

v\_check BOOLEAN := false;

v\_nume\_produs produse.nume%TYPE;

CURSOR v\_cursor(c\_id\_produs produse.id\_produs%TYPE)

IS

SELECT clienti.nume clienti\_nume, clienti.prenume clienti\_prenume, angajati.nume angajati\_nume, angajati.prenume angajati\_prenume, nr\_inmatriculare, data\_primire

FROM produse\_comanda

JOIN comenzi ON comenzi.id\_comanda = produse\_comanda.id\_comanda

JOIN clienti ON clienti.id\_client = comenzi.id\_client

JOIN angajati ON angajati.id\_angajat = produse\_comanda.id\_angajat

JOIN masini\_livrare ON masini\_livrare.serie\_sasiu = angajati.serie\_sasiu

WHERE produse\_comanda.id\_produs = c\_id\_produs;

BEGIN

SELECT MAX(COUNT(\*)) INTO v\_nr\_ingrediente FROM contine GROUP BY id\_produs;

SELECT id\_produs INTO v\_id\_produs FROM (SELECT id\_produs, COUNT(\*) as nr FROM contine GROUP BY id\_produs) c WHERE nr = v\_nr\_ingrediente;

SELECT nume INTO v\_nume\_produs FROM produse WHERE id\_produs = v\_id\_produs;

FOR i IN v\_cursor(v\_id\_produs) LOOP

v\_check := true;

DBMS\_OUTPUT.PUT\_LINE('Clientul ' || i.clienti\_nume || ' ' || i.clienti\_prenume || ' a comandat produsul ' || v\_nume\_produs || ' pe ' || TO\_CHAR(i.data\_primire, 'YYYY-MM-DD HH24:MI:SS') || ' si a fost livrata de ' || i.angajati\_nume || ' ' || i.angajati\_prenume || ' cu masina ' || i.nr\_inmatriculare);

END LOOP;

IF v\_check = false THEN

DBMS\_OUTPUT.PUT\_LINE('Nu a comandat inca nimeni produsul ' || v\_nume\_produs);

END IF;

EXCEPTION

WHEN too\_many\_rows THEN

RAISE\_APPLICATION\_ERROR(-20060, 'Exista mai multe produse ce au numar maxim de ingrediente');

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20061, 'Nu exista nici un produs');

WHEN others THEN

RAISE\_APPLICATION\_ERROR(-20062, 'A aparut alta eroare');

END ex\_9;

END pachet\_ex\_13;

DECLARE

v\_nume clienti.nume%TYPE := 'Maru';

v\_prenume clienti.prenume%TYPE :='Andrei';

v\_data comenzi.data\_primire%TYPE := '09-FEB-22 07.00.00.000000000 AM';

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Exercitiul 6:');

pachet\_ex\_13.ex\_6;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Exercitiul 7:');

pachet\_ex\_13.ex\_7(v\_nume, v\_prenume);

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Exercitiul 8:');

DBMS\_OUTPUT.PUT\_LINE('Valoarea comenzii este de ' || pachet\_ex\_13.ex\_8(v\_nume, v\_prenume, v\_data) || 'RON');

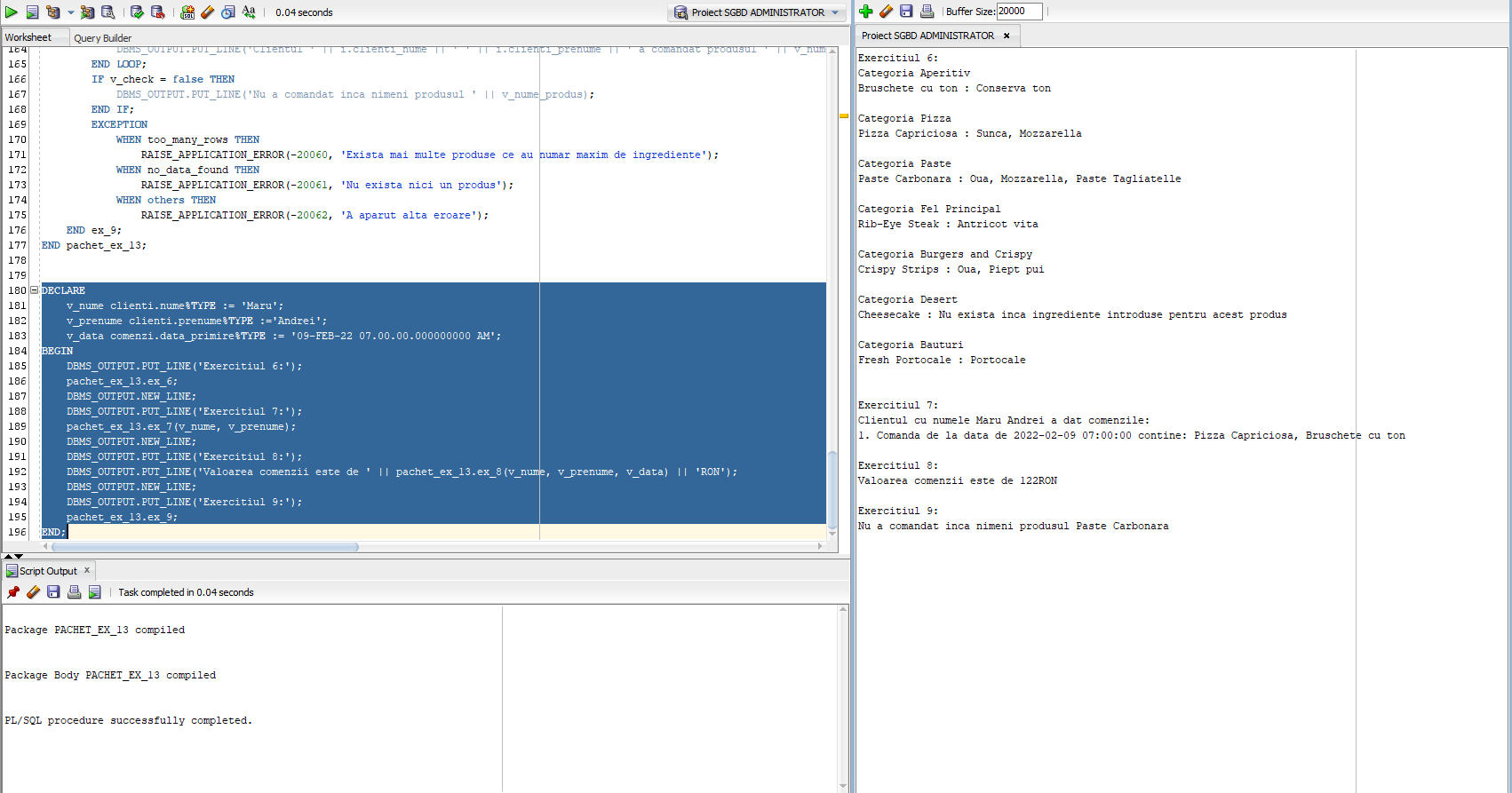
DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Exercitiul 9:');

pachet\_ex\_13.ex\_9;

END;

Exemplu rulare subprograme din pachet:



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

În acest pachet se vor regasi subprograme pentru:

* Afișarea a top N clienți care au comandat cel mai mult dintr-un an (N și anul sunt dați ca parametri)
* Afișarea a top N produse comandate cel mai mult dintr-un an (N și anul sunt dați ca parametri)
* O funcție care returnează încasările dintr-un an (anul este dat ca parametru)
* Afișarea a top N furnizori de la care restaurantul s-a aprovizionat cel mai des într-un anumit an (N și anul sunt dați ca parametri)
* O funcție care returnează anul cu cele mai mari încasări și veniturile din acel an
* O funcție care returnează o lista cu cantitatea primită de la furnizori pentru fiecare ingredient și costurile asociate într-un an (anul este dat ca parametru)

Acest pachet va include și tratarea excepțiilor necesare.

CREATE OR REPLACE PACKAGE pachet\_ex\_14 AS

TYPE best\_year\_structure IS RECORD

(g\_year VARCHAR2(4),

g\_revenue NUMBER(8, 2));

TYPE ingredients\_providers\_structure IS RECORD

(g\_id\_ingredient ingrediente.id\_ingredient%TYPE,

g\_nume ingrediente.nume%TYPE,

g\_cantitate NUMBER(8, 2),

g\_cost NUMBER(8, 2));

TYPE ingredients\_providers\_vector IS TABLE OF ingredients\_providers\_structure;

PROCEDURE top\_clients

(l\_n NUMBER,

l\_an VARCHAR2);

PROCEDURE top\_products

(l\_n NUMBER,

l\_an VARCHAR2);

FUNCTION get\_income

(l\_an VARCHAR2)

RETURN NUMBER;

PROCEDURE top\_providers

(l\_n NUMBER,

l\_an VARCHAR2);

FUNCTION best\_year RETURN best\_year\_structure;

FUNCTION ingredients\_providers

(l\_an VARCHAR2)

RETURN ingredients\_providers\_vector;

END pachet\_ex\_14;

CREATE OR REPLACE PACKAGE BODY pachet\_ex\_14 AS

-- Afișarea a top N clienți care au comandat cel mai mult dintr-un an (N și anul sunt dați ca parametri)

PROCEDURE top\_clients

(l\_n NUMBER,

l\_an VARCHAR2)

IS

TYPE record\_clienti IS RECORD

(nume clienti.nume%TYPE,

prenume clienti.prenume%TYPE,

numar\_comenzi NUMBER(8));

TYPE vector\_clienti IS TABLE OF record\_clienti;

top\_clienti vector\_clienti;

nr NUMBER(8);

not\_enough EXCEPTION;

BEGIN

SELECT COUNT(\*) INTO nr FROM

(SELECT DISTINCT(nr) FROM

(SELECT COUNT(\*) nr

FROM comenzi

WHERE TO\_CHAR(data\_primire, 'YYYY') = l\_an AND id\_client IS NOT NULL

GROUP BY id\_client

ORDER BY nr DESC)

WHERE ROWNUM <= l\_n);

IF nr != l\_n THEN

RAISE not\_enough;

END IF;

SELECT \*

BULK COLLECT INTO top\_clienti

FROM

(SELECT clienti.nume, clienti.prenume, c.nr

FROM clienti

JOIN

(SELECT id\_client, COUNT(\*) nr

FROM comenzi

WHERE TO\_CHAR(data\_primire, 'YYYY') = l\_an AND id\_client IS NOT NULL

GROUP BY id\_client) c

ON c.id\_client = clienti.id\_client)

WHERE nr IN

(SELECT DISTINCT(nr) FROM

(SELECT COUNT(\*) nr

FROM comenzi

WHERE TO\_CHAR(data\_primire, 'YYYY') = l\_an AND id\_client IS NOT NULL

GROUP BY id\_client

ORDER BY nr DESC)

WHERE ROWNUM <= l\_n);

FOR i IN top\_clienti.FIRST..top\_clienti.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(top\_clienti(i).nume || ' ' || top\_clienti(i).prenume || ' ' || top\_clienti(i).numar\_comenzi);

END LOOP;

EXCEPTION

WHEN not\_enough OR no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20101, 'Nu sunt destui oameni care au dat un numar diferit de comenzi pentru a forma un top-' || l\_n || ', top-ul maxim este de ' || nr);

END top\_clients;

--Afișarea a top N produse comandate cel mai mult dintr-un an (N este dat ca parametru)

PROCEDURE top\_products

(l\_n NUMBER,

l\_an VARCHAR2)

IS

TYPE record\_produse IS RECORD

(nume produse.nume%TYPE,

cantitate\_totala\_comandata NUMBER(8));

TYPE vector\_produse IS TABLE OF record\_produse;

top\_produse vector\_produse;

nr NUMBER(8);

not\_enough EXCEPTION;

BEGIN

SELECT COUNT(\*) INTO nr FROM

(SELECT quantity

FROM

(SELECT DISTINCT(SUM(cantitate)) quantity

FROM

(SELECT produse\_comanda.id\_produs, cantitate

FROM produse\_comanda

LEFT JOIN comenzi

ON comenzi.id\_comanda = produse\_comanda.id\_comanda

WHERE TO\_CHAR(data\_primire, 'YYYY') = l\_an)

GROUP BY id\_produs

ORDER BY quantity DESC)

WHERE ROWNUM <= l\_n);

IF nr != l\_n THEN

RAISE not\_enough;

END IF;

SELECT \* BULK COLLECT INTO top\_produse

FROM

(SELECT MAX(produse.nume), SUM(c.cantitate) nr\_comenzi

FROM

(SELECT cantitate, produse\_comanda.id\_produs

FROM produse\_comanda

LEFT JOIN comenzi

ON produse\_comanda.id\_comanda = comenzi.id\_comanda

WHERE TO\_CHAR(comenzi.data\_primire, 'YYYY') = l\_an) c

JOIN produse

ON produse.id\_produs = c.id\_produs

GROUP BY produse.id\_produs

ORDER BY nr\_comenzi DESC)

WHERE nr\_comenzi IN

(SELECT quantity

FROM

(SELECT DISTINCT(SUM(cantitate)) quantity

FROM

(SELECT produse\_comanda.id\_produs, cantitate

FROM produse\_comanda

LEFT JOIN comenzi

ON comenzi.id\_comanda = produse\_comanda.id\_comanda

WHERE TO\_CHAR(data\_primire, 'YYYY') = l\_an)

GROUP BY id\_produs

ORDER BY quantity DESC)

WHERE ROWNUM <= l\_n);

FOR i IN top\_produse.FIRST..top\_produse.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(top\_produse(i).nume || ' ' || top\_produse(i).cantitate\_totala\_comandata);

END LOOP;

EXCEPTION

WHEN no\_data\_found OR not\_enough THEN

RAISE\_APPLICATION\_ERROR(-20102, 'Nu sunt destule produse care au un numar cantitati totale comandate diferite pentru a forma un top-' || l\_n || ', top-ul maxim este de ' || nr);

END top\_products;

--O funcție care returnează încasările dintr-un an (anul este dat ca parametru)

FUNCTION get\_income

(l\_an VARCHAR2)

RETURN NUMBER

IS

incasari NUMBER(8, 2);

BEGIN

SELECT SUM(valoare) INTO incasari FROM mod\_plata

JOIN comenzi ON comenzi.id\_comanda = mod\_plata.id\_comanda

WHERE TO\_CHAR(data\_primire, 'YYYY') = l\_an;

IF incasari IS NULL THEN

RAISE no\_data\_found;

END IF;

RETURN incasari;

EXCEPTION

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20105, 'Nu exista comenzi in anul ' || l\_an);

END get\_income;

--Afișarea a top N furnizori de la care restaurantul s-a aprovizionat cel mai des intr-un an (N este dat ca parametru)

PROCEDURE top\_providers

(l\_n NUMBER,

l\_an VARCHAR2)

IS

TYPE furnizori\_record IS RECORD

(nume furnizori.nume%TYPE,

nr\_receptii NUMBER(8));

TYPE furnizori\_vector IS TABLE OF furnizori\_record;

top\_furnizori furnizori\_vector;

not\_enough EXCEPTION;

nr NUMBER(8);

BEGIN

SELECT COUNT(\*) INTO nr

FROM

(SELECT \* FROM

(SELECT DISTINCT(nr\_receptii)

FROM

(SELECT COUNT(\*) nr\_receptii FROM receptii

WHERE TO\_CHAR(data\_primire, 'YYYY') = l\_an

GROUP BY id\_furnizor

ORDER BY nr\_receptii DESC))

WHERE ROWNUM <= l\_n);

IF nr!= l\_n THEN

RAISE not\_enough;

END IF;

SELECT nume, nr\_receptii BULK COLLECT INTO top\_furnizori

FROM

(SELECT COUNT(\*) nr\_receptii, id\_furnizor FROM receptii

WHERE TO\_CHAR(data\_primire, 'YYYY') = l\_an

GROUP BY id\_furnizor) c

LEFT JOIN furnizori ON furnizori.id\_furnizor = c.id\_furnizor

WHERE nr\_receptii IN

(SELECT \* FROM

(SELECT DISTINCT(nr\_receptii)

FROM

(SELECT COUNT(\*) nr\_receptii FROM receptii

WHERE TO\_CHAR(data\_primire, 'YYYY') = l\_an

GROUP BY id\_furnizor

ORDER BY nr\_receptii DESC))

WHERE ROWNUM <= l\_n)

ORDER BY nr\_receptii DESC;

FOR i IN top\_furnizori.FIRST..top\_furnizori.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(top\_furnizori(i).nume || ' ' || top\_furnizori(i).nr\_receptii);

END LOOP;

EXCEPTION

WHEN not\_enough OR no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20103, 'Nu exista suficienti furnizori care sa fi trimis un numar total de receptii diferit pentru a forma un top-' || l\_n || ' top-ul maxim este de ' || nr);

END top\_providers;

--O funcție care returnează anul cu cele mai mari încasări și veniturile din acel an

FUNCTION best\_year

RETURN best\_year\_structure

IS

ret\_value best\_year\_structure;

TYPE vector\_ani IS TABLE OF VARCHAR2(4) INDEX BY PLS\_INTEGER;

ani vector\_ani;

nr NUMBER(8, 2);

no\_orders EXCEPTION;

BEGIN

SELECT DISTINCT(TO\_CHAR(data\_primire, 'YYYY')) BULK COLLECT INTO ani FROM comenzi;

IF ani.COUNT < 1 THEN

RAISE no\_orders;

END IF;

ret\_value.g\_revenue := 0;

FOR i IN ani.FIRST..ani.LAST LOOP

nr := get\_income(ani(i));

IF nr > ret\_value.g\_revenue THEN

ret\_value.g\_revenue := nr;

ret\_value.g\_year := ani(i);

END IF;

END LOOP;

RETURN ret\_value;

EXCEPTION

WHEN no\_orders THEN

RAISE\_APPLICATION\_ERROR(-20104, 'Restaurantul nu are inca nici o comanda');

END best\_year;

--O funcție care returnează o lista cu cantitatea primită de la furnizori pentru fiecare ingredient și costurile asociate într-un an (anul este dat ca parametru)

FUNCTION ingredients\_providers

(l\_an VARCHAR2)

RETURN ingredients\_providers\_vector

IS

vec\_ingrediente ingredients\_providers\_vector;

no\_receptions EXCEPTION;

BEGIN

SELECT c.id\_ingredient, ingrediente.nume, c.quantity, c.price BULK COLLECT INTO vec\_ingrediente FROM

(SELECT produse\_receptie.id\_ingredient, SUM(produse\_receptie.pret\_buc \* produse\_receptie.cantitate) price, SUM (produse\_receptie.cantitate) quantity

FROM produse\_receptie

JOIN receptii

ON receptii.id\_receptie = produse\_receptie.id\_receptie

WHERE TO\_CHAR(receptii.data\_primire, 'YYYY') = l\_an

GROUP BY id\_ingredient) c

LEFT JOIN ingrediente

ON c.id\_ingredient = ingrediente.id\_ingredient;

IF vec\_ingrediente.COUNT < 1 THEN

RAISE no\_receptions;

END IF;

RETURN vec\_ingrediente;

EXCEPTION

WHEN no\_receptions THEN

RAISE\_APPLICATION\_ERROR(-20106, 'Restaurantul nu a primit nici o receptie in anul ' || l\_an);

END ingredients\_providers;

END pachet\_ex\_14;

DECLARE

v\_value pachet\_ex\_14.best\_year\_structure;

v\_ingrediente pachet\_ex\_14.ingredients\_providers\_vector;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Top clienti in anul 2022:');

pachet\_ex\_14.top\_clients(2, '2022');

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Top produse vandute in anul 2022:');

pachet\_ex\_14.top\_products(5, '2022');

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Total incasari pe anul 2022: ' || pachet\_ex\_14.get\_income('2022'));

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Top furnizori pe anul 2022:');

pachet\_ex\_14.top\_providers(2, '2022');

DBMS\_OUTPUT.NEW\_LINE;

v\_value := pachet\_ex\_14.best\_year;

DBMS\_OUTPUT.PUT\_LINE('Anul cu cele mai bune incasari a fost ' || v\_value.g\_year || ' cu venituri in valoare de ' || v\_value.g\_revenue);

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE('Totalul cantitatii si al costului dat pe fiecare ingredient in anul 2022:');

v\_ingrediente := pachet\_ex\_14.ingredients\_providers('2022');

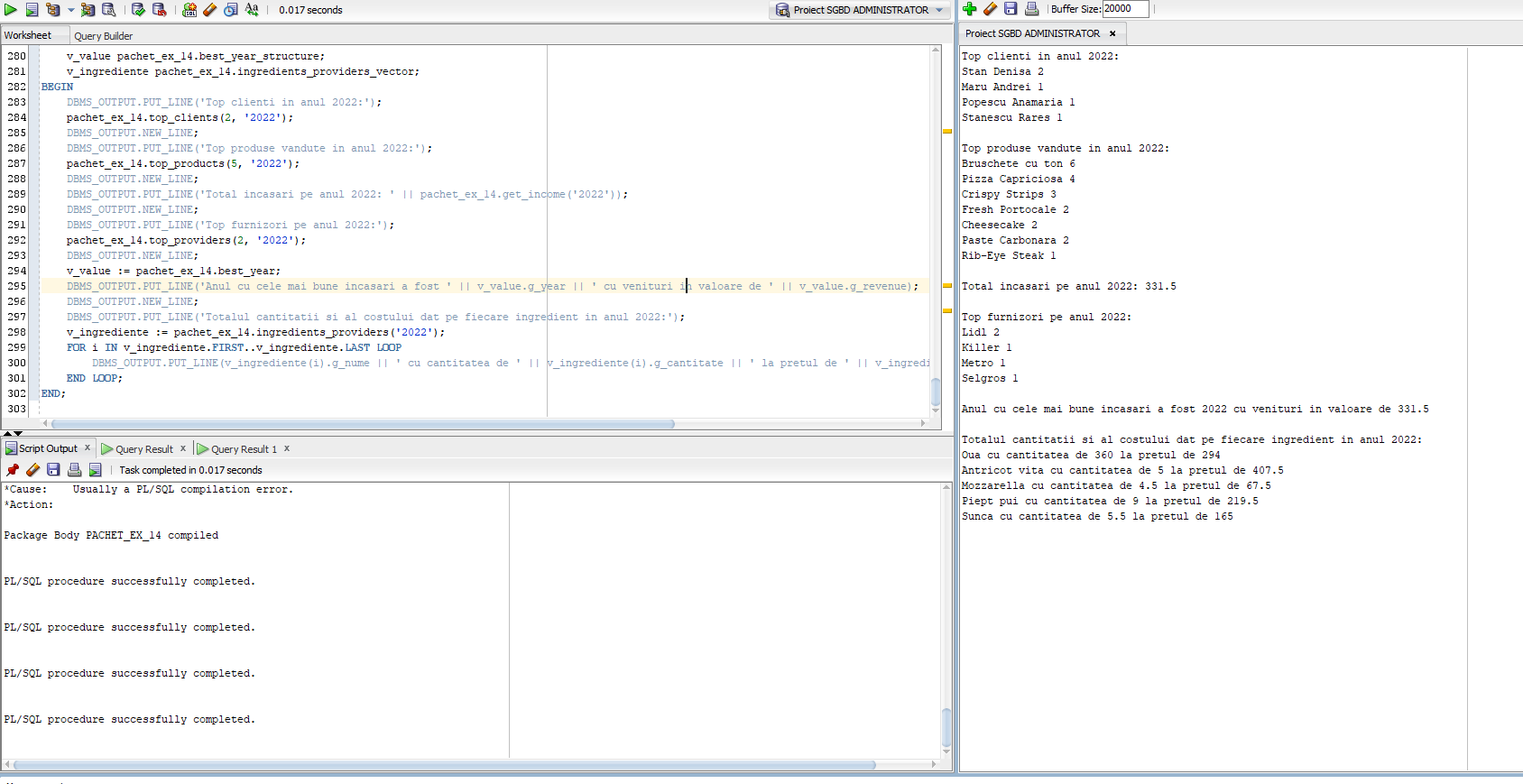
FOR i IN v\_ingrediente.FIRST..v\_ingrediente.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(v\_ingrediente(i).g\_nume || ' cu cantitatea de ' || v\_ingrediente(i).g\_cantitate || ' la pretul de ' || v\_ingrediente(i).g\_cost);

END LOOP;

END;

Exemplu de situație în care nu apare nici o excepție:



Exemplu de situații în care apar excepții:

