Proiect - Sisteme de Baze de Date

- Gestiunea unui lanț de restaurante -

Popescu Paullo Robertto Karloss

Grupa 406

2025 Ianuarie

Cuprins:

1. Prezentarea bazei de date.
   1. Tehnologii folosite pentru realizarea proiectului.
   2. Descrierea temei alese.
   3. Prezentarea constrângerilor impuse asupra modelului.
   4. Descrierea entităților.
   5. Descrierea relațiilor.
   6. Descrierea atributelor.
2. Diagrama Entitate-Relație (ER).
3. Diagrama Conceptuală.
4. Schemele relaționale corespunzătoare diagramei conceptuale.
5. Normalizarea până în forma normal 3 (FN1-FN3).
6. Implementarea bazei de date în Oracle.
   1. Crearea tabelelor și a constrângerilor.
   2. Inserarea datelor coerente în tabele (minium 5 înregistrări în fiecare tabel neasociativ, minimum 10 înregistrări în tabelele asociative) + crearea unei secvențe.
   3. Diagrama generată în SQL Developer după crearea tabelelor și inserarea datelor.
7. Crearea a 15 interogări complexe în SQL.
8. Crearea tabelului MESAJE pentru excepții.
9. PL / SQL.
   1. Subprogram stocat independent (inclusiv apelare) care utilizează toate cele 3 tipuri de colecții învățate.
   2. Subprogram stocat independent (inclusiv apelare) care utilizează toate cele 2 tipuri de cursoare învățate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor.
   3. Subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite și tratarea excepțiilor care pot apărea.
   4. Crearea unui Trigger LMD la nivel de comandă.
   5. Crearea unui Trigger LMD la nivel de linie (TRIGGER COMPOUND).
   6. Crearea unui Trigger LDD.
   7. Crearea unui pachet pentru obiectele definite.
10. Inserarea excepțiilor în tabelul mesaje.

**1. Prezentarea bazei de date**

* 1. ***Tehnologii folosite pentru realizarea proiectului.***

Pentru proiectul din cadrul cursului de Baze de Date am folosit versiunea **21c** a *Oracle Database*.

Ce aplicații am folosit?

* Oracle SQL Developer
  1. ***Descrierea temei alese. Pentru ce ar fi folosită***

În cadrul acestui proiect, am ales ca temă ***Gestiunea unui lanț de restaurante***. Scopul ei este de a face mai ușoară ținerea în evidență a mai multor restaurante (comenzi, angajați, facturi etc).

* 1. ***Prezentarea constrângerilor impuse asupra modelului.***

Un angajat trebuie să lucreze la un singur restaurant.

Un restaurant poate avea mai mulți angajați.

Angajații pot fi chelneri, casieri, bucătari sau manageri.

Un angajat poate avea subordonați, iar subordonații pot avea la rândul lor alți subordonați.

Într-o locație se poate găsii un singur restaurant.

Produsele sunt făcute cu ingrediente care sunt fabricate de un producător.

Într-o factură se poate găsii o singură comandă.

O comandă conține unul sau mai multe produse.

Un produs poate să aparțină mai multor restaurante.

Produsele pot fi preparate de un bucătar, dar există și produse care nu trebuie preparate (de exemplu vin, șampanie etc).

La o factură se pot preciza detalii, cum ar fi: dacă a fost achitată cash sau cu ajutorul unui card de credit.

* 1. ***Descrierea entităților.***

Cum am precizat mai sus, ideea proiectului este *Gestionarea unui lanț de restaurante*, de la care clienții pot comanda diferite produse. Acesta cuprinde următoarele entități:

* Entitatea **Restaurant**, care va conține numele restaurantului. Cheia primară fiind *id-ul restaurantului*.
* Entitatea **Locație,** care va conține numele țării unde se află restaurantul, orașul, respectiv codul poștal și strada. Cheia primară fiind *id-ul locației*.
* Entitatea **Produs**, care va conține numele fiecărui produs, cantitatea și o scurtă descriere a sa. Cheia primară fiind *id-ul produsului*.
* Entitatea **Ingredient**, care va conține numele ingredientului. Cheia primară fiind *id-ul ingredientului*.
* Entitatea **Producător**, care va conține numele și numărul de telefon al unui producător de ingrediente. Cheia primară fiind *id-ul producătorului*.
* Entitatea **Comandă**, care va conține prețul și data pentru fiecare comandă plasată. Cheia primară fiind *id-ul comenzii*.
* Entitatea **Angajat**, care va conține id-ul șefului (dacă are un șef), numele, prenumele și data angajării. Cheia primară fiind id-ul angajatului.
* Subentitatea **Bucătar**, care va conține numărul de stele al fiecărui bucătar. Cheia primară fiind id-ul angajatului.
* Subentitatea **Casier**, care va conține numărul de ani de studii al fiecărui casier. Cheie primară fiind id-ul angajatului.
* Subentitatea **Chelner**, care va conține numărul de ani de experiență al fiecărui chelner. Cheie primară fiind id-ul angajatului.
* Subentitatea **Manager**, care nu are atribute, are doar cheia primară id-ul angajatului.
* Entitatea **Client**, care va conține numele, prenumele și numărul personal de telefon, întrucât acestea sunt necesare atunci când se plasează o comandă. Cheia primară fiind *id-ul clientului*.
* Entitatea **Factură**, care va conține valoarea pe care clientul este nevoit să o plătească pentru comanda sa, dar și câteva detalii cum ar fi dacă a fost achitată cash sau cu ajutorul unui card de credit. Cheia primară fiind *id-ul facturii*.
  1. ***Descrierea relațiilor.***

Un anumit produs poate aparține mai multor restaurante.

Într-o locație poate fi găsit un singur restaurant.

Un produs se poate regăsii în mai multe comenzi.

Într-o comandă se pot găsii mai multe produse, dar cel puțin unul.

Un produs poate fi preparat de mai mulți bucătari.

Un ingredient trebuie să fie produs de un producător.

Produsele trebuie să conțină măcar un ingredient.

Un bucătar poate să prepare mai multe produse, dar trebuie să prepare minim un produs.

Un client poate plasa una sau mai multe comenzi (devine client după ce plasează minim o comandă).

O factură poate conține o singură comandă.

La o factură este atașat un singur casier, dar un casier poate fi atașat la mai multe facturi, nu doar una.

Un angajat trebuie să fie chelner, casier, bucătar sau manager.

* 1. ***Descrierea atributelor.***
* Entitatea **restaurant** va avea ca atribute id-ul și numele restaurantului curent (de exemplu “Yamas”, “Ivans” etc).

Tipul de date pentru id-ul restaurantului va fi *number(10)*, id-locație *number(10)*, iar pentru nume va fi *varchar2(32)*.

Cheia externă va fi id\_locație (provenită din tabelul locație), cu tipul de date *number(10).*

PK-ul va fi id-ul restaurantului (cheia primară).

Constrângere pentru numele restaurantului: Unique.

* Entitatea **locație** va avea ca atribute: id-ul locației, țara, orașul, codul poștal și strada pentru restaurantul respectiv.

Tipul de date pentru id-ul locației va fi *number(10),* țara *varchar2(20)*, orașul *varchar2(20)*, codul poștal *varchar2(15)* și strada *varchar2(50)*.

PK-ul va fi id-ul locației (cheia primară).

Constrângere pentru codul poștal: Unique.

* Entitatea **produs** va avea ca atribute: id-ul produsului, numele, cantitatea și o scurtă descriere pentru produsul comercializat.

Tipul de date pentru id-ul produsului va fi *number(10)*, nume *varchar2(25)*, gramaj *number(10)* și descriere *varchar2(100)*.

PK-ul va fi id-ul produsului (cheia primară).

Constrângere pentru gramajul produsului: check gramaj > 0.

* Tabelul asociativ **meniu** va avea ca atribute: id-ul produsului, id-ul restaurantului și prețul. Avem o cheie primară compusă formată din id-ul produsului (FK din tabelul produs) și id-ul restaurantului (FK din tabelul restaurant).

Tipul de date pentru id-ul produsului va fi *number(10)*, id-ul restaurantului *number(10)* și prețul *number(10)*.

Constrângere pentru prețul meniului: check preț > 0.

* Tabelul asociativ **cantitate**\_**produs** va avea ca atribute: id-ul produsului, id-ul ingredientului și cantitatea.

Avem o cheie primară compusă formată din id-ul produsului (FK din tabelul produs) și id-ul ingredient (FK din tabelul ingredient).

Tipul de date pentru id-ul produsului va fi *number(10)*, id-ul ingredientului *number(12)* și cantitatea *number(10)*.

Constrângere pentru cantitate: check cantitate >= 0.

* Entitatea **ingredient** va avea ca atribute: id-ul și numele ingredientului.

Ingredient va avea ca cheie externă id-producător (provenită din tabelul producător).

Tipul de date pentru id-ul ingredientului va fi *number(12)*, id-ul producătorului *number(10)*, iar pentru numele ingredientului *varchar2(32)*.

PK-ul va fi id-ul ingredientului (cheia primară).

* Entitatea **producător** va avea ca atribute: id-ul, numele și numărul de telefon al producătorului respectiv.

Tipul de date pentru id va fi *number(10)*, numele *varchar2(32)* și numărul de telefon *varchar2(15)*.

PK-ul va fi id-ul producătorului (cheia primară).

Constrângere pentru numărul de telefon: Unique.

* Entitatea **comandă** va avea ca atribute: id-ul, prețul și data unei comenzi plasate de client, și va avea ca cheie externă id-ul clientului (provenită din tabelul client), dar și id-ul facturii (provenită din tabelul factură).

Tipul de date pentru id-ul comenzii va fi *number(20)*, id-ul clientului *number(10)*, id-ul facturii *number(15)*, data *date*,iar prețul *number(10)*.

PK-ul va fi id-ul comenzii (cheia primară).

Constrângere pentru prețul comenzii: NOT\_NULL, dar și check preț > 0.

Atributul data va avea setat ca default ziua curentă (sysdate).

* Tabelul asociativ **conținut**\_**comandă** va avea ca atribute: id-ul produsului, id-ul comenzii și numărul de produse.

Avem o cheie primară compusă formată din id-ul produsului (FK din tabelul produs) și id-ul comenzii (FK din comandă).

Tipul de date pentru id-ul produsului va fi *number(10)*, id-ul comenzii *number(20)* și numărul de produse din comandă *number(10)*.

Constrângere pentru numărul de produse: check număr\_produse > 0.

* Entitatea **angajat** va avea ca atribute: id-ul, id-ul șefului, numele, prenumele și data angajării(am ales această abordare pentru că niciun subtip nu avea atribute separate de angajat).

Angajat va avea ca cheie externă id-restaurant (provenită din tabelul restaurant) și id-șef ca fiind cheie externă auto-referențială (provenită din același tabel, angajat).

Tipul de date pentru id-ul angajatului va fi *number(10)*, id-ul șefului *number(10)* dar care poate fi NULL, id-ul restaurantului *number(10)*, nume *varchar2(32)*, prenume *varchar2(32)*, iar data angajării *date* .

Atributul data\_angajare va avea setat ca default ziua curentă (sysdate).

* Subentitatea **bucătar** va avea ca atribute id-ul angajatului și numărul de stele.

Tipul de date pentru id-ul angajatului va fi *number(10)*, iar pentru numărul de stele *number(7)*.

PK va fi id\_angajat (care provine din tabelul angajat).

* Subentitatea **casier** va avea ca atribute id-ul angajatului și numărul de ani de studii.

Tipul de date pentru id-ul angajatului va fi *number(10)*, iar pentru numărul de ani de studii *number(5)*.

PK va fi id\_angajat (care provine din tabelul angajat).

* Subentitatea **chelner** va avea ca atribute id-ul angajatului și numărul de ani de experiență.

Tipul de date pentru id-ul angajatului va fi *number(10)*, iar pentru numărul de ani de experiență *number(5)*.

PK va fi id\_angajat (care provine din tabelul angajat).

* Subentitatea **manager** va avea ca atribut id-ul angajatulul.

Tipul de date pentru id-ul angajatului va fi *number(10)*.

PK va fi id\_angajat (care provine din tabelul angajat).

* Tabelul asociativ **preparare** va avea ca atribute: id-ul produsului, id-ul comenzii, id-ul bucătarului și durata.

Avem o cheie primară compusă formată din id-ul produsului (FK din tabelul produs), id-ul comenzii (FK din comandă) și id-ul bucătarului (FK din angajat), acestea reies din relația de tip 3 dintre produs, comandă și bucătar.

Tipul de date pentru id-ul produsului va fi *number(10)*, id-ul comenzii *number(20)*, id-ul bucătarului *number(10)* și durata *number(10)*.

* Entitatea **client** va avea ca atribute: id-ul, numele, prenumele și numărul de telefon al unui client.

Tipul de date pentru id-ul clientului va fi *number(10)*, numele *varchar2(32)*, prenumele *varchar2(32)* și numărul de telefon *varchar2(15)*.

Constrângere pentru numele și prenumele: NOT\_NULL.

PK-ul va fi id-ul clientului (cheia primară).

* Entitatea **factură** va avea ca atribute: id-ul, valoarea și câteva mici detalii legate de factură.

Constrângere pentru valoare : NOT\_NULL. Va avea cheia externă id-ul id-ul casierului.

Tipul de date pentru id-ul facturii va fi *number(15)*, valoarea *number(10)*, detalii *varchar2(50)* și id-ul casierului *number(10)*.

Constrângere pentru valoarea facturii: check valoare > 0.

PK-ul va fi id-ul facturii (cheia primară).

**2. Diagrama Entitate-Relație (ER):**

1

furnizează

PRODUCĂTOR

INGREDIENT

se\_află

LOCAȚIE

1

M(0)

M(1)

are

1

M(0)

conține

COMANDĂ

are

PRODUS

RESTAURANT

M(1)

1

M(1)

M(0)

1

M(1)

M(1)

M(1)

M(0)

atașat\_la

FACTURĂ

CLIENT

lucrează

ISA

ISA

ISA

ISA

1

1

1

1

1(0)

1(0)

1(0)

1(0)

CASIER

MANAGER

ANGAJAT

prepară

are

cere

M(1)

M(0)

BUCĂTAR

1

1

1

M(0)

CHELNER

PRODUCĂTOR

#ID-PRODUCĂTOR

Nume

Nr\_telefon

**3. Diagrama Conceptuală:**

INGREDIENT

#ID-INGREDIENT

Nume



LOCAȚIE

#ID-LOCAȚIE

Țara

Oraș

Cod\_Poștal

Strada

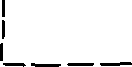


CANTITATE\_PRODUS

#ID-PRODUS

#ID-INGREGDIENT

Cantitate



RESTAURANT

#ID-RESTAURANT

Nume



COMANDĂ

#ID-COMANDĂ

Preț

Data



CONȚINUT\_COMANDĂ

#ID-PRODUS

#ID-COMANDĂ

Număr\_Produse

PRODUS

#ID-PRODUS

Nume

Gramaj

Descriere

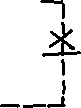


MENIU

#ID-RESTAURANT

#ID-PRODUS

Preț



ANGAJAT

#ID-ANGAJAT

ID-Șef

Nume

Prenume

Data\_Angajare



PREPARARE

#ID-PRODUS

#ID-COMANDĂ

#ID-BUCĂTAR

Durată



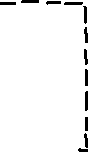
CLIENT

#ID-CLIENT

Nume

Prenume

Nr\_telefon



BUCĂTAR

nr\_stele



CASIER

ani\_studiu

FACTURĂ

#ID-FACTURĂ

Valoare

Detalii



CHELNER

ani\_experiență



MANAGER



**4. Schema relațională:**

* RESTAURANT (#ID-RESTAURANT, id-locație, Nume)
* LOCAȚIE (#ID-LOCAȚIE, Țara, Oraș, Cod\_Poștal, Strada)
* MENIU (#ID-RESTAURANT, #ID-PRODUS, Preț)
* PRODUS (#ID-PRODUS, Nume, Gramaj, Descriere)
* CANTITATE\_PRODUS (#ID-PRODUS, #ID-INGREDIENT, Cantitate)
* INGREDIENT (#ID-INGREDIENT, id-producător, Nume)
* PRODUCĂTOR (#ID-PRODUCĂTOR, Nume, Nr\_telefon)
* CONȚINUT\_COMANDĂ (#ID-PRODUS, #ID-COMANDĂ, Număr\_Produse)
* COMANDĂ (#ID-COMANDĂ, Preț, Data, id-factură, id-client)
* CLIENT (#ID-CLIENT, Nume, Prenume, Nr\_telefon)
* ANGAJAT (#ID-ANGAJAT, id-șef, id-restaurant, Nume, Prenume, data\_angajare)
* BUCĂTAR (#ID-ANGAJAT, nr\_stele)
* CASIER (#ID-ANGAJAT, ani\_studiu)
* CHELNER (#ID-ANGAJAT, ani\_experiență)
* MANAGER (#ID-ANGAJAT)
* FACTURĂ (#ID-FACTURĂ, Valoare, Detalii, id-casier)
* PREPARARE (#ID-PRODUS, #ID-COMANDĂ, #ID-BUCĂTAR, Durată)

**5. Normalizarea până la forma normal 3 (FN1 – FN3):**

Schema noastră ar fi scoasă din FN1 dacă entitatea produs ar conține atributul ingrediente, întrucât un produs are mai multe ingrediente (de exemplu piper, boia, sare etc).

**Schema Non-FN1:**

**PRODUS** (#ID-PRODUS, Nume, Gramaj, Descriere, ***Ingrediente***)

Schema noastră ar fi scoasă din FN2 dacă în loc să existe două entități ingredient și producător ar exista doar entitatea Ingredient (#id-ingredient, #id-producător, nume\_ingredient, nume\_producător, nr\_telefon\_producător), pentru ca nr de telefon nu depinde de id-ul ingredientului.

**Schema Non-FN2:**

**INGREDIENT** (#ID-INGREDIENT, #ID-PRODUCĂTOR, Nume\_ingredient, Nume\_Producător, ***Nr\_telefon***)

Schema noastră ar fi scoasă din FN3 dacă entitatea comandă ar conține ca atribut numărul de telefon al clientului (dependența tranzitivă între oraș-țara).

**Schema Non-FN3:**

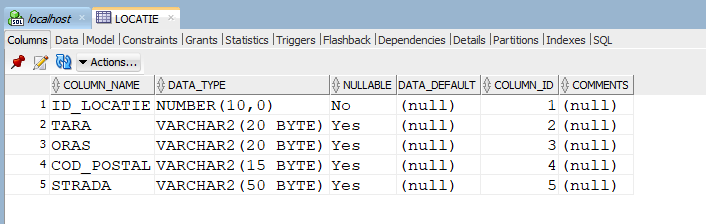
**COMANDĂ** (#ID-COMANDĂ, Preț, Data, id\_factură, id\_client, ***nr\_telefon\_client***)

**6. Implementarea bazei de date în Oracle:**

***6.1 Crearea tabelelor și a constrângerilor.***

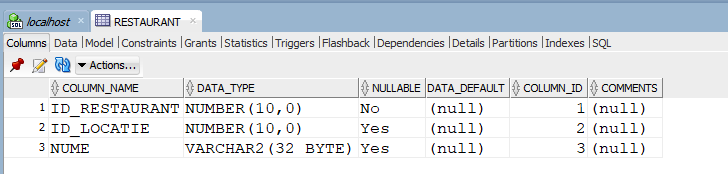
|  |
| --- |
| --- CREAREA TABELELOR ---  -- LOCATIE --  CREATE TABLE locatie (  id\_locatie NUMBER(10) PRIMARY KEY,  tara VARCHAR2(20),  oras VARCHAR2(20),  cod\_postal VARCHAR2(15) UNIQUE,  strada VARCHAR2(50)  ); |

***Print-Screen:***

******

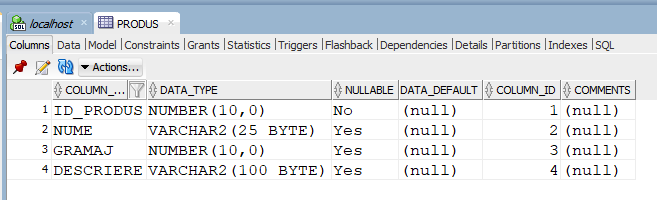
|  |
| --- |
| -- RESTAURANT --  CREATE TABLE restaurant (  id\_restaurant NUMBER(10) PRIMARY KEY,  id\_locatie NUMBER(10),  nume VARCHAR2(32) UNIQUE,  CONSTRAINT fk\_restaurant\_locatie FOREIGN KEY ( id\_locatie )  REFERENCES locatie ( id\_locatie )  ); |

***Print-Screen:***

******

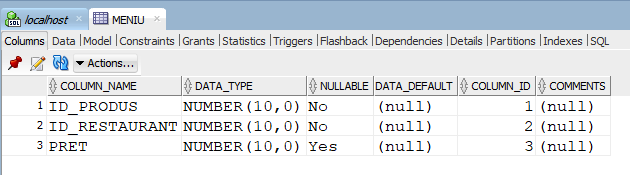
|  |
| --- |
| -- PRODUS --  CREATE TABLE produs (  id\_produs NUMBER(10) PRIMARY KEY,  nume VARCHAR2(25),  gramaj NUMBER(10),  descriere VARCHAR2(100),  CONSTRAINT chk\_gramaj CHECK ( gramaj > 0 )  ); |

***Print-Screen:***

******

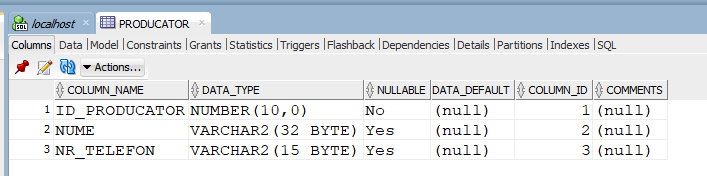
|  |
| --- |
| -- MENIU --  CREATE TABLE meniu (  id\_produs NUMBER(10),  id\_restaurant NUMBER(10),  pret NUMBER(10),  CONSTRAINT meniu\_pk PRIMARY KEY ( id\_produs,  id\_restaurant ),  CONSTRAINT meniu\_produs\_fk FOREIGN KEY ( id\_produs )  REFERENCES produs ( id\_produs ),  CONSTRAINT meniu\_restaurant\_fk FOREIGN KEY ( id\_restaurant )  REFERENCES restaurant ( id\_restaurant ),  CONSTRAINT chk\_pret CHECK ( pret > 0 )  ); |

***Print-Screen:***



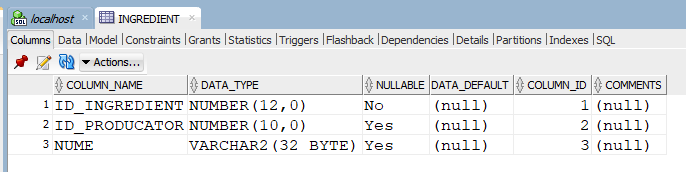
|  |
| --- |
| -- PRODUCATOR --  CREATE TABLE producator (  id\_producator NUMBER(10) PRIMARY KEY,  nume VARCHAR2(32),  nr\_telefon VARCHAR2(15) UNIQUE  ); |

***Print-Screen:***

******

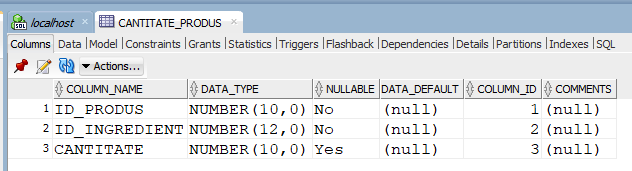
|  |
| --- |
| -- INGREDIENT --  CREATE TABLE ingredient (  id\_ingredient NUMBER(12) PRIMARY KEY,  id\_producator NUMBER(10),  nume VARCHAR2(32),  CONSTRAINT fk\_ingredient\_producator FOREIGN KEY ( id\_producator )  REFERENCES producator ( id\_producator )  ); |

***Print-Screen:***

******

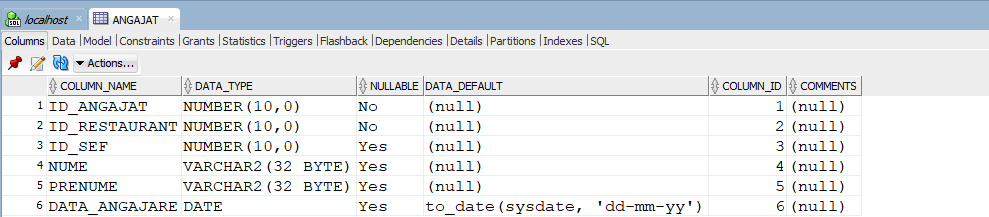
|  |
| --- |
| -- CANTITATE\_PRODUS --  CREATE TABLE cantitate\_produs (  id\_produs NUMBER(10),  id\_ingredient NUMBER(12),  cantitate NUMBER(10),  CONSTRAINT cantitate\_produs\_pk PRIMARY KEY ( id\_produs,  id\_ingredient ),  CONSTRAINT cantitate\_produs\_fk FOREIGN KEY ( id\_produs )  REFERENCES produs ( id\_produs ),  CONSTRAINT cantitate\_ingredient\_fk FOREIGN KEY ( id\_ingredient )  REFERENCES ingredient ( id\_ingredient ),  CONSTRAINT chk\_cantitate\_produs CHECK ( cantitate > 0 )  ); |

***Print-Screen:***

******

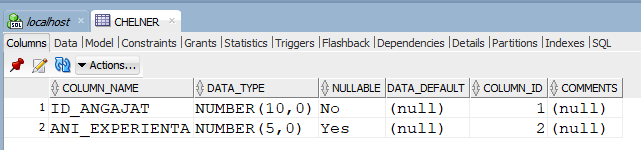
|  |
| --- |
| -- ANGAJAT --  CREATE TABLE angajat (  id\_angajat NUMBER(10) PRIMARY KEY,  id\_restaurant NUMBER(10) NOT NULL,  id\_sef NUMBER(10),  nume VARCHAR2(32),  prenume VARCHAR2(32),  data\_angajare DATE DEFAULT to\_date(sysdate, 'dd-mm-yy'),  CONSTRAINT angajat\_restaurant\_fk FOREIGN KEY ( id\_restaurant )  REFERENCES restaurant (id\_restaurant),  CONSTRAINT angajat\_sef\_fk FOREIGN KEY (id\_sef)  REFERENCES angajat (id\_angajat)  ); |

***Print-Screen:***



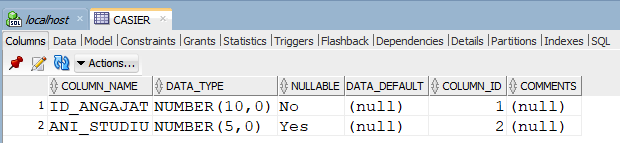
|  |
| --- |
| -- CHELNER --  CREATE TABLE chelner (  id\_angajat NUMBER(10),  ani\_experienta NUMBER(5),  CONSTRAINT chelner\_pk PRIMARY KEY ( id\_angajat ),  CONSTRAINT angajat\_fk FOREIGN KEY ( id\_angajat )  REFERENCES angajat ( id\_angajat )  ); |

***Print-Screen:***



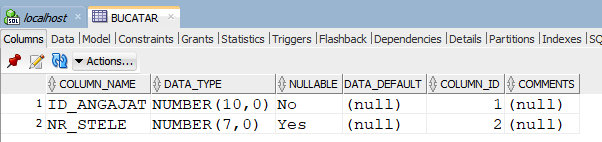
|  |
| --- |
| -- CASIER --  CREATE TABLE casier (  id\_angajat NUMBER(10),  ani\_studiu NUMBER(5),  CONSTRAINT casier\_pk PRIMARY KEY ( id\_angajat ),  CONSTRAINT angajat\_casier\_fk FOREIGN KEY ( id\_angajat )  REFERENCES angajat ( id\_angajat )  ); |

***Print-Screen:***

******

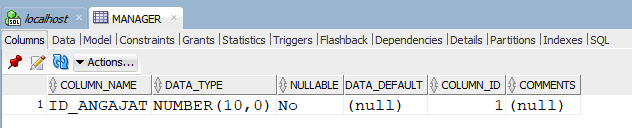
|  |
| --- |
| -- BUCATAR --  CREATE TABLE bucatar (  id\_angajat NUMBER(10),  nr\_stele NUMBER(7),  CONSTRAINT bucatar\_pk PRIMARY KEY ( id\_angajat ),  CONSTRAINT angajat\_bucatar\_fk FOREIGN KEY ( id\_angajat )  REFERENCES angajat ( id\_angajat )  ); |

***Print-Screen:***

******

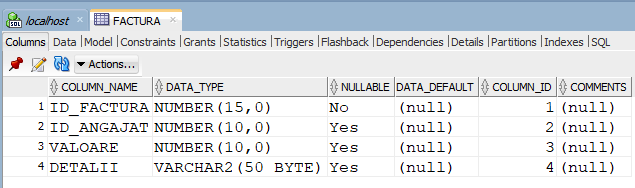
|  |
| --- |
| -- MANAGER --  CREATE TABLE manager (  id\_angajat NUMBER(10),  CONSTRAINT manager\_pk PRIMARY KEY ( id\_angajat ),  CONSTRAINT angajat\_manager\_fk FOREIGN KEY ( id\_angajat )  REFERENCES angajat ( id\_angajat )  ); |

***Print-Screen:***



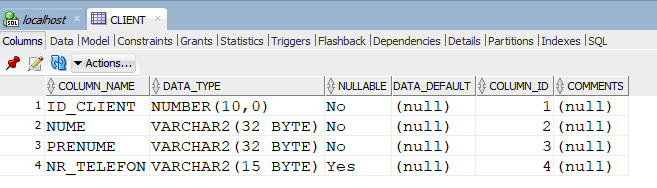
|  |
| --- |
| -- FACTURA --  CREATE TABLE factura (  id\_factura NUMBER(15) PRIMARY KEY,  id\_angajat NUMBER(10),  valoare NUMBER(10),  detalii VARCHAR2(50),  CONSTRAINT fk\_factura\_casier FOREIGN KEY ( id\_angajat )  REFERENCES casier ( id\_angajat ),  CONSTRAINT chk\_factura\_valoare CHECK ( valoare > 0 )  ); |

***Print-Screen:***



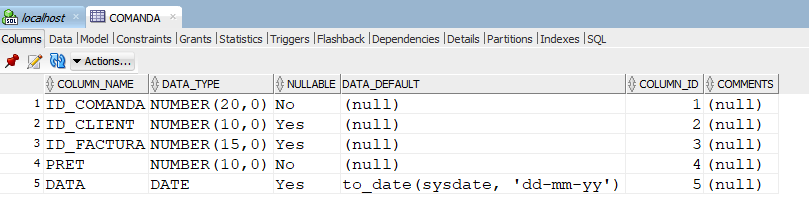
|  |
| --- |
| -- CLIENT --  CREATE TABLE client (  id\_client NUMBER(10) PRIMARY KEY,  nume VARCHAR2(32) NOT NULL,  prenume VARCHAR2(32) NOT NULL,  nr\_telefon VARCHAR2(15)  ); |

***Print-Screen:***

******

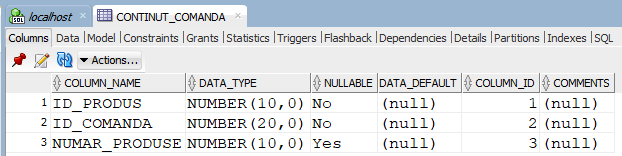
|  |
| --- |
| -- COMANDA --  CREATE TABLE comanda (  id\_comanda NUMBER(20) PRIMARY KEY,  id\_client NUMBER(10),  id\_factura NUMBER(15),  pret NUMBER(10) NOT NULL,  data DATE DEFAULT to\_date(sysdate, 'dd-mm-yy'),  CONSTRAINT fk\_comanda\_client FOREIGN KEY ( id\_client )  REFERENCES client ( id\_client ),  CONSTRAINT fk\_comanda\_factura FOREIGN KEY ( id\_factura )  REFERENCES factura ( id\_factura ),  CONSTRAINT chk\_comanda CHECK ( pret > 0 )  ); |

***Print-Screen:***

******

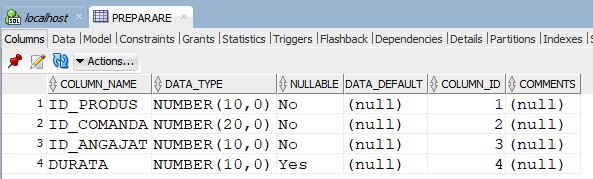
|  |
| --- |
| -- CONTINUT\_COMANDA --  CREATE TABLE continut\_comanda (  id\_produs NUMBER(10),  id\_comanda NUMBER(20),  numar\_produse NUMBER(10),  CONSTRAINT continut\_comanda\_pk PRIMARY KEY ( id\_produs,  id\_comanda ),  CONSTRAINT continut\_produs\_fk FOREIGN KEY ( id\_produs )  REFERENCES produs ( id\_produs ),  CONSTRAINT continut\_comanda\_fk FOREIGN KEY ( id\_comanda )  REFERENCES comanda ( id\_comanda ),  CONSTRAINT chk\_continut\_comanda CHECK ( numar\_produse > 0 )  ); |

***Print-Screen:***

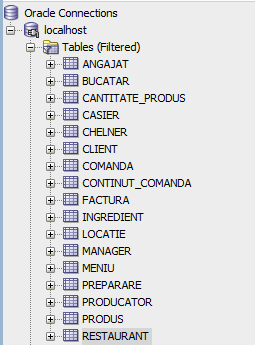
******

|  |
| --- |
| -- PREPARARE --  CREATE TABLE preparare (  id\_produs NUMBER(10),  id\_comanda NUMBER(20),  id\_angajat NUMBER(10),  durata NUMBER(10),  CONSTRAINT preparare\_pk PRIMARY KEY ( id\_produs,  id\_comanda,  id\_angajat ),  CONSTRAINT preparare\_casier\_fk FOREIGN KEY ( id\_angajat )  REFERENCES bucatar ( id\_angajat ),  CONSTRAINT preparare\_produs\_fk FOREIGN KEY ( id\_produs )  REFERENCES produs ( id\_produs ),  CONSTRAINT preparare\_comanda\_fk FOREIGN KEY ( id\_comanda )  REFERENCES comanda ( id\_comanda )  ); |

***Print-Screen:***

******

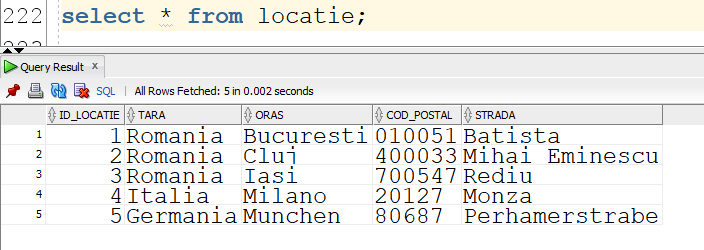
***Print-Screen:***



***6.2 Inserarea datelor coerente în tabele (minimum 5 înregistrări în fiecare tabel neasociativ, minimum 10 înregistrări în tabelele asociative) + crearea unei secvențe.***

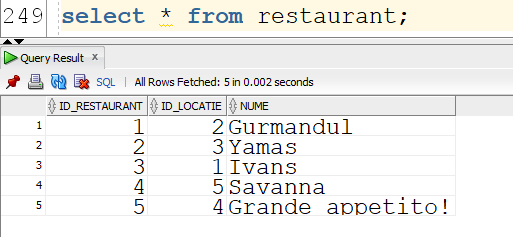
|  |
| --- |
| --- INSERAREA DATELOR IN TABELE ---  -- PENTRU TABELUL LOCATIE --  create sequence id\_locatie  start with 1  increment by 1  minvalue 0  maxvalue 9999  nocycle;  insert into locatie  values (id\_locatie.nextval, 'Romania' , 'Bucuresti', '010051', 'Batista'); --1  insert into locatie  values (id\_locatie.nextval, 'Romania', 'Cluj', '400033', 'Mihai Eminescu'); --2  insert into locatie  values (id\_locatie.nextval, 'Romania', 'Iasi', '700547', 'Rediu'); --3  insert into locatie  values (id\_locatie.nextval, 'Italia', 'Milano', '20127', 'Monza'); --4  insert into locatie  values (id\_locatie.nextval, 'Germania', 'Munchen', '80687', 'Perhamerstrabe'); --5 |

***Print-Screen:***



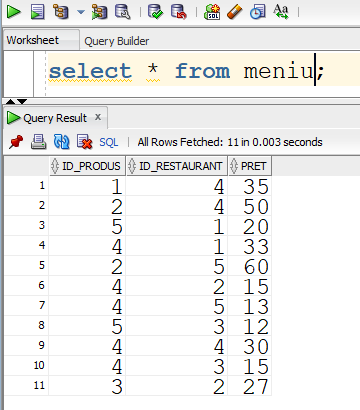
|  |
| --- |
| -- PENTRU TABELUL RESTAURANT --  create sequence id\_restaurant  start with 1  increment by 1  minvalue 0  maxvalue 9999  nocycle;  insert into restaurant  values (id\_restaurant.nextval, 2, 'Gurmandul'); --1  insert into restaurant  values (id\_restaurant.nextval, 3, 'Yamas'); --2  insert into restaurant  values (id\_restaurant.nextval, 1, 'Ivans'); --3  insert into restaurant  values (id\_restaurant.nextval, 5, 'Savanna'); --4  insert into restaurant  values (id\_restaurant.nextval, 4, 'Grande appetito!'); --5 |

***Print-Screen:***



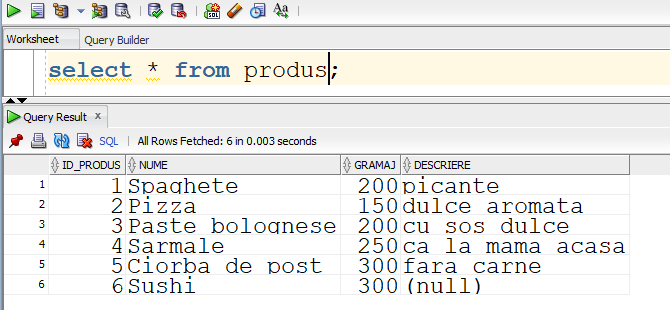
|  |
| --- |
| -- PENTRU TABELUL MENIU --  -- pretul este in lei --  insert into meniu (id\_restaurant, id\_produs, pret)  values (4, 1, 35); --1  insert into meniu (id\_restaurant, id\_produs, pret)  values (4, 2, 50); --2  insert into meniu (id\_restaurant, id\_produs, pret)  values (1, 5, 20); --3  insert into meniu (id\_restaurant, id\_produs, pret)  values (1, 4, 33); --4  insert into meniu (id\_restaurant, id\_produs, pret)  values (5, 2, 60); --5  insert into meniu (id\_restaurant, id\_produs, pret)  values (2, 4, 15); --6  insert into meniu (id\_restaurant, id\_produs, pret)  values (5, 4, 13); --7  commit;  insert into meniu (id\_restaurant, id\_produs, pret)  values (3, 5, 12); --8  insert into meniu (id\_restaurant, id\_produs, pret)  values (4, 4, 30); --9  insert into meniu (id\_restaurant, id\_produs, pret)  values (3, 4, 15); --10  insert into meniu (id\_restaurant, id\_produs, pret)  values (2, 3, 27); --11 |

***Print-Screen:***



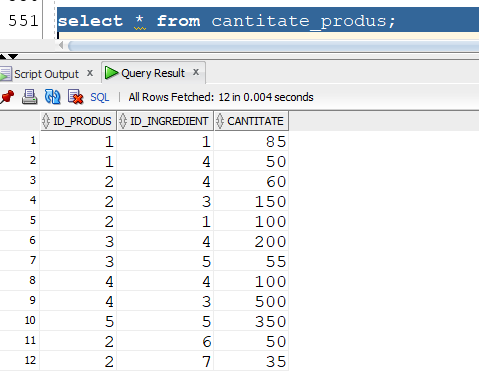
|  |
| --- |
| -- PENTRU TABELUL PRODUS --  -- aici cantitatea este in grame  create sequence id\_produs  start with 1  increment by 1  minvalue 0  maxvalue 9999  nocycle;  insert into produs  values (id\_produs.nextval, 'Spaghete', 200, 'picante'); --1  insert into produs  values (id\_produs.nextval, 'Pizza', 150, 'dulce aromata'); --2  insert into produs  values (id\_produs.nextval, 'Spaghete', 200, 'picante'); --3  -- aici am updatat linia pentru paste bolognese  update produs  set nume = 'Paste bolognese', descriere = 'cu sos dulce'  where id\_produs = 3;  insert into produs  values (id\_produs.nextval, 'Sarmale', 250, 'ca la mama acasa'); --4  insert into produs  values (id\_produs.nextval, 'Ciorba de post', 300, 'fara carne'); --5  insert into produs(id\_produs,nume,gramaj)  values (id\_produs.nextval, 'Sushi', 300); --6 |

***Print-Screen:***



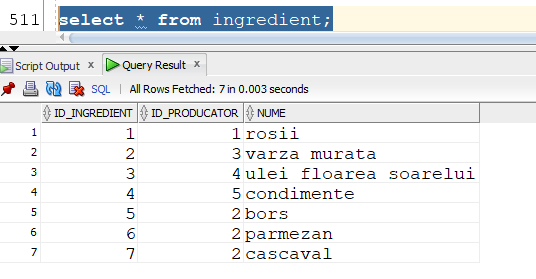
|  |
| --- |
| -- PENTRU TABELUL CANTITATE\_PRODUS --  -- cantitatea este masurata in grame --  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (1, 1, 85); --1  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (1, 4, 50); --2  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (2, 4, 60); --3  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (2, 3, 150); --4  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (2, 1, 100); --5  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (3, 4, 200); --6  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (3, 5, 55); --7  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (4, 4, 100); --8  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (4, 3, 500); --9  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (5, 5, 350); --10  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (2, 6, 50); --11  insert into cantitate\_produs (id\_produs, id\_ingredient, cantitate)  values (2, 7, 35); --12 |

***Print-Screen:***



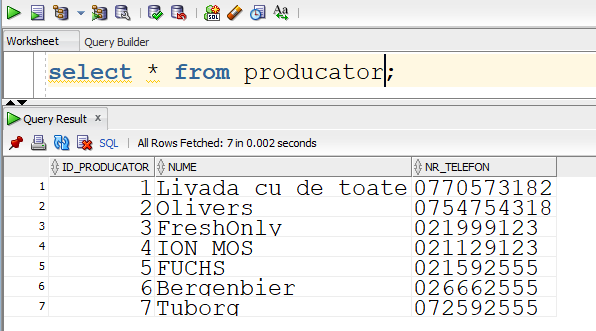
|  |
| --- |
| -- PENTRU TABELUL INGREDIENT --  create sequence id\_ingredient  start with 1  increment by 1  minvalue 0  maxvalue 9999  nocycle;  insert into ingredient  values (id\_ingredient.nextval, 1, 'rosii'); --1  insert into ingredient  values (id\_ingredient.nextval, 3, 'varza murata'); --2  insert into ingredient  values (id\_ingredient.nextval, 4, 'ulei floarea soarelui'); --3  insert into ingredient  values (id\_ingredient.nextval, 5, 'condimente'); --4  insert into ingredient  values (id\_ingredient.nextval, 2, 'bors'); --5  insert into ingredient  values (id\_ingredient.nextval, 2, 'parmezan'); --6  insert into ingredient  values (id\_ingredient.nextval, 2, 'cascaval'); --7 |

***Print-Screen:***



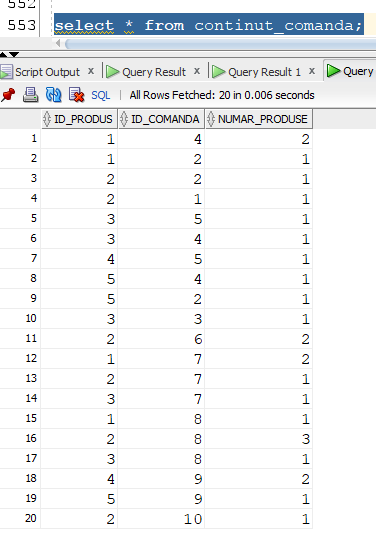
|  |
| --- |
| -- PENTRU TABELUL PRODUCATOR --  create sequence id\_producator  start with 1  increment by 1  minvalue 0  maxvalue 9999  nocycle;  insert into producator  values (id\_producator.nextval, 'Livada cu de toate', '0770573182'); --1  insert into producator  values (id\_producator.nextval, 'Olivers', '0754754318'); --2  insert into producator  values (id\_producator.nextval, 'FreshOnly', '021999123'); --3  insert into producator  values (id\_producator.nextval, 'ION MOS', '021129123'); --4  insert into producator  values (id\_producator.nextval, 'FUCHS', '021592555'); --5  insert into producator  values (id\_producator.nextval, 'Bergenbier', '026662555'); --6  insert into producator  values (id\_producator.nextval, 'Tuborg', '072592555'); --7 |

***Print-Screen:***



|  |
| --- |
| -- PENTRU TABELUL CONTINUT\_COMANDA --  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (1, 4, 2); --1  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (1, 2, 1); --2  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (2, 2, 1); --3  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (2, 1, 1); --4  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (3, 5, 1); --5  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (3, 4, 1); --6  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (4, 5, 1); --7  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (5, 4, 1); --8  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (5, 2, 1); --9  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (3, 3, 1); --10  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (2, 6, 2); --11  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (1, 7, 2); --12  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (2, 7, 1); --13  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (3, 7, 1); --14  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (1, 8, 1); --15  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (2, 8, 3); --16  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (3, 8, 1); --17  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (4, 9, 2); --18  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (5, 9, 1); --19  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (2, 10, 1); --20 |

***Print-Screen:***

******

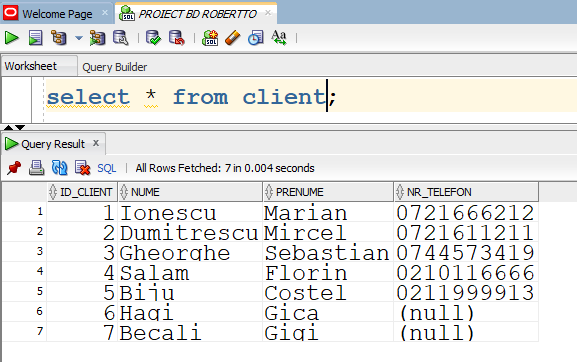
|  |
| --- |
| -- PENTRU TABELUL COMANDA --  create sequence id\_comanda  start with 1  increment by 1  minvalue 0  maxvalue 9999  nocycle;  insert into comanda (id\_comanda, id\_client, id\_factura, pret)  values (id\_comanda.nextval, 1, 4, 50); --1  insert into comanda (id\_comanda, id\_client, id\_factura, pret)  values (id\_comanda.nextval, 3, 5, 130); --2  insert into comanda (id\_comanda, id\_client, id\_factura, pret)  values (id\_comanda.nextval, 2, 3, 20); --3  insert into comanda (id\_comanda, id\_client, id\_factura, pret)  values (id\_comanda.nextval, 4, 2, 230); --4  insert into comanda (id\_comanda, id\_client, id\_factura, pret)  values (id\_comanda.nextval, 5, 1, 80); --5  insert into comanda (id\_comanda, id\_client, id\_factura, pret)  values (id\_comanda.nextval, 1, 4, 100); --6  insert into comanda (id\_comanda, id\_client, id\_factura, pret)  values (id\_comanda.nextval, 1, 1, 50); --7  insert into comanda (id\_comanda, id\_client, id\_factura, pret)  values (id\_comanda.nextval, 1, 2, 30); --8  insert into comanda (id\_comanda, id\_client, id\_factura, pret)  values (id\_comanda.nextval, 1, 3, 70); --9  insert into comanda (id\_comanda, id\_client, id\_factura, pret)  values (id\_comanda.nextval, 1, 4, 20); --10 |

***Print-Screen:***

******

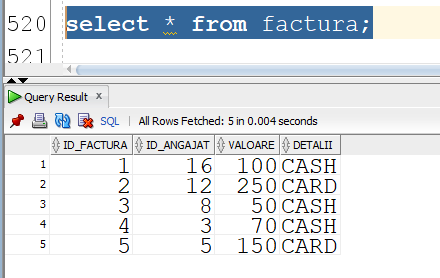
|  |
| --- |
| -- PENTRU TABELUL CLIENT --  create sequence id\_client  start with 1  increment by 1  minvalue 0  maxvalue 9999  nocycle;  insert into client  values (id\_client.nextval, 'Ionescu', 'Marian', '0721666212'); --1  insert into client (id\_client, nume, prenume, nr\_telefon)  values (id\_client.nextval, 'Dumitrescu', 'Mircel', '0721611211'); --2  insert into client (id\_client, nume, prenume, nr\_telefon)  values (id\_client.nextval, 'Gheorghe', 'Sebastian', '0744573419'); --3  insert into client (id\_client, nume, prenume, nr\_telefon)  values (id\_client.nextval, 'Salam','Florin', '0210116666'); --4  insert into client (id\_client, nume, prenume, nr\_telefon)  values (id\_client.nextval, 'Biju', 'Costel', '0211999913'); --5  insert into client (id\_client, nume, prenume)  values (id\_client.nextval, 'Hagi', 'Gica'); --6  insert into client (id\_client, nume, prenume)  values (id\_client.nextval, 'Becali', 'Gigi'); --7 |

***Print-Screen:***



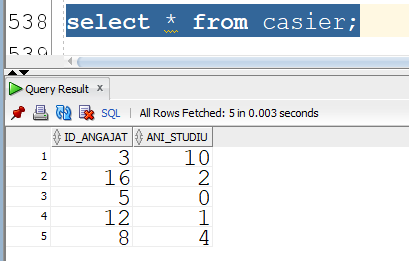
|  |
| --- |
| -- PENTRU TABELUL FACTURA --  -- fiecare factura are o valoare care include mai mult taxe etc --  create sequence id\_factura  start with 1  increment by 1  minvalue 0  maxvalue 9999  nocycle;  insert into factura (id\_factura, id\_angajat, valoare, detalii)  values (id\_factura.nextval, 16, 100, 'CASH'); --1  insert into factura (id\_factura, id\_angajat, valoare, detalii)  values (id\_factura.nextval, 12, 250, 'CARD'); --2  insert into factura (id\_factura, id\_angajat, valoare, detalii)  values (id\_factura.nextval, 8, 50, 'CASH'); --3  insert into factura (id\_factura, id\_angajat, valoare, detalii)  values (id\_factura.nextval, 3, 70, 'CASH'); --4  insert into factura (id\_factura, id\_angajat, valoare, detalii)  values (id\_factura.nextval, 5, 150, 'CARD'); --5 |

***Print-Screen:***



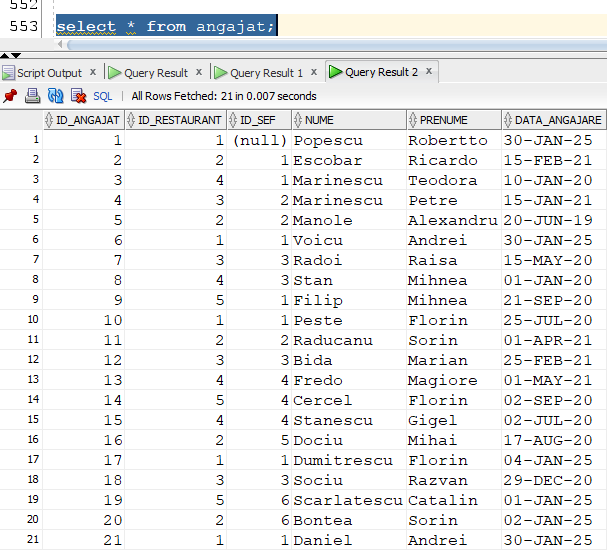
|  |
| --- |
| -- PENTRU TABELUL CASIER --  insert into casier(id\_angajat, ani\_studiu)  values (3, 10); --1  insert into casier(id\_angajat, ani\_studiu)  values (16, 2); --2  insert into casier(id\_angajat, ani\_studiu)  values (5, 0); --3  insert into casier(id\_angajat, ani\_studiu)  values (12, 1); --4  insert into casier(id\_angajat, ani\_studiu)  values (8, 4); --5 |

***Print-Screen:***



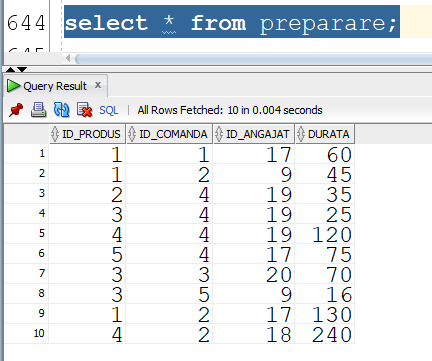
|  |
| --- |
| -- PENTRU TABELUL ANGAJAT --  create sequence id\_angajat  start with 1  increment by 1  minvalue 0  maxvalue 9999  nocycle;  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume)  values (id\_angajat.nextval, 1, NULL, 'Popescu', 'Robertto'); --1  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 2, 1, 'Escobar', 'Ricardo', to\_date('15-02-21', 'dd-mm-yy')); --2  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 4, 1, 'Marinescu', 'Teodora', to\_date('10-01-20','dd-mm-yy')); --3  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 3, 2, 'Marinescu', 'Petre', to\_date('15-01-21','dd-mm-yy')); --4  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 2, 2, 'Manole', 'Alexandru', to\_date('20-06-19','dd-mm-yy')); --5  insert into angajat (id\_angajat, id\_sef, id\_restaurant, nume, prenume)  values (id\_angajat.nextval, 1, 1, 'Voicu', 'Andrei'); --6  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 3, 3, 'Radoi', 'Raisa', to\_date('15-05-20','dd-mm-yy')); --7  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 4, 3, 'Stan', 'Mihnea', to\_date('01-01-20','dd-mm-yy')); --8  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 5, 1, 'Filip', 'Mihnea', to\_date('21-09-20','dd-mm-yy')); --9  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 1, 1, 'Peste', 'Florin', to\_date('25-07-20','dd-mm-yy')); --10  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 2, 2, 'Raducanu', 'Sorin', to\_date('01-04-21', 'dd-mm-yy')); --11  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 3, 3, 'Bida', 'Marian', to\_date('25-02-21', 'dd-mm-yy')); --12  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 4, 4, 'Fredo', 'Magiore', to\_date('01-05-21', 'dd-mm-yy')); --13  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 5, 4, 'Cercel', 'Florin', to\_date('02-09-20','dd-mm-yy')); --14  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 4, 4, 'Stanescu', 'Gigel', to\_date('02-07-20', 'dd-mm-yy')); --15  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 2, 5, 'Dociu', 'Mihai', to\_date('17-08-20', 'dd-mm-yy')); --16  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 1, 1, 'Dumitrescu', 'Florin', to\_date('04-01-25', 'dd-mm-yy')); --17  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 3, 3, 'Sociu', 'Razvan', to\_date('29-12-20', 'dd-mm-yy')); --18  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 5, 6, 'Scarlatescu', 'Catalin', to\_date('01-01-25', 'dd-mm-yy')); --19  insert into angajat (id\_angajat, id\_restaurant, id\_sef, nume, prenume, data\_angajare)  values (id\_angajat.nextval, 2, 6, 'Bontea', 'Sorin', to\_date('02-01-25', 'dd-mm-yy')); --20  insert into angajat (id\_angajat, id\_sef, id\_restaurant, nume, prenume)  values (id\_angajat.nextval, 1, 1, 'Daniel', 'Andrei'); --21 |

***Print-Screen:***

******

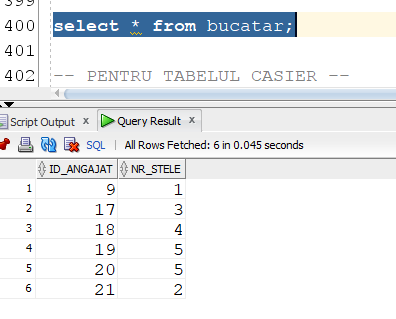
|  |
| --- |
| -- PENTRU TABELUL PREPARARE --  -- durata este masurata in minute --  insert into preparare (id\_produs, id\_comanda, id\_angajat, durata)  values (1, 1, 17, 60); --1  insert into preparare (id\_produs, id\_comanda, id\_angajat, durata)  values (1, 2, 9, 45); --2  insert into preparare (id\_produs, id\_comanda, id\_angajat, durata)  values (2, 4, 19, 35); --3  insert into preparare (id\_produs, id\_comanda, id\_angajat, durata)  values (3, 4, 19, 25); --4  insert into preparare (id\_produs, id\_comanda, id\_angajat, durata)  values (4, 4, 19, 120); --5  insert into preparare (id\_produs, id\_comanda, id\_angajat, durata)  values (5, 4, 17, 75); --6  insert into preparare (id\_produs, id\_comanda, id\_angajat, durata)  values (3, 3, 20, 70); --7  insert into preparare (id\_produs, id\_comanda, id\_angajat, durata)  values (3, 5, 9, 16); --8  insert into preparare (id\_produs, id\_comanda, id\_angajat, durata)  values (1, 2, 17, 130); --9  insert into preparare (id\_produs, id\_comanda, id\_angajat, durata)  values (4, 2, 18, 240); --10 |

***Print-Screen:***



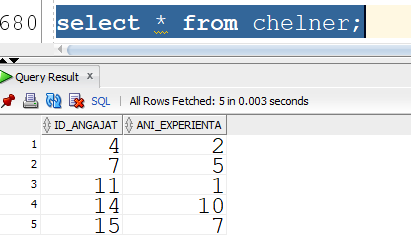
|  |
| --- |
| -- PENTRU TABELUL BUCATAR --  insert into bucatar (id\_angajat, nr\_stele)  values (9, 1); --1  insert into bucatar (id\_angajat, nr\_stele)  values (17, 3); --2  insert into bucatar (id\_angajat, nr\_stele)  values (18, 4); --3  insert into bucatar (id\_angajat, nr\_stele)  values (19, 5); --4  insert into bucatar (id\_angajat, nr\_stele)  values (20, 5); --5  insert into bucatar (id\_angajat, nr\_stele)  values (21, 2); --6 |

***Print-Screen:***



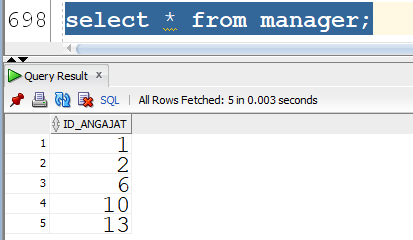
|  |
| --- |
| -- PENTRU TABELUL CHELNER --  insert into chelner (id\_angajat, ani\_experienta)  values (4, 2); --1  insert into chelner (id\_angajat, ani\_experienta)  values (7, 5); --2  insert into chelner (id\_angajat, ani\_experienta)  values (11, 1); --3  insert into chelner (id\_angajat, ani\_experienta)  values (14, 10); --4  insert into chelner (id\_angajat, ani\_experienta)  values (15, 7); --5 |

***Print-Screen:***



|  |
| --- |
| -- PENTRU TABELUL MANAGER --  insert into manager (id\_angajat)  values (1); --1  insert into manager (id\_angajat)  values (2); --2  insert into manager (id\_angajat)  values (6); --3  insert into manager (id\_angajat)  values (10); --4  insert into manager (id\_angajat)  values (13); --5 |

***Print-Screen:***



***6.3 Diagrama generată în SQL Developer după crearea tabelelor și inserarea datelor.***

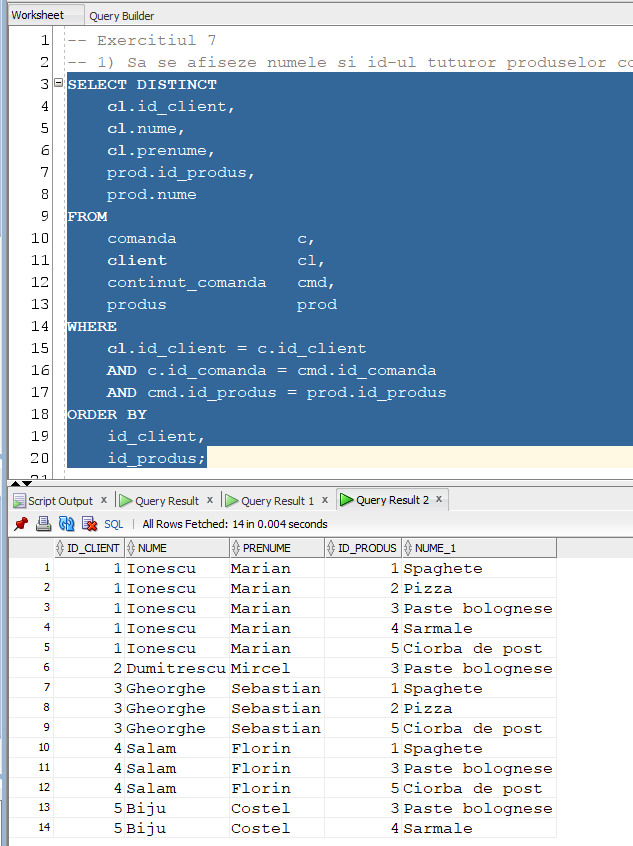


**7. Crearea a 15 interogări complexe în SQL:**

1) Să se afișeze numele și id-ul tuturor produselor comandate de un client vreodată.

|  |
| --- |
| SELECT DISTINCT  cl.id\_client,  cl.nume,  cl.prenume,  prod.id\_produs,  prod.nume  FROM  comanda c,  client cl,  continut\_comanda cmd,  produs prod  WHERE  cl.id\_client = c.id\_client  AND c.id\_comanda = cmd.id\_comanda  AND cmd.id\_produs = prod.id\_produs  ORDER BY  id\_client,  id\_produs; |

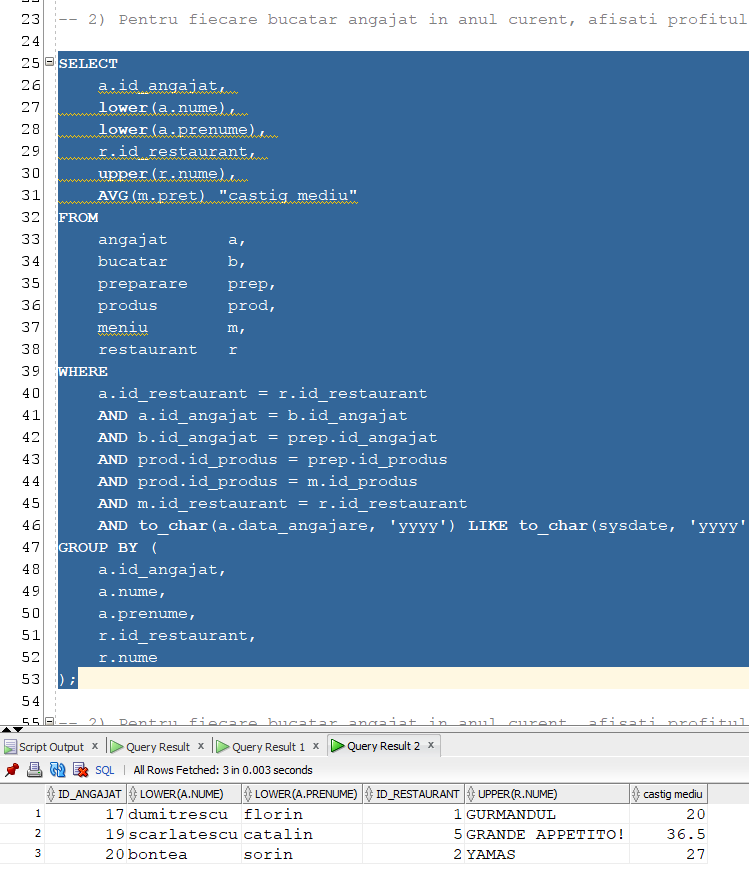
***Print-Screen:***



2) Pentru fiecare bucătar angajat în anul curent, afișați profitul mediu pe care îl poate aduce dacă ar vinde din fiecare mâncare pe care știe să o prepare exact o bucată.

|  |
| --- |
| SELECT  a.id\_angajat,  lower(a.nume),  lower(a.prenume),  r.id\_restaurant,  upper(r.nume),  AVG(m.pret) "castig mediu"  FROM  angajat a,  bucatar b,  preparare prep,  produs prod,  meniu m,  restaurant r  WHERE  a.id\_restaurant = r.id\_restaurant  AND a.id\_angajat = b.id\_angajat  AND b.id\_angajat = prep.id\_angajat  AND prod.id\_produs = prep.id\_produs  AND prod.id\_produs = m.id\_produs  AND m.id\_restaurant = r.id\_restaurant  AND to\_char(a.data\_angajare, 'yyyy') LIKE to\_char(sysdate, 'yyyy')  GROUP BY (  a.id\_angajat,  a.nume,  a.prenume,  r.id\_restaurant,  r.nume  ); |

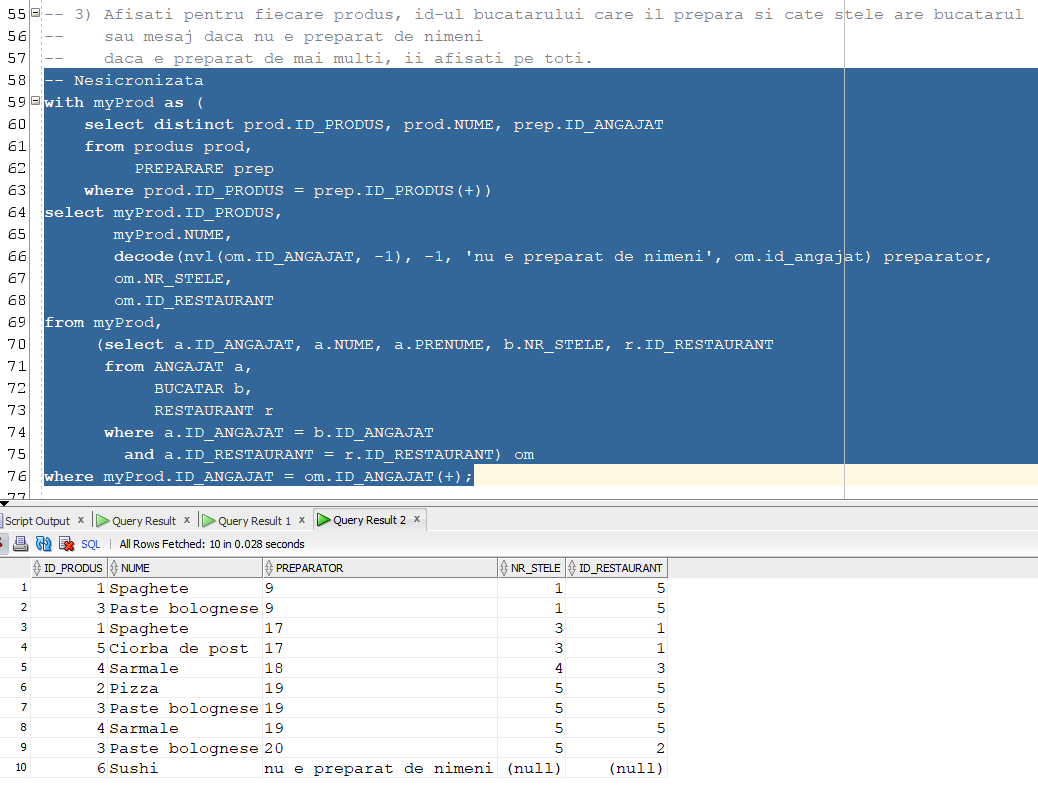
***Print-Screen:***



3) Afișați pentru fiecare produs, id-ul bucătarului care îl prepară și câte stele are bucătarul sau mesaj dacă nu e preparat de nimeni, dacă e preparat de mai mulți, îi afișați pe toți.

|  |
| --- |
| -- Nesicronizata  with myProd as (  select distinct prod.ID\_PRODUS, prod.NUME, prep.ID\_ANGAJAT  from produs prod,  PREPARARE prep  where prod.ID\_PRODUS = prep.ID\_PRODUS(+))  select myProd.ID\_PRODUS,  myProd.NUME,  decode(nvl(om.ID\_ANGAJAT, -1), -1, 'nu e preparat de nimeni', om.id\_angajat) preparator,  om.NR\_STELE,  om.ID\_RESTAURANT  from myProd,  (select a.ID\_ANGAJAT, a.NUME, a.PRENUME, b.NR\_STELE, r.ID\_RESTAURANT  from ANGAJAT a,  BUCATAR b,  RESTAURANT r  where a.ID\_ANGAJAT = b.ID\_ANGAJAT  and a.ID\_RESTAURANT = r.ID\_RESTAURANT) om  where myProd.ID\_ANGAJAT = om.ID\_ANGAJAT(+); |

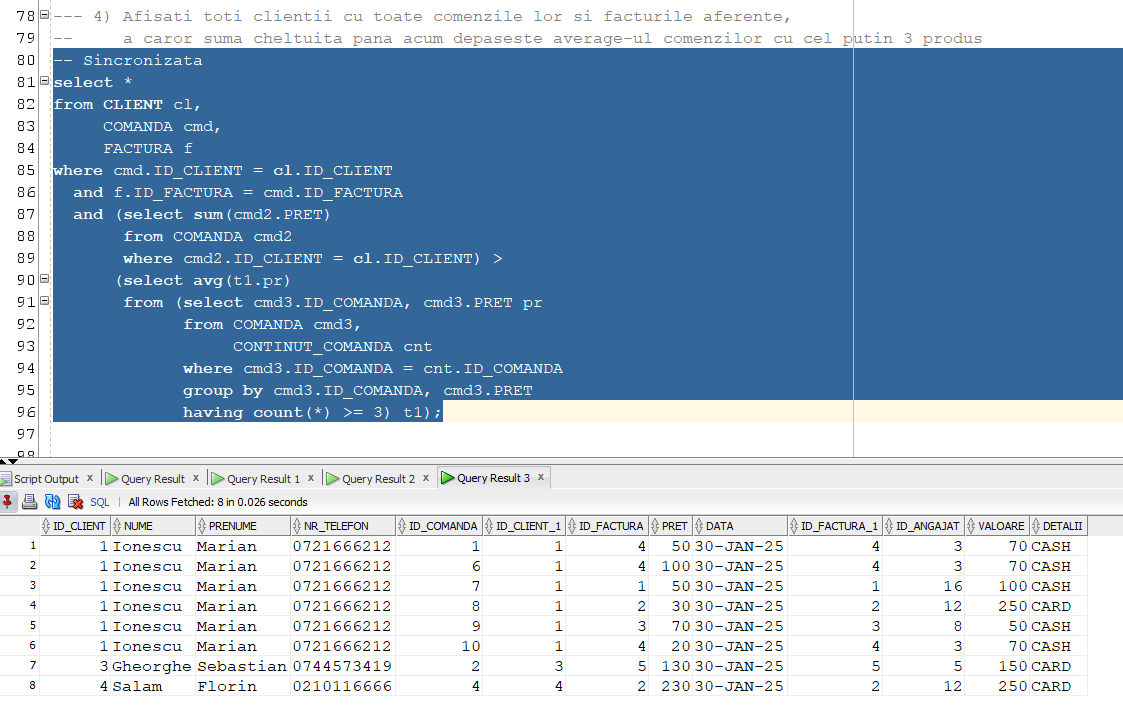
***Print-Screen:***



4) Afișați toți clienții cu toate comenzile lor și facturile aferente, a căror sumă cheltuită până acum depășește average-ul comenzilor cu cel puțin 3 produse.

|  |
| --- |
| -- Sincronizata  select \*  from CLIENT cl,  COMANDA cmd,  FACTURA f  where cmd.ID\_CLIENT = cl.ID\_CLIENT  and f.ID\_FACTURA = cmd.ID\_FACTURA  and (select sum(cmd2.PRET)  from COMANDA cmd2  where cmd2.ID\_CLIENT = cl.ID\_CLIENT) >  (select avg(t1.pr)  from (select cmd3.ID\_COMANDA, cmd3.PRET pr  from COMANDA cmd3,  CONTINUT\_COMANDA cnt  where cmd3.ID\_COMANDA = cnt.ID\_COMANDA  group by cmd3.ID\_COMANDA, cmd3.PRET  having count(\*) >= 3) t1); |

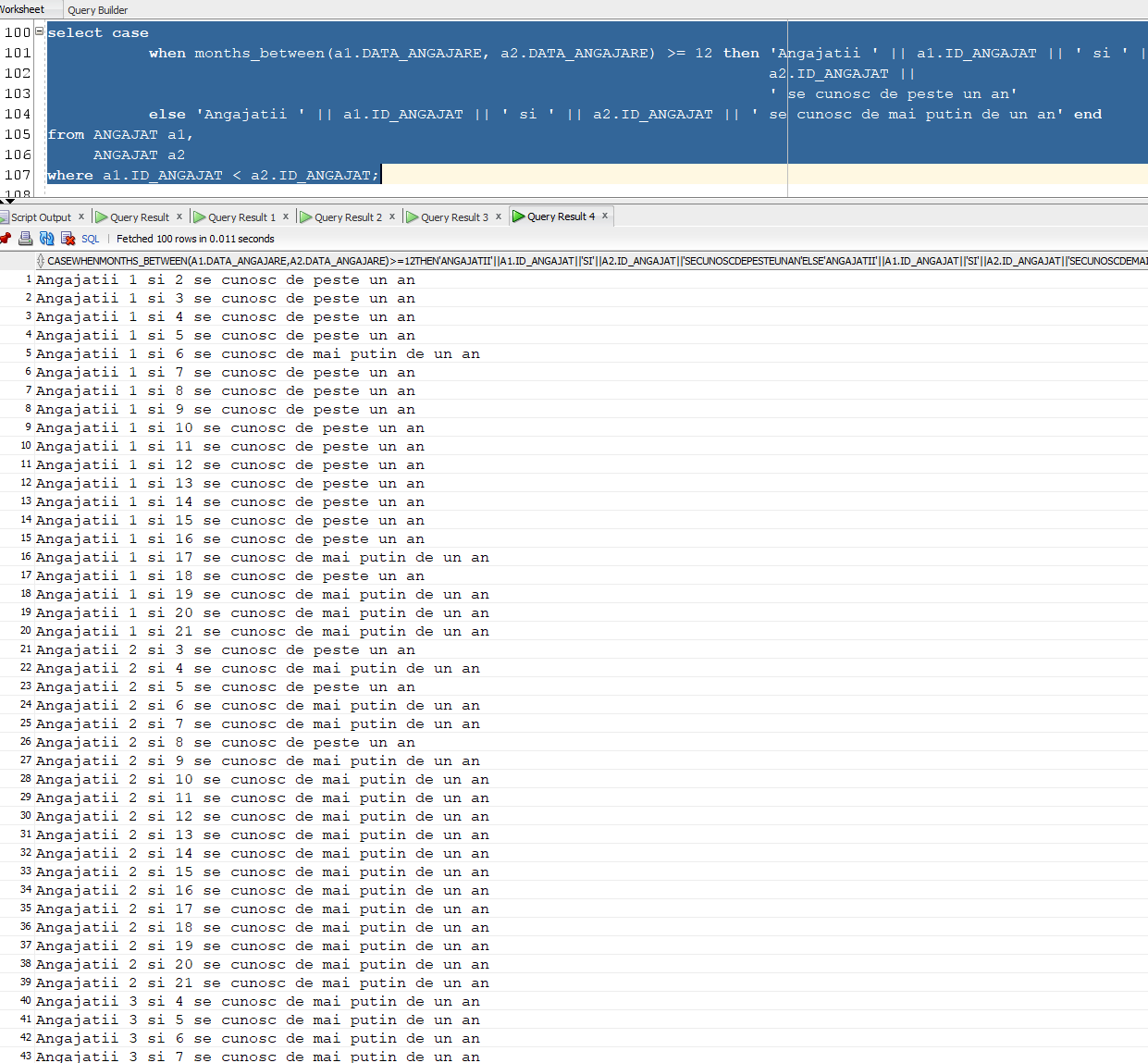
***Print-Screen:***



5) Pentru fiecare pereche de angajați, afișați dacă aceștia se cunosc de cel puțin un an.

|  |
| --- |
| select case  when months\_between(a1.DATA\_ANGAJARE, a2.DATA\_ANGAJARE) >= 12 then 'Angajatii ' || a1.ID\_ANGAJAT || ' si ' ||  a2.ID\_ANGAJAT ||  ' se cunosc de peste un an'  else 'Angajatii ' || a1.ID\_ANGAJAT || ' si ' || a2.ID\_ANGAJAT || ' se cunosc de mai putin de un an' end  from ANGAJAT a1,  ANGAJAT a2  where a1.ID\_ANGAJAT < a2.ID\_ANGAJAT; |

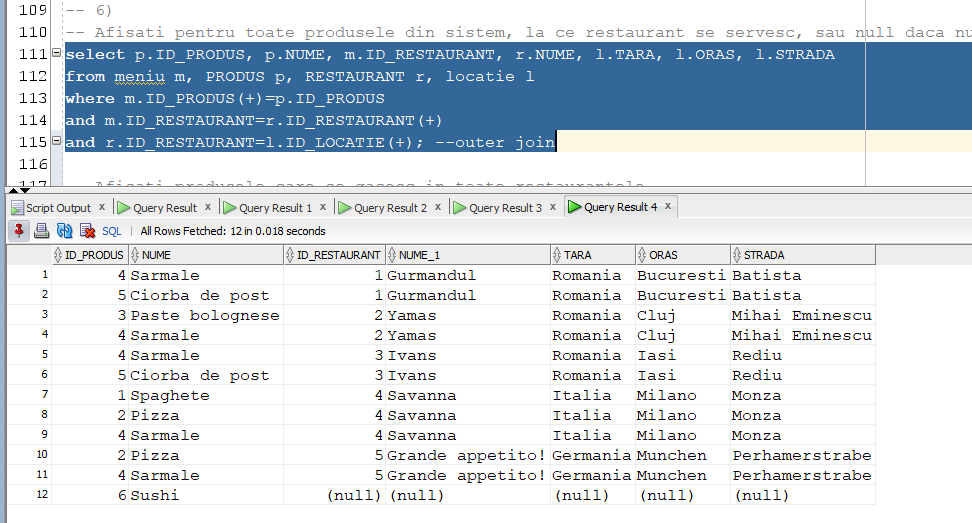
***Print-Screen:***

******

6) Afișați pentru toate produsele din sistem, la ce restaurant se servesc, sau null dacă nu e servit nicăieri.

|  |
| --- |
| select p.ID\_PRODUS, p.NUME, m.ID\_RESTAURANT, r.NUME, l.TARA, l.ORAS, l.STRADA  from meniu m, PRODUS p, RESTAURANT r, locatie l  where m.ID\_PRODUS(+)=p.ID\_PRODUS  and m.ID\_RESTAURANT=r.ID\_RESTAURANT(+)  and r.ID\_RESTAURANT=l.ID\_LOCATIE(+); --outer join |

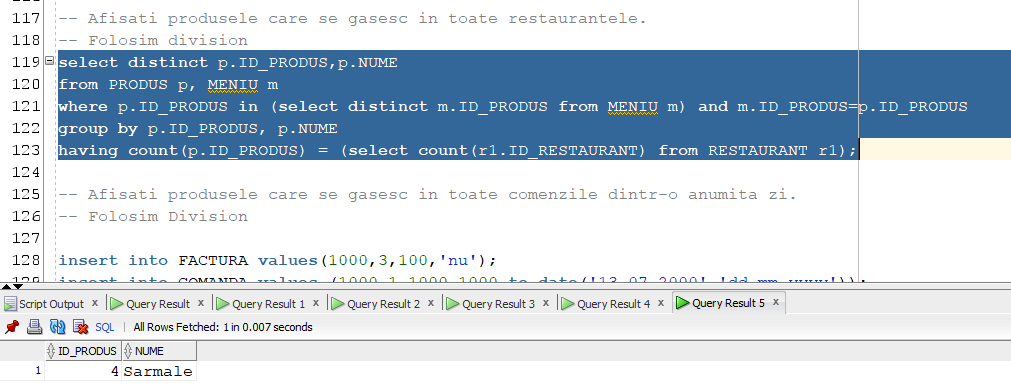
***Print-Screen:***



Afișați produsele care se găsesc în toate restaurantele.

|  |
| --- |
| -- Folosim division  select distinct p.ID\_PRODUS,p.NUME  from PRODUS p, MENIU m  where p.ID\_PRODUS in (select distinct m.ID\_PRODUS from MENIU m) and m.ID\_PRODUS=p.ID\_PRODUS  group by p.ID\_PRODUS, p.NUME  having count(p.ID\_PRODUS) = (select count(r1.ID\_RESTAURANT) from RESTAURANT r1); |

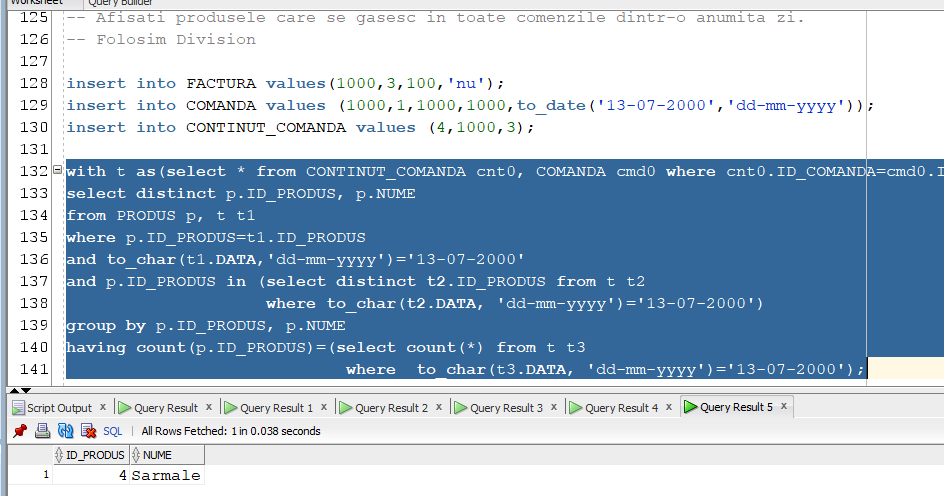
***Print-Screen:***



Afișați produsele care se găsesc în toate comenzile dintr-o anumită zi. Folosim Division.

|  |
| --- |
| insert into FACTURA values(1000,3,100,'nu');  insert into COMANDA values (1000,1,1000,1000,to\_date('13-07-2000','dd-mm-yyyy'));  insert into CONTINUT\_COMANDA values (4,1000,3);  with t as(select \* from CONTINUT\_COMANDA cnt0, COMANDA cmd0 where cnt0.ID\_COMANDA=cmd0.ID\_COMANDA)  select distinct p.ID\_PRODUS, p.NUME  from PRODUS p, t t1  where p.ID\_PRODUS=t1.ID\_PRODUS  and to\_char(t1.DATA,'dd-mm-yyyy')='13-07-2000'  and p.ID\_PRODUS in (select distinct t2.ID\_PRODUS from t t2  where to\_char(t2.DATA, 'dd-mm-yyyy')='13-07-2000')  group by p.ID\_PRODUS, p.NUME  having count(p.ID\_PRODUS)=(select count(\*) from t t3  where to\_char(t3.DATA, 'dd-mm-yyyy')='13-07-2000'); |

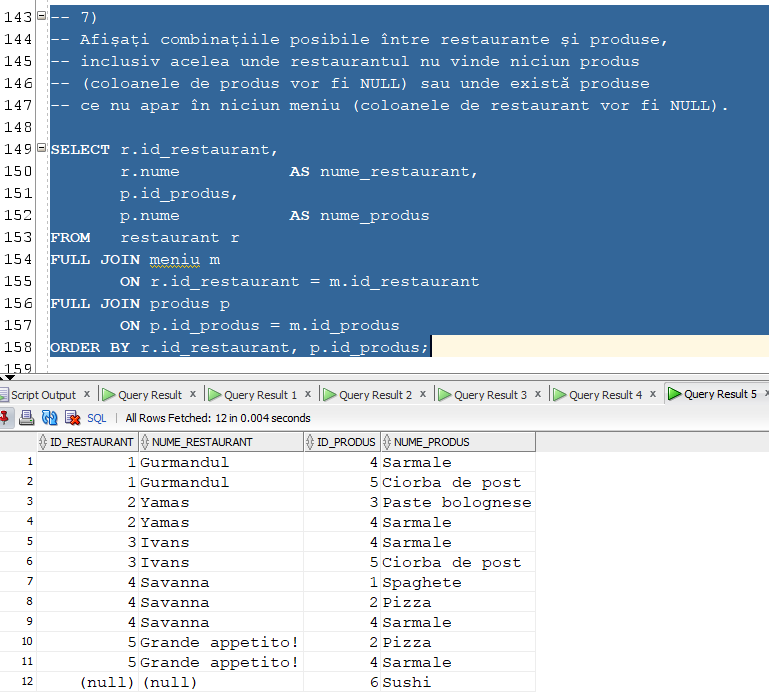
***Print-Screen:***



7) Afișați combinațiile posibile între restaurante și produse, inclusiv acelea unde restaurantul nu vinde niciun produs (coloanele de produs vor fi NULL) sau unde există produse ce nu apar în niciun meniu (coloanele de restaurant vor fi NULL).

|  |
| --- |
| SELECT r.id\_restaurant,  r.nume AS nume\_restaurant,  p.id\_produs,  p.nume AS nume\_produs  FROM restaurant r  FULL JOIN meniu m  ON r.id\_restaurant = m.id\_restaurant  FULL JOIN produs p  ON p.id\_produs = m.id\_produs  ORDER BY r.id\_restaurant, p.id\_produs; |

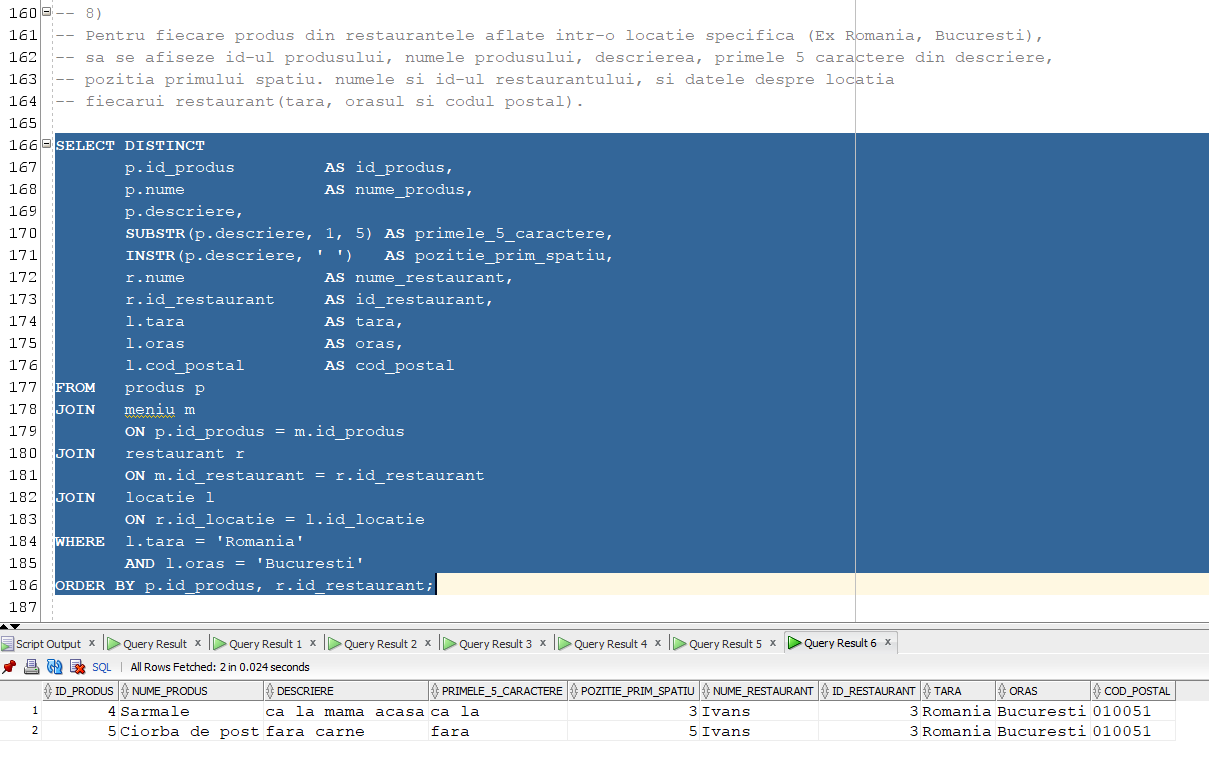
***Print-Screen:***



8) Pentru fiecare produs din restaurantele aflate într-o locație specifică (Ex România, București), să se afișeze id-ul produsului, numele produsului, descrierea, primele 5 caractere din descriere, poziția primului spațiu. numele și id-ul restaurantului, și datele despre locația fiecărui restaurant(țara, orașul și codul poștal).

|  |
| --- |
| SELECT DISTINCT  p.id\_produs AS id\_produs,  p.nume AS nume\_produs,  p.descriere,  SUBSTR(p.descriere, 1, 5) AS primele\_5\_caractere,  INSTR(p.descriere, ' ') AS pozitie\_prim\_spatiu,  r.nume AS nume\_restaurant,  r.id\_restaurant AS id\_restaurant,  l.tara AS tara,  l.oras AS oras,  l.cod\_postal AS cod\_postal  FROM produs p  JOIN meniu m  ON p.id\_produs = m.id\_produs  JOIN restaurant r  ON m.id\_restaurant = r.id\_restaurant  JOIN locatie l  ON r.id\_locatie = l.id\_locatie  WHERE l.tara = 'Romania'  AND l.oras = 'Bucuresti'  ORDER BY p.id\_produs, r.id\_restaurant; |

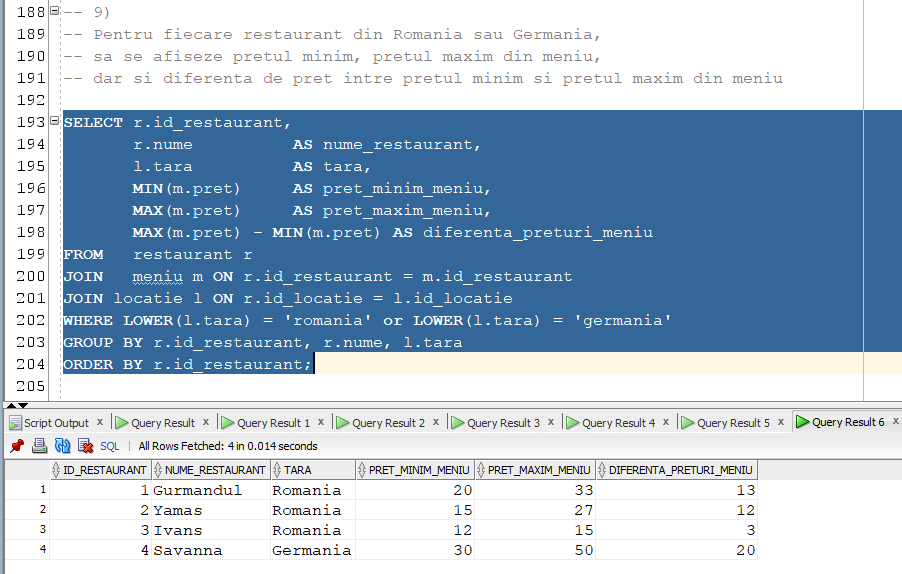
***Print-Screen:***



9) Pentru fiecare restaurant din România sau Germania, să se afișeze prețul minim, prețul maxim din meniu, dar și diferența de preț între prețul minim și prețul maxim din meniu.

|  |
| --- |
| SELECT r.id\_restaurant,  r.nume AS nume\_restaurant,  l.tara AS tara,  MIN(m.pret) AS pret\_minim\_meniu,  MAX(m.pret) AS pret\_maxim\_meniu,  MAX(m.pret) - MIN(m.pret) AS diferenta\_preturi\_meniu  FROM restaurant r  JOIN meniu m ON r.id\_restaurant = m.id\_restaurant  JOIN locatie l ON r.id\_locatie = l.id\_locatie  WHERE LOWER(l.tara) = 'romania' or LOWER(l.tara) = 'germania'  GROUP BY r.id\_restaurant, r.nume, l.tara  ORDER BY r.id\_restaurant; |

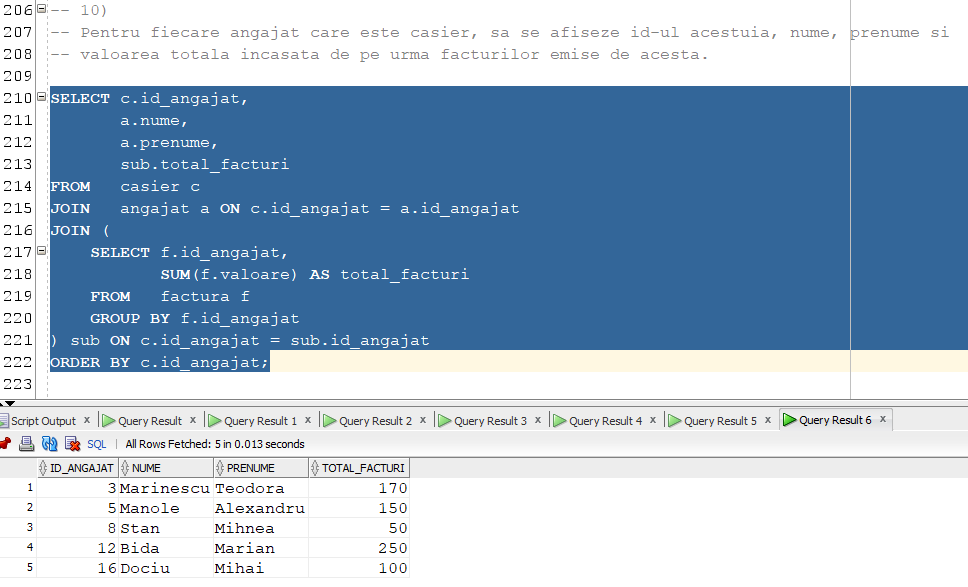
***Print-Screen:***



10) Pentru fiecare angajat care este casier, să se afișeze id-ul acestuia, nume, prenume și valoarea totală încasată de pe urma facturilor emise de acesta.

|  |
| --- |
| SELECT c.id\_angajat,  a.nume,  a.prenume,  sub.total\_facturi  FROM casier c  JOIN angajat a ON c.id\_angajat = a.id\_angajat  JOIN (  SELECT f.id\_angajat,  SUM(f.valoare) AS total\_facturi  FROM factura f  GROUP BY f.id\_angajat  ) sub ON c.id\_angajat = sub.id\_angajat  ORDER BY c.id\_angajat; |

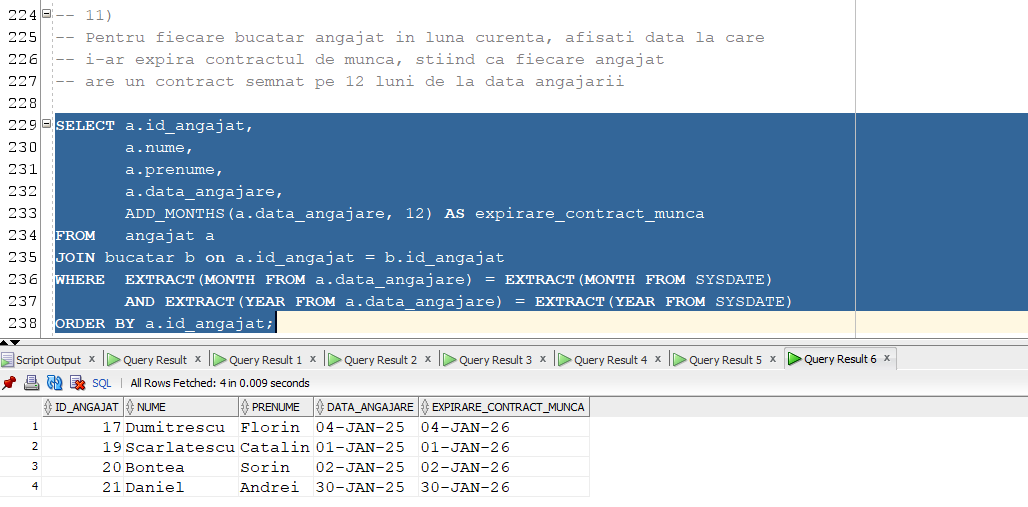
***Print-Screen:***



11) Pentru fiecare bucătar angajat în luna curentă, afișați data la care i-ar expira contractul de muncă, știind că fiecare angajat are un contract semnat pe 12 luni de la data angajării.

|  |
| --- |
| SELECT a.id\_angajat,  a.nume,  a.prenume,  a.data\_angajare,  ADD\_MONTHS(a.data\_angajare, 12) AS expirare\_contract\_munca  FROM angajat a  JOIN bucatar b on a.id\_angajat = b.id\_angajat  WHERE EXTRACT(MONTH FROM a.data\_angajare) = EXTRACT(MONTH FROM SYSDATE)  AND EXTRACT(YEAR FROM a.data\_angajare) = EXTRACT(YEAR FROM SYSDATE)  ORDER BY a.id\_angajat; |

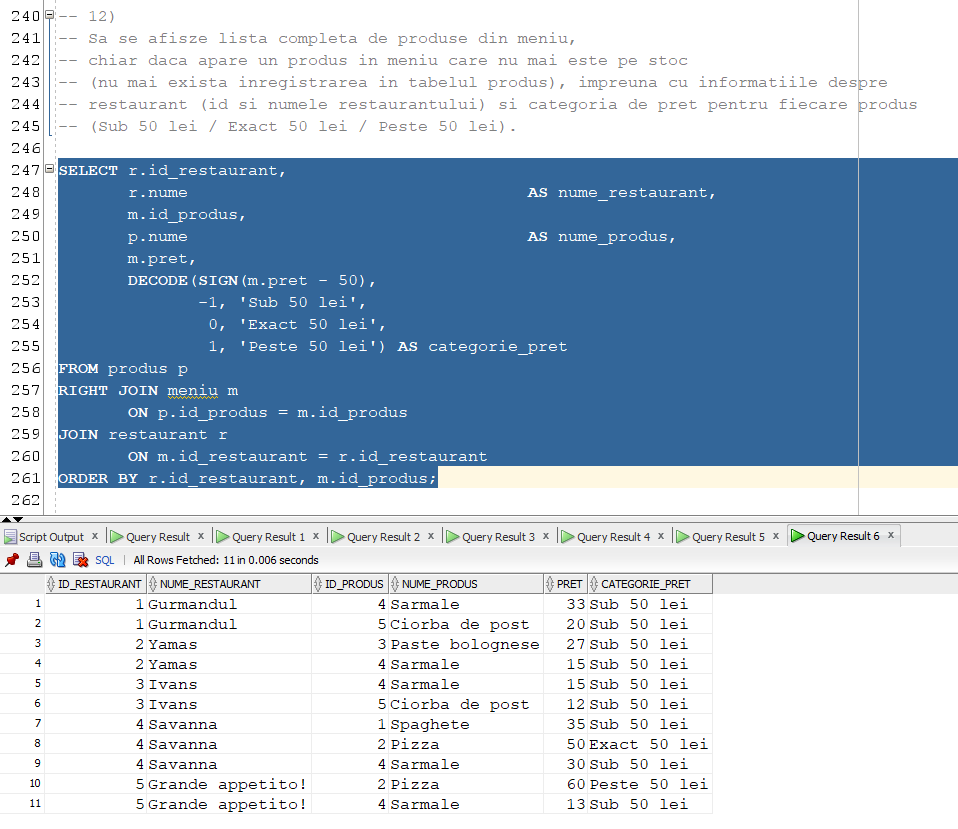
***Print-Screen:***



12) Să se afișeze lista completă de produse din meniu, chiar dacă apare un produs în meniu care nu mai este pe stoc (nu mai există înregistrarea în tabelul produs), împreună cu informațiile despre restaurant (id și numele restaurantului) și categoria de preț pentru fiecare produs (Sub 50 lei / Exact 50 lei / Peste 50 lei).

|  |
| --- |
| SELECT r.id\_restaurant,  r.nume AS nume\_restaurant,  m.id\_produs,  p.nume AS nume\_produs,  m.pret,  DECODE(SIGN(m.pret - 50),  -1, 'Sub 50 lei',  0, 'Exact 50 lei',  1, 'Peste 50 lei') AS categorie\_pret  FROM produs p  RIGHT JOIN meniu m  ON p.id\_produs = m.id\_produs  JOIN restaurant r  ON m.id\_restaurant = r.id\_restaurant  ORDER BY r.id\_restaurant, m.id\_produs; |

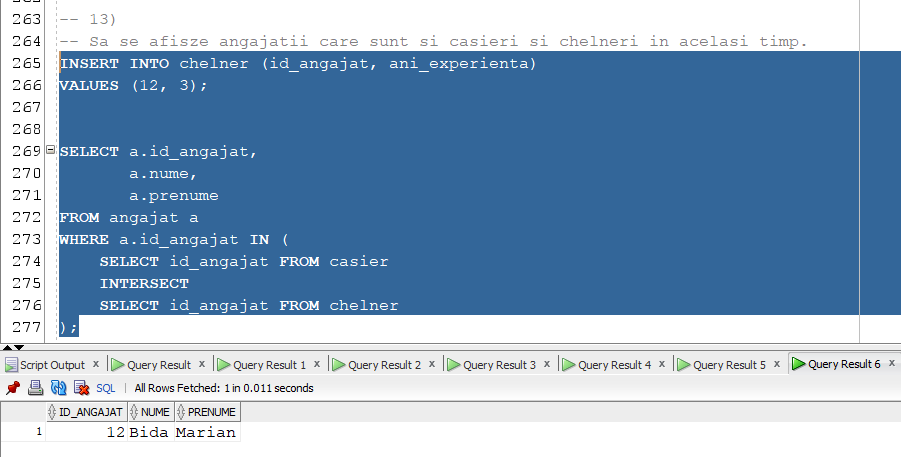
***Print-Screen:***



13) Să se afișeze angajații care sunt și casieri și chelneri în același timp.

|  |
| --- |
| INSERT INTO chelner (id\_angajat, ani\_experienta)  VALUES (12, 3);  SELECT a.id\_angajat,  a.nume,  a.prenume  FROM angajat a  WHERE a.id\_angajat IN (  SELECT id\_angajat FROM casier  INTERSECT  SELECT id\_angajat FROM chelner  ); |

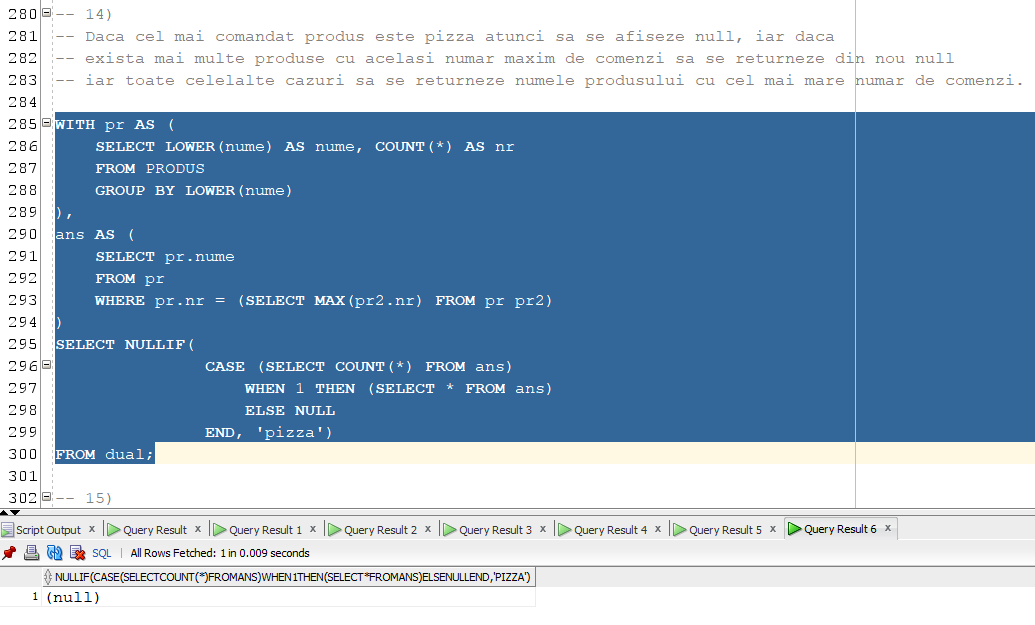
***Print-Screen:***



14) Dacă cel mai comandat produs este pizza atunci să se afișeze null, iar dacă există mai multe produse cu același număr maxim de comenzi să se returneze din nou null iar toate celelalte cazuri să se returneze numele produsului cu cel mai mare număr de comenzi.

|  |
| --- |
| WITH pr AS (  SELECT LOWER(nume) AS nume, COUNT(\*) AS nr  FROM PRODUS  GROUP BY LOWER(nume)  ),  ans AS (  SELECT pr.nume  FROM pr  WHERE pr.nr = (SELECT MAX(pr2.nr) FROM pr pr2)  )  SELECT NULLIF(  CASE (SELECT COUNT(\*) FROM ans)  WHEN 1 THEN (SELECT \* FROM ans)  ELSE NULL  END, 'pizza')  FROM dual; |

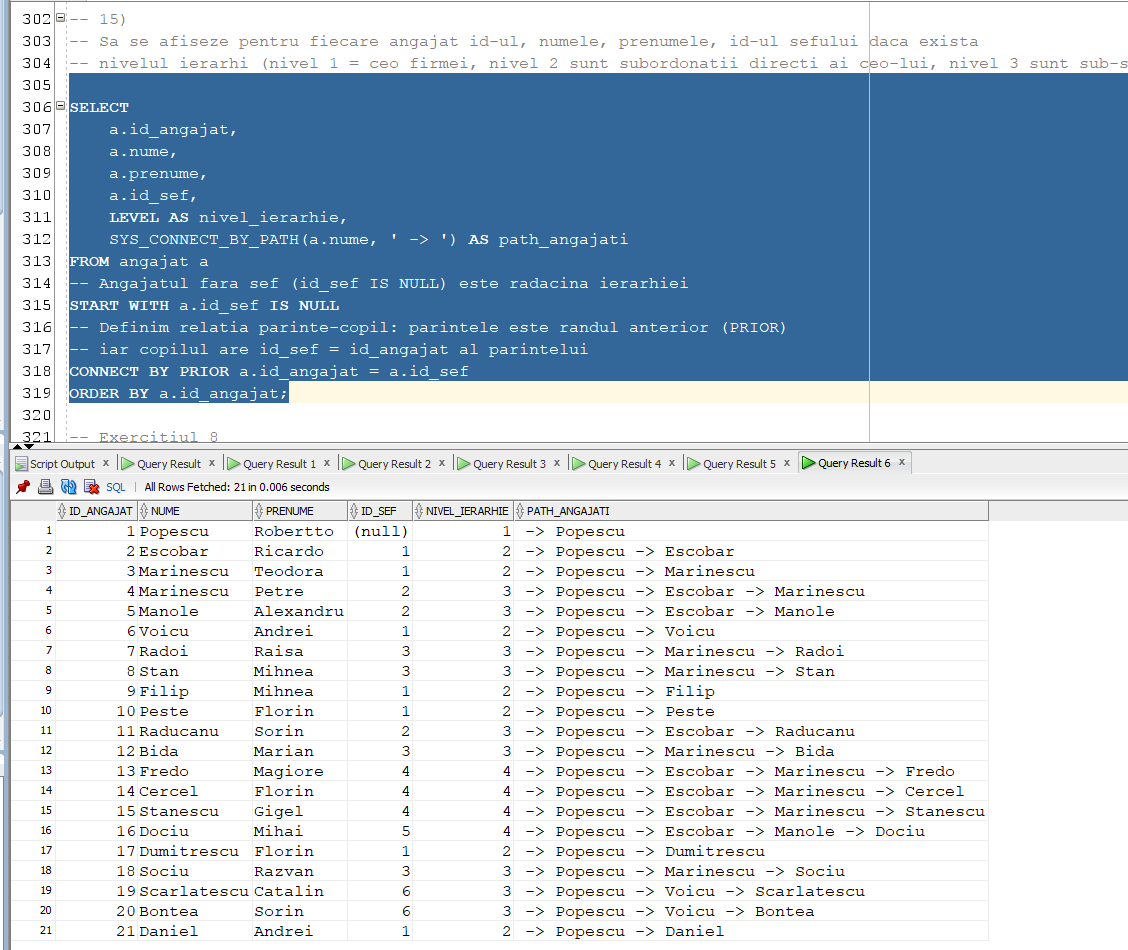
***Print-Screen:***



15) Să se afișeze pentru fiecare angajat id-ul, numele, prenumele, id-ul șefului dacă există nivelul ierarhic (nivel 1 = ceo firmei, nivel 2 sunt subordonații direcți ai ceo-lui, nivel 3 sunt sub-subordonații etc.) și drumul complet de la CEO până la angajatul curent.

|  |
| --- |
| SELECT  a.id\_angajat,  a.nume,  a.prenume,  a.id\_sef,  LEVEL AS nivel\_ierarhie,  SYS\_CONNECT\_BY\_PATH(a.nume, ' -> ') AS path\_angajati  FROM angajat a  -- Angajatul fara sef (id\_sef IS NULL) este radacina ierarhiei  START WITH a.id\_sef IS NULL  -- Definim relatia parinte-copil: parintele este randul anterior (PRIOR)  -- iar copilul are id\_sef = id\_angajat al parintelui  CONNECT BY PRIOR a.id\_angajat = a.id\_sef  ORDER BY a.id\_angajat; |

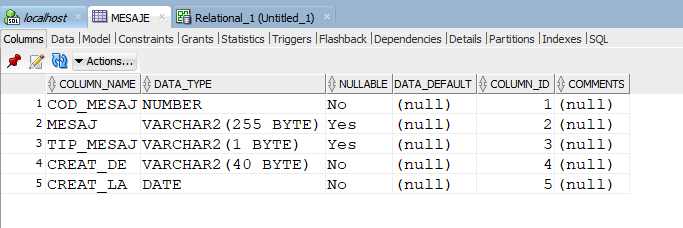
***Print-Screen:***



**8. Crearea tabelului MESAJE pentru excepții:**

|  |
| --- |
| --- CREAREA TABELULUI MESAJE ---  CREATE TABLE MESAJE  (  cod\_mesaj NUMBER,  mesaj VARCHAR2(255),  tip\_mesaj VARCHAR2(1),  creat\_de VARCHAR2(40) NOT NULL,  creat\_la DATE NOT NULL,  CONSTRAINT pk\_message PRIMARY KEY (cod\_mesaj),  CHECK (tip\_mesaj = 'E' OR tip\_mesaj = 'W' OR tip\_mesaj = 'I')  );  -- PENTRU TABELUL MESAJE --  create sequence cod\_mesaj  start with 1  increment by 1  minvalue 0  maxvalue 9999  nocycle; |

***Print-Screen:***



**9. PL / SQL:**

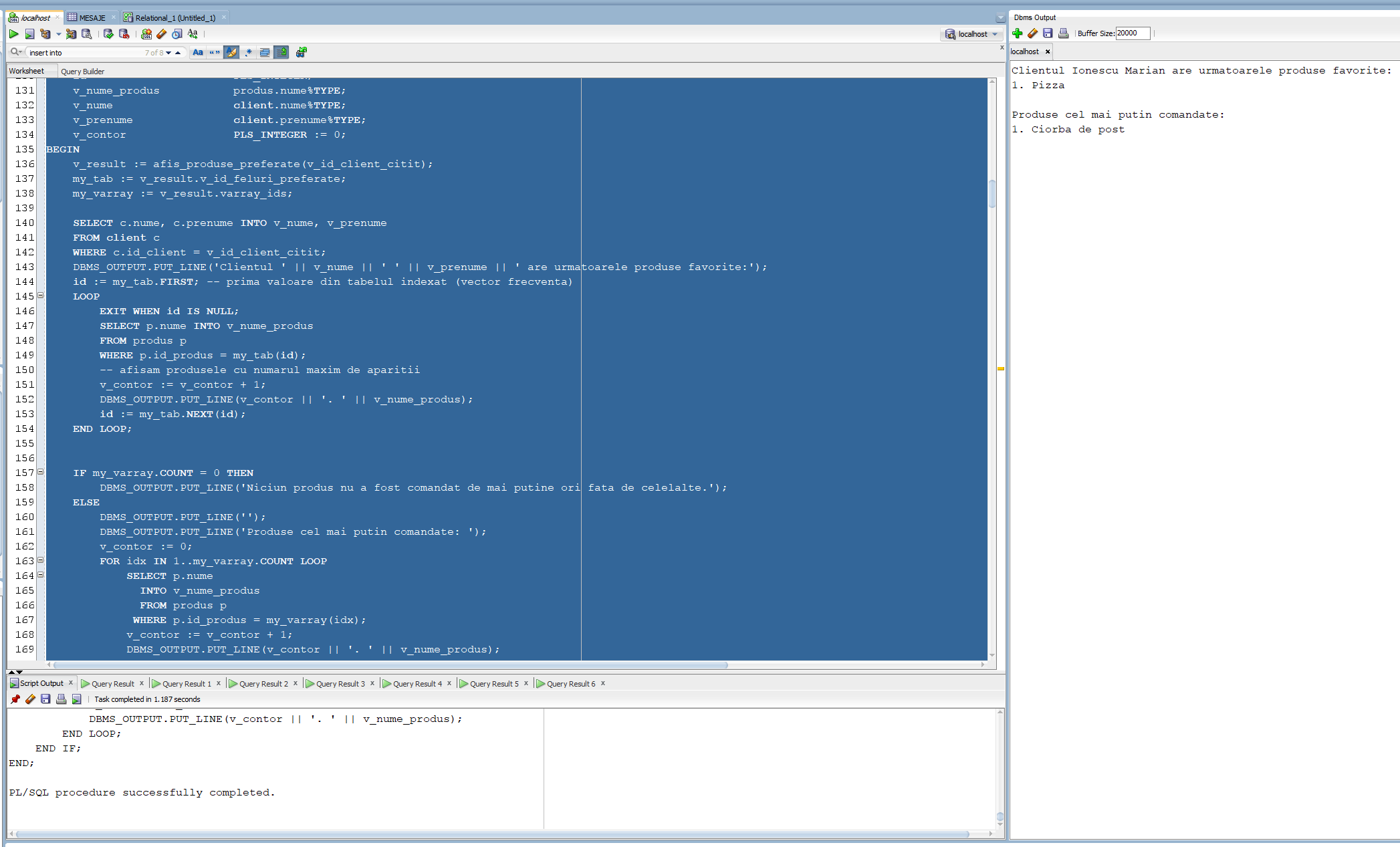
***9.1 Subprogram stocat independent (inclusiv apelare) care utilizează toate cele 3 tipuri de colecții învățate.***

Produsele preferate pentru un client citit de la tastatură și produsele cele mai puțin comandate vreodată de acesta. Se afișează numele și prenumele clientului, produsele preferate și de câte ori le-a comandat. Se afișează numele și prenumele clientului, produsele cele mai puțin comandate vreodată și de câte ori le-a comandat.

Aici voi simula cu ajutorul unui tabel indexat, un vector de frecvență, iar return type-ul funcției va fi un obiect format dintr-un tabel imbricat și un vector varray.

|  |
| --- |
| CREATE OR REPLACE TYPE tab\_imb IS  TABLE OF NUMBER(10);  /  CREATE OR REPLACE TYPE t\_array IS  VARRAY(10) OF NUMBER(10);  /  CREATE OR REPLACE TYPE result\_obj AS OBJECT (  v\_id\_feluri\_preferate tab\_imb,  varray\_ids t\_array  );  /  CREATE OR REPLACE FUNCTION afis\_produse\_preferate (my\_id\_client IN client.id\_client%TYPE)  RETURN result\_obj IS  TYPE tab\_ind IS TABLE OF PLS\_INTEGER INDEX BY PLS\_INTEGER;  v\_contor tab\_ind;  varray\_ids t\_array;  v\_index\_max PLS\_INTEGER;  v\_index\_min PLS\_INTEGER;  id PLS\_INTEGER;  v\_id\_feluri\_preferate tab\_imb;  ok BINARY\_INTEGER := 0;  client\_fara\_comenzi EXCEPTION;  v\_err\_msg VARCHAR2(4000);  BEGIN  v\_id\_feluri\_preferate := tab\_imb();  varray\_ids := t\_array();    FOR i IN (SELECT p.id\_produs, cont.numar\_produse -- CURSOR IMPLICIT  FROM produs p, continut\_comanda cont, comanda cmd  WHERE my\_id\_client = cmd.id\_client  AND cont.id\_produs = p.id\_produs  AND cmd.id\_comanda = cont.id\_comanda) LOOP BEGIN  v\_contor(i.id\_produs) := v\_contor(i.id\_produs) + i.numar\_produse; -- simulez un vector de frecventa, unde tin de cate ori a fost comandat produsul respectiv  IF ok = 1 THEN  IF v\_contor(i.id\_produs) > v\_contor(v\_index\_max) THEN  v\_index\_max := i.id\_produs;  END IF;  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN -- intr-un tabel indexat daca nu exista indexul respectiv  -- se arunca exceptia no data found  v\_contor(i.id\_produs) := i.numar\_produse; -- initializam numarul de aparitii cu 1  IF ok = 0 THEN -- daca e primul produs selectat de cursor  ok := 1;  v\_index\_max := i.id\_produs;  END IF;  END;  END LOOP;  IF ok = 0 THEN  RAISE client\_fara\_comenzi;  ELSE  v\_index\_min := v\_contor.FIRST;    IF v\_index\_min IS NOT NULL THEN  id := v\_index\_min;  LOOP  EXIT WHEN id IS NULL;  IF v\_contor(id) < v\_contor(v\_index\_min) THEN  v\_index\_min := id;  END IF;    id := v\_contor.NEXT(id);  END LOOP;  END IF;    IF v\_contor(v\_index\_max) != v\_contor(v\_index\_min) THEN  -- DBMS\_OUTPUT.PUT\_LINE('v\_index\_max: ' || v\_index\_max);  -- DBMS\_OUTPUT.PUT\_LINE('v\_contor(v\_index\_max): ' || v\_contor(v\_index\_max));  -- DBMS\_OUTPUT.PUT\_LINE('v\_index\_min: ' || v\_index\_min);  -- DBMS\_OUTPUT.PUT\_LINE('v\_contor(v\_index\_min): ' || v\_contor(v\_index\_min));  id := v\_contor.first; -- prima valoare din tabelul indexat (vector frecventa)  LOOP  EXIT WHEN id IS NULL;  IF v\_contor(id) = v\_contor(v\_index\_max) THEN  v\_id\_feluri\_preferate.extend;  v\_id\_feluri\_preferate(v\_id\_feluri\_preferate.LAST) := id;  END IF;    IF v\_contor(id) = v\_contor(v\_index\_min) THEN  IF varray\_ids.COUNT < varray\_ids.LIMIT THEN  varray\_ids.EXTEND;  varray\_ids(varray\_ids.LAST) := id;  END IF;  END IF;  id := v\_contor.NEXT(id);  END LOOP;  ELSE  id := v\_contor.first; -- prima valoare din tabelul indexat (vector frecventa)  LOOP  EXIT WHEN id IS NULL;  IF v\_contor(id) = v\_contor(v\_index\_max) THEN  v\_id\_feluri\_preferate.extend;  v\_id\_feluri\_preferate(v\_id\_feluri\_preferate.LAST) := id;  END IF;  id := v\_contor.NEXT(id);  END LOOP;  END IF;  END IF;  RETURN result\_obj(v\_id\_feluri\_preferate, varray\_ids);  EXCEPTION  WHEN client\_fara\_comenzi THEN  INSERT INTO mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  VALUES (cod\_mesaj.NEXTVAL, 'Pentru clientul dat de la tastatura cu id ' || my\_id\_client || ' nu s-a gasit nicio comanda!', 'W', USER, SYSDATE);  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Acest client nu a comandat nimic in viata lui!');  WHEN OTHERS THEN  v\_err\_msg := 'Alta eroare in afis\_produse\_preferate: ' || SQLERRM;  INSERT INTO mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  VALUES (cod\_mesaj.NEXTVAL, 'Alta eroare in afis\_produse\_preferate: ' || v\_err\_msg, 'E', USER, SYSDATE);  COMMIT;  END;  / |

***Print-Screen:***



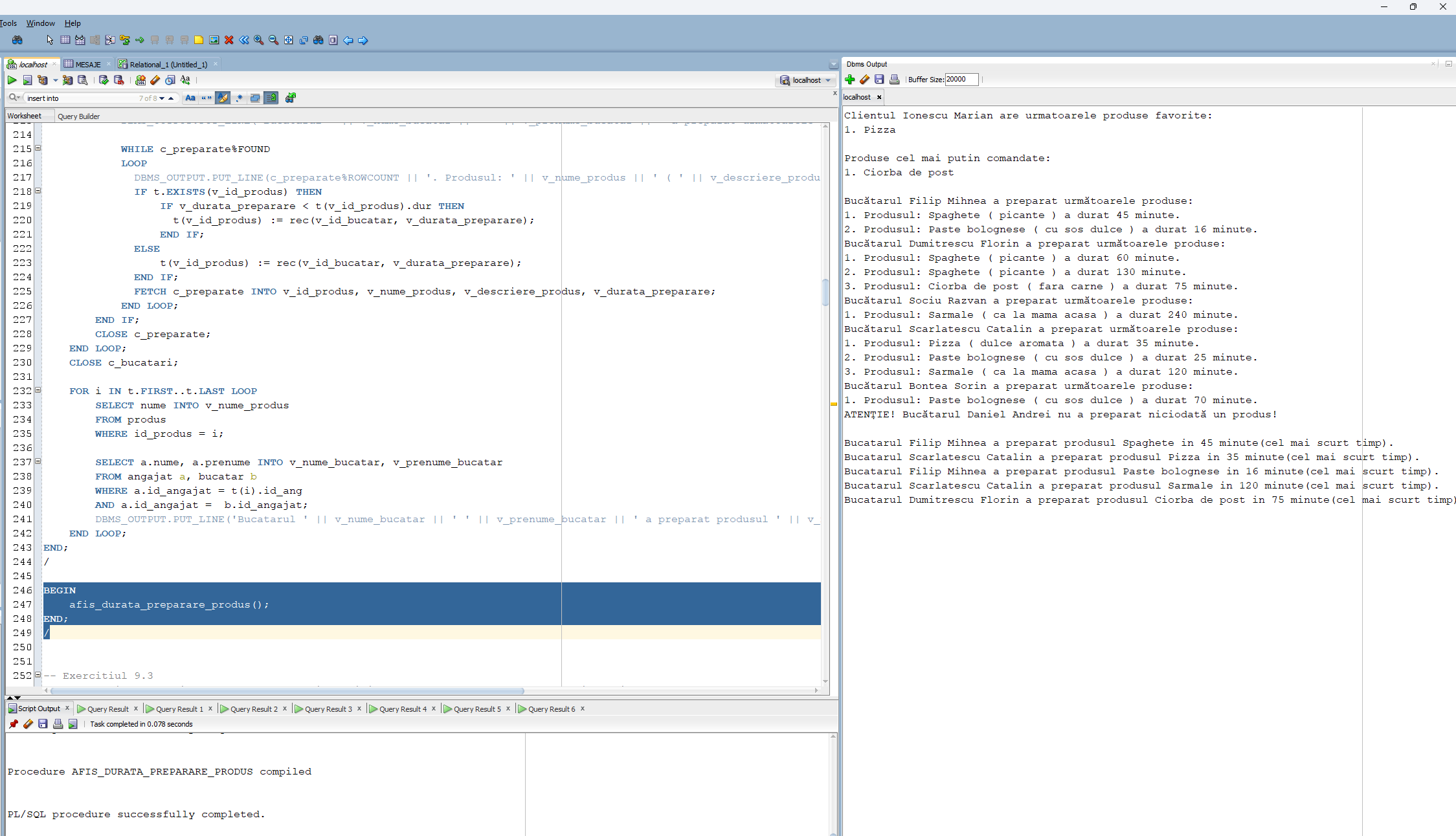
***9.2 Subprogram stocat independent (inclusiv apelare) care utilizează toate cele 2 tipuri de cursoare învățate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor.***

Să se afișeze pentru fiecare bucătar numele și prenumele acestuia, dacă a preparat vreun produs de când s-a angajat.

Dacă a preparat minim un produs se vor afișa date corespunzătoare legate de acesta (numele produsului, descrierea acestuia și cât a durat să îl prepare).

|  |
| --- |
| CREATE OR REPLACE PROCEDURE afis\_durata\_preparare\_produs  IS  CURSOR c\_bucatari IS SELECT a.id\_angajat, a.nume, a.prenume, b.nr\_stele  FROM angajat a  JOIN bucatar b ON a.id\_angajat = b.id\_angajat;  CURSOR c\_preparate(p\_id\_angajat in angajat.id\_angajat%TYPE) IS SELECT p.id\_produs, p.nume, p.descriere, prep.durata  FROM preparare prep  JOIN produs p ON p.id\_produs = prep.id\_produs  WHERE prep.id\_angajat = p\_id\_angajat;  TYPE rec IS RECORD (id\_ang angajat.id\_angajat%TYPE, dur preparare.durata%TYPE);  TYPE tab\_ind IS TABLE OF rec INDEX BY PLS\_INTEGER;  t tab\_ind;  v\_id\_bucatar angajat.id\_angajat%TYPE;  v\_nume\_bucatar angajat.nume%TYPE;  v\_prenume\_bucatar angajat.prenume%TYPE;  v\_nr\_stele\_bucatar bucatar.nr\_stele%TYPE;  v\_durata\_preparare preparare.durata%TYPE;  v\_id\_produs produs.id\_produs%TYPE;  v\_nume\_produs produs.nume%TYPE;  v\_descriere\_produs produs.descriere%TYPE;  BEGIN  OPEN c\_bucatari;  LOOP  FETCH c\_bucatari INTO v\_id\_bucatar, v\_nume\_bucatar, v\_prenume\_bucatar, v\_nr\_stele\_bucatar;  EXIT WHEN c\_bucatari%NOTFOUND;    OPEN c\_preparate(v\_id\_bucatar);  FETCH c\_preparate INTO v\_id\_produs, v\_nume\_produs, v\_descriere\_produs, v\_durata\_preparare;    IF c\_preparate%NOTFOUND THEN  DBMS\_OUTPUT.PUT\_LINE('ATENȚIE! Bucătarul ' || v\_nume\_bucatar || ' ' || v\_prenume\_bucatar || ' nu a preparat niciodată un produs!');  DBMS\_OUTPUT.NEW\_LINE();  ELSE  DBMS\_OUTPUT.PUT\_LINE('Bucătarul ' || v\_nume\_bucatar || ' ' || v\_prenume\_bucatar || ' a preparat următoarele produse:');    WHILE c\_preparate%FOUND  LOOP  DBMS\_OUTPUT.PUT\_LINE(c\_preparate%ROWCOUNT || '. Produsul: ' || v\_nume\_produs || ' ( ' || v\_descriere\_produs || ' ) a durat ' || v\_durata\_preparare || ' minute.');  IF t.EXISTS(v\_id\_produs) THEN  IF v\_durata\_preparare < t(v\_id\_produs).dur THEN  t(v\_id\_produs) := rec(v\_id\_bucatar, v\_durata\_preparare);  END IF;  ELSE  t(v\_id\_produs) := rec(v\_id\_bucatar, v\_durata\_preparare);  END IF;  FETCH c\_preparate INTO v\_id\_produs, v\_nume\_produs, v\_descriere\_produs, v\_durata\_preparare;  END LOOP;  END IF;  CLOSE c\_preparate;  END LOOP;  CLOSE c\_bucatari;    FOR i IN t.FIRST..t.LAST LOOP  SELECT nume INTO v\_nume\_produs  FROM produs  WHERE id\_produs = i;    SELECT a.nume, a.prenume INTO v\_nume\_bucatar, v\_prenume\_bucatar  FROM angajat a, bucatar b  WHERE a.id\_angajat = t(i).id\_ang  AND a.id\_angajat = b.id\_angajat;  DBMS\_OUTPUT.PUT\_LINE('Bucatarul ' || v\_nume\_bucatar || ' ' || v\_prenume\_bucatar || ' a preparat produsul ' || v\_nume\_produs || ' in ' || t(i).dur || ' minute(cel mai scurt timp).');  END LOOP;  END;  /  BEGIN  afis\_durata\_preparare\_produs();  END;  / |

***Print-Screen:***

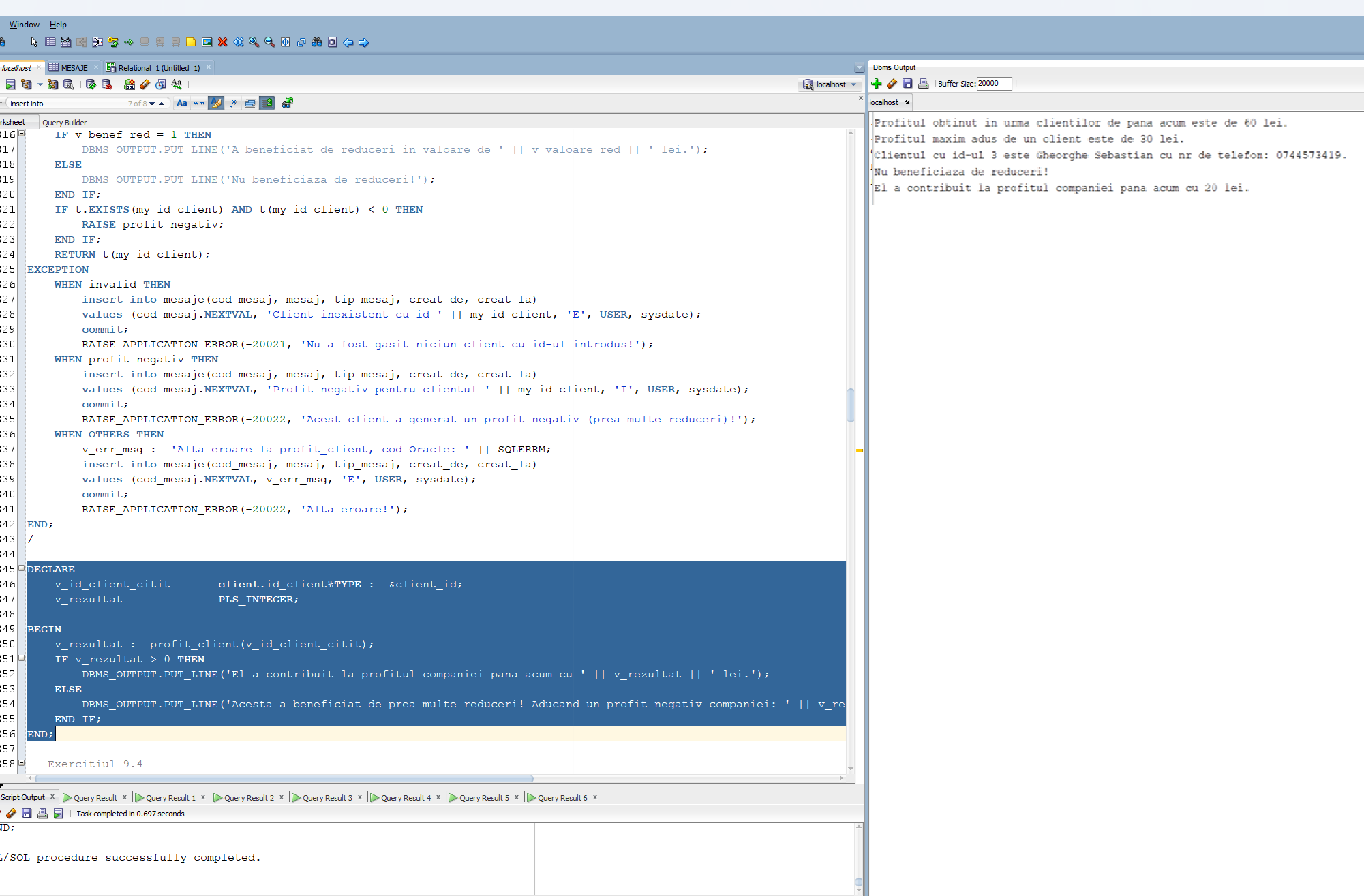


***9.3 Subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite și tratarea excepțiilor care pot apărea.***

Să se afișeze profitul generat de un client citit de la tastatură, de-a lungul timpului. Dacă acesta a beneficiat de reduceri de-a lungul timpului se va afișa valoarea totală a acestor reduceri. Dacă profitul adus de-a lungul timpului este unul negativ înseamnă că clientul a beneficiat de prea multe reduceri și se va afișa un mesaj corespunzător și suma (chiar dacă este negativă). De asemenea se va afișa profitul maxim oferit de un client și profitul obținut de companie până la momentul actual.

|  |
| --- |
| CREATE OR REPLACE FUNCTION profit\_client (my\_id\_client IN client.id\_client%TYPE)  RETURN PLS\_INTEGER IS  TYPE tab\_ind IS TABLE OF PLS\_INTEGER INDEX BY PLS\_INTEGER;  v\_nume\_client client.nume%TYPE;  v\_prenume\_client client.nume%TYPE;  v\_nr\_telefon\_client client.nr\_telefon%TYPE;  v\_profit\_maxim\_companie PLS\_INTEGER := 0;  v\_profit\_maxim\_client PLS\_INTEGER := 0;  v\_benef\_red BINARY\_INTEGER := 0;  v\_valoare\_red PLS\_INTEGER := 0;  invalid EXCEPTION;  profit\_negativ EXCEPTION;  v\_exista PLS\_INTEGER := 0;  t tab\_ind;  v\_err\_msg VARCHAR2(4000);  BEGIN  SELECT COUNT(\*) INTO v\_exista  FROM client  WHERE id\_client = my\_id\_client;  IF v\_exista = 0 THEN  RAISE invalid;  END IF;    SELECT nume, prenume, nr\_telefon INTO v\_nume\_client, v\_prenume\_client, v\_nr\_telefon\_client  FROM client  WHERE id\_client = my\_id\_client;    FOR i IN (SELECT cli.id\_client, cli.nume, cli.prenume, cmd.pret, f.valoare, f.detalii  FROM client cli, comanda cmd, factura f  WHERE cli.id\_client = cmd.id\_client  AND cmd.id\_factura = f.id\_factura) LOOP  BEGIN  IF i.id\_client = my\_id\_client AND i.valoare < i.pret THEN  v\_benef\_red := 1;  v\_valoare\_red := v\_valoare\_red + (i.pret - i.valoare);  -- ELSIF i.id\_client = my\_id\_client THEN  -- t(i.id\_client) := t(i.id\_client) + (i.valoare - i.pret);  END IF;  v\_profit\_maxim\_companie := v\_profit\_maxim\_companie + (i.valoare - i.pret);  t(i.id\_client) := t(i.id\_client) + (i.valoare - i.pret);  EXCEPTION -- pentru prima inserare in tabel  WHEN NO\_DATA\_FOUND THEN  insert into mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  values (cod\_mesaj.NEXTVAL, 'Prima inserare in tabelul pentru clientul cu id: ' || i.id\_client, 'I', USER, sysdate);  commit;  t(i.id\_client) := i.valoare - i.pret;  END;  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('Profitul obtinut in urma clientilor de pana acum este de ' || v\_profit\_maxim\_companie || ' lei.');  FOR i IN 1..t.LAST LOOP  IF t(i) > v\_profit\_maxim\_client THEN  v\_profit\_maxim\_client := t(i);  END IF;  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('Profitul maxim adus de un client este de ' || v\_profit\_maxim\_client || ' lei.');  DBMS\_OUTPUT.PUT\_LINE('Clientul cu id-ul ' || my\_id\_client || ' este ' || v\_nume\_client || ' ' || v\_prenume\_client || ' cu nr de telefon: ' || v\_nr\_telefon\_client || '.');  IF v\_benef\_red = 1 THEN  DBMS\_OUTPUT.PUT\_LINE('A beneficiat de reduceri in valoare de ' || v\_valoare\_red || ' lei.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Nu beneficiaza de reduceri!');  END IF;  IF t.EXISTS(my\_id\_client) AND t(my\_id\_client) < 0 THEN  RAISE profit\_negativ;  END IF;  RETURN t(my\_id\_client);  EXCEPTION  WHEN invalid THEN  insert into mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  values (cod\_mesaj.NEXTVAL, 'Client inexistent cu id=' || my\_id\_client, 'E', USER, sysdate);  commit;  RAISE\_APPLICATION\_ERROR(-20021, 'Nu a fost gasit niciun client cu id-ul introdus!');  WHEN profit\_negativ THEN  insert into mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  values (cod\_mesaj.NEXTVAL, 'Profit negativ pentru clientul ' || my\_id\_client, 'I', USER, sysdate);  commit;  RAISE\_APPLICATION\_ERROR(-20022, 'Acest client a generat un profit negativ (prea multe reduceri)!');  WHEN OTHERS THEN  v\_err\_msg := 'Alta eroare la profit\_client, cod Oracle: ' || SQLERRM;  insert into mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  values (cod\_mesaj.NEXTVAL, v\_err\_msg, 'E', USER, sysdate);  commit;  RAISE\_APPLICATION\_ERROR(-20022, 'Alta eroare!');  END;  /  DECLARE  v\_id\_client\_citit client.id\_client%TYPE := &client\_id;  v\_rezultat PLS\_INTEGER;    BEGIN  v\_rezultat := profit\_client(v\_id\_client\_citit);  IF v\_rezultat > 0 THEN  DBMS\_OUTPUT.PUT\_LINE('El a contribuit la profitul companiei pana acum cu ' || v\_rezultat || ' lei.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Acesta a beneficiat de prea multe reduceri! Aducand un profit negativ companiei: ' || v\_rezultat || ' lei.');  END IF;  END; |

***Print-Screen:***



Pentru id\_client = 100;

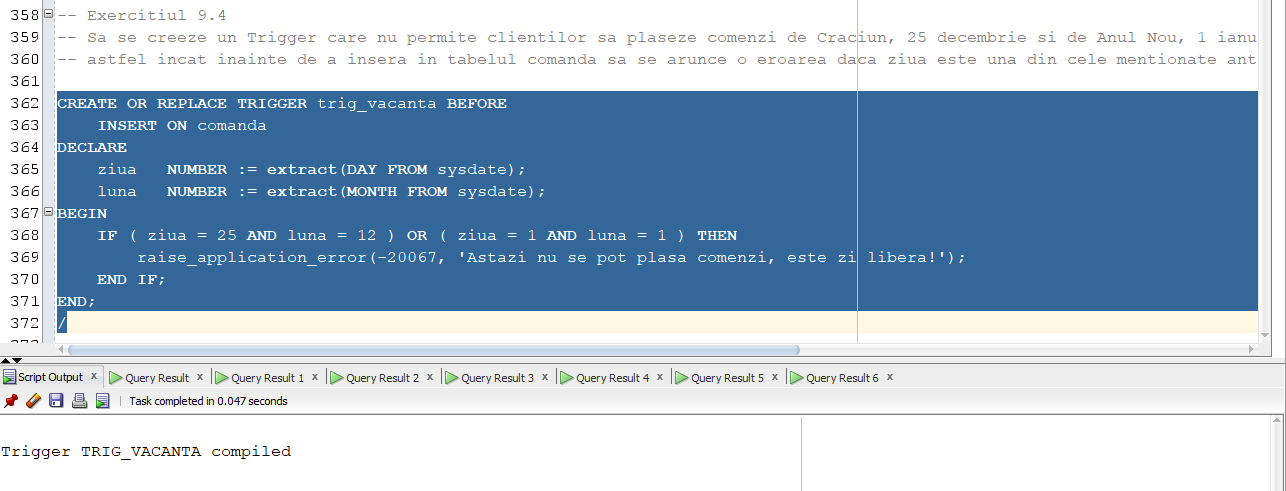


***9.4 Crearea unui Trigger LMD la nivel de comandă.***

Să se creeze un Trigger care nu permite clienților să plaseze comenzi de Crăciun, 25 decembrie și de Anul Nou, 1 ianuarie; astfel încât înainte de a insera în tabelul comandă să se arunce o eroarea dacă ziua este una din cele menționate anterior.

|  |
| --- |
| CREATE OR REPLACE TRIGGER trig\_vacanta BEFORE  INSERT ON comanda  DECLARE  ziua NUMBER := extract(DAY FROM sysdate);  luna NUMBER := extract(MONTH FROM sysdate);  BEGIN  IF ( ziua = 25 AND luna = 12 ) OR ( ziua = 1 AND luna = 1 ) THEN  raise\_application\_error(-20067, 'Astazi nu se pot plasa comenzi, este zi libera!');  END IF;  END;  / |

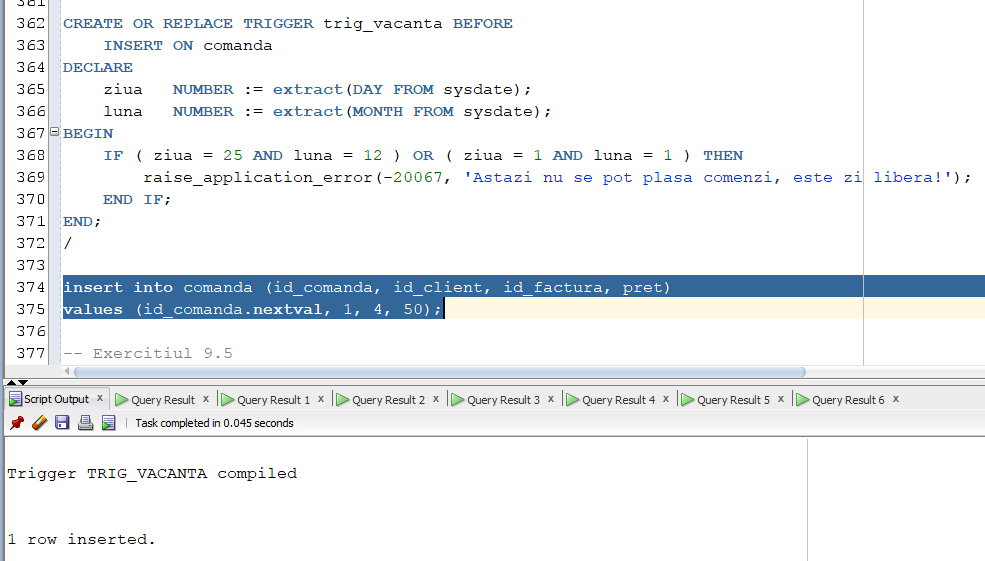
***Print-Screen:***

****

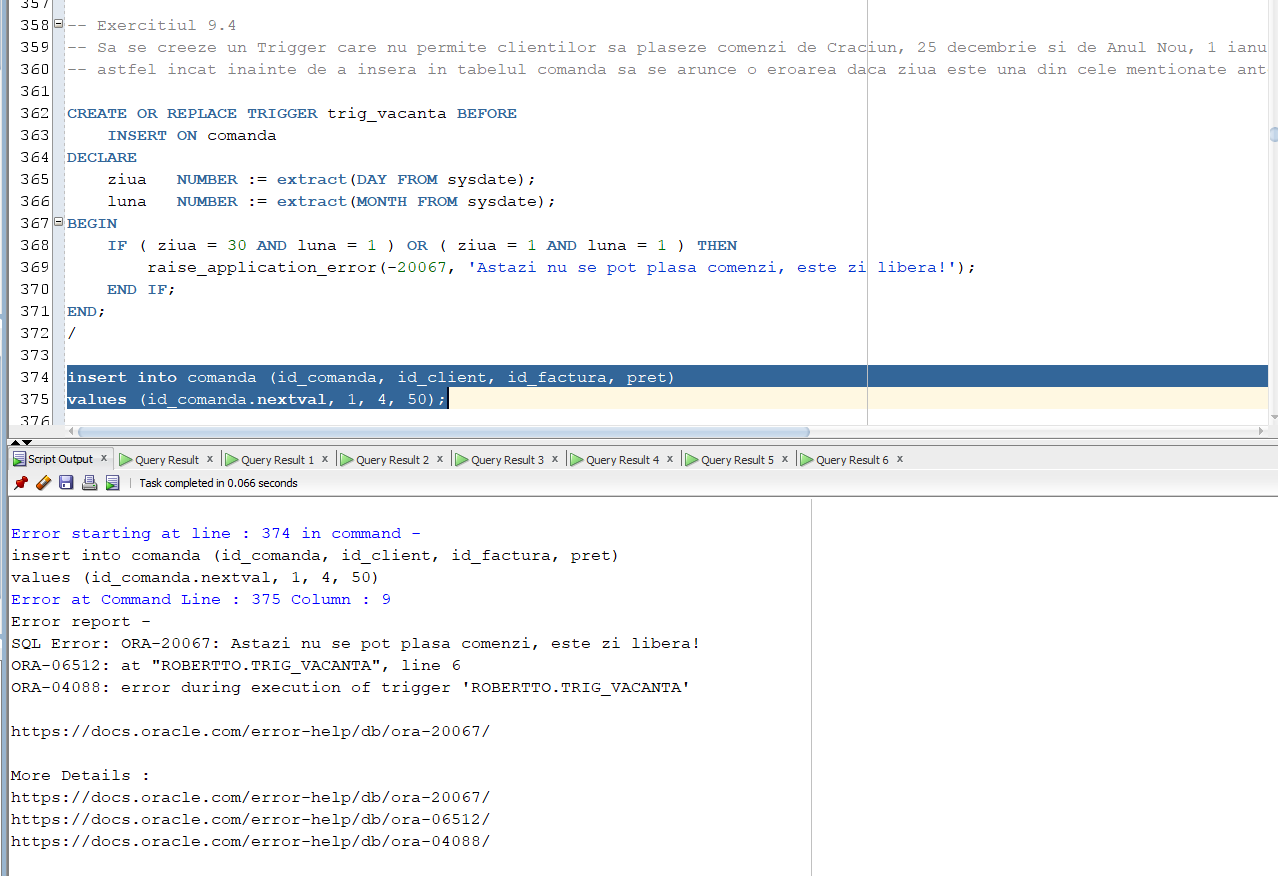
**Declanșarea acestuia:**

|  |
| --- |
| insert into comanda (id\_comanda, id\_client, id\_factura, pret)  values (id\_comanda.nextval, 1, 4, 50); |

***Print-Screen:***



Am putut insera pentru că data de astăzi este 7 ianuarie. Dacă modificăm trigger-ul la data de astăzi, 30 ianuarie, se va declanșa:

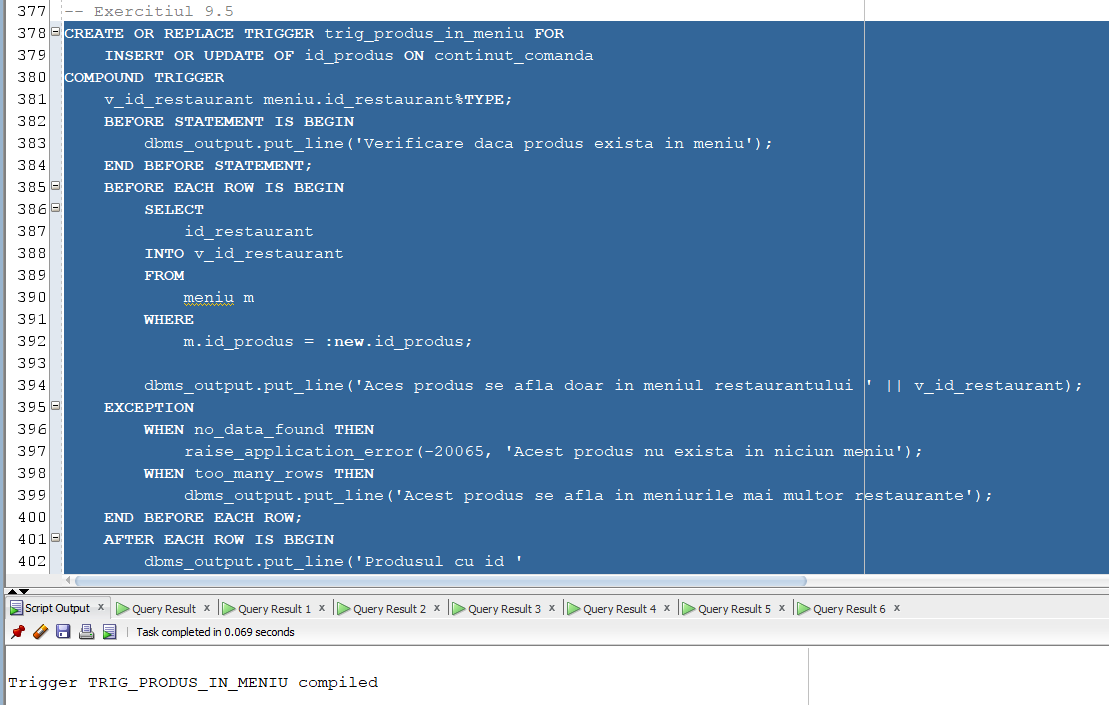


***9.5 Crearea unui Trigger LMD la nivel de linie (TRIGGER COMPOUND).***

Aici voi folosi un Trigger Compound pentru a nu putea insera un produs care nu se află în niciun meniu.

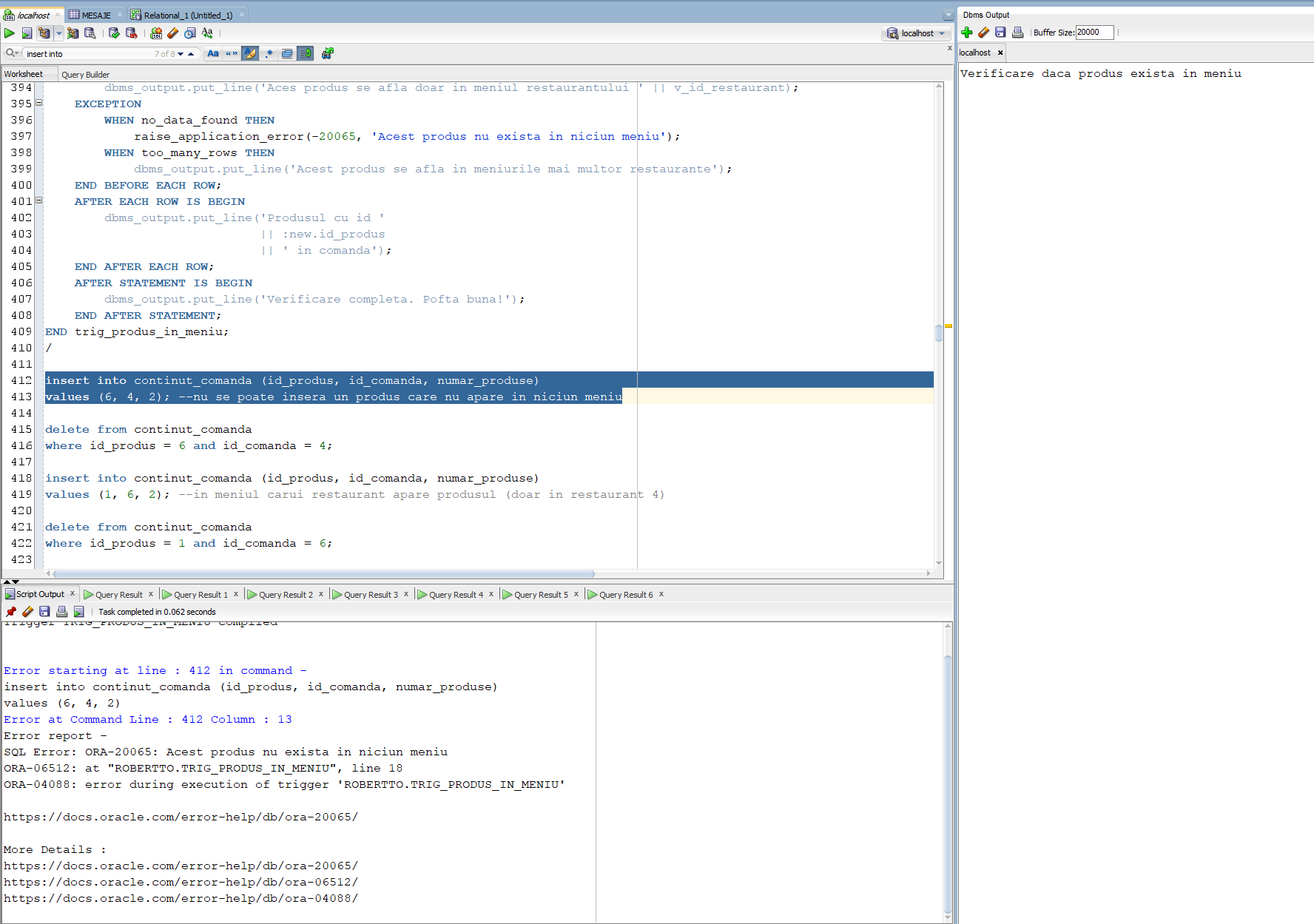
|  |
| --- |
| CREATE OR REPLACE TRIGGER trig\_produs\_in\_meniu FOR  INSERT OR UPDATE OF id\_produs ON continut\_comanda  COMPOUND TRIGGER  v\_id\_restaurant meniu.id\_restaurant%TYPE;  BEFORE STATEMENT IS BEGIN  dbms\_output.put\_line('Verificare daca produs exista in meniu');  END BEFORE STATEMENT;  BEFORE EACH ROW IS BEGIN  SELECT  id\_restaurant  INTO v\_id\_restaurant  FROM  meniu m  WHERE  m.id\_produs = :new.id\_produs;  dbms\_output.put\_line('Aces produs se afla doar in meniul restaurantului ' || v\_id\_restaurant);  EXCEPTION  WHEN no\_data\_found THEN  raise\_application\_error(-20065, 'Acest produs nu exista in niciun meniu');  WHEN too\_many\_rows THEN  dbms\_output.put\_line('Acest produs se afla in meniurile mai multor restaurante');  END BEFORE EACH ROW;  AFTER EACH ROW IS BEGIN  dbms\_output.put\_line('Produsul cu id '  || :new.id\_produs  || ' in comanda');  END AFTER EACH ROW;  AFTER STATEMENT IS BEGIN  dbms\_output.put\_line('Verificare completa. Pofta buna!');  END AFTER STATEMENT;  END trig\_produs\_in\_meniu;  / |

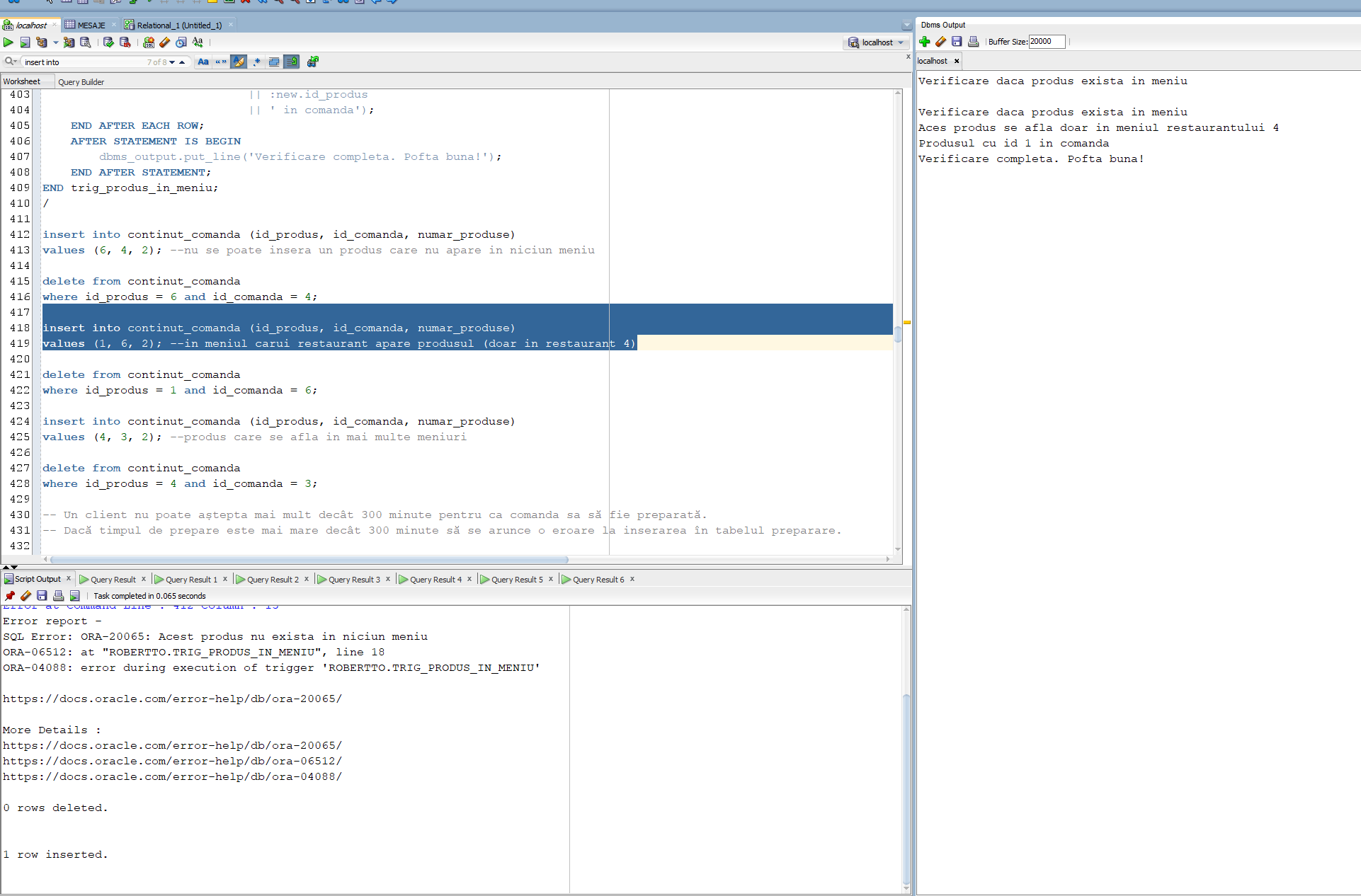
***Print-Screen:***

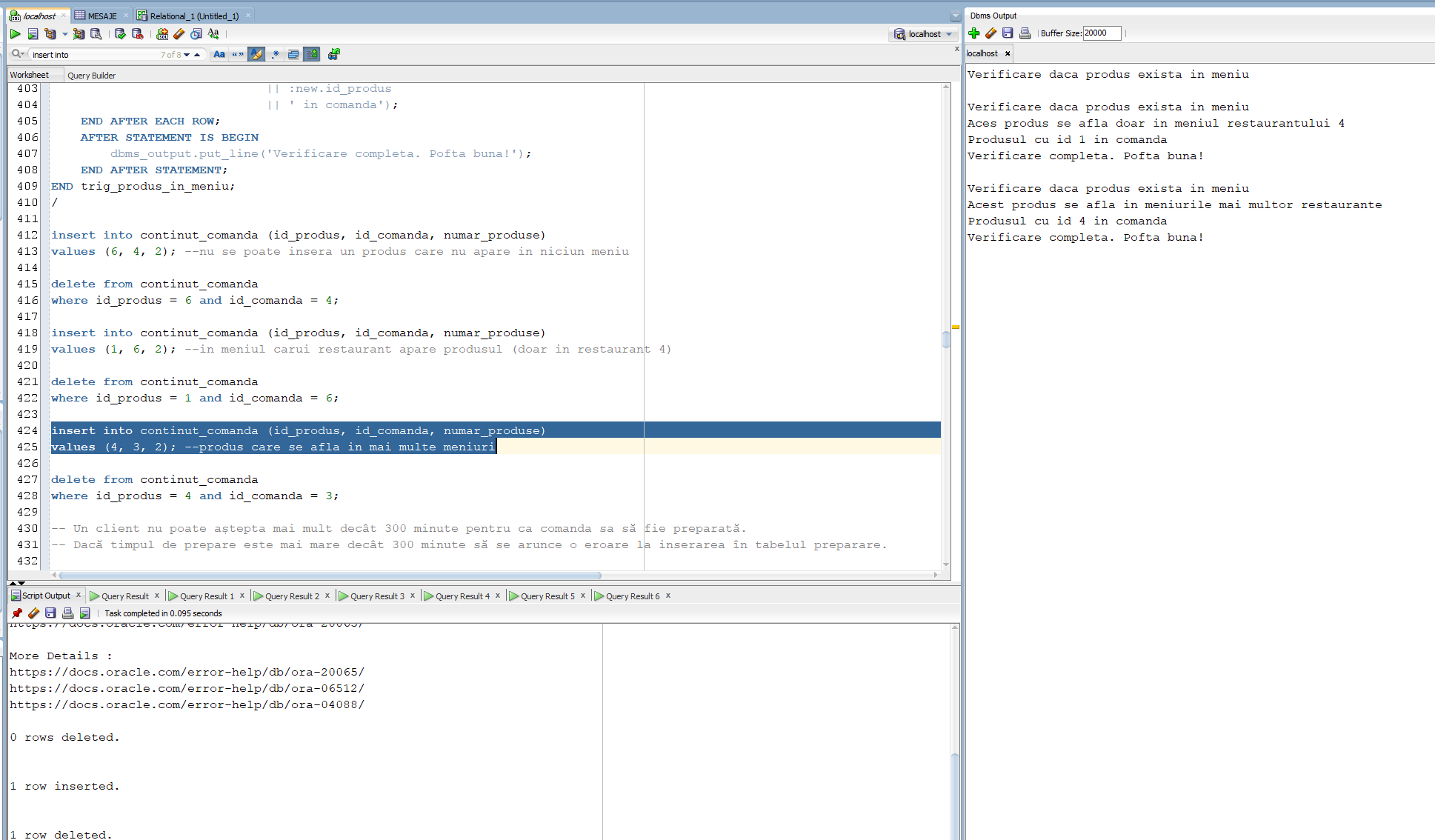


**Declanșarea acestuia:**

|  |
| --- |
| insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (6, 4, 2); --nu se poate insera un produs care nu apare in niciun meniu  delete from continut\_comanda  where id\_produs = 6 and id\_comanda = 4;  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (1, 6, 2); --in meniul carui restaurant apare produsul (doar in restaurant 4)  delete from continut\_comanda  where id\_produs = 1 and id\_comanda = 6;  insert into continut\_comanda (id\_produs, id\_comanda, numar\_produse)  values (4, 3, 2); --produs care se afla in mai multe meniuri  delete from continut\_comanda  where id\_produs = 4 and id\_comanda = 3; |





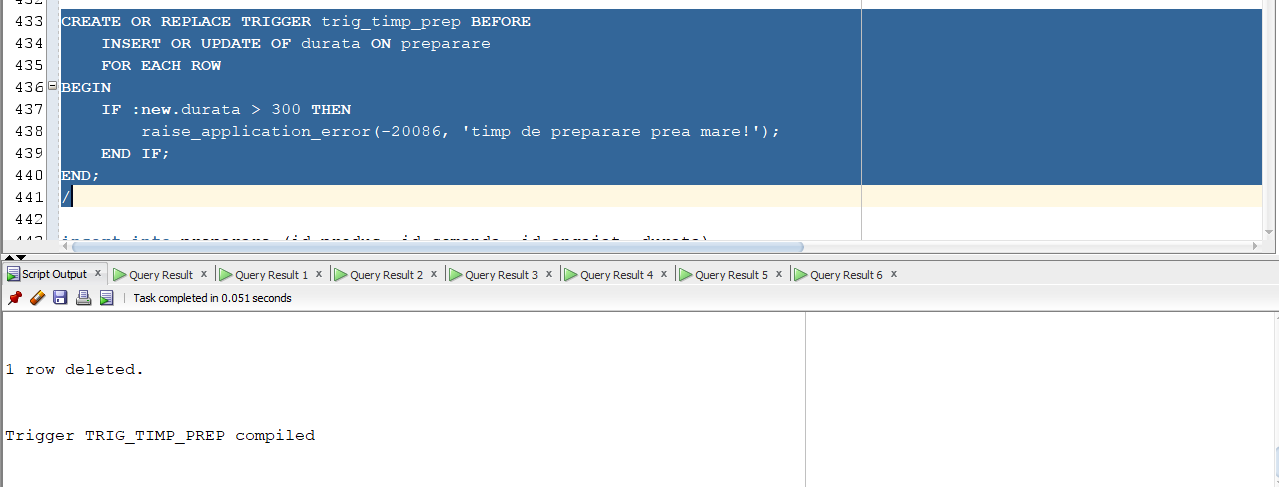


***Aici am mai făcut încă un Trigger:***

Un client nu poate aștepta mai mult decât 300 minute pentru ca comanda sa să fie preparată. Dacă timpul de prepare este mai mare decât 300 minute să se arunce o eroare la inserarea în tabelul preparare.

|  |
| --- |
| CREATE OR REPLACE TRIGGER trig\_timp\_prep BEFORE  INSERT OR UPDATE OF durata ON preparare  FOR EACH ROW  BEGIN  IF :new.durata > 300 THEN  raise\_application\_error(-20086, 'timp de preparare prea mare!');  END IF;  END;  / |

***Print-Screen:***



***L-am declanșat:***

|  |
| --- |
| insert into preparare (id\_produs, id\_comanda, id\_angajat, durata)  values (1, 1, 17, 600); |

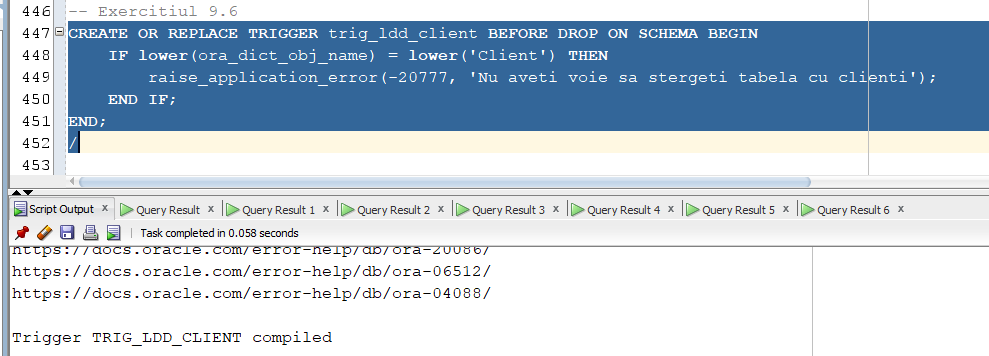
***Print-Screen:***



***9.6 Crearea unui Trigger LDD.***

|  |
| --- |
| CREATE OR REPLACE TRIGGER trig\_ldd\_client BEFORE DROP ON SCHEMA BEGIN  IF lower(ora\_dict\_obj\_name) = lower('Client') THEN  raise\_application\_error(-20777, 'Nu aveti voie sa stergeti tabela cu clienti');  END IF;  END;  / |

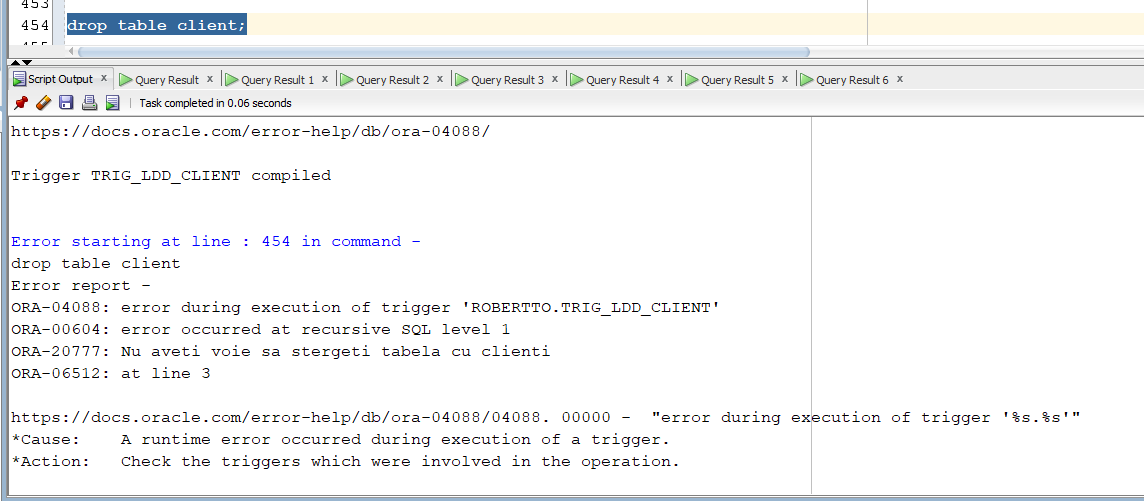
***Print-Screen:***



**Declanșarea acestuia:**

|  |
| --- |
| drop table client; |

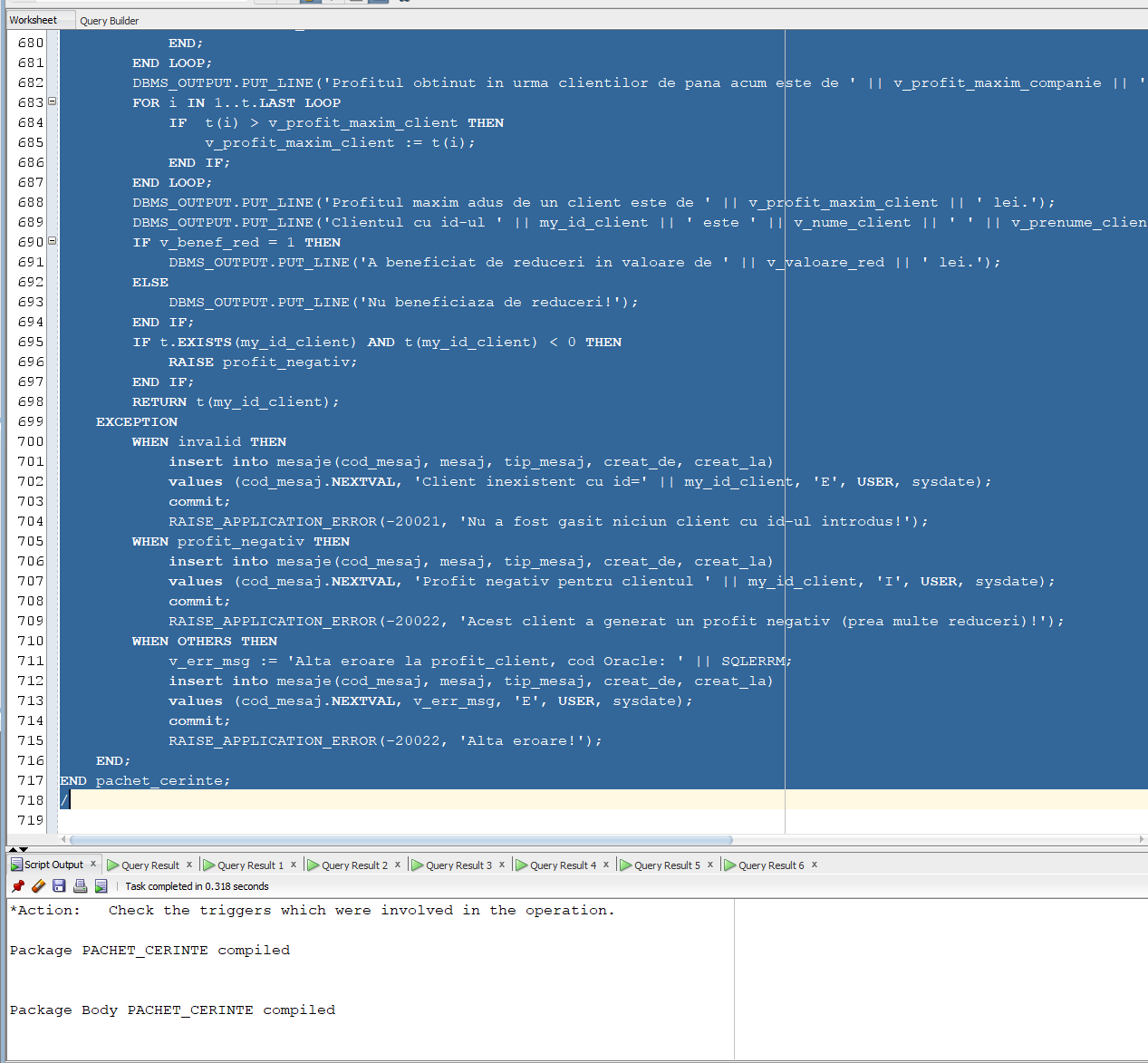
***Print-Screen:***



***9.7 Crearea unui unui pachet pentru obiectele definite.***

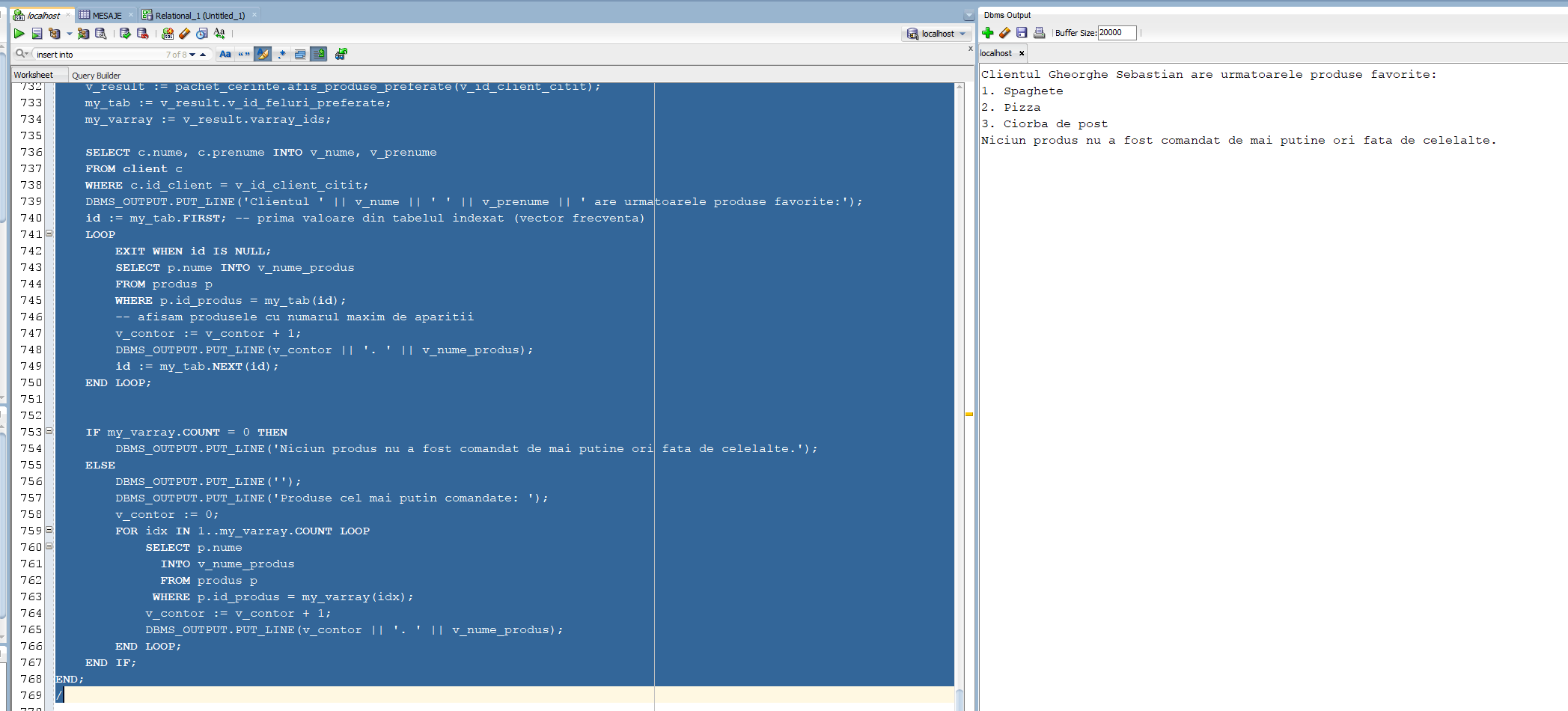
|  |
| --- |
| CREATE OR REPLACE PACKAGE pachet\_cerinte IS  FUNCTION afis\_produse\_preferate (my\_id\_client IN client.id\_client%TYPE) RETURN result\_obj;  PROCEDURE afis\_durata\_preparare\_produs;  FUNCTION profit\_client (my\_id\_client IN client.id\_client%TYPE) RETURN PLS\_INTEGER;  END pachet\_cerinte;  /  CREATE OR REPLACE PACKAGE BODY pachet\_cerinte AS  FUNCTION afis\_produse\_preferate (my\_id\_client IN client.id\_client%TYPE)  RETURN result\_obj IS  TYPE tab\_ind IS TABLE OF PLS\_INTEGER INDEX BY PLS\_INTEGER;  v\_contor tab\_ind;  varray\_ids t\_array;  v\_index\_max PLS\_INTEGER;  v\_index\_min PLS\_INTEGER;  id PLS\_INTEGER;  v\_id\_feluri\_preferate tab\_imb;  ok BINARY\_INTEGER := 0;  client\_fara\_comenzi EXCEPTION;  v\_err\_msg VARCHAR2(4000);  BEGIN  v\_id\_feluri\_preferate := tab\_imb();  varray\_ids := t\_array();    FOR i IN (SELECT p.id\_produs, cont.numar\_produse -- CURSOR IMPLICIT  FROM produs p, continut\_comanda cont, comanda cmd  WHERE my\_id\_client = cmd.id\_client  AND cont.id\_produs = p.id\_produs  AND cmd.id\_comanda = cont.id\_comanda) LOOP BEGIN  v\_contor(i.id\_produs) := v\_contor(i.id\_produs) + i.numar\_produse; -- simulez un vector de frecventa, unde tin de cate ori a fost comandat produsul respectiv  IF ok = 1 THEN  IF v\_contor(i.id\_produs) > v\_contor(v\_index\_max) THEN  v\_index\_max := i.id\_produs;  END IF;  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN -- intr-un tabel indexat daca nu exista indexul respectiv  -- se arunca exceptia no data found  v\_contor(i.id\_produs) := i.numar\_produse; -- initializam numarul de aparitii cu 1  IF ok = 0 THEN -- daca e primul produs selectat de cursor  ok := 1;  v\_index\_max := i.id\_produs;  END IF;  END;  END LOOP;  IF ok = 0 THEN  RAISE client\_fara\_comenzi;  ELSE  v\_index\_min := v\_contor.FIRST;    IF v\_index\_min IS NOT NULL THEN  id := v\_index\_min;  LOOP  EXIT WHEN id IS NULL;  IF v\_contor(id) < v\_contor(v\_index\_min) THEN  v\_index\_min := id;  END IF;    id := v\_contor.NEXT(id);  END LOOP;  END IF;    IF v\_contor(v\_index\_max) != v\_contor(v\_index\_min) THEN  -- DBMS\_OUTPUT.PUT\_LINE('v\_index\_max: ' || v\_index\_max);  -- DBMS\_OUTPUT.PUT\_LINE('v\_contor(v\_index\_max): ' || v\_contor(v\_index\_max));  -- DBMS\_OUTPUT.PUT\_LINE('v\_index\_min: ' || v\_index\_min);  -- DBMS\_OUTPUT.PUT\_LINE('v\_contor(v\_index\_min): ' || v\_contor(v\_index\_min));  id := v\_contor.first; -- prima valoare din tabelul indexat (vector frecventa)  LOOP  EXIT WHEN id IS NULL;  IF v\_contor(id) = v\_contor(v\_index\_max) THEN  v\_id\_feluri\_preferate.extend;  v\_id\_feluri\_preferate(v\_id\_feluri\_preferate.LAST) := id;  END IF;    IF v\_contor(id) = v\_contor(v\_index\_min) THEN  IF varray\_ids.COUNT < varray\_ids.LIMIT THEN  varray\_ids.EXTEND;  varray\_ids(varray\_ids.LAST) := id;  END IF;  END IF;  id := v\_contor.NEXT(id);  END LOOP;  ELSE  id := v\_contor.first; -- prima valoare din tabelul indexat (vector frecventa)  LOOP  EXIT WHEN id IS NULL;  IF v\_contor(id) = v\_contor(v\_index\_max) THEN  v\_id\_feluri\_preferate.extend;  v\_id\_feluri\_preferate(v\_id\_feluri\_preferate.LAST) := id;  END IF;  id := v\_contor.NEXT(id);  END LOOP;  END IF;  END IF;  RETURN result\_obj(v\_id\_feluri\_preferate, varray\_ids);    EXCEPTION  WHEN client\_fara\_comenzi THEN  INSERT INTO mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  VALUES (cod\_mesaj.NEXTVAL, 'Pentru clientul dat de la tastatura cu id ' || my\_id\_client || ' nu s-a gasit nicio comanda!', 'W', USER, SYSDATE);  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Acest client nu a comandat nimic in viata lui!');  WHEN OTHERS THEN  v\_err\_msg := 'Alta eroare in afis\_produse\_preferate: ' || SQLERRM;  INSERT INTO mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  VALUES (cod\_mesaj.NEXTVAL, 'Alta eroare in afis\_produse\_preferate: ' || v\_err\_msg, 'E', USER, SYSDATE);  COMMIT;  END;    PROCEDURE afis\_durata\_preparare\_produs  IS  CURSOR c\_bucatari IS SELECT a.id\_angajat, a.nume, a.prenume, b.nr\_stele  FROM angajat a  JOIN bucatar b ON a.id\_angajat = b.id\_angajat;  CURSOR c\_preparate(p\_id\_angajat in angajat.id\_angajat%TYPE) IS SELECT p.id\_produs, p.nume, p.descriere, prep.durata  FROM preparare prep  JOIN produs p ON p.id\_produs = prep.id\_produs  WHERE prep.id\_angajat = p\_id\_angajat;  TYPE rec IS RECORD (id\_ang angajat.id\_angajat%TYPE, dur preparare.durata%TYPE);  TYPE tab\_ind IS TABLE OF rec INDEX BY PLS\_INTEGER;  t tab\_ind;  v\_id\_bucatar angajat.id\_angajat%TYPE;  v\_nume\_bucatar angajat.nume%TYPE;  v\_prenume\_bucatar angajat.prenume%TYPE;  v\_nr\_stele\_bucatar bucatar.nr\_stele%TYPE;  v\_durata\_preparare preparare.durata%TYPE;  v\_id\_produs produs.id\_produs%TYPE;  v\_nume\_produs produs.nume%TYPE;  v\_descriere\_produs produs.descriere%TYPE;  BEGIN  OPEN c\_bucatari;  LOOP  FETCH c\_bucatari INTO v\_id\_bucatar, v\_nume\_bucatar, v\_prenume\_bucatar, v\_nr\_stele\_bucatar;  EXIT WHEN c\_bucatari%NOTFOUND;    OPEN c\_preparate(v\_id\_bucatar);  FETCH c\_preparate INTO v\_id\_produs, v\_nume\_produs, v\_descriere\_produs, v\_durata\_preparare;    IF c\_preparate%NOTFOUND THEN  DBMS\_OUTPUT.PUT\_LINE('ATENȚIE! Bucătarul ' || v\_nume\_bucatar || ' ' || v\_prenume\_bucatar || ' nu a preparat niciodată un produs!');  DBMS\_OUTPUT.NEW\_LINE();  ELSE  DBMS\_OUTPUT.PUT\_LINE('Bucătarul ' || v\_nume\_bucatar || ' ' || v\_prenume\_bucatar || ' a preparat următoarele produse:');    WHILE c\_preparate%FOUND  LOOP  DBMS\_OUTPUT.PUT\_LINE(c\_preparate%ROWCOUNT || '. Produsul: ' || v\_nume\_produs || ' ( ' || v\_descriere\_produs || ' ) a durat ' || v\_durata\_preparare || ' minute.');  IF t.EXISTS(v\_id\_produs) THEN  IF v\_durata\_preparare < t(v\_id\_produs).dur THEN  t(v\_id\_produs) := rec(v\_id\_bucatar, v\_durata\_preparare);  END IF;  ELSE  t(v\_id\_produs) := rec(v\_id\_bucatar, v\_durata\_preparare);  END IF;  FETCH c\_preparate INTO v\_id\_produs, v\_nume\_produs, v\_descriere\_produs, v\_durata\_preparare;  END LOOP;  END IF;  CLOSE c\_preparate;  END LOOP;  CLOSE c\_bucatari;    FOR i IN t.FIRST..t.LAST LOOP  SELECT nume INTO v\_nume\_produs  FROM produs  WHERE id\_produs = i;    SELECT a.nume, a.prenume INTO v\_nume\_bucatar, v\_prenume\_bucatar  FROM angajat a, bucatar b  WHERE a.id\_angajat = t(i).id\_ang  AND a.id\_angajat = b.id\_angajat;  DBMS\_OUTPUT.PUT\_LINE('Bucatarul ' || v\_nume\_bucatar || ' ' || v\_prenume\_bucatar || ' a preparat produsul ' || v\_nume\_produs || ' in ' || t(i).dur || ' minute(cel mai scurt timp).');  END LOOP;  END;  FUNCTION profit\_client (my\_id\_client IN client.id\_client%TYPE)  RETURN PLS\_INTEGER IS  TYPE tab\_ind IS TABLE OF PLS\_INTEGER INDEX BY PLS\_INTEGER;  v\_nume\_client client.nume%TYPE;  v\_prenume\_client client.nume%TYPE;  v\_nr\_telefon\_client client.nr\_telefon%TYPE;  v\_profit\_maxim\_companie PLS\_INTEGER := 0;  v\_profit\_maxim\_client PLS\_INTEGER := 0;  v\_benef\_red BINARY\_INTEGER := 0;  v\_valoare\_red PLS\_INTEGER := 0;  invalid EXCEPTION;  profit\_negativ EXCEPTION;  v\_exista PLS\_INTEGER := 0;  t tab\_ind;  v\_err\_msg VARCHAR2(4000);  BEGIN  SELECT COUNT(\*) INTO v\_exista  FROM client  WHERE id\_client = my\_id\_client;  IF v\_exista = 0 THEN  RAISE invalid;  END IF;    SELECT nume, prenume, nr\_telefon INTO v\_nume\_client, v\_prenume\_client, v\_nr\_telefon\_client  FROM client  WHERE id\_client = my\_id\_client;    FOR i IN (SELECT cli.id\_client, cli.nume, cli.prenume, cmd.pret, f.valoare, f.detalii  FROM client cli, comanda cmd, factura f  WHERE cli.id\_client = cmd.id\_client  AND cmd.id\_factura = f.id\_factura) LOOP  BEGIN  IF i.id\_client = my\_id\_client AND i.valoare < i.pret THEN  v\_benef\_red := 1;  v\_valoare\_red := v\_valoare\_red + (i.pret - i.valoare);  -- ELSIF i.id\_client = my\_id\_client THEN  -- t(i.id\_client) := t(i.id\_client) + (i.valoare - i.pret);  END IF;  v\_profit\_maxim\_companie := v\_profit\_maxim\_companie + (i.valoare - i.pret);  t(i.id\_client) := t(i.id\_client) + (i.valoare - i.pret);  EXCEPTION -- pentru prima inserare in tabel  WHEN NO\_DATA\_FOUND THEN  insert into mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  values (cod\_mesaj.NEXTVAL, 'Prima inserare in tabelul pentru clientul cu id: ' || i.id\_client, 'I', USER, sysdate);  commit;  t(i.id\_client) := i.valoare - i.pret;  END;  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('Profitul obtinut in urma clientilor de pana acum este de ' || v\_profit\_maxim\_companie || ' lei.');  FOR i IN 1..t.LAST LOOP  IF t(i) > v\_profit\_maxim\_client THEN  v\_profit\_maxim\_client := t(i);  END IF;  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('Profitul maxim adus de un client este de ' || v\_profit\_maxim\_client || ' lei.');  DBMS\_OUTPUT.PUT\_LINE('Clientul cu id-ul ' || my\_id\_client || ' este ' || v\_nume\_client || ' ' || v\_prenume\_client || ' cu nr de telefon: ' || v\_nr\_telefon\_client || '.');  IF v\_benef\_red = 1 THEN  DBMS\_OUTPUT.PUT\_LINE('A beneficiat de reduceri in valoare de ' || v\_valoare\_red || ' lei.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Nu beneficiaza de reduceri!');  END IF;  IF t.EXISTS(my\_id\_client) AND t(my\_id\_client) < 0 THEN  RAISE profit\_negativ;  END IF;  RETURN t(my\_id\_client);  EXCEPTION  WHEN invalid THEN  insert into mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  values (cod\_mesaj.NEXTVAL, 'Client inexistent cu id=' || my\_id\_client, 'E', USER, sysdate);  commit;  RAISE\_APPLICATION\_ERROR(-20021, 'Nu a fost gasit niciun client cu id-ul introdus!');  WHEN profit\_negativ THEN  insert into mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  values (cod\_mesaj.NEXTVAL, 'Profit negativ pentru clientul ' || my\_id\_client, 'I', USER, sysdate);  commit;  RAISE\_APPLICATION\_ERROR(-20022, 'Acest client a generat un profit negativ (prea multe reduceri)!');  WHEN OTHERS THEN  v\_err\_msg := 'Alta eroare la profit\_client, cod Oracle: ' || SQLERRM;  insert into mesaje(cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)  values (cod\_mesaj.NEXTVAL, v\_err\_msg, 'E', USER, sysdate);  commit;  RAISE\_APPLICATION\_ERROR(-20022, 'Alta eroare!');  END;  END pachet\_cerinte;  / |

***Print-Screen:***

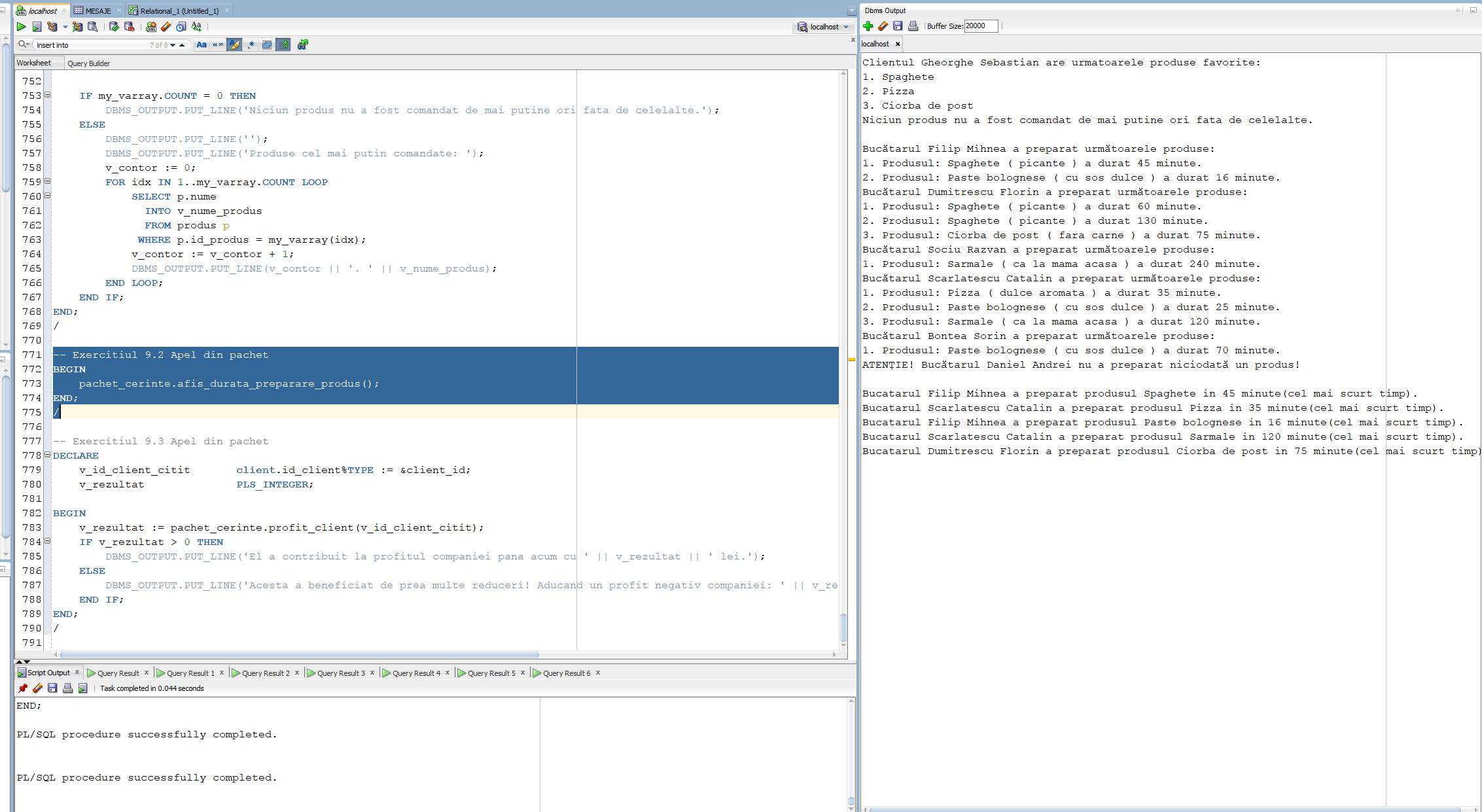


***Apelare din pachet:***

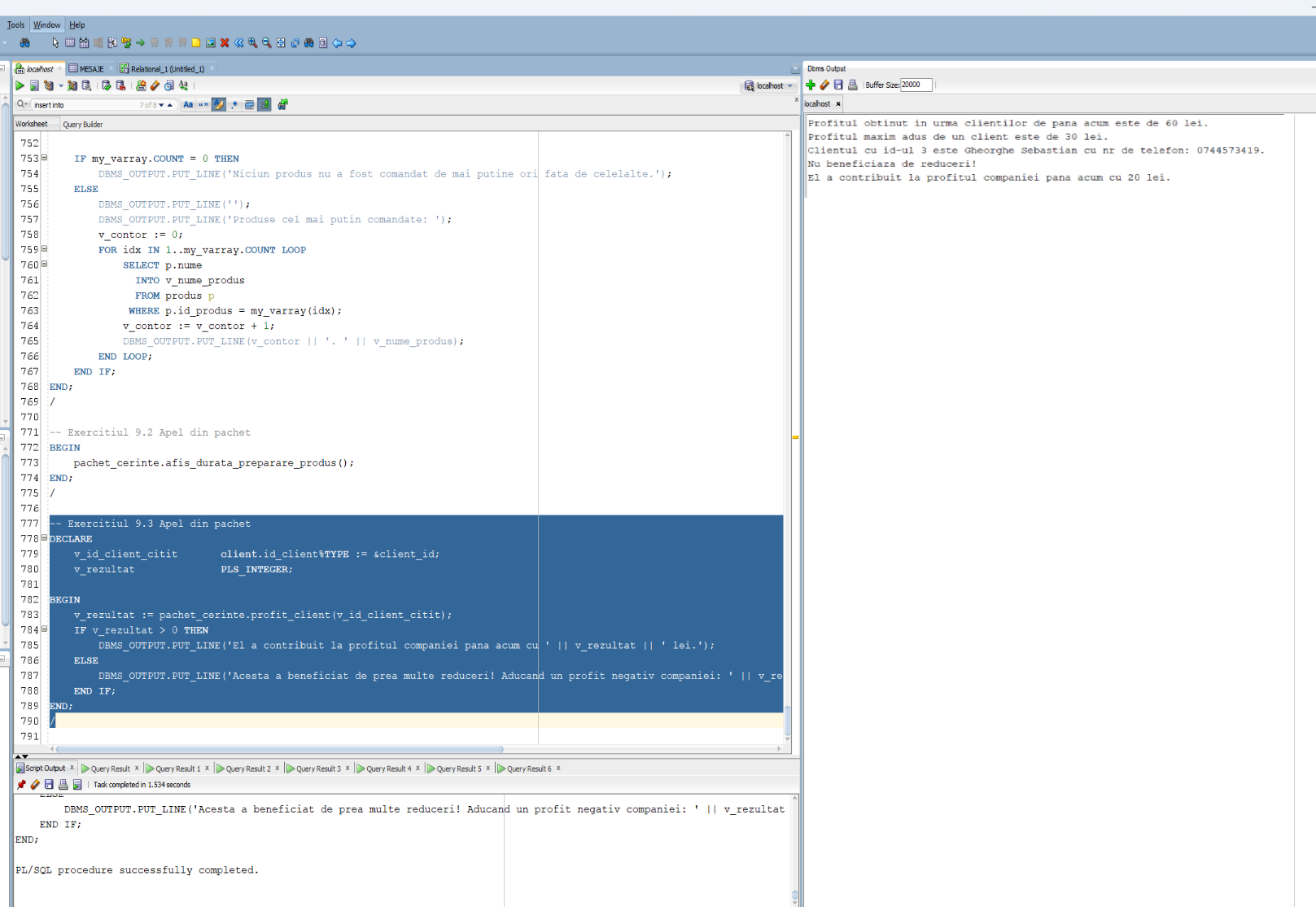
|  |
| --- |
| -- Exercitiul 9.1 Apel din pachet  DECLARE  v\_id\_client\_citit client.id\_client%TYPE := &client\_id;  v\_result result\_obj;  my\_tab tab\_imb;  my\_varray t\_array;  id PLS\_INTEGER;  v\_nume\_produs produs.nume%TYPE;  v\_nume client.nume%TYPE;  v\_prenume client.prenume%TYPE;  v\_contor PLS\_INTEGER := 0;  BEGIN  v\_result := pachet\_cerinte.afis\_produse\_preferate(v\_id\_client\_citit);  my\_tab := v\_result.v\_id\_feluri\_preferate;  my\_varray := v\_result.varray\_ids;    SELECT c.nume, c.prenume INTO v\_nume, v\_prenume  FROM client c  WHERE c.id\_client = v\_id\_client\_citit;  DBMS\_OUTPUT.PUT\_LINE('Clientul ' || v\_nume || ' ' || v\_prenume || ' are urmatoarele produse favorite:');  id := my\_tab.FIRST; -- prima valoare din tabelul indexat (vector frecventa)  LOOP  EXIT WHEN id IS NULL;  SELECT p.nume INTO v\_nume\_produs  FROM produs p  WHERE p.id\_produs = my\_tab(id);  -- afisam produsele cu numarul maxim de aparitii  v\_contor := v\_contor + 1;  DBMS\_OUTPUT.PUT\_LINE(v\_contor || '. ' || v\_nume\_produs);  id := my\_tab.NEXT(id);  END LOOP;    IF my\_varray.COUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('Niciun produs nu a fost comandat de mai putine ori fata de celelalte.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('Produse cel mai putin comandate: ');  v\_contor := 0;  FOR idx IN 1..my\_varray.COUNT LOOP  SELECT p.nume  INTO v\_nume\_produs  FROM produs p  WHERE p.id\_produs = my\_varray(idx);  v\_contor := v\_contor + 1;  DBMS\_OUTPUT.PUT\_LINE(v\_contor || '. ' || v\_nume\_produs);  END LOOP;  END IF;  END;  / |



|  |
| --- |
| -- Exercitiul 9.2 Apel din pachet  BEGIN  pachet\_cerinte.afis\_durata\_preparare\_produs();  END;  / |

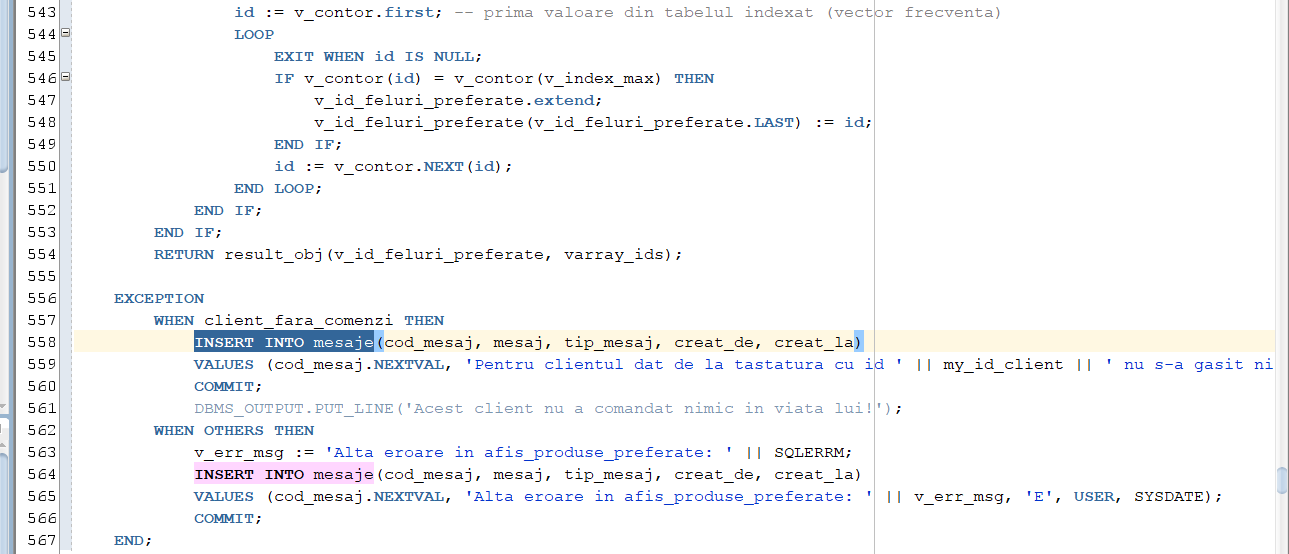


|  |
| --- |
| -- Exercitiul 9.3 Apel din pachet  DECLARE  v\_id\_client\_citit client.id\_client%TYPE := &client\_id;  v\_rezultat PLS\_INTEGER;    BEGIN  v\_rezultat := pachet\_cerinte.profit\_client(v\_id\_client\_citit);  IF v\_rezultat > 0 THEN  DBMS\_OUTPUT.PUT\_LINE('El a contribuit la profitul companiei pana acum cu ' || v\_rezultat || ' lei.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Acesta a beneficiat de prea multe reduceri! Aducand un profit negativ companiei: ' || v\_rezultat || ' lei.');  END IF;  END;  / |

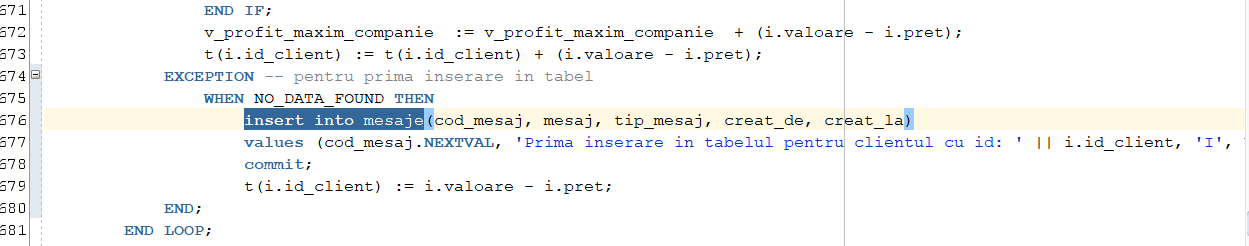
â

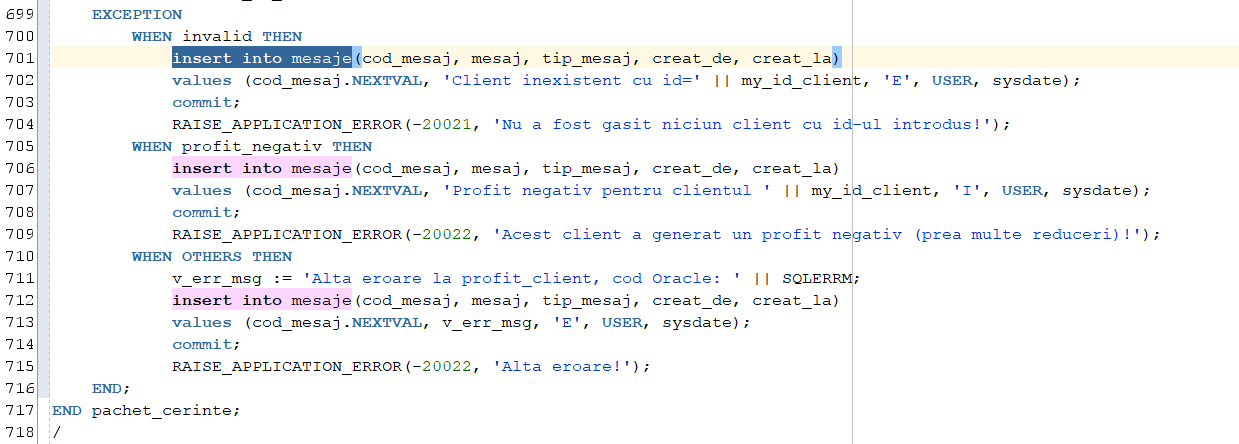
**10. Inserarea excepțiilor în tabelul mesaje:**

La exercițiul 9.1:



La exercițiul 9.3:





Mesajele stocate în tabelul mesaje:

